



Fakultät für Informatik

Lehrstuhl für Bildverarbeitung und Mustererkennung

# Convex Relaxation of Variational Models with Applications in Image Analysis

**Evgeny Strekalovskiy**

Vollständiger Abdruck der von der Fakultät für Informatik  
der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktors der Naturwissenschaften**

genehmigten Dissertation.

Vorsitzender:

Univ.-Prof. Dr. Nils Thuerey

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Daniel Cremers
2. Prof. Dr. Antonin Chambolle  
École Polytechnique, Paris/Frankreich

Die Dissertation wurde am 02.07.2015 bei der Technischen Universität München  
eingereicht und durch die Fakultät für Informatik am 22.10.2015 angenommen.



*Für meine Mutter.*



# Zusammenfassung

Variationelle Methoden sind zu einem etablierten Modellierungsmittel in der Bildverarbeitung geworden. Beliebte Einsatzgebiete sind Bildentrauschung, Segmentierung, Tiefenrekonstruktion, Berechnung des optischen Flusses und 3D-Rekonstruktion. Die meisten Energien sind jedoch nicht konvex, was ihre Optimierung deutlich erschwert. Lokale Methoden liefern dann mit großer Wahrscheinlichkeit suboptimale Lösungen, die auch von der Initialisierung abhängen.

In dieser Arbeit präsentieren wir neue Konvexifizierungsmethoden für mehrere allgemeine Problemklassen in der Bildverarbeitung. Diese ermöglichen es, Lösungen zu erhalten, die unabhängig von der Initialisierung sind und in einer verifizierbaren Nähe des Optimums liegen. Dadurch nähert man sich dem Ziel, für praktische Bildverarbeitungsprobleme zuverlässige und automatische Algorithmen zu entwickeln. Wir erweitern und verallgemeinern bestehende Techniken zur Konvexifizierung von Energien. Für jeden betrachteten Fall schlagen wir spezielle Relaxierungen vor, die nah an den optimalen und dennoch effizient berechenbar sind. Der Fokus der Beiträge dieser Arbeit liegt auf zwei allgemeinen Energieklassen: Multilabel-Segmentierung und vektorielle Funktionale.

Der erste Teil der Arbeit beschäftigt sich mit Multilabelproblemen. Dies sind Probleme, in denen die gesuchte Lösungsfunktion in jedem Bildpunkt einen von nur endlich vielen vordefinierten Werten annehmen kann. Diese Energien stehen im Zentrum von fundamentalen Problemen der Bildverarbeitung wie Segmentierung und Berechnung des optischen Flusses. Zum anderen sind sie jedoch sehr schwierig zu optimieren, da sie nach der Diskretisierung meist auf NP-schwere Probleme hinausführen. Wir geben einen Überblick über die modernen Ansätze zur Lösung von Multilabelproblemen und präsentieren anschließend neue effiziente konvexe Relaxierungen für drei spezielle Anwendungen. In jedem Fall gehen wir auf die verschiedenen Weisen ein, wie High-Level-Wissen über die wahrscheinlichen Labelkonfigurationen effizient integriert werden kann.

Zuerst betrachten wir die Multilabelsegmentierung mit Längenregularisierung. Hier bestimmt eine Labeltransferfunktion die Wahrscheinlichkeit, dass zwei verschiedene Labels benachbart sind. Klassische Methoden gehen von der Annahme aus, dass diese Funktion eine Metrik ist. Wir schlagen einen neuen Ansatz vor, der diese Annahme nicht braucht und stattdessen allgemeine Transferfunktionen erlaubt. Dies ermöglicht es, allgemeinere Energien zu betrachten, und verbessert die Segmentierungen im Vergleich zum Metrikfall. Als zweites beschreiben wir, wie man Bedingungen an die relative geometrische Lage der Labelregionen auf eine konvexe Weise formulieren kann. Dadurch wird die Modellierung von Labelanordnungen in der Bildebene möglich. Unser Ansatz ver-

einheitlich unterschiedliche existierende Lösungen von drei speziellen Anordnungsproblemen und lässt zudem verschiedene Verallgemeinerungen wie die Bevorzugung der Konvexform zu. Als drittes betrachten wir die Segmentierung von Bildsequenzen. Die Aufgabe ist es, ein bestimmtes Objekt gleichzeitig in einer Vielzahl von Bildern zu segmentieren. Zu diesem Zweck schlagen wir Proportionalitätsregularisierer vor. Diese zielen darauf ab, die relative Größe von je zwei Einzelteilen des Objekts über alle Bilder hinweg im Wesentlichen konstant zu halten. In jedem Einzelfall betrachten wir verschiedene Weisen zu einer Konvexifizierung zu gelangen. Die proportionalitätserhaltenden Regularisierer führen zur Robustheit bezüglich starker Form- und Größenänderungen des gesuchten Objekts, was eine genauere Segmentierungen ermöglicht.

Im zweiten Teil der Arbeit betrachten wir Energieminimierungsprobleme, in denen die gesuchte Lösungsfunktion vektorwertig ist und einen kontinuierlichen Wertebereich hat. Solche Energien tauchen in der Praxis oft auf, z.B. im Zusammenhang mit Farbbildentrauschung, Tiefenrekonstruktion oder dem optischen Fluss. Im skalaren Fall kann der "Functional-Lifting" Ansatz benutzt werden, um eine große Klasse von Energien auf eine effiziente Weise zu konvexifizieren. Jedoch lässt sich dieses leistungsfähige Verfahren nicht auf den vektoriellen Fall verallgemeinern, so dass die Optimierung von vektoriellen Energien immer noch eine Herausforderung darstellt. Wir betrachten drei Spezialfälle von vektoriellen Funktionalen und schlagen jeweils individuell entwickelte konvexe Relaxierungen vor, die auf dem skalaren "Functional-Lifting" Ansatz beruhen.

Im ersten Spezialfall betrachten wir separable Regularisierer. In diesen wird die Kopplung zwischen den einzelnen Kanälen der vektorwertigen Lösungsfunktion nur über den Datenterm hergestellt. Wir schlagen eine effiziente, auf einer neuen Reduzierungstechnik beruhende Relaxierung vor. Im Vergleich zu bestehenden Methoden werden der Speicherverbrauch und die Berechnungszeit dadurch um Größenordnungen reduziert. Diese Technik erlaubt es uns, neue und größere Probleme zu betrachten, die bisher mit den früheren Methoden aufgrund des zu hohen Ressourcenbedarfs nicht lösbar waren. Ein wichtiges Beispiel ist die Berechnung des optischen Flusses in den Fällen, die eine Bewegung über große Bildbereiche hinweg aufweisen. Im zweiten Spezialfall betrachten wir Regularisierer mit einer speziellen Kanalkopplung. Wir stellen eine neue Relaxierung vor, die in der Klasse der effizient berechenbaren eine der optimalen ist. Dadurch wird es zum ersten Mal möglich, einen nichtkonvexen Datenterm in Kombination mit einem koppelnden Regularisierer zu optimieren. Ein wichtiges Anwendungsbeispiel dieser Methode ist die vektorielle Totalvariation. Als dritten Spezialfall schlagen wir die erste effizient umsetzbare konvexe Relaxierung für das vektorielle Mumford-Shah-Funktional vor. Dadurch kann man bei der Lösungsberechnung die Kanalkopplung unter im Wesentlichen den gleichen Kosten wie im skalaren Fall miteinbeziehen. Dies liefert hochqualitative Lösungen für kontrasterhaltende und kantenverstärkende Regularisierung.

Die Doktorarbeit demonstriert die allgemeinen Vorteile von Konvexifizierungsmethoden wie Optimalitätsgarantien, Initialisierungsunabhängigkeit, und Parallelisierung von resultierenden Algorithmen. Diese Methoden stellen so ein effektives Werkzeug zur Lösung von einer Vielzahl von Optimierungsaufgaben in der Bildverarbeitung dar.

# Summary

Variational methods have become an established way to approach a multitude of image analysis problems such as image denoising, segmentation, stereo reconstruction, optical flow and 3D reconstruction. However, most practical energies are not convex and thus hard to optimize, and local methods may lead to sub-optimal solutions and will depend on appropriate initialization.

In this work, we present novel convexification techniques for several classes of energies frequently encountered in imaging problems. This allows to obtain solutions which do not depend on initialization and are within a computable bound of the global optimum, thus coming closer to the goal of having reliable and automatic computer vision algorithms. We extend and generalize existing convexification techniques, and in each case strive to find convex relaxations as tight as possible but which still allow efficient optimization. The contributions of this thesis are concentrated on two broad areas of energies: Multilabel segmentation, and vectorial functionals.

The first part of the thesis deals with the multilabel problem. These are energies where the value of the solution at each pixel is constrained to a predefined finite set of possible values. Such energies lie at the heart of fundamental problems like image segmentation and optical flow, but their optimization is difficult since they typically lead to NP-hard problems after discretization. After giving an overview of the multilabel problem and the state-of-the-art approaches, subsequently we present efficient novel convex relaxations for three different types of priors derived from higher-level knowledge about likely label configurations.

As the first prior, we revisit the length regularized multi-object segmentation, where a label transition function defines which pairs of labels are likely to occur next to each other. While classical approaches assume this function to be a metric, we propose an approach which allows to remove this restriction and impose nonmetric distances between labels. These more general energies lead to improved results in comparison to the metric case. As the second prior, we describe how constraints on the relative geometric location of label regions can be cast in a convex way. This makes it possible to model geometric layouts of labels, and provides a unified solution to three different problems, while generalizing to new cases such as the convex shape prior. Third, we consider image sequence segmentation, where one and the same object is to be segmented throughout a potentially large series of images. We introduce proportion priors to ensure that the relative size of different objects or object parts remains mainly the same across all images, and propose corresponding convexifications. Such priors are shown to increase the robustness with respect to strong deformations

in shape and to size changes over the images, thus leading to more accurate segmentations.

The second part of the thesis is about energies defined on vectorial functions with a continuous range. Such energies naturally arise in problems such as color denoising, depth reconstruction and optical flow. While the general functional lifting convexification method can be used to obtain a convexification in the scalar case, it does not generalize to the vectorial case. Thus, optimization of vectorial energies remains a challenge. We focus on three special cases of vectorial functionals and propose an especially tailored convex relaxation in each case based on the functional lifting idea.

In the first case, we consider separable regularizers, so that there is no channel coupling except for the data term. We propose a reduction technique resulting in a relaxation with run time and memory requirements lower by orders of magnitude in comparison to previous methods. This enables to consider previously infeasible large-scale problems such as wide-range optical flow, while being able to provide optimality bounds on the solution. In the second case, we consider special coupling regularizers such as vectorial total variation and propose a convex relaxation which is as tight as possible in the class of tractable relaxations. For the first time this makes possible to optimize energies with a nonconvex data term and a coupling regularizer. As the third special case, we provide a first tractable convexification for the vectorial Mumford-Shah functional. The proposed relaxation allows one to take channel coupling into account at essentially the same cost as in the scalar case. This yields efficiently computable high-quality solutions for contrast-preserving and edge-enhancing regularization.

The developed methods highlight the general advantages of convexification methods such as optimality guarantees, independence of initialization, and the major parallelization capabilities of the resulting algorithms. This shows that these methods are a viable way to tackle important optimization problems in computer vision.



# Acknowledgments

Throughout the last years I had a fortunate situation to work with very talented people. This doctoral thesis would not have been possible without their help.

Foremost, I am very grateful to my advisor Daniel Cremers for his great personal guidance through his experience, inspiration and the many insightful and motivating discussions, for the continuous support throughout my doctoral studies, as well as for providing me with plenty of freedom and many opportunities for my development as a researcher.

I also wish to thank Antonin Chambolle for the fruitful collaboration and inspiring discussions, his pleasant hospitality during my two months research stay at École Polytechnique, for being my second supervisor, as well as for always being ready to help. My sincere thank goes to Thomas Pock for sharing his experience on convexification and primal-dual methods and for many valuable comments. Additionally, I want to thank Olga Veksler for fruitful and enjoyable conversations as well as for the continuous support.

I would like to thank all my paper co-authors the excellent cooperation. In particular, I thank Bastian Goldlücke for our joint work on vectorial relaxations, and Claudia Nieuwenhuis for our research on multilabel problems. A substantial part of this thesis is based on joint work with both of them. Furthermore, I thank Jan Lellmann and Thomas Möllenhoff for the great collaboration. Thanks also to my colleagues in the Munich computer vision group for interesting discussions and for making my time there very enjoyable in and outside of office.

Finally, I would like to express my deep gratitude to my family for the constant support and love during these years.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Convex Relaxation Methods . . . . .	3
1.3	Contributions . . . . .	8
1.4	Publications . . . . .	12
<b>2</b>	<b>Fundamentals</b>	<b>15</b>
2.1	Convex Analysis . . . . .	15
2.2	Functions of Bounded Variation . . . . .	19
2.3	The Primal-Dual Algorithm . . . . .	23
<b>I</b>	<b>Multilabel Segmentation</b>	<b>33</b>
<b>3</b>	<b>The Multilabel Problem: Common Approaches</b>	<b>35</b>
3.1	Introduction . . . . .	35
3.2	Computational Approaches . . . . .	39
3.3	Convex Relaxation . . . . .	40
3.4	Relaxations for the Length Prior . . . . .	43
3.5	Implementation . . . . .	47
<b>4</b>	<b>Multilabel Segmentation with Nonmetric Priors</b>	<b>53</b>
4.1	Introduction . . . . .	53
4.2	Continuous Multilabel Optimization with Nonmetric Priors . . . . .	55
4.3	Implementation . . . . .	58
4.4	Experimental Results . . . . .	59
4.5	Conclusion . . . . .	64
4.6	Appendix: Proofs of Propositions and Theorems . . . . .	64
<b>5</b>	<b>Multilabel Segmentation with Ordering Constraints</b>	<b>67</b>
5.1	Introduction . . . . .	67
5.2	General Framework . . . . .	68
5.3	Constraint Sets . . . . .	75
5.4	Implementation . . . . .	76
5.5	Experimental Results . . . . .	78
5.6	Conclusion . . . . .	86
5.7	Appendix: Proofs of Propositions and Theorems . . . . .	86

<b>6</b>	<b>Multilabel Segmentation with Proportion Priors</b>	<b>89</b>
6.1	Introduction . . . . .	89
6.2	Multilabel Image Sequence Segmentation . . . . .	91
6.3	Proportion Preserving Priors . . . . .	93
6.4	Implementation . . . . .	95
6.5	Experimental Results . . . . .	98
6.6	Conclusion . . . . .	102
6.7	Appendix: Proofs and Proximal Operators . . . . .	103
<b>II</b>	<b>Vectorial Functionals</b>	<b>105</b>
<b>7</b>	<b>Functional Lifting: Convex Relaxation for Scalar Problems</b>	<b>107</b>
7.1	Introduction . . . . .	107
7.2	Functional Lifting . . . . .	109
7.3	Convex Relaxation . . . . .	113
7.4	Implementation . . . . .	115
<b>8</b>	<b>Vectorial Problems with Separable Regularization</b>	<b>119</b>
8.1	Introduction . . . . .	119
8.2	Multidimensional Label Spaces . . . . .	122
8.3	Convex Relaxation of the Data Term . . . . .	125
8.4	Convex Relaxation of the Regularizer . . . . .	128
8.5	Implementation . . . . .	134
8.6	Experimental Results . . . . .	140
8.7	Conclusion . . . . .	148
8.8	Appendix: Proofs of Propositions and Theorems . . . . .	149
<b>9</b>	<b>Vectorial Problems with Coupled Regularization</b>	<b>155</b>
9.1	Introduction . . . . .	155
9.2	Convex Relaxation . . . . .	157
9.3	General Vectorial Case . . . . .	159
9.4	Special Cases . . . . .	164
9.5	Implementation . . . . .	173
9.6	Experimental Results . . . . .	181
9.7	Conclusion . . . . .	190
9.8	Appendix: Proof of the Main Lemma . . . . .	190
9.9	Appendix: Projections . . . . .	194
<b>10</b>	<b>The Vectorial Mumford-Shah Functional</b>	<b>199</b>
10.1	Introduction . . . . .	199
10.2	The Convex Representation . . . . .	202
10.3	Implementation . . . . .	206
10.4	Experimental Results . . . . .	210
10.5	Conclusion . . . . .	214
<b>11</b>	<b>Conclusion</b>	<b>215</b>

# Chapter 1

## Introduction

### 1.1 Motivation

#### 1.1.1 Variational Methods

Starting in the 1980's *variational methods* have revolutionized the way to approach computer vision problems. Each candidate solution  $u$  is assigned a real number, which is called its *energy*  $E(u)$ . The idea is that lower energies should correspond to better quality solutions, and that the optimal solution is the *minimizer* of this cost function:

$$\min_u E(u). \tag{1.1}$$

This way, computer vision problems become optimization problems, and modeling the energy for each specific problem is a crucial part to obtain high-quality solutions. It becomes possible to look for solutions with specific desired properties: Candidates which fit the input data well or are regular in some way are assigned a lower energy, while those which are less fitting and irregular should have a higher energy. The energy is typically modelled as a sum of two terms, the *data term* and *regularization term*:

$$E(u) = E_{\text{data}}(u) + E_{\text{reg}}(u). \tag{1.2}$$

The first term describes how well the candidate  $u$  fits the input data. The second term is the prior, or regularity term, which sorts out unlikely “wild” candidates  $u$  in favor of more regular ones. The idea is that a minimizer  $u$  of the overall energy  $E$  should keep both of these terms low.

Variational methods have a number of key advantages, in contrast to heuristic procedural “step-by-step-manipulation” methods. Foremost, the model is mathematically transparent. There is a clear mathematical relation between inputs and outputs, and it is possible to derive and prove properties of the solutions depending on the inputs. Another advantage is that variational models usually have only few parameters.

Over the last decades, *Bayesian inference* [127] has become one of the established ways to model energies in an intuitive and direct way. Given input data

$I$ , one looks for an optimal solution  $u$  by maximizing the conditional probability of  $u$  given  $I$ , and rewrites this problem using the Bayes rule:

$$\max_u \mathcal{P}(u|I) = \max_u \frac{\mathcal{P}(I|u)\mathcal{P}(u)}{\mathcal{P}(I)}. \quad (1.3)$$

The denominator  $\mathcal{P}(I)$  is the same for each  $u$  and can thus be omitted in the optimization since we are only interested in the minimizer itself and not in the energy values. Taking the negative logarithm on both sides, we arrive at the corresponding energy (1.2) for  $u$  with

$$E_{\text{data}}(u) = -\log \mathcal{P}(I|u) \quad \text{and} \quad E_{\text{reg}}(u) = -\log \mathcal{P}(u). \quad (1.4)$$

Thus, the data term plays the role of a *generating model*: It is (the negative logarithm of) the probability to obtain the input data  $I$  given the underlying model  $u$ , for example in application of image denoising the probability of obtaining a noisy image given the clean one. Furthermore, the regularizer term is directly interpreted as a *prior term*, being (the negative logarithm of) the probability of each candidate solution  $u$ , independently of the input data  $I$ . This enables one to directly favor solutions with certain a-priori properties such as smoothness.

Applications of variational methods in computer vision and image analysis are extremely broad. Initial break-throughs popularizing this approach include, among others, variational formulations for optical flow estimation [59], image segmentation [13, 63, 92, 34, 64, 26] and multiview stereo reconstruction [45]. By now, functional optimization has become an established paradigm to solve a multitude of image analysis problems ranging from image denoising [100] and segmentation [33, 76, 29], to stereo [140, 101], optical flow estimation [23] and 3D reconstruction [38].

### 1.1.2 Discrete and Continuous Viewpoints

In practice, one always deals with digital images, which are discrete consisting of a regular grid of pixels. Thus it is natural to assume that images are given on a *discrete domain*, and to model computer vision problems directly on the underlying grid using the framework of Markov random fields (MRFs). Such models are very popular due to many efficient combinatorial algorithms which can be devised in this setting [17, 134, 70]. Nevertheless, digital images are only discrete representations of actually spatially continuous real-world objects. Because of this, ideally, computer vision and image processing algorithms should be independent of the underlying image grid. However, this is typically not the case for MRF based models. For example, in the case of image segmentation the resulting object boundary tends to consist of pieces aligned along the coordinate axes, a phenomenon known as “metrication error” or “grid bias” [68].

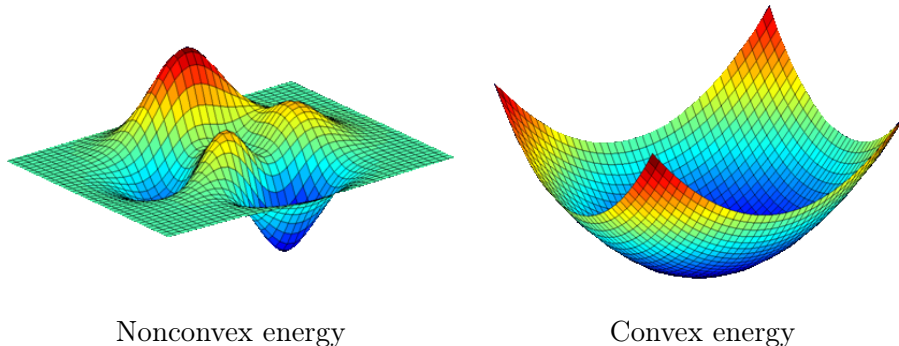
In this thesis we will focus on the *spatially continuous* viewpoint. This means that images are viewed as functions  $I : \Omega \rightarrow \mathbb{R}^k$  defined on a spatially continuous domain  $\Omega \subset \mathbb{R}^m$  (where  $k \geq 1$  is the number of channels and  $m \geq 1$  is the dimension of the image domain). Energies are modeled by means of integrals, which are independent of any discretization into a pixel grid. The discretization only occurs as the last step in order to actually compute the solutions. Spatially

continuous methods provide a powerful alternative to MRFs, as they typically require less memory [68], and are able to provide more accurate solutions for the underlying vision problems. Furthermore, spatially continuous models are amenable to *massive parallelization* after discretization of the image domain, so that they can be solved efficiently on graphics processing units (GPUs).

### 1.1.3 Local Minimization Methods

Unfortunately, most practical cost functions are very hard to optimize. The reason is that they are *not convex* and that the problem is usually very *high-dimensional*, see Figure 1.1. For instance, this is the case for all of the above mentioned pioneering variational approaches. Although general global optimization methods exist [50, 49, 13], they tend to be prohibitively slow for very high-dimensional problems and require a careful tuning of parameters.

Classical optimization techniques are usually based on local optimization via *gradient descent*. They also may utilize suitable approximations, e.g. the classical Horn-Schunck approach for optical flow [59] first linearizes the data term to make the overall problem tractable. Such local methods inevitably get trapped in *local minima* of the functional. This has crucial practical implications. First, the obtained solutions will potentially be of inferior quality. In general, there is no way to provide any guarantees concerning the optimality of the solution. Second, the solution will highly depend on the used initialization. This makes large data processing only possible under a constant interaction with a user who would provide appropriate initializations. As a consequence, local methods cannot be used in a fully automatic “black-box” manner to solve computer vision problems. As for *general* functionals, their practical usefulness is limited.



**Figure 1.1: Convexity is crucial for optimization.** While nonconvex energies exhibit numerous local minima so that local methods inevitably get trapped in suboptimal solutions, for convex energies any local minimum is automatically a global one.

## 1.2 Convex Relaxation Methods

The situation changes dramatically if the functional to minimize is *convex*, see Figure 1.1. The distinctive characteristic of convex models is that every local

minimum is automatically a global one. Such functionals can be efficiently minimized and even local methods now can be employed to solve the problem in a globally optimal way. However, most practical cost functions are nonconvex.

Starting around 2005, researchers have revisited the variational approaches and developed a variety of *convex relaxation* techniques. These aim at casting the respective computer vision problems in terms of convex functionals. In other words, one tries to replace the original nonconvex functional by a convex formulation. Convexification has a number of key advantages:

- one can compute *globally optimal solutions* (or solutions with bounded optimality),
- and these solutions are *independent of the initialization*.

One of the first convexification examples is the Chan et al. approach [33] for foreground-background segmentation, which yields globally optimal solutions. The advantages of convex models have led to a surge of new developments, with applications covering a large variety of practical problems, see Section 1.1.1.

### 1.2.1 Convex Relaxation

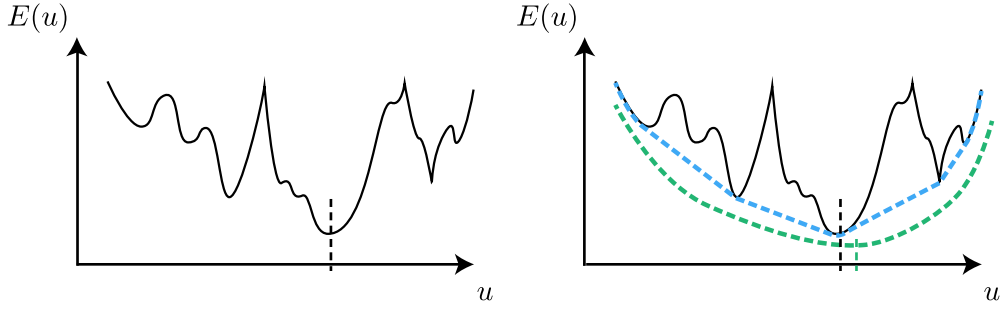
The method of convex relaxation, or convexification, is to try to find good solutions of nonconvex energies by finding suitable *convex energies* with *the same* global minimizers. Let us illustrate the idea on 1D examples.

**Continuous Optimization Domain.** Assume a nonconvex energy such as in Figure 1.2 (black solid line), which has many local minima. Clearly, gradient descent would need a good initial guess to arrive at the global minimum. The idea of convexification is to find an approximating convex energy with *the same* global minimum. This approximation is usually constructed as a convex lower bound. The *convex envelope* of the energy, i.e. the highest convex function below or equal to the original one, is the tightest such bound and always ensures that the global minima are the same, see the dashed blue line in Figure 1.2. To solve the original problem one now focuses on the minimization of the convex approximation, which is a tractable optimization task for which many efficient state-of-the-art methods can be employed.

In practical large-scale image processing applications, the cost functions are usually very high-dimensional and it is hard or currently even impossible to compute the exact convex envelope. One then must resort to less tight but still computable convex lower bounds, see the dashed green line in Figure 1.2. Still, many problem classes allow good convex approximations or representations, so that the original problem can be solved optimally or almost optimally.

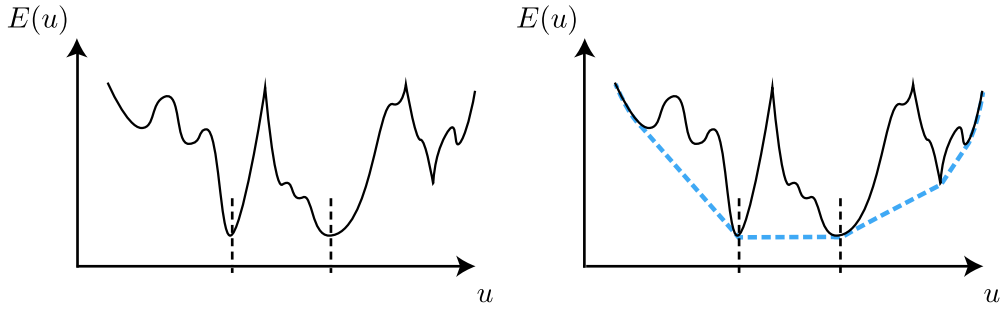
In the case that the minimum of the original functional is not unique, the convex envelope is minimized by all arguments which lie in the convex hull of the original minimizers, see Figure 1.3. This means that the solution obtained through convex relaxation in general can be a convex combination of true solutions, e.g. a mixture of two solutions. However, this rarely happens in practice.





**Figure 1.2: Convex relaxation idea.** While the original nonconvex function (*black*) has many local minima, its convex envelope (*blue*) only has global ones. Minimizing the latter automatically minimizes the former. A less tight convex relaxation is shown as a *green line*.

Due to the data term, which gives the local value preference of the solution at each pixel, the solution will be unique in most cases.

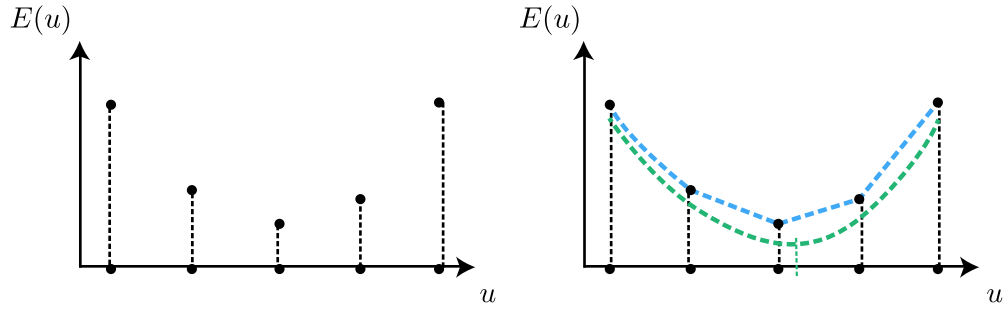


**Figure 1.3: Effect of non-unique minimizers on convex relaxation.** For the tightest convex relaxation (*blue*), the set of the minimizers is the convex hull of the set of the minimizers of the original functional (*black*).

**Discrete Optimization Domain.** A different scenario occurs when the energy is minimized over a *discrete domain*  $\mathcal{D}_0$ , see Figure 1.4. An example is when the energy is defined only on the set of 0/1-valued functions  $\{u \mid u : \Omega \rightarrow \{0, 1\}\}$ , which is a discrete set because of the discrete range  $\{0, 1\}$ . Note that we do not mean a discrete domain of the images  $u$  as in Section 1.1.2, but rather a discrete domain of the *energy*  $E$ , i.e. that the *set of images*  $\{u\}$  is discrete.

This situation poses a challenge because for efficient optimization not only the energy itself but also the domain of optimization must be convex. A typical way to cope with this is to enlarge the domain by replacing it by its *convex envelope*  $\mathcal{D} = \text{co } \mathcal{D}_0$ , and then to suitably extend the energy definition to the new arguments in such a way that the extended energy is convex. This is the idea of *relaxation*.

The definition of the extended energy, i.e. of its values on the extended domain, is not unique. A common requirement is that the extension should preserve the original values on  $\mathcal{D}_0$ . The best choice is, again, to take the *con-*



**Figure 1.4: Convex relaxation for discrete domains.** Convex relaxations are considered on the convex hull of the original discrete domain (*dots on the  $u$ -axis*). Shown are the convex envelope (*blue*) and a less tight relaxation (*green*).

*convex envelope*. After having found a suitable extension, minimizing this convex relaxation yields a *relaxed minimizer*. It is possible that it lies outside of the original domain  $\mathcal{D}_0$  since we optimize in the extended domain  $\mathcal{D} \supset \mathcal{D}_0$ . This can occur even if the relaxation is the tightest one. More precisely, in this case the relaxed minimizer will always lie in the convex hull  $\text{co} S \subset \mathcal{D}$  of the set  $S \subset \mathcal{D}_0$  of all true minimizers over  $\mathcal{D}_0$ , but not necessarily in  $\mathcal{D}_0$ . Therefore, as a postprocessing step, one needs to *project back* to the original discrete domain. This step is not unique, but in practice there are straightforward ways to find a discrete nearby point.

We note that this second scenario with a discrete domain is actually a special case of the first scenario, where the domain is continuous. In fact, one can consider the *indicator function* of the discrete domain  $\mathcal{D}_0$ , which is defined over whole  $\mathcal{D}$  and takes the value 0 for arguments from  $\mathcal{D}_0$  and  $\infty$  otherwise. Adding this to the original energy and minimizing over the convex hull  $\mathcal{D}$  is equivalent to the original problem.

## 1.2.2 Optimality Guarantees

**A-Posteriori Energy Bounds.** As discussed above, when using a convex lower bound less tight than the convex envelope or in the case of a discrete optimization domain, convex relaxation methods are not always guaranteed to optimally solve the original functional. However, these methods naturally come with *energy bounds* which provide *optimality guarantees*. This means that, after having computed a relaxed minimizer, one can assess how far one is from the global optimum in terms of the energy values.

Suppose the original energy  $E$  is to be minimized over a convex domain, and we do this by minimizing a convex relaxation  $\bar{E}$  of  $E$ . Then the following estimate holds:

$$\bar{E}(\bar{u}^*) \leq E(u^*) \leq E(\bar{u}^*). \quad (1.5)$$

Here  $\bar{u}^*$  is the computed relaxed minimizer of  $\bar{E}$ , and  $u^*$  a true minimizer (which is unknown to us) of  $E$ . The first inequality follows from the optimality of  $\bar{u}^*$  for  $\bar{E}$  and since  $\bar{E}$  is a lower bound for  $E$ , so that  $\bar{E}(\bar{u}^*) \leq \bar{E}(u^*) \leq E(u^*)$ . The second one follows from the optimality of  $u^*$  for  $E$ .

In the case that the original energy  $E$  is defined over a discrete domain  $\mathcal{D}_0$ , suppose we minimize a convex relaxation  $\bar{E}$  of  $E$  over the convex hull  $\mathcal{D}$  of  $\mathcal{D}_0$ . Since the relaxed minimizer  $\bar{u}^* \in \mathcal{D}$  may lie outside of the original domain  $\mathcal{D}_0$ , we need to *project back* to  $\mathcal{D}_0$ . Suppose we have computed such a projection  $(\bar{u}^*)_{\mathcal{D}_0} \in \mathcal{D}_0$  which lies nearby  $\bar{u}^*$  in some sense. Then the estimate (1.5) can be adapted to:

$$\bar{E}(\bar{u}^*) \leq E(u^*) \leq E((\bar{u}^*)_{\mathcal{D}_0}) \quad (1.6)$$

and is obtained analogously.

In the estimates (1.5) and (1.6) both the left and right hand side values are explicitly computable, after we have computed a relaxed minimizer  $\bar{u}^*$ .

**A-Priori Energy Bounds.** A-priori bounds give estimates independently of the actually computed solution. Such bounds are hard to establish and currently have been achieved only for very specific functionals, as will be detailed in Chapter 3.

Certain classes of energies are always guaranteed to yield optimal solutions through convex relaxation. The most prominent example is the Chan et al. two-phase segmentation approach [33]. Another one is the minimization of general data terms with the total variation regularization. See Chapters 3 and 7 for more details.

### 1.2.3 Established Convex Relaxation Techniques

For special classes of cost functions one can establish concrete ways, or at least general guidelines, of finding convex approximations or representations. Let us give two important examples.

**Multilabel Problems.** Multilabel problems are a very general and important class of energies, as many computer vision problems can be directly expressed in this way. Given a finite set of *labels*, represented as  $\mathcal{L} = \{1, \dots, n\}$  (with a  $n \geq 1$ ), the task is to assign each point in the image domain  $\Omega$  one of these labels, in a way such that the overall *labeling*  $u : \Omega \rightarrow \mathcal{L}$  is optimal, i.e. minimizing an energy

$$E(u) = E_{\text{data}}(u) + E_{\text{reg}}(u) = \int_{\Omega} c(x, u(x)) dx + E_{\text{reg}}(u). \quad (1.7)$$

The data term usually has a pointwise structure, given by a function  $c$ . Depending on the application at hand, certain label configurations may be more likely, or more preferable than others. This information can be included as part of the regularizer  $E_{\text{reg}}$ , which then acts as a prior term.

Because of the vast generality, there is no general method to arrive at a convex representation. Each concrete energy  $E$  requires an especially tailored approach in order to convexify it. Nonetheless, recent developments suggest that at least a certain *representation* of the labeling is more suited than others to find convex relaxations. Namely, the idea is to use *indicator functions*: For every label  $i \in \mathcal{L}$  one considers the characteristic function  $\chi_{u=i}$  for this label, which

has value 1 in the image points where this label is attained and 0 otherwise. See Chapter 3 for more details.

**Functional Lifting.** One class of energies with a well-developed convexification theory are energies of the form

$$E(u) = E_{\text{data}}(u) + E_{\text{reg}}(u) = \int_{\Omega} c(x, u(x)) \, dx + \int_{\Omega} f(x, \nabla u(x)) \, dx, \quad (1.8)$$

which are defined on smooth scalar functions  $u : \Omega \rightarrow \mathbb{R}$  and depend on the values  $u$  and the gradient  $\nabla u$ . The general approach to convexify such energies is *functional lifting* [1, 101]. The idea is to express  $E$  in terms of the *graph* of  $u$ , or, more precisely, in terms of the characteristic function  $1_u(x, t) = \chi_{t < u}$  of the subgraph of  $u$ . Assuming certain conditions on  $f$ , for instance that it is convex in  $\nabla u$ , the new energy then turns out to be a convex function of  $1_u$ . This can be effectively used to arrive at optimal or near-optimal solutions. The approach can be generalized to energies defined on a possibly discontinuous  $u$ . Functional lifting is related to the Ishikawa construction [60] in the discrete setting.

The key advantage of this method is that it allows arbitrarily complicated data terms  $c$ . In view of (1.4), this enables one to work with general generative models  $\mathcal{P}(I|u)$  of the data  $I$  given the solution  $u$  as long as the dependency is *pointwise*, i.e. for each image point  $x \in \Omega$  the image value  $I(x)$  only depends on  $u(x)$ . On the other hand, the problem dimension is increased since one now minimizes over functions defined on  $\Omega \times \mathbb{R}$  instead of  $\Omega$ , which increases the computational complexity. Furthermore, for the implementation the range of  $u$  must be discretized into a finite number of levels, typically 32 or 64. We will give more details in Chapter 7.

The main and essential limitation of the lifting approach is that it only works for *scalar* functions  $u$ .

## 1.3 Contributions

The goal of this thesis is to extend and generalize existing convexification methods introduced in Section 1.2.3 to convexify a number of practical functionals. The contributions are divided into two broad areas: Multilabel segmentation using various priors, and extensions of functional lifting to vectorial functions. These areas are covered respectively in Part I and Part II of the thesis.

Central ideas of this thesis were developed in various conferences and journal papers, in particular [125, 121, 94] and [124, 55, 120, 119].

### 1.3.1 Part I: Multilabel Segmentation

The first part deals with the multilabeling problem and how to incorporate specific higher-level knowledge of valid segmentations into the segmentation process.

In Chapter 3 we will first give a brief introduction to multilabel approaches and common relaxations in the case of length regularity. Subsequently, in Chapters 4, 5 and 6 we will present three novel priors and propose respective convex relaxations for their efficient optimization.

**Multilabel Segmentation with Nonmetric Priors.** In Chapter 4 we will revisit the well studied instance of the multilabel problem (1.7) when the regularizer  $E_{\text{reg}}$  acts as length penalization. That is, considering the regions  $\Omega_i = \{x \mid u(x) = i\}$  where each label is assigned, one seeks to minimize the data term plus the overall boundary length

$$\sum_{i < j} d(i, j) \text{Length}(\partial\Omega_i \cap \partial\Omega_j), \quad (1.9)$$

where boundary lengths are weighted by a label transition function  $d(i, j)$ . In order for classical relaxations to work, one limitation is that  $d$  must be a metric.

We propose a novel convex approach which allows to remove this restriction and impose arbitrary *nonmetric distances*  $d$  between labels. The only remaining constraints are symmetry  $d(i, j) = d(j, i)$ , nonnegativity  $d(i, j) \geq 0$ , and reflexivity  $d(i, i) = 0$ . We show that the model can be applied to provide a convex relaxation of the Mumford-Shah functional, where the nonmetric distance function is a truncated quadratic potential. The proposed prior is formulated in the continuous setting, so that the grid artifacts of the corresponding grid based approaches for the nonmetric case are avoided. Furthermore, experiments on the MSRC segmentation database yield comparable or superior results comparing to metric or grid based approaches.

**Multilabel Segmentation with Ordering Constraints.** In Chapter 5 we propose a novel framework for imposing *label ordering constraints* in multilabel optimization. This gives a very general prior on the relative geometric location of the distinct regions. In particular, label jumps can be penalized differently depending on the jump direction. For example, it is possible to restrict that one specific label is always to the left of another one.

The proposed method provides a unified solution to three different problems which are otherwise solved by three separate approaches [48, 82, 131], as part of one common convexification framework. The method gives a generalization beyond these MRF-based approaches, making it possible to consider new label layouts. For example, we show that we can impose the novel *convex shape prior*. Since we work in the spatially continuous setting, it is possible to consider the correct Euclidean boundary length, in contrast to grid-based approaches. The method naturally extends to three and higher dimensions of the image domain. We provide an exact characterization of the label distance penalization functions that are expressible with our approach.

Despite the generality, the implementation is straightforward and can be easily adjusted to various label layouts. Experiments show comparable and superior results compared to the previous methods.

**Multilabel Segmentation with Proportion Priors.** In Chapter 6 we propose a convex multilabel framework for image sequence segmentation. In this application, the task is to segment one and the same object which is present in every image of the sequence. Because the appearance can change significantly from image to image, including changes in color, size, shape and position

within the image frame, this poses a challenge even for current state-of-the-art segmentation approaches.

We propose to impose a novel *proportion preserving prior* on object parts. It ensures that the *relative* size of each object part w.r.t. the size of the whole object remains roughly the same across all images. The key idea is that, even for strong deformations in shape and size, these relative sizes are still well preserved. We introduce two different kinds of such a prior based on the Bayesian framework for image segmentation. Several ways for their convexification are proposed and compared, yielding a convex relaxation of the overall image sequence segmentation problem. The resulting algorithm is easy to implement and can provide proportion-consistent segmentations with run times of about one second.

Quantitative and qualitative experiments demonstrate the effectiveness of proportion priors to achieve accurate segmentations. For example, semantically relevant small-scale object parts are better preserved from disappearing, and background areas with similar colors as the object are less likely to be labeled as foreground. The method naturally applies to multiple object instances, e.g. when segmenting all players of one team in a sports game. Finally, the same method can be equally applied to constrain the relative size of different objects w.r.t. each other, such as several organs in medical imaging.

### 1.3.2 Part II: Vectorial Functionals

In the second part we present several approaches of how the functional lifting idea can be used to provide convex relaxations for *vectorial* functionals. Functional lifting is essentially limited to energies depending on *scalar* functions  $u : \Omega \rightarrow \mathbb{R}$ . Therefore, energies defined on *vectorial* functions  $u : \Omega \rightarrow \mathbb{R}^k$ ,  $k \geq 2$ , require entirely different convexification strategies.

In Chapter 7 we will first introduce and review the classical functional lifting idea. The next Chapters 8, 9 and 10 are devoted to various special cases of vectorial functionals. In each case, we will propose an especially tailored convex relaxation. Using suitable decouplings and reformulations, we will also highlight a number of ways to make the proposed convexifications efficiently solvable.

**Vectorial Problems with Separable Regularization.** In Chapter 8 we will first start with the case that there is *no channel coupling* in the regularizer, i.e. the channels are regularized independently of each other:

$$E_{\text{reg}}(u) = \sum_{i=1}^k R_i(u_i). \quad (1.10)$$

In order to arrive at a convex relaxation, all previous methods require that the whole vectorial range of  $u$  is discretized into a finite set of points. For example, if the range of  $u : \Omega \rightarrow \mathbb{R}^k$  is the unit box  $[0, 1]^k$  and each channel range  $[0, 1]$  is discretized into  $n \geq 1$  levels, say  $n = 32$  or  $n = 64$ , the discretization of the overall range will require  $n^k$  points. Unfortunately, in the previous methods both run time and memory scale linearly with the total number  $n^k$  of labels.

This makes them inefficient for  $k \geq 2$  and often not even applicable for  $k \geq 3$  due to memory constraints.

Assuming that the regularizer is *separable*, we propose a reduction technique so that the overall time and memory complexity scales only as  $k \cdot n$  instead of  $n^k$ . For typical real-world problems, this means that the resulting convex relaxation requires orders of magnitude less time as well as memory in comparison to previous methods. This enables us to consider large-scale problems which were previously infeasible.

The method is demonstrated on applications such as optical flow, stereo with occlusion detection, color image segmentation into a large number of regions, and joint denoising and local noise estimation. For the first time, our approach enables one to efficiently compute solutions to the optical flow functional which are within provable bounds (typically 5%) of the global optimum.

**Vectorial Problems with Coupled Regularization.** In Chapter 9 we consider how specific *channel couplings* can be incorporated in the regularizer while still being able to devise an efficient convex relaxation. Namely, we will consider regularizers

$$E_{\text{reg}}(u) = \int_{\Omega} f(x, \nabla u(x)) \, dx \quad (1.11)$$

with an  $f$  convex in  $\nabla u$ . This is first defined for smooth functions  $u$  and then suitably extended to functions of bounded variation. For instance, this includes the  $l^2$ -coupled total variation  $TV(u) = \sqrt{\sum_{i=1}^k |\nabla u_i|^2}$ , which couples the channels and is not of the form (1.10).

For nonconvex energies with regularizers as above, we propose a convex relaxation which is both *tractable* and yet as tight as possible. Rather than separately treating the data term and the regularizer, the key idea is to consider the collection of graph functions of  $u$ , one for each channel, and to devise a relaxation that takes into account the entire functional as a whole. We provide a theoretical analysis of the relaxation, and give implementation details for a number of regularizers enabled with our approach. In each case, we show how the arising constraints can be decoupled to make the relaxation efficiently implementable.

In contrast to previous relaxations, it now becomes possible to handle the combination of nonconvex data terms with coupled regularizers such as  $l^2$ -total variation. Experiments demonstrate that such regularizers systematically yield improved results in denoising, inpainting and optical flow applications.

**The Vectorial Mumford-Shah Functional.** In Chapter 10 we focus on the *vectorial Mumford-Shah functional*, which is devised to obtain *piecewise smooth* or *piecewise constant* approximations of input images. The vectorial functional couples the channels in such a way that the discontinuities in the different color channels preferably coincide.

We propose the first tractable convex formulation, which allows one to efficiently compute high-quality solutions independently of the initialization. The approach is a generalization of the scalar case [100] to a general number of

channels. Furthermore, we propose an efficient reformulation of the arising constraints which makes the overall optimization problem *as tractable as in the scalar case*.

Numerous experiments confirm the proposed convexification to be superior for contrast-preserving and edge-enhancing regularization, comparing with the naive channel-wise approach, the well-known Ambrosio-Tortorelli approximation, as well as the classical total variation.

One can think of this functional as being in the “middle” between the cases of regularizers covered in the previous two Chapters 8 and 9. It is not representable in the form (1.10) because of channel coupling, although (1.10) does allow *channel-wise* Mumford-Shah regularization. And though (1.11) allows channel coupling, the vectorial Mumford-Shah regularizer can be seen to be not of this form, since it explicitly allows jumps in  $u$ .

### 1.3.3 Thesis Structure

In Chapter 2 we will introduce the basic notions and concepts which will be used throughout the thesis, including convexity, convex duality, functions of bounded variation and the primal-dual algorithm. This is followed by the main two Parts I and II of the thesis as described in the previous two sections. Finally, we will conclude in Chapter 11 with a summary of the achieved results, as well as notes about possible directions for future research.

## 1.4 Publications

In the process of the doctoral study, a number of papers have been written and published in various international journals and conferences:

### Journal Publications

- Moellenhoff, T., Strelakovski, E., Moeller, M., Cremers, D.: The primal-dual hybrid gradient method for semiconvex splittings. *SIAM Journal on Imaging Sciences* 8(2), 827–857 (2015)
- Strelakovski, E., Chambolle, A., Cremers, D.: Convex relaxation of vectorial problems with coupled regularization. *SIAM Journal on Imaging Sciences* 7(1), 294–336 (2014)
- Goldluecke, B., Strelakovski, E., Cremers, D.: Tight convex relaxations for vector-valued labeling. *SIAM Journal on Imaging Sciences* 6(3), 1626–1664 (2013)
- Cremers, D., Strelakovski, E.: Total cyclic variation and generalizations. *Journal of Mathematical Imaging and Vision* 47(3), 258–277 (2012)
- Goldluecke, B., Strelakovski, E., Cremers, D.: The natural total variation which arises from geometric measure theory. *SIAM Journal on Imaging Sciences* 5(2), 537–563 (2012)



**Conference Publications**

- Möllenhoff, T., Strelakovsky, E., Möller, M., Cremers, D.: Low rank priors for color image regularization. In: International Conference on Energy Minimization Methods for Computer Vision and Pattern Recognition (EMMCVPR) (2015)
- Strelakovsky, E., Cremers, D.: Real-time minimization of the piecewise smooth Mumford-Shah functional. In: European Conference on Computer Vision (ECCV) (2014)
- Nieuwenhuis, C., Strelakovsky, E., Cremers, D.: Proportion priors for image sequence segmentation. In: IEEE International Conference on Computer Vision (ICCV) (2013)
- Lellmann, J., Strelakovsky, E., Koetter, S., Cremers, D.: Total variation regularization for functions with values in a manifold. In: IEEE International Conference on Computer Vision (ICCV) (2013)
- Souiai, M., Strelakovsky, E., Nieuwenhuis, C., Cremers, D.: A co-occurrence prior for continuous multi-label optimization. In: International Conference on Energy Minimization Methods for Computer Vision and Pattern Recognition (EMMCVPR) (2013)
- Souiai, M., Nieuwenhuis, C., Strelakovsky, E., Cremers, D.: Convex optimization for scene understanding. In: ICCV Workshop on Graphical Models for Scene Understanding (2013)
- Strelakovsky, E., Chambolle, A., Cremers, D.: A convex representation for the vectorial Mumford-Shah functional. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
- Strelakovsky, E., Nieuwenhuis, C., Cremers, D.: Nonmetric priors for continuous multilabel optimization. In: European Conference on Computer Vision (ECCV) (2012)
- Strelakovsky, E., Cremers, D.: Generalized ordering constraints for multi-label optimization. In: IEEE International Conference on Computer Vision (ICCV) (2011)
- Strelakovsky, E., Goldluecke, B., Cremers, D.: Tight convex relaxations for vector-valued labeling problems. In: IEEE International Conference on Computer Vision (ICCV) (2011)
- Strelakovsky, E., Cremers, D.: Total variation for cyclic structures: Convex relaxation and efficient minimization. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2011)



# Chapter 2

## Fundamentals

In this chapter we want to introduce the basic instruments which will be used throughout the thesis. This thesis is all about convexification approaches, thus in Section 2.1 we will first give a brief overview of the foundations of convex analysis. Next, in Section 2.2 we will introduce the main function spaces that will serve as the basic domain to which the solutions of all of our considered practical problems will belong, namely the space of functions of bounded variations. Finally, in order to practically compute solutions of a given convex relaxation model, in Section 2.3 we will present the main primal-dual minimization algorithm and give notes about common dualization strategies to render efficient optimization possible.

### 2.1 Convex Analysis

#### 2.1.1 Basic Definitions

For a detailed introduction to convex analysis we refer the reader to [108].

When dealing with convex functions, it is convenient to extend the set of real numbers by adding an infinity element to it:  $\overline{\mathbb{R}} := \mathbb{R} \cup \{+\infty\}$ . For instance, this will be useful to model hard constraints, by assigning energy value  $+\infty$  to functions which violate the constraints. In order to simplify the notation we will use the same notation  $\mathbb{R}$  instead of  $\overline{\mathbb{R}}$  throughout the thesis.

Let  $X$  be some fixed Hilbert space with scalar product  $\langle \cdot, \cdot \rangle$ . For the purpose of this introduction, one can think of  $X$  as being  $\mathbb{R}^m$  for some  $m \geq 1$ , but in general  $X$  can also be infinitely dimensional. For example,  $X$  can be the function space  $L^2(\Omega; \mathbb{R})$  with its usual scalar product  $\langle f, g \rangle = \int_{\Omega} f(x)g(x) dx$ . Throughout the thesis, general norms are denoted as usual by  $\|\cdot\|$ , while we use the simpler notation  $|\cdot|$  for the Euclidean norm of vectors and the Frobenius norm of matrices.

**Convexity and Lower-semicontinuity.** A function  $f : X \rightarrow \mathbb{R}$  is called *convex* if it satisfies the inequality

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y) \quad (2.1)$$

for all points  $x, y \in X$  and every  $0 < \alpha < 1$ . One calls  $f$  *proper* if  $f$  is not the constant  $\infty$ , i.e. if  $f$  is finite for at least one point in  $X$ . Furthermore,  $f$  is called *lower-semicontinuous* if for any sequence  $(x_n)_{n \geq 1} \subset X$  converging to a  $x \in X$  it holds

$$f(x) \leq \liminf_{n \rightarrow \infty} f(x_n). \quad (2.2)$$

Functions  $f$  satisfying  $f(sx) = |s|f(x)$  for any  $x \in X$  and  $s \in \mathbb{R}$  are called *one-homogeneous*. If this holds at least for all  $s > 0$ , then  $f$  is called *positive one-homogeneous*.

**Set Indicator Functions.** A very useful kind of convex functions are the *indicator functions* of sets. For a subset  $A \subset X$  its indicator function is defined as

$$\delta_A(x) = \begin{cases} 0 & \text{if } x \in A, \\ \infty & \text{else.} \end{cases} \quad (2.3)$$

We can check right away that this is a convex function whenever  $A$  is convex. Namely, if either  $x \notin A$  or  $y \notin A$ , then the right hand side of (2.1) is  $\infty$ , so that the inequality is trivially fulfilled. Otherwise, i.e. if both  $x, y$  lie in  $A$ , then so does their convex combination  $\alpha x + (1 - \alpha)y$  because  $A$  is convex, i.e. both sides of (2.1) are zero. More generally,  $\delta_A$  is convex if and only if  $A$  is convex. Moreover,  $\delta_A$  is lower-semicontinuous if and only if  $A$  is closed.

Thus, an example of a convex function which is not lower-semicontinuous can be given by  $\delta_A$  where  $A$  is convex but not closed, e.g.  $\delta_{(0,1)}$  when  $X = \mathbb{R}$ .

**Recession Function.** For convex functions  $f : X \rightarrow \mathbb{R}$ , its *recession function*  $f_\infty : X \rightarrow \mathbb{R}$  is defined as

$$f_\infty(x) = \limsup_{t \rightarrow \infty} \frac{f(tx)}{t}. \quad (2.4)$$

This is a convex and positive one-homogeneous function, and it always holds  $f_\infty(0) = 0$ . It captures the behavior of  $f$  at infinity when going along different directions  $x$ , ignoring the behavior at small values. Intuitively,  $f_\infty$  is obtained by first drawing the graph of  $f$  and then “zooming out”, so that only the behavior at large values will matter. For example, if  $f$  is some norm,  $f(x) = \|x\|$ , we get  $f_\infty(x) = \|x\|$ , and in general  $f_\infty = f$  for any positive one-homogeneous  $f$ . If  $f$  has superlinear growth at infinity, for example  $f(x) = \|x\|^2$ , then  $f_\infty(x)$  is equal to  $\infty$  everywhere except for  $x = 0$ , where  $f_\infty(0) = 0$ .

**Subdifferential.** Convex functions  $f : X \rightarrow \mathbb{R}$  are not necessarily differentiable everywhere, so that one generalizes the notion of differentiability and considers the *subdifferential* instead. For every  $x \in X$  it is the set  $(\partial f)(x) \subset X$  defined by

$$(\partial f)(x) = \left\{ a \in X \mid f(y) \geq f(x) + \langle a, y - x \rangle \quad \forall y \in X \right\}. \quad (2.5)$$

In other words,  $(\partial f)(x)$  is the set of all *slopes*  $a$  such that the corresponding linear approximation at  $x$  with that slope is equal to or below  $f$ . If  $f$  is convex

and differentiable at  $x$ , then  $(\partial f)(x) = \{(\nabla f)(x)\}$  is a set containing just the gradient as its single element. The converse is also true, in that if  $f$  is convex and  $(\partial f)(x)$  contains only one element, then  $f$  is differentiable at  $x$ . As an example, for the Euclidean norm  $f(x) = |x|$  we get  $(\partial f)(x) = \{x/|x|\}$  for any  $x \neq 0$ , while  $f$  is not differentiable at  $x = 0$  and  $(\partial f)(0) = \{a \mid |a| \leq 1\}$ .

The subdifferential is very useful when dealing with optimization problems. Namely, for any convex function  $f$ ,  $x \in X$  is a minimizer of  $f$  if and only if  $0 \in (\partial f)(x)$ . This can be seen directly since  $0 \in (\partial f)(x)$  by definition means  $f(y) \geq f(x)$  for all  $y$ , which is just the definition of  $x$  being a minimizer of  $f$ . This is a direct generalization of the necessary condition  $(\nabla f)(x) = 0$  for differentiable  $f$ .

### 2.1.2 Convex Duality

A fundamentally important concept in convex analysis is that of the *dual function*. Given a function  $f : X \rightarrow \mathbb{R}$ , its dual function  $f^* : X \rightarrow \mathbb{R}$  is defined by

$$f^*(y) := \sup_{x \in X} \langle x, y \rangle - f(x). \quad (2.6)$$

It is also known as the *Legendre-Fenchel dual*, the *convex dual* or the *convex conjugate* of  $f$ . Being a pointwise supremum of linear functions, this is always a *convex* function, even if  $f$  itself is not convex. This can be readily seen directly:

$$\begin{aligned} f^*(\alpha y_1 + (1 - \alpha)y_2) &= \sup_{x \in X} \langle x, \alpha y_1 + (1 - \alpha)y_2 \rangle - f(x) \\ &= \sup_{x \in X} \alpha \left( \langle x, y_1 \rangle - f(x) \right) + (1 - \alpha) \left( \langle x, y_2 \rangle - f(x) \right) \\ &\leq \alpha \sup_{x \in X} \left( \langle x, y_1 \rangle - f(x) \right) + (1 - \alpha) \sup_{x \in X} \left( \langle x, y_2 \rangle - f(x) \right) \\ &= \alpha f^*(y_1) + (1 - \alpha) f^*(y_2) \end{aligned} \quad (2.7)$$

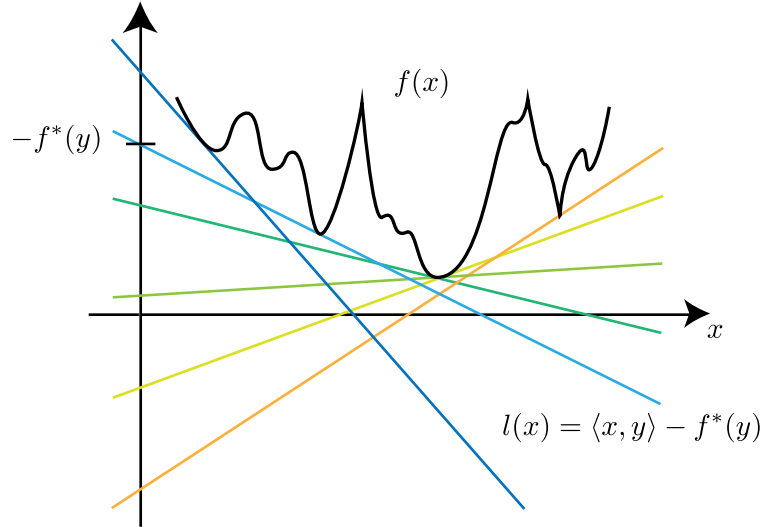
for any  $y_1, y_2 \in X$  and  $0 < \alpha < 1$ . Moreover,  $f^*$  is always *lower-semicontinuous*.

We can iterate this process and consider the convex conjugate of  $f^*$ , giving the *bidual*  $f^{**} = (f^*)^*$ . For *convex*  $f$ , it turns out that doing so we get back to the original function:  $f^{**} = f$ . More precisely,  $f$  needs to be convex and lower-semicontinuous. This important property is referred to as *convex duality*. Thus, writing out the definition of the convex conjugate  $(f^*)^*$ , we have the following proposition:

**Proposition 2.1** (Convex Duality). *For every convex and lower-semicontinuous function  $f$  we have the representation*

$$f(x) = \sup_{y \in X} \langle x, y \rangle - f^*(y). \quad (2.8)$$

For *nonconvex* functions  $f$  it turns out that  $f^{**}$  is the *convex envelope* of  $f$ , more precisely the lower-semicontinuous convex envelope. It is defined as the greatest convex and lower-semicontinuous function which is pointwise below or equal to  $f$ .



**Figure 2.1: Convex conjugate interpretation.** The highest affine lower bound to  $f$  with slope  $y$  has value  $-f^*(y)$  at zero.

Thus, computing the double conjugate  $f^{**}$  gives us an *explicit construction to find convex relaxations* of nonconvex functions  $f$ : Starting with  $f$ , compute  $f^*$  and then the right hand side of (2.8), which is nothing but the definition of  $f^{**}(x)$ . However, although this always works in theory, in practice the computation of  $f^{**}$  can be done explicitly only for very few special cases.

**Intuitive Interpretation.** The convex conjugate arises naturally in the context of optimal affine lower bounds, see Figure 2.1.

Consider a (convex or nonconvex) function  $f$ . We want to approximate  $f$  from below by affine functions  $l : X \rightarrow \mathbb{R}$ . For a fixed slope  $y \in X$  they have the form  $l(x) = \langle x, y \rangle + a$  with an  $a \in \mathbb{R}$ . The question is how large can we choose  $a$  so that  $l$  is still a lower bound for  $f$ . For this we must have

$$\langle x, y \rangle + a \leq f(x) \quad \forall x, \quad \text{or equivalently} \quad -a \geq \langle x, y \rangle - f(x) \quad \forall x. \quad (2.9)$$

These constraints over all  $x$  are equivalent to a single one:

$$-a \geq \sup_{x \in X} (\langle x, y \rangle - f(x)) = f^*(y). \quad (2.10)$$

Therefore, the maximal  $a$  is  $a = -f^*(y)$  and the best affine lower bound with slope  $y$  is

$$l(x) = \langle x, y \rangle - f^*(y). \quad (2.11)$$

The convex (and lower-semicontinuous) envelope of  $f$  is the supremum over all affine lower bounds. This is the same as taking the supremum over all slopes  $y$  in (2.11), which directly gives  $f^{**}(x)$ . Thus,  $f^{**}$  can indeed be interpreted as the convex and lower-semicontinuous envelope of  $f$ .

## 2.2 Functions of Bounded Variation

The space of bounded variation functions plays an important role for many practical computer vision and image processing applications, serving as the basis space to which the solutions belong. On one hand, it is general enough including smooth as well as discontinuous functions. On the other hand, it is small enough in the sense that the functions possess many important and useful properties, for instance, certain regularity of the distributional gradient. We refer to [5] and [6] for a comprehensive introduction to functions of bounded variation.

**The Space  $BV(\Omega, \mathbb{R}^k)$ .** Let  $\Omega \subset \mathbb{R}^m$ ,  $m \geq 1$ , be a bounded open set. The usual  $m$ -dimensional Lebesgue measure is denoted by  $\mathcal{L}^m$  (with  $dx := d\mathcal{L}^m$ ), and for  $a > 0$  the  $a$ -dimensional Hausdorff-measure by  $\mathcal{H}^a$ .  $B_r(x)$  is the open ball of radius  $r$  around  $x$ .

For vectorial functions  $u \in L^1(\Omega, \mathbb{R}^k)$  with  $k \geq 1$  channels, the *total variation* of  $u$  is defined by

$$TV(u) = \sup \left\{ \int_{\Omega} - \sum_{i=1}^k u_i(x) \operatorname{div} \varphi_i(x) dx \mid \varphi \in C_c^1(\Omega; \mathbb{R}^{m \times k}), |\varphi(x)| \leq 1 \ \forall x \in \Omega \right\}. \quad (2.12)$$

Functions  $u$  with  $TV(u) < \infty$  are called *functions of bounded variation*, and the space of all such  $u$  is denoted by  $BV(\Omega, \mathbb{R}^k)$ .

We use the notation  $\varphi(x) = (\varphi_1(x), \dots, \varphi_k(x)) \in \mathbb{R}^{m \times k}$ , i.e.  $\varphi_i(x) \in \mathbb{R}^m$  is the vector  $(\varphi_{ji}(x))_{1 \leq j \leq m}$  for each  $i$ , and  $|\cdot|$  is the Euclidean norm on vectors or matrices. As usual,  $C_c^1(\Omega; \mathbb{R}^{m \times k})$  is the space of continuously differentiable vectorial functions with compact support.

If  $u$  is smooth with gradient  $\nabla u$ , total variation can be expressed as

$$TV(u) = \int_{\Omega} |\nabla u(x)| dx. \quad (2.13)$$

This can serve as a starting point to motivate the general definition (2.12). Namely, for each  $x \in \Omega$  the absolute value of  $\nabla u(x) \in \mathbb{R}^{m \times k}$  can be written as

$$|\nabla u(x)| = \sup_{\varphi(x) \in \mathbb{R}^{m \times k}, |\varphi(x)| \leq 1} \langle \varphi(x), \nabla u(x) \rangle, \quad (2.14)$$

which explains the appearance of the variables  $\varphi(x) \in \mathbb{R}^{m \times k}$  in (2.12). Plugging this into (2.13), assuming smoothness of  $\varphi$  and integrating by parts, the formulation (2.12) is obtained.

It is useful to consider bounded variation functions where the range is constrained to some set. For sets  $E \subset \mathbb{R}^k$ , the space  $BV(\Omega; E)$  is defined as the set of all functions  $u \in BV(\Omega; \mathbb{R}^k)$  with  $u(x) \in E$  for a.e.  $x \in \Omega$ .

**The Jump Set  $S_u$ .** One of the most important properties of bounded variation functions is the regular structure of their values. Essentially, they are continuous everywhere except for possibly only jump discontinuities, occurring along regular hypersurfaces. Furthermore, they are weakly differentiable almost everywhere.

Although functions  $u \in L^1(\Omega, \mathbb{R}^k)$  are defined only a.e., one can try to define a “standard” representative for each  $x$  by averaging over small neighborhoods:  $\bar{u}(x) = \lim_{r \rightarrow 0} |B_r(x)|^{-1} \int_{B_r(x)} u(y) dy$ . For general  $u \in L^1(\Omega, \mathbb{R}^k)$ , this limit exists for every  $x \in \Omega$  except possibly for a set  $S_u \subset \Omega$  of measure zero.

If  $u \in \text{BV}(\Omega, \mathbb{R}^k)$ , this set  $S_u$  has a certain regularity. Namely, up to a set of zero  $\mathcal{H}^{m-1}$ -measure, it is a  $(m-1)$ -dimensional set and consists of a countable collection of pieces of smooth hypersurfaces. Furthermore, it can be seen as the *jump set* of  $u$ , in that  $u$  has a jump discontinuity at  $\mathcal{H}^{m-1}$ -a.e.  $x \in S_u$ : One can define two values  $u^-(x) = (u_1^-(x), \dots, u_k^-(x))$  and  $u^+(x) = (u_1^+(x), \dots, u_k^+(x))$  and a normal vector  $\nu_u(x) \in \mathbb{S}^{m-1} := \{z \in \mathbb{R}^m \mid |z| = 1\}$  indicating the direction of the jump and pointing towards the  $u^+(x)$  side. Specifically, for these  $x$  it holds

$$u(x + \varepsilon z) \xrightarrow{\varepsilon \rightarrow 0} u^+(x) \chi_{\langle z, \nu_u(x) \rangle > 0} + u^-(x) \chi_{\langle z, \nu_u(x) \rangle < 0} \quad (2.15)$$

w.r.t. the  $L^1(B_1(0), \mathbb{R}^k)$ -norm, as functions of  $z$ . There could be some points on  $S_u$  where (2.15) does not hold (they form a set of zero  $\mathcal{H}^{m-1}$ -measure, e.g. isolated points for  $m = 2$ ), these are for example “corners” of  $S_u$ , or points where three or more values of  $u$  meet at  $x$ .

Note that when two or more components  $u_i$  jump at a point  $x \in S_u$ , for  $\mathcal{H}^{m-1}$ -a.e.  $x$  all  $k$  components jump in one and the same direction, though this common jump direction  $\nu_u(x)$  of course might be different for different  $x$ .

**The Distributional Gradient.** Functions in  $\text{BV}(\Omega, \mathbb{R}^k)$  are not necessarily smooth since they may have jump discontinuities, so that the gradient  $\nabla u$  may not exist in the ordinary sense. Therefore one is lead to consider the distributional gradient instead.

For functions  $u \in L^1(\Omega, \mathbb{R}^k)$ , the distributional gradient is a mapping  $Du : C_c^1(\Omega; \mathbb{R}^{m \times k}) \rightarrow \mathbb{R}$  defined on smooth compactly supported vectorial functions  $\varphi$  by

$$(Du)(\varphi) = \int_{\Omega} \langle -\text{div } \varphi, u \rangle dx. \quad (2.16)$$

Here  $\text{div } \varphi(x) \in \mathbb{R}^k$  at each  $x \in \Omega$  is formed by applying the divergence separately to each component  $\varphi_i : \Omega \rightarrow \mathbb{R}^m$ ,  $1 \leq i \leq k$ , of  $\varphi : \Omega \rightarrow \mathbb{R}^{m \times k} = (\mathbb{R}^m)^k$ .

For  $u \in \text{BV}(\Omega, \mathbb{R}^k)$  it can be represented as a  $\mathbb{R}^{m \times k}$ -valued Radon measure [5, Proposition 3.6], which we again denote by  $Du$ . This means that for any  $\varphi \in C_c^1(\Omega; \mathbb{R}^{m \times k})$  we can write

$$(Du)(\varphi) = \int_{\Omega} \langle \varphi, dDu \rangle. \quad (2.17)$$

In the last integral,  $\langle \cdot, \cdot \rangle$  denotes the standard scalar product on  $\mathbb{R}^{m \times k}$ . We will



also write this in any of the following forms:

$$\int_{\Omega} \langle \varphi, dDu \rangle = \int_{\Omega} \varphi \cdot dDu = \sum_{ij} \int_{\Omega} \varphi_{ij} dDu_{ij} = \int_{\Omega} \langle \varphi, \frac{Du}{|Du|} \rangle d|Du| \quad (2.18)$$

where  $\frac{Du}{|Du|}|Du|$  with  $\frac{Du}{|Du|} \in L^1_{|Du|}(\Omega; \mathbb{R}^{m \times k})$  is the polar decomposition of  $Du$  [5, Corollary 1.19].

In the case that  $u$  is smooth, i.e. the gradient  $\nabla u$  exists in the classical sense, (2.16) can be rewritten as

$$(Du)(\varphi) = \int_{\Omega} \langle -\operatorname{div} \varphi, u \rangle dx = \int_{\Omega} \langle \varphi, \nabla u \rangle dx, \quad (2.19)$$

so that the measure  $Du$  in (2.17) is just the usual Lebesgue measure weighted by the gradient:

$$Du = \nabla u(x) \mathcal{L}^m. \quad (2.20)$$

For a general  $u$ , additional terms appear on the right hand side of (2.20), according to the general structure of  $Du$ . It can be decomposed into three parts [5, Definition 3.91]:

$$Du = \nabla u(x) \mathcal{L}^m + \nu_u(x) \otimes (u^+(x) - u^-(x)) \mathcal{H}^{m-1} \llcorner S_u + Cu. \quad (2.21)$$

The first one  $\nabla u(x) \mathcal{L}^m$  is the absolutely continuous part w.r.t.  $m$ -dimensional Lebesgue measure  $\mathcal{L}^m$ . It is called the “smooth part” of  $Du$ . The density  $\nabla u \in L^1(\Omega, \mathbb{R}^{m \times k})$  is called the “approximate gradient” of  $u$  [5, Definition 3.70].

The second one is the *jump part*, which is the  $(m-1)$ -dimensional Hausdorff measure restricted to the jump set  $S_u$  (i.e.  $(\mathcal{H}^{m-1} \llcorner S_u)(A) := \mathcal{H}^{m-1}(A \cap S_u)$  for all  $A$ ) and weighted by jump height  $u^+ - u^-$  and jump direction  $\nu_u$ . Here  $\nu_u(x) \otimes (u^+(x) - u^-(x)) := \nu_u(u^+ - u^-)^T \in \mathbb{R}^{m \times k}$  is the Kronecker product between the column vectors  $\nu_u \in \mathbb{R}^m$  and  $u^+ - u^- \in \mathbb{R}^k$ . This collects the jumps for all  $k$  channels into one  $m \times k$  matrix of  $k$  columns  $\nu_u$  multiplied by the corresponding individual scalar jump heights for each channel:

$$\nu_u(x) \otimes (u^+(x) - u^-(x)) = \left( (u_1^+ - u_1^-) \nu_u \quad \cdots \quad (u_k^+ - u_k^-) \nu_u \right). \quad (2.22)$$

Intuitively, the jump part can be derived from the smooth part  $\nabla u \mathcal{L}^m$  by imagining a smooth transition becoming steeper and steeper until it becomes a jump discontinuity: The gradient  $\nabla u$  becomes infinitely large at jumps, but the jump area is infinitely small; these two effects cancel out effectively yielding  $\mathcal{H}^{m-1}$ -integration along the jump discontinuity  $S_u$ .

The last part in (2.21) is the “Cantor” part and corresponds to an uncountable accumulation of “zero-height jumps” on a set of measure zero. It is a measure absolutely continuous w.r.t.  $\mathcal{L}^m$  (i.e.  $Cu(A) = 0$  if  $\mathcal{L}^m(A) = 0$ ), and zero for any  $(m-1)$ -dimensional set. It is supported on a set  $E \subset \Omega$  of Hausdorff-dimension  $m-1 \leq \dim_{\mathcal{H}}(E) < m$ .

**The Space  $SBV(\Omega, \mathbb{R}^k)$ .** One says that  $u$  is a *special function of bounded variation*,  $u \in SBV(\Omega, \mathbb{R}^k)$ , if the distributional gradient has no Cantor part  $Cu$  in the decomposition (2.21). Intuitively, there will be no “strange pathological” jumps of  $u$ , such as present in the Cantor-Vitali function. Instead, there will be only regions of continuous variation and jump discontinuities in-between these regions. This makes it a natural space for computer vision problems with possibly discontinuous solutions.

Functions  $u \in SBV(\Omega, \mathbb{R}^k)$  are absolutely continuous up to a  $(m-1)$ -dimensional jump set  $S_u$ , where they assume values  $u^-, u^+$  on the two sides of the interface with normal vector  $\nu_u$ . As a special case of (2.21), for  $u \in SBV(\Omega, \mathbb{R}^k)$  the distributional gradient decomposes as

$$Du = \nabla u(x) \mathcal{L}^m + \nu_u(x) \otimes (u^+(x) - u^-(x)) \mathcal{H}^{m-1} \llcorner S_u \quad (2.23)$$

into a smooth part and a jump part.

**The Perimeter.** The definition of  $TV$  is not only applicable to smooth functions  $f$ , but also to ones which may have discontinuities. For instance, one can apply it to characteristic functions  $u = \chi_E$  of sets  $E \subset \Omega$ . The value

$$\text{Per}(E; \Omega) := TV(\chi_E) \quad (2.24)$$

is called the *perimeter* of  $E$ . Sets  $E$  with  $\text{Per}(E; \Omega) < \infty$  are called *sets of finite perimeter*. For finite perimeter sets  $E$  by the definition of  $BV(\Omega, \mathbb{R})$  we have  $u = \chi_E \in BV(\Omega, \mathbb{R})$ , for instance there is a  $(m-1)$ -dimensional jump set  $S_u$  for  $u$ .

One calls

$$\partial_* E := S_u \quad (2.25)$$

the *essential boundary* of  $E$ . It is a regular hypersurface up to a set of  $\mathcal{H}^{m-1}$ -measure zero. On one side of  $\partial_* E$  there are only points inside of  $E$  and on the other side only points outside of  $E$  in the sense of (2.15). Because of this property, the essential boundary may differ from the *topological* boundary  $\partial E$ . For example, if  $E$  is a smooth  $(m-1)$ -dimensional surface, then  $\partial E = E$  and  $\partial_* E = \emptyset$ . If  $E$  is open and has a sufficiently smooth (topological) boundary, say Lipschitz, then  $\partial_* E = \partial E$  up to a  $\mathcal{H}^{m-1}$ -negligible set. Note that the definition of  $\partial_* E$  depends on  $\Omega$ ; for instance, for an open set  $E$  the boundary parts that coincide with the boundary of  $\Omega$ , or which lie outside of  $\Omega$ , are not part of  $\partial_* E$ .

A remarkable property of total variation is that the perimeter is equal to the  $(m-1)$ -dimensional Hausdorff measure of the (essential) boundary:

$$\text{Per}(E; \Omega) = \mathcal{H}^{m-1}(\partial_* E) = \text{boundary length of } E. \quad (2.26)$$

This makes it very useful for geometric applications like segmentation or 3D reconstruction, where one wants to penalize the length (in 2D, or area in 3D) of the foreground-background boundary.

We will frequently also consider the *weighted perimeter* with a weighting function  $g : \Omega \rightarrow \mathbb{R}$ :

$$\text{Per}_g(E; \Omega) = \int_{\partial_* E} g(x) d\mathcal{H}^{m-1}(x). \quad (2.27)$$

This can be represented analogously to (2.24) by

$$\text{Per}_g(E; \Omega) = TV_g(\chi_E), \quad (2.28)$$

where the weighted total variation is defined similarly as in (2.12) but constraining  $\varphi$  by  $|\varphi(x)| \leq g(x)$  instead of  $|\varphi(x)| \leq 1$ . The usual perimeter (2.24) is recovered using the weight  $g(x) = 1$  for all  $x$ .

## 2.3 The Primal-Dual Algorithm

### 2.3.1 Min-Max Problems

**Continuous Image Domain.** Convex optimization techniques to tackle computer vision problems typically transform the minimization problem (1.1) to a *min-max-problem*:

$$\min_{v \in V} \max_{p \in P} E(v, p), \quad (2.29)$$

with some reflexive Banach spaces  $V$  and  $P$ . The energy  $E(v, p)$  is assumed to be *convex in  $v$*  for fixed  $p$ , and *concave in  $p$*  for fixed  $v$ . This form naturally arises by using various convexification techniques, which typically rewrite the energy or parts of it as a supremum over some new additional variables. The most prominent example of such a technique is the convex duality (2.8).

Problems of the form (2.29) are also called *saddle-point problems*. A pair  $(v, p) \in V \times P$  is said to be a solution, or a *saddle-point* of (2.29) if  $v$  is a global minimum of  $E(\cdot, p)$  and  $p$  a global maximum of  $E(v, \cdot)$ . Note that there might be several distinct saddle-points, i.e. the solution is not necessarily unique. In general, it is more preferable to replace “min” with “inf” as well as “max” with “sup” in order to deal with cases where e.g. the max over  $p$  is not attained for any  $p$ . We will use either forms interchangeably in this thesis, since for a solution  $(v, p)$  the min and max are always attained in practice. The order of min and max can be freely exchanged, without altering the set of solutions.

The variable  $v$ , over which the minimum is taken, is called the *primal variable*, while  $p$  is called the *dual variable*. Evaluating the maximum in (2.29) for a fixed  $v$  defines the *primal energy*  $E_{\text{prim}} : V \rightarrow \mathbb{R}$ ,

$$E_{\text{prim}}(v) = \max_{p \in P} E(v, p). \quad (2.30)$$

Analogously, the minimum for a fixed  $p$  defines the *dual energy*  $E_{\text{dual}} : P \rightarrow \mathbb{R}$ ,

$$E_{\text{dual}}(p) = \min_{v \in V} E(v, p). \quad (2.31)$$

In all of our considered practical problems in this thesis, the energy (2.29) will have the following special form:

$$\min_{v \in V} \max_{p \in P} D(v) + \langle Kv, p \rangle - F(p). \quad (2.32)$$

Here, the functions  $D : V \rightarrow \mathbb{R}$  and  $F : P \rightarrow \mathbb{R}$  are convex and lower-semicontinuous,  $K : V \rightarrow P$  is a linear operator, and  $\langle \cdot, \cdot \rangle$  denotes the scalar product on  $P$ , resp.  $V$ . Thus, the energy (2.32) is comprised of a convex part  $D$  that only depends on the primal variable  $v$ , a concave part  $-F$  only depending on the dual  $p$ , and a bilinear term which mixes the primal and dual variables.

**The Discretized Min-Max-Problem.** As mentioned in the introduction Section 1.1.2, throughout this thesis we will assume the continuous viewpoint, modeling vision problems through functions defined on a spatially continuous domain  $\Omega$ . After having established a convex model (2.32) for the application at hand, the final step is then, of course, to discretize the model to be able to practically solve it. The image domain  $\Omega$  is then discretized into a finite set of pixels forming a regular rectangular grid.

Let us mention here the discretization of two frequently occurring differential operators. For the gradient  $\nabla$  we will use *forward differences* with Neumann boundary conditions. In image domain dimension  $m = 2$ , this means setting

$$(\nabla^+ f)(x, y) = ((\partial_x^+ f)(x, y), (\partial_y^+ f)(x, y)) \quad (2.33)$$

with

$$\begin{aligned} (\partial_x^+ f)(x, y) &= (f(x+1, y) - f(x, y))\chi_{(x+1, y) \in \Omega}, \\ (\partial_y^+ f)(x, y) &= (f(x, y+1) - f(x, y))\chi_{(x, y+1) \in \Omega}. \end{aligned} \quad (2.34)$$

The indicator function  $\chi_{(x+1, y) \in \Omega}$  is defined as 1 if  $(x+1, y) \in \Omega$ , i.e. if  $(x+1, y)$  is part of the discretization grid, and as 0 otherwise. This means that  $(\partial_x^+ f)(x, y)$  is nonzero only if  $(x+1, y) \in \Omega$ .

For the divergence  $\text{div}$  of functions  $g : \Omega \rightarrow \mathbb{R}^2$  we will instead use *backward differences* with Dirichlet boundary conditions:

$$(\text{div}^- g)(x, y) = (\partial_x^- g_1)(x, y) + (\partial_y^- g_2)(x, y) \quad (2.35)$$

with

$$\begin{aligned} (\partial_x^- g_1)(x, y) &= g_1(x, y)\chi_{(x+1, y) \in \Omega} - g_1(x-1, y)\chi_{(x-1, y) \in \Omega}, \\ (\partial_y^- g_2)(x, y) &= g_2(x, y)\chi_{(x, y+1) \in \Omega} - g_2(x, y-1)\chi_{(x, y-1) \in \Omega}. \end{aligned} \quad (2.36)$$

In the implementation, the  $g_{1/2}$  values should only be evaluated if their corresponding  $\chi$ -factor is 1. The above  $\text{div}$  discretization is automatically obtained from that of  $\nabla$  by requiring the partial integration theorem to be preserved in the discrete setting:

$$\sum_{(x, y) \in \Omega} \langle (\nabla^+ f)(x, y), g(x, y) \rangle = \sum_{(x, y) \in \Omega} f(x, y) (\text{div}^- g)(x, y), \quad (2.37)$$

so that  $\text{div} = -\nabla^T$  is then the negative adjoint operator of  $\nabla$ .

Ultimately, we end up with a *finite-dimensional* problem, a discrete form of (2.32):

$$\min_{x \in X} \max_{y \in Y} D(x) + \langle Kx, y \rangle - F(y). \quad (2.38)$$

The spaces  $X$  and  $Y$  are now some finite dimensional vector spaces. In practice, they will be of the form  $X = \mathbb{R}^N$  and  $Y = \mathbb{R}^M$  for some  $N, M \geq 1$ . With the standard bases of  $X$  and  $Y$ , the linear operator  $K : X \rightarrow Y$  can be viewed as a matrix  $K \in \mathbb{R}^{M \times N}$  with entries  $K = (K_{ij})_{ij}$ .

To solve the overall convex-concave min-max optimization problem (2.38), we will utilize the fast state-of-the-art *primal-dual algorithm* [98, 31]. A special case of it was first proposed in [100], and the general version subsequently

appeared in [31]. Essentially, the idea is to iteratively apply a *gradient descent* step in the primal variable  $x$  and a *gradient ascent* step in the dual variable  $y$ , with subsequent applications of so called *proximal operators*, which act as generalized projections onto constraint sets.

Before we proceed with the general primal-dual algorithm in Section 2.3.3, let us first introduce the proximal operator of functions, which is used quite extensively in the primal-dual framework.

### 2.3.2 The Prox-Operator

When minimizing convex functions  $f : X \rightarrow \mathbb{R}$ , local methods such as gradient descent can be applied to arrive at an optimal solution, but only if  $f$  is differentiable. If this is not the case, proximal operators can be used as a substitute for one gradient descent step with time step  $\tau > 0$ . More importantly in the context of this thesis, they always arise as a basic computational step in the primal-dual algorithm of Section 2.3.3.

Given a function  $f : X \rightarrow \mathbb{R}$ , its corresponding *proximal operator* [31], with parameter  $\tau > 0$ , is a mapping  $\text{prox}_{\tau, f} : X \rightarrow X$  defined by

$$\text{prox}_{\tau, f}(x) = \underset{y \in X}{\operatorname{argmin}} \left( \frac{\|y - x\|^2}{2\tau} + f(y) \right). \quad (2.39)$$

The norm  $\|\cdot\|$  used here is the one derived from the scalar product on  $X$ :  $\|x\| = \sqrt{\langle x, x \rangle}$ . The proximal operator is well-defined for  $f$  convex and lower-semicontinuous (and proper), since then a minimum always exists and is unique.

In practice, through the use of the dualization (2.8) one is often faced with the need to evaluate the proximal operator for the convex conjugate  $f^*$ . It turns out that this task can be reduced to instead computing the proximal operator of  $f$ . This formula is known as the *Moreau-Identity* [108]:

$$\text{prox}_{\tau, f^*}(x) = x - \tau \text{prox}_{\frac{1}{\tau}, f}\left(\frac{x}{\tau}\right). \quad (2.40)$$

Thus, the proximal operators of  $f^*$  and  $f$  can be easily expressed in terms of each other.

In the special case that  $f$  is the *indicator function*  $\delta_A$  of some set  $A \subset X$ , the prox operator is just the *Euclidean projection* onto  $A$ , independently of the time step  $\tau$ :

$$\text{prox}_{\tau, \delta_A}(x) = \underset{y \in A}{\operatorname{argmin}} \|y - x\|^2 =: \pi_A(x). \quad (2.41)$$

### 2.3.3 The General Primal-Dual Algorithm

This section describes the basic primal-dual algorithm [31, ‘‘Alg. 1’] for problems of the form (2.38).

**Algorithm 1** (Basic PD). *Starting values*: Choose arbitrary  $x^0 \in X$ ,  $y^0 \in Y$ , and set  $\bar{x}^0 = x^0$ . Choose some time steps  $\tau, \sigma > 0$  with  $\tau\sigma < \frac{1}{\|K\|^2}$ , where  $\|K\|$  is the operator norm of  $K : X \rightarrow Y$ . Finally, set  $\theta = 1$ .

*Iteration:* Iterate until convergence for  $k \geq 0$ :

$$y^{k+1} = \text{prox}_{\sigma, F} \left( y^k + \sigma K \bar{x}^k \right), \quad (2.42)$$

$$x^{k+1} = \text{prox}_{\tau, D} \left( x^k - \tau K^T y^{k+1} \right), \quad (2.43)$$

$$\bar{x}^{k+1} = x^{k+1} + \theta(x^{k+1} - x^k). \quad (2.44)$$

The algorithm converges to a saddle-point of (2.38) with asymptotic energy rate  $\mathcal{O}(1/N_{\text{iter}})$ , where  $N_{\text{iter}}$  is the number of iterations. This means that  $E(x^k, y^k) \leq E(x^*, y^*) + C/k$  for some saddle-point  $(x^*, y^*)$  and some constant  $C > 0$ . This rate is actually optimal, as one can show that no first-order algorithm can achieve a better rate if it is able to solve the general class of problems (2.38), see [31]. The iterates  $(x^k, y^k)$  themselves are not guaranteed to converge, but at least the averaged sequence  $\frac{1}{N} \sum_{k=1}^N (x^k, y^k)$  does converge to some (as there might be several) saddle-point of (2.38). Furthermore, every point of accumulation of  $(x^k, y^k)$  is also a saddle-point. In practice, convergence is always observed.

The structure of the algorithm is very simple, which is one of the reasons making it a popular choice for practical problems. The dual update step (2.42) consists of two parts: The first one is a usual gradient ascent w.r.t. the bilinear (and, of course, differentiable) part  $\langle y, Kx \rangle$  of the energy. Then, at this intermediate result, the proximal operator corresponding to  $y$  is computed, using the same time step parameter  $\sigma$  as has been used in the gradient ascent step.

Beside  $x^k$  and  $y^k$  the algorithm also introduces new primal variables  $\bar{x}^k$ . Essentially, doing so makes it possible to prove convergence of the overall algorithm. Taking  $\theta = 0$  the algorithm would become simpler, since then  $\bar{x}^k = x^k$  for all  $k$  and the “bar”-copies would be eliminated, but one then has the disadvantage that convergence cannot be guaranteed anymore.

We note that essentially the same algorithm can be given but where the roles of  $x$  and  $y$  are reversed:

**Algorithm 2** (Basic PD, dual-bar). Iterate until convergence for  $k \geq 0$ :

$$x^{k+1} = \text{prox}_{\tau, D} \left( x^k - \tau K^T \bar{y}^k \right), \quad (2.45)$$

$$y^{k+1} = \text{prox}_{\sigma, F} \left( y^k + \sigma K x^{k+1} \right), \quad (2.46)$$

$$\bar{y}^{k+1} = y^{k+1} + \theta(y^{k+1} - y^k). \quad (2.47)$$

This is derived by applying the above primal-dual Algorithm 1 to the problem (2.38) “multiplied by  $-1$ ”:

$$\min_{y \in Y} \max_{x \in X} F(y) + \langle -K^T y, x \rangle - D(x). \quad (2.48)$$

In the altered Algorithm 2, the “bar”-copies  $\bar{y}$  are now introduced for the duals  $y$  rather than for the primals  $x$ , and analogous convergence properties hold. In general, one should use the version of the algorithm which results in the least amount of overall scalar variables, which will then decrease both memory and

computation time. In other words, if there are less primal variables  $x$  than duals  $y$ , one should use Algorithm 1, otherwise Algorithm 2.

As for the stopping criterion, one can consider the relative *primal-dual gap*  $(E_{\text{prim}}(x^k) - E_{\text{dual}}(y^k))/E_{\text{prim}}(x^k)$  and stop when this is smaller than some predefined  $\varepsilon > 0$ . Unfortunately, this is not always possible in the case that the primal or dual energy involves *hard constraints* which have been dualized in the primal-dual formulation, see Section 2.3.6. The iterates  $x^k$  and  $y^k$  could then violate these hard constraints in every iteration  $k$ , resulting in infinite energy values and thus a useless primal dual gap. In such cases, one can either just revert to choosing a fixed number of iterations, or stop when the primal iterates do not change significantly anymore. For the latter option, we can consider the average change “per pixel”  $\frac{1}{|\Omega|} \sum_j |x_j^{k+1} - x_j^k|$  (note that there may be more than one primal variable defined for each pixel) and stop when this is less than some small  $\varepsilon$ .

Recently, [32] gave significantly simplified proofs of convergence for the above primal-dual algorithm. The convergence rates are also proved with better constants and the algorithm was generalized to somewhat more general min-max problems.

Though the above algorithm is quite versatile, it has the disadvantage that one always needs an estimate of the norm  $\|K\|$  to be able to suitably set the time steps  $\tau$  and  $\sigma$ . Next, we will describe a preconditioned variant of the algorithm which resolves this drawback.

### 2.3.4 The Preconditioned Primal-Dual Algorithm

To set the time steps automatically, we will use the convenient *preconditioning* scheme [98] for [31]. Instead of using the same time steps  $\tau$  and  $\sigma$ , the idea is to set them *individually adapted* for each of the scalar variables  $x_j$  and  $y_i$ , namely as follows:

$$\tau_j = \frac{\tau_0}{\sum_i |K_{ij}|}, \quad \sigma_i = \frac{\sigma_0}{\sum_j |K_{ij}|} \quad (2.49)$$

with some constants  $\tau_0, \sigma_0 > 0$  satisfying  $\tau_0 \sigma_0 = 1$ . In this thesis we slightly generalize the original scheme [98] introducing additional multipliers  $\tau_0$  and  $\sigma_0$ , while [98] uses  $\tau_0 = \sigma_0 = 1$ . The convergence proof of [98] can be easily extended to this case. Depending on the problem at hand, choosing  $\tau_0$  appropriately (and then  $\sigma_0 = \frac{1}{\tau_0}$ ) may significantly speed up the convergence compared to  $\tau_0 = 1$ , in some cases even by orders of magnitude.

We combine the time steps into diagonal matrices  $\mathbf{T} = \text{diag}(\tau_j)_{1 \leq j \leq N}$  and  $\mathbf{\Sigma} = \text{diag}(\sigma_i)_{1 \leq i \leq M}$ . The algorithm then goes as follows:

**Algorithm 3** (Preconditioned PD). Iterate until convergence for  $k \geq 0$ :

$$y^{k+1} = \text{prox}_{\Sigma, F} \left( y^k + \Sigma K \bar{x}^k \right), \quad (2.50)$$

$$x^{k+1} = \text{prox}_{\mathbf{T}, D} \left( x^k - \mathbf{T} K^T y^{k+1} \right), \quad (2.51)$$

$$\bar{x}^{k+1} = x^{k+1} + \theta(x^{k+1} - x^k). \quad (2.52)$$

Here the proximal operators are also extended to using individual time steps for each scalar variable, e.g.

$$\text{prox}_{\mathbb{T}, D}(\tilde{x}) = \underset{x \in X}{\operatorname{argmin}} \sum_j \frac{(x_j - \tilde{x}_j)^2}{2\tau_j} + D(x). \quad (2.53)$$

Note that the constants  $\tau_j$  in the prox step (2.53) must be exactly the same as in the linear gradient descent step for  $x^k$ . This is crucial in order to have guaranteed convergence.

For the implementation there is a quick and intuitive way of how to incorporate the adapted time steps. For example when updating  $y^k$  in (2.50), having computed the scalar update  $(K\bar{x}^k)_i = \sum_j K_{ij}\bar{x}_j^k$  for each individual  $y_i^k$ , the corresponding time step  $\sigma_j$  is equal to  $\sigma_0$  divided by the sum  $\sum_j |K_{ij}|$  of the absolute values of the used coefficients. This can be easily computed along with the updates  $(K\bar{x}^k)_i$ .

Generally, the preconditioned Algorithm 3 is always preferable to the basic one, Algorithm 1 in Section 2.3.3. Furthermore, just as in the previous section one can also give the analogous version of the algorithm where the “bar”-copies are introduced for the duals  $y$  rather than for the primals, and same comments apply:

**Algorithm 4** (Preconditioned PD, dual-bar). Iterate until convergence for all  $k \geq 0$ :

$$x^{k+1} = \text{prox}_{\mathbb{T}, D} \left( x^k - \mathbb{T}K^T\bar{y}^k \right), \quad (2.54)$$

$$y^{k+1} = \text{prox}_{\Sigma, F} \left( y^k + \Sigma K x^{k+1} \right), \quad (2.55)$$

$$\bar{y}^{k+1} = y^{k+1} + \theta(y^{k+1} - y^k). \quad (2.56)$$

### 2.3.5 The Accelerated Primal-Dual Algorithm

If either  $D$  or  $F$  in (2.38) are *uniformly convex*, then the convergence rate  $\mathcal{O}(1/N_{\text{iter}})$  can be improved to  $\mathcal{O}(1/N_{\text{iter}}^2)$  by an acceleration scheme.

One says that  $D : X \rightarrow \mathbb{R}$  is uniformly convex with a constant  $\gamma > 0$  if for any  $x, y \in X$  it holds

$$D(y) \geq D(x) + \langle a, y - x \rangle + \frac{\gamma}{2} \|y - x\|^2 \quad (2.57)$$

for every  $a \in \partial D(x)$ . Intuitively, this means that  $D(x)$  has a quadratic part and grows at least as  $\frac{\gamma}{2} \|x - x_0\|^2$  for some  $x_0 \in X$ .

The accelerated algorithm [31, “Alg. 2”] for  $D$  uniformly convex is the standard one, Algorithm 1, but with time steps  $\tau, \sigma$  and the  $\theta$ -parameter *adapting* with iterations:



**Algorithm 5** (Accelerated PD). Iterate until convergence for  $k \geq 0$ :

$$y^{k+1} = \text{prox}_{\sigma, F} \left( y^k + \sigma_k K \bar{x}^k \right), \quad (2.58)$$

$$x^{k+1} = \text{prox}_{\tau, D} \left( x^k - \tau_k K^T y^{k+1} \right), \quad (2.59)$$

$$\theta_k = \frac{1}{\sqrt{1 + 2\gamma\tau_k}}, \quad \tau_{k+1} = \tau_k \theta_k, \quad \sigma_{k+1} = \sigma_k / \theta_k, \quad (2.60)$$

$$\bar{x}^{k+1} = x^{k+1} + \theta_k (x^{k+1} - x^k). \quad (2.61)$$

The same constraints for the initial time steps apply:  $\tau_0, \sigma_0 > 0$  with  $\tau_0 \sigma_0 < \frac{1}{\|K\|^2}$ . One can show that  $\tau_k$  goes to 0 asymptotically as  $\frac{1}{\gamma k}$ , i.e.  $\tau_k \gamma k \rightarrow 1$ . The dual time steps  $\sigma_k = \frac{\sigma_0 \tau_0}{\tau_k}$  then go to  $\infty$  asymptotically as  $\gamma \tau_0 \sigma_0 k$ .

If instead  $F$  is uniformly convex with a constant  $\omega$ , one can easily write down an analogous accelerated algorithm by applying the above scheme to (2.48).

### 2.3.6 Dualization and Decoupling of Energy Terms

Having already found a convex model for a concrete vision problem at hand, and having discretized it arriving at the form (2.38), the primal-dual algorithm update equations from Sections 2.3.3, 2.3.4, or 2.3.5 can then be derived in a straightforward manner. Assuming the linear updates  $K \bar{x}^k$  and  $K^T y^{k+1}$  etc. are easily calculable, the main question is then only how *feasible and quick* it is to compute the *proximal operators* for the functions  $D$  and  $F$ . They must be computed in each iteration, so that the overall primal-dual algorithm can only be efficient if the prox operators are easy to compute, with an explicit or straightforward solution. For example, though  $D$  only depends on the primals  $x$ , it may couple many individual scalar variables  $x_j$  in a nontrivial way involving convoluted or otherwise complicated expressions. In the worst case,  $\text{prox}_{\tau, D}$  may be as hard to evaluate as the overall problem itself, making the resulting primal-dual algorithm infeasible in practice.

Therefore, having established an initial primal-dual form (2.38) for our optimization problem, the next step is always to simplify the functions  $D$  and  $F$  until their prox operators become easy to compute. This is done by means of *decoupling* and *dualizing* of complicated energy terms, which can either appear directly in the energy or in the form of hard constraints. The idea is to simplify at the cost of introducing additional variables in the problem. The final optimization is then performed also over these new variables. In the end, the algorithm will still remain easy to implement since only additional energy terms and variables are added to the problem.

**Writing Hard Constraints as Energy Terms.** Although in (2.38) the variables  $x$  and  $y$  can vary in the whole of the space  $X$  resp.  $Y$ , hard constraints are still easily possible. To incorporate a hard constraint  $x \in A$ , the idea is that the constrained minimization  $\min_{x \in A} E(x)$  of an energy  $E$  is equivalent to unconstrained minimization  $\min_x (E(x) + \delta_A(x))$  of the same energy plus the *indicator function*  $\delta_A$ , defined in (2.3). Thus, a primal-dual problem

$$\min_{x \in X \text{ s.t. } x \in A} \max_{y \in Y \text{ s.t. } y \in B} D(x) + \langle Kx, y \rangle - F(y) \quad (2.62)$$

with hard constraints  $x \in A$  and  $y \in B$  is equivalent to

$$\min_{x \in X} \max_{y \in Y} (D(x) + \delta_A(x)) + \langle Kx, y \rangle - (F(y) + \delta_B(y)), \quad (2.63)$$

which is just the form (2.38) with different  $D$  and  $F$ .

Because of this equivalence, in the following we will only consider the treatment of how to dualize a general *energy term*  $f(x)$ .

**Dualizing Primal Terms.** The general idea of dualization/decoupling is to use the property (2.8) of the convex dual. Assume that  $D(x) = D_0(x) + f(x)$  contains a (convex) term  $f(x)$  which couples many variables in a nontrivial way, thus making the prox for  $D$  hard to compute. One then first computes the convex dual  $f^*(z)$  of  $f$  by formula (2.6), and then replaces the occurrence of  $f(x)$  in the energy by its dual representation  $f(x) = \sup_{z \in X} \langle x, z \rangle - f^*(z)$ . Thus,

$$\min_{x \in X} \max_{y \in Y} (D_0(x) + f(x)) + \langle Kx, y \rangle - F(y) \quad (2.64)$$

equivalently becomes

$$\min_{x \in X} \max_{y \in Y} \left( D_0(x) + \left( \sup_{z \in X} \langle x, z \rangle - f^*(z) \right) \right) + \langle Kx, y \rangle - F(y), \quad (2.65)$$

or

$$\min_{x \in X} \max_{(y, z) \in Y \times X} D_0(x) + (\langle Kx, y \rangle + \langle x, z \rangle) - (F(y) + f^*(z)). \quad (2.66)$$

This is again a problem of the general form (2.38). New *dual* variables  $z$  have been introduced into optimization, and the overall set of dual variables is  $(y, z)$ .

The effect of the reformulation is that the prox for  $x$  is now *easier* to compute as we now only need to deal with  $\text{prox}_{\tau, D_0}$  instead of  $\text{prox}_{\tau, D_0+f}$ . The bilinear term is extended from  $\langle Kx, y \rangle$  to  $\langle Kx, y \rangle + \langle x, z \rangle$  but is still easy to handle. Finally, the prox for the new duals  $(y, z)$  *always* decomposes into separately computing the prox for the “old” duals  $y$  and the prox for the new ones  $z$ . In fact,

$$(y, z) = \underset{(y, z) \in Y \times X}{\operatorname{argmin}} \frac{\|y - y_0\|^2}{2\sigma} + \frac{\|z - z_0\|^2}{2\sigma} + F(y) + f^*(z) \quad (2.67)$$

is equivalent to  $y = \text{prox}_{\sigma, F}(y_0)$  and  $z = \text{prox}_{\sigma, f^*}(z_0)$ . The number of additionally needed variables  $z$  is the number of real arguments of  $f$ . For example, if we have  $D(x) = D_0(x) + f(x_2, x_5)$ , i.e.  $f$  depends only on two variables, the dualization of  $f$  also only requires two new real variables, even though the primal vector  $x$  may consist of thousands of individual real variables.

In practice, variable coupling often originates from *linear subexpressions*, i.e. the coupling terms  $f$  have the form  $f(x) = g(l(x))$  with some linear vectorial mapping  $l : X \rightarrow \mathbb{R}^L$ ,  $L \geq 1$ , and a convex  $g : \mathbb{R}^L \rightarrow \mathbb{R}$ . In this case, it is more convenient to dualize w.r.t. the linear term  $l(x)$  instead of  $x$ , i.e. the idea is to apply the above dualization to  $g$  instead of  $f$ . Thus, one first computes the dual

$g^*$  of  $g$  through (2.6), and then replaces the term  $g(l(x))$  in the energy by its dual representation

$$g(l(x)) = \sup_{z \in \mathbb{R}^L} \langle l(x), z \rangle - g^*(z). \quad (2.68)$$

This way,

$$\min_{x \in X} \max_{y \in Y} (D_0(x) + g(l(x))) + \langle Kx, y \rangle - F(y) \quad (2.69)$$

equivalently becomes

$$\min_{x \in X} \max_{(y,z) \in Y \times \mathbb{R}^L} D_0(x) + (\langle Kx, y \rangle + \langle l(x), z \rangle) - (F(y) + g^*(z)). \quad (2.70)$$

The difference to (2.66) is that (2.70) only requires to compute the prox of the simpler  $g^*$  instead of  $(g \circ l)^*$ . For example,  $f(x) = \sqrt{(x_1 - x_2)^2 + (x_1 - x_3)^2}$  can be written as  $f(x) = g(l(x))$  with  $l(x) = (x_1 - x_2, x_1 - x_3) \in \mathbb{R}^2$  and  $g(w) = \sqrt{w_1^2 + w_2^2}$ , and  $g$  is much easier to handle w.r.t. the prox operator computation than  $f$ . Note that it is actually not needed to compute  $g^*$  explicitly, one could use the Moreau-Identity (2.40) to reduce the prox of  $g^*$  to that of  $g$ .

This dualization technique can of course be applied several times over, separating more and more of the individual coupling terms  $f$  out of  $D$ , each of which should have an easy to compute prox. For instance, a hard constraint defined through many equations can be handled by dualizing each of them individually.

**Dualizing Dual Terms.** The same dualization/decoupling technique can of course be equally used for the dual terms in  $F$  in (2.38). The only difference is that, because of the minus sign in front of  $F$ , one ends up with new *primal* rather than dual variables.

**Dualization for Special Cases.** In the following we will give the dualizations (2.68) for some special cases of hard constraints or coupling energy terms, which will frequently occur in the applications in this thesis. Each case reduces to computing the corresponding convex conjugate  $g^*$  and can be easily verified. Below,  $x \in \mathbb{R}^n$  is a general vectorial variable of  $n \geq 1$  elements (not necessarily equal to the primal variable  $x$  in (2.38)).

- Hard constraints for scalar linear expressions. Let  $l : X \rightarrow \mathbb{R}$  be linear. In each case,  $t \in \mathbb{R}$  can be a constant or one of the variables of optimization.

$$\delta_{l(x)=t} = \sup_{b \in \mathbb{R}} b(l(x) - t), \quad (2.71)$$

$$\delta_{l(x) \leq t} = \sup_{\substack{b \in \mathbb{R} \\ b \geq 0}} b(l(x) - t), \quad (2.72)$$

$$\delta_{l(x) \geq t} = \sup_{\substack{b \in \mathbb{R} \\ b \leq 0}} b(l(x) - t). \quad (2.73)$$

- Hard constraints for vectorial linear subexpressions. Let  $l : \mathbb{R}^n \rightarrow \mathbb{R}^m$  with some  $m \geq 1$  be linear, and  $\lambda > 0$  a constant. In (2.74), the vector

$t \in \mathbb{R}^m$  can be a constant or one of the optimization variables, while in (2.76)  $t \geq 0$  is assumed to be scalar. The norm  $\|\cdot\|$  on  $\mathbb{R}^m$  can be any norm, and  $\|z\|_* = \sup_{\|a\| \leq 1} \langle a, z \rangle$  denotes its dual norm.

$$\delta_{l(x)=t} = \sup_{a \in \mathbb{R}^m} \langle a, l(x) - t \rangle, \quad (2.74)$$

$$\delta_{\|l(x)\| \leq \lambda} = \sup_{a \in \mathbb{R}^m} \langle a, l(x) \rangle - \lambda \|a\|_*, \quad (2.75)$$

$$\delta_{\lambda \|l(x)\| \leq t} = \sup_{\substack{a \in \mathbb{R}^m, b \in \mathbb{R} \\ \|a\|_* \leq \lambda b}} \langle a, l(x) \rangle - bt. \quad (2.76)$$

- Norms and quadratic expressions. Let  $l : \mathbb{R}^n \rightarrow \mathbb{R}^m$  be linear,  $\lambda > 0$  a constant, and  $t \geq 0$  a constant or a variable of optimization.

$$\lambda \|l(x)\| = \sup_{\substack{a \in \mathbb{R}^m \\ \|a\|_* \leq \lambda}} \langle a, l(x) \rangle, \quad (2.77)$$

$$\lambda \|l(x)\|^2 = \sup_{a \in \mathbb{R}^m} \langle a, l(x) \rangle - \frac{1}{4\lambda} \|a\|_*^2, \quad (2.78)$$

$$\delta_{t \geq 0} + \lambda \frac{\|l(x)\|^2}{t} = \sup_{\substack{a \in \mathbb{R}^m, b \in \mathbb{R} \\ \frac{1}{4\lambda} \|a\|_*^2 \leq b}} \langle a, l(x) \rangle - bt. \quad (2.79)$$

The dualizations (2.71) and (2.74) of linear constraints are also known as the method of ‘‘Lagrange multipliers’’. These Lagrange multipliers are here the new variables  $b$  resp.  $a$ . An alternative way to impose linear constraints in the form of equivalent energy terms is the ‘‘augmented Lagrangian’’ method, which in addition to the linear term also includes a quadratic penalization  $\frac{1}{2\theta} |l(x) - t|^2$  in the energy. For certain special problems, [74] shows that this can lead to a faster convergence by a factor of 2 (smaller constant in the convergence rate), though the asymptotic convergence rate  $\mathcal{O}(1/N_{\text{iter}})$  still remains the same.

## Part I

# Multilabel Segmentation



## Chapter 3

# The Multilabel Problem: Common Approaches

This first part, consisting of Chapters 3, 4, 5 and 6, will be about minimizing energies

$$\min_l E(l) \tag{3.1}$$

for functions  $l$  with a fixed *finite* range:

$$l : \Omega \rightarrow \{1, \dots, n\}, \quad n \geq 1, \tag{3.2}$$

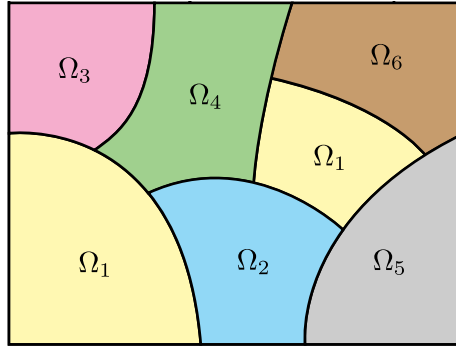
defined over an image domain  $\Omega \subset \mathbb{R}^m$ ,  $m \geq 1$ . Such energies are called *multiregion segmentation*, *image partitioning*, or simply *multilabel* problems. The range itself could have any structure other than (3.2), e.g. be some multidimensional rectangular grid of points. The essential part is that  $l$  assumes only finitely many values.

In the current Chapter 3 we will first give an overview over the multilabel problem and present some established state-of-the-art approaches for the *length regularity prior*. Subsequently, in the next three chapters we will present novel convex relaxations for multilabel problems which each take a different *higher-level knowledge prior* into account. Chapter 4 extends the common length regularity prior to nonmetric distance functions. Chapter 5 describes how constraints on the global geometric ordering of labels can be cast in a convex way. Finally, Chapter 6 provides a convex relaxation for the proportion priors to achieve a more robust and scale-invariant segmentation.

### 3.1 Introduction

#### 3.1.1 The Multilabel Problem

Let a set of possible labels be given:  $\mathcal{L} := \{1, \dots, n\}$  with a  $n \geq 1$ . The linear label enumeration is used only for the convenience of notation. The labels themselves may have any meaning. They could carry some structure such as being some multidimensional vectors, or denote any abstract notions or concrete objects such as “background” or “sky”. The only requirement is that there are finitely many labels, so that we can enumerate them.



**Figure 3.1: Image partitioning problem.** The goal is to find a subdivision of the image domain  $\Omega$  into  $n$  subregions  $\Omega_1, \dots, \Omega_n$  in an optimal way according to an energy. The subregions can have any shape or consist of separate regions (e.g. the yellow region  $\Omega_1$ ).

The task of the *multilabeling* problem is to assign each point  $x \in \Omega$  in the image domain a certain *label*  $i \in \mathcal{L}$  in an optimal way. That is, we look for a *labeling* function  $l : \Omega \rightarrow \mathcal{L}$ , which assigns each point  $x$  its corresponding label  $l(x) \in \mathcal{L}$ , minimizing some predefined energy  $E(l)$ .

Another viewpoint is as an *image partitioning* problem. The goal here is to decompose the image domain  $\Omega$  into meaningful separate subregions,  $\Omega_1, \dots, \Omega_n \subset \Omega$ . These must form a *partition* of the image domain:

$$\bigcup_{i=1}^n \Omega_i = \Omega, \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j. \quad (3.3)$$

In other words, each point  $x \in \Omega$  is contained in some region  $\Omega_i$ , and this is then the only region containing  $x$ , see Figure 3.1. One seeks for a partition with a minimal energy  $E(\Omega_1, \dots, \Omega_n)$ .

The two viewpoints are entirely equivalent. Given a labeling  $l : \Omega \rightarrow \mathcal{L}$ , we can consider the corresponding regions  $\Omega_i = l^{-1}(\{i\}) = \{x \in \Omega \mid l(x) = i\} \subset \Omega$  where each label  $i$  is assigned. They then form a partitioning of the image domain. Conversely, each partitioning (3.3) directly corresponds to a labeling, setting  $l(x)$  to the index  $i$  of the subregion  $\Omega_i$  containing  $x$ . In the following we will mainly focus on the multilabel view of the problem.

### 3.1.2 Energy: Data Term and Regularizer

The energy is usually formulated as a sum of a *data term* and a *regularizer* as in (1.2):

$$E(l) = E_{\text{data}}(l) + E_{\text{reg}}(l). \quad (3.4)$$

The data term measures the local cost of assigning a label  $i$  at a certain point  $x$ , while the regularizer is responsible for regularity and global consistency of the labeling.



**Data term.** For each point  $x \in \Omega$ , one considers a specific *local cost*  $c_i(x) \in \mathbb{R}$  of assigning a label  $i$  at this point. In other words, at each  $x$  one has a local preference as to which labels are more likely, namely those  $i$  for which the value  $c_i(x)$  is lower. The data term is generally modelled as being pointwise, accumulating the local costs incurred by assigning pixels to their labels:

$$E_{\text{data}}(l) = \int_{\Omega} c_{l(x)}(x) \, dx = \sum_{i=1}^n \int_{\Omega_i} c_i(x) \, dx. \quad (3.5)$$

We assume  $c_i \in L^2(\Omega; \mathbb{R})$  for each  $i$  to make sure the integrals in (3.5) are finite. In computer vision applications, the costs  $c_i$  specify how well a given label  $i$  fits some observed data. They can be arbitrarily sophisticated, derived from statistical models or complicated local matching scores.

**Regularizer.** The values  $c_i(x)$  are typically only an estimate of which label is *locally* more preferable. In real-world scenarios with noisy data, minimizing only the data term, setting  $l(x) = \operatorname{argmin}_{i \in \mathcal{L}} c_i(x)$  independently for all  $x \in \Omega$  may result in a labeling with undesired local fluctuations or violating some global consistency constraints.

Because of this, we also want the optimal assignment to exhibit a certain *regularity*, by penalizing each candidate labeling with a regularizer

$$E_{\text{reg}}(l). \quad (3.6)$$

It acts as a *prior* on label configurations, in other words it reflects our knowledge about which label configurations are a-priori more likely. This is where higher-level knowledge about possible or likely labelings, depending on each application at hand, can be accounted for.

Often one enforces a kind of *local* spatial coherence, where one wants neighboring points to likely have the same labels. This can be achieved by preferring labelings with regular *interface boundaries*  $\partial\Omega_i \cap \partial\Omega_j$  between the partition regions. For example, this can be achieved by penalizing the total interface length. Another way is to additionally minimize the curvature of the boundaries. More global features can additionally be taken into account, such as imposing restrictions on the area, shape, or relative geometric location of each region. Even “fully global” information can be considered, such as whether a label occurs at all anywhere in the image or whether two certain labels occur simultaneously.

### 3.1.3 Examples

Multilabel problems are a very important class of energies in computer vision. A multitude of fundamental problems can be expressed in this way. A prominent example is *image segmentation* [33, 76, 29, 140], which directly corresponds to the partitioning problem. In *foreground-background* image segmentation the image is to be divided into two non-overlapping parts which are homogeneous in some predefined way. The foreground is usually an object of interest one wishes to extract from or detect in the image, such as a ‘car’ in a street scene. In

3D reconstruction [38], which can be seen as a specific instance of foreground-background segmentation, the foreground is the 3D object. An extension is *multiregion* image segmentation, where the image is to be decomposed into several non-overlapping regions, one of which serves as image background.

Another common source of multilabel problems is *range discretization*: Starting with a problem where solutions are functions with a continuous range, constraining the range to a predefined and meaningful *finite* set yields a multilabel problem. This approach is motivated by the observation that it is often possible to derive convex formulations of the range discretized problem. Examples where this can be successfully applied include stereo reconstruction [101] where the range is a bounded interval, and optic flow estimation [53] where the range is a bounded set of 2D-vectors. In image applications such as computing piecewise-smooth or piecewise-constant approximations using the Mumford-Shah functional [100], the image range can be discretized into  $n$  grayscale values, resp.  $n$  most dominant colors.

### 3.1.4 Length Regularity Prior

By far the most common approach to label regularity is *interface length minimization* (resp. *interface area* in the case that the image domain  $\Omega$  has dimension  $m = 3$ ). This choice is popular in practical applications since it can be easily formulated in a convex way and allows for quick optimization schemes. We will focus on this prior in the current chapter.

The length regularizer measures the total length of all interfaces between each pair of two labels:

$$E_{\text{reg}}(l) = \sum_{1 \leq i < j \leq n} d(i, j) \text{Length}(\partial\Omega_i \cap \partial\Omega_j). \quad (3.7)$$

More precisely, by length we mean the  $\mathcal{H}^{m-1}$ -measure and by  $\partial\Omega_i$  the essential boundary  $\partial_*\Omega_i$ , see (2.25). The boundary lengths in (3.7) are *weighted* by a label transition function  $d(i, j) \in \mathbb{R}$  for each pair  $(i, j)$ . This way it is possible to ensure that certain labels are more likely to occur next to each other than others. Choosing  $d(i, j)$  small favors that the labeling can locally switch from  $i$  and  $j$ , while a large  $d(i, j)$  has the opposite effect.

For the case that there is a natural *linear ordering* for the labels, a simple prior is the linear label cost:  $d(i, j) = \lambda|i - j|$  for some constant  $\lambda > 0$ . This corresponds to the total variation regularization of the labeling function  $l : \Omega \rightarrow \{1, \dots, n\}$ , i.e. then  $E_{\text{reg}}(l) = \lambda TV(l)$ .

A popular prior is the *Potts model* [104]. Here all label transitions are penalized equally:

$$d(i, j) = \begin{cases} \lambda & \text{if } i \neq j, \\ 0 & \text{else} \end{cases} \quad (3.8)$$

with a parameter  $\lambda > 0$ . Since each interface  $\partial\Omega_i \cap \partial\Omega_j$  lies between exactly two subregions, we can rewrite (3.7) equivalently as

$$E_{\text{reg}}(l) = \frac{\lambda}{2} \sum_{i=1}^n \text{Length}(\partial\Omega_i). \quad (3.9)$$

## 3.2 Computational Approaches

Finding an optimal labeling  $l$  is a hard computational challenge as the overall energy is generally not convex. For some cases, good results may be obtained by local minimization, starting from a good initialization, possibly further improved by coarse-to-fine strategies commonly employed in optical flow estimation. Yet, such methods cannot guarantee any form of quality of the result and performance typically depends on data, on initialization and on the choice of the algorithmic minimization scheme (number of levels in the coarse-to-fine hierarchy, number of iterations per level, etc.).

In recent years, researchers have made substantial progress regarding algorithms which allow to compute optimal and near-optimal solutions for certain problem classes.

### 3.2.1 Discrete Approaches

In the discrete setting with a discrete domain (a grid of pixels) and range of the solution functions, energy functionals can be formulated in the framework of Markov random fields (MRFs).

In the case that the label space is *binary*,  $n = 2$ , the energy can be discretized using a graph representation, where the nodes denote discrete pixel locations and the edge weights encode the energy functional. Fast combinatorial graph cut based algorithms are frequently employed to compute the minimizers. If the energy is *submodular*, a global solution can be found by computing a minimum cut [56, 19, 71]. Continuous variants of this minimum cut problem have also been studied [118].

Among the first computational paradigms for efficiently solving multilabel problems with  $n \geq 3$  labels, was the graph cut approach of Ishikawa [60]. It is applicable in the important case of a *linearly ordered* label set and a *convex* regularizer. For the regularizer (3.7), this case occurs when the distance function has the form  $d(i, j) = f(|i - j|)$  with some convex function  $f$ , e.g. the linear distance  $d(i, j) = \lambda |i - j|$ . The global optimum then corresponds to a cut in a multi-layered graph which can be computed in polynomial time [60]. A different discrete encoding scheme for this problem was also presented in [112].

For general cost functions with  $n \geq 3$ , only approximate solutions can be found. For instance, for the regularizer (3.7) this is the case for the Potts prior (3.8). Boykov et al. [19] introduced the concepts of  $\alpha$ -expansion and  $\alpha$ - $\beta$ -swap to approximate the hard multilabel problems through a sequence of binary problems. Other approaches for the general case include linear programming relaxations [134, 72] and quadratic pseudo-boolean optimization [70].

### 3.2.2 Continuous Approaches

Continuous approaches are based on a spatially continuous representation of the image domain. This avoids typical drawbacks of the discrete graph-based approaches such as anisotropy and metrication errors. The multilabel problem is addressed by means of *convex relaxation*. The approaches strive to find a convex lower bound as close as possible to the original functional, and this bound can

then be minimized globally. How good the bound is depends on the *tightness* of the relaxation, i.e. how close the energy is to the convex envelope relaxation.

For particular labeling problems, this strategy leads to globally optimal solutions. An example is the two-label problem,  $n = 2$ , with length regularity, which can be solved in a globally optimal way [33], as was also the case in the discrete setting.

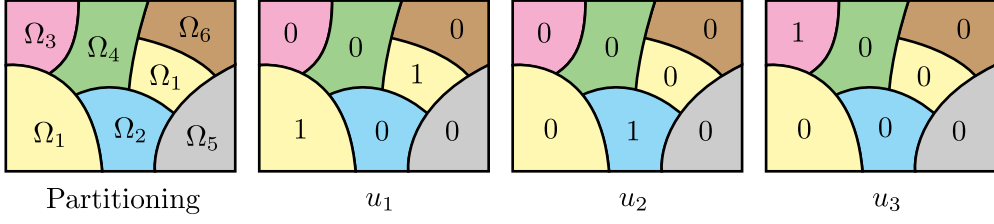
Another instance where optimal solutions can be found are special multilabel problems with  $n \geq 3$  as described above for the discrete setting, namely the special case of a linearly ordered set of labels and convex regularizers. In [102, 101] it was shown that globally optimal solutions can be achieved in this case. The convex relaxation is constructed based on the idea of *functional lifting*, first assuming a continuous labeling space and in the end discretizing it to a finite space. This is related to the corresponding discrete approach of Ishikawa [60], but the resulting complexity may be different from case to case. For example, for the quadratic distance function  $d(i, j) = \lambda(i - j)^2$ , the Ishikawa graph construction [60] requires a quadratic number of edges, while the convex relaxation approach results in a linear number of constraints.

For the general multilabel case  $n \geq 3$ , no relaxation is known that would lead to provably optimal solutions. However, it seems unlikely that such a relaxation exists at all, since the corresponding discrete formulations typically pose NP-hard problems [19]. Relaxations of different tightness and computational efficiency have been proposed for the general length regularity prior (3.7), respectively for the Potts special case (3.9). The first approaches appeared around the same time in 2008, and include [29, 140, 76], and more recently [79, 143, 141]. In [140] a relaxation is given specifically for the Potts model (3.9), and [76] covers the more general case (3.7) with so called Euclidean-representable label distances  $d$ . The relaxation [29, 30] gives the tightest possible local relaxation for a certain general class of distance functions  $d$  including the Potts model (3.9), but this relaxation has a higher computational complexity. It was generalized to all metric distances  $d$  in [79]. In [95] the authors give a detailed survey with publicly available code.

### 3.3 Convex Relaxation

#### 3.3.1 Indicator Function Representation

Classically, a popular way to approach partitioning problems was through *levelset methods* [96]. In the case of two labels, the idea is to represent the foreground boundary as the zero level set  $\{x \mid \varphi(x) = 0\}$  of a scalar function  $\varphi : \Omega \rightarrow \mathbb{R}$  defined on the image domain  $\Omega$ , and this was also extended to the multilabel case. The main disadvantage is that the resulting energies are not convex in  $\varphi$ , so that only local optimization methods could be applied. In the pioneering work [33] Chan et al. observed that the energy can be made convex by using the *indicator function* representation instead. In fact, they showed that the two-label segmentation problem can be solved optimally this way. This has popularized the use of this representation and has made it an established way to approach multilabel problems.



**Figure 3.2: Indicator function representation.** The indicator function  $u_i : \Omega \rightarrow \{0, 1\}$  for each label  $i$  is defined as 1 in image points where this label is attained, and 0 otherwise.

Instead of minimizing (3.4) directly in terms of the labeling function  $l : \Omega \rightarrow \mathcal{L}$ , we will work with the individual *label indicator functions*, defined as the characteristic functions

$$u_i = \chi_{\Omega_i} : \Omega \rightarrow \mathbb{R}, \quad i = 1, \dots, n, \quad (3.10)$$

of regions where each individual label  $i$  is attained, see Figure 3.2:

$$u_i(x) = \begin{cases} 1 & \text{if } l(x) = i, \\ 0 & \text{else.} \end{cases} \quad (3.11)$$

Being an implicit representation, this allows for any shape or topology of the regions  $\Omega_i$ , for instance  $\Omega_i$  may consist of multiple disjoint regions. We remark that the name “indicator function” is also used for *set* indicator functions  $\delta_A$  from convex analysis in (2.3), but there will not be any confusion with *label* indicator functions as defined here.

Since each point  $x \in \Omega$  belongs to one and only one region  $\Omega_i$ , the label indicator functions must satisfy a *label-uniqueness constraint*. Namely, exactly one of the label indicator function values  $u_1(x), \dots, u_n(x)$  is 1 while all others are 0. We can write this as  $u : \Omega \rightarrow \Delta_0$  with the binary simplex

$$\Delta_0 = \left\{ z \in \{0, 1\}^n \mid \sum_{i=1}^n z_i = 1 \right\} \subset \mathbb{R}^n. \quad (3.12)$$

The data term (3.5) can be directly formulated in terms of  $u_i = \chi_{\Omega_i}$  as

$$E_{\text{data}}(u) = \sum_{i=1}^n \int_{\Omega} c_i(x) u_i(x) dx. \quad (3.13)$$

The regularizer in terms of  $u$  is given by the expression (3.7) indirectly through the relation (3.10). For the Potts special case (3.9) it is possible give a simpler and more direct expression. Namely, we can use the property (2.24) to rewrite the boundary lengths directly in terms of the corresponding indicator functions  $u_i = \chi_{\Omega_i}$  through the use of the total variation:

$$E_{\text{reg}}(u) = \frac{\lambda}{2} \sum_{i=1}^n TV(u_i). \quad (3.14)$$

In order for  $TV$  to be applicable, we need to assume that  $u : \Omega \rightarrow \Delta_0$  is a function of bounded variation, so that the following is a natural domain set for the indicator functions:

$$\mathcal{D}_0 = \text{BV}(\Omega; \Delta_0). \quad (3.15)$$

The overall energy (3.4) becomes

$$\min_{u \in \mathcal{D}_0} \sum_{i=1}^n \int_{\Omega} c_i(x) u_i(x) dx + E_{\text{reg}}(u) \quad (3.16)$$

with  $E_{\text{reg}}(u)$  in (3.7), resp. (3.14) for the Potts special case.

### 3.3.2 Relaxed Indicator Functions and Convex Relaxation

Disregarding the constraints on  $u$ , the energy (3.16) is already convex, since the data term is linear and the regularizer is based on the convex total variation. However, the domain for  $u$  is not convex and thus does not allow efficient optimization. The reason is the discrete range  $\{0, 1\}$  of the indicator functions.

To overcome this, the idea is to *relax* the range  $\{0, 1\}$  to the interval  $[0, 1]$ . This essentially replaces the binary simplex  $\Delta_0$  in (3.12) with its convex hull

$$\Delta = \left\{ z \in [0, 1]^n \mid \sum_{i=1}^n z_i = 1 \right\} \subset \mathbb{R}^n. \quad (3.17)$$

Thus, we will consider the relaxed indicator functions  $u$  from the domain

$$\mathcal{D} = \text{BV}(\Omega; \Delta). \quad (3.18)$$

Later in Section 3.4.5 we will discuss how a “binary” solution of the original problem, i.e. with values in  $\Delta_0$ , can be recovered from a relaxed one.

With the transition from  $\mathcal{D}_0$  to  $\mathcal{D}$ , the domain of definition of the energy has become larger by including also nonbinary  $u$ . The question is now how to properly define the energy  $E$  for these new arguments. For the extension of the data term, the same linear expression can be chosen. Fixing also a convex relaxation  $\bar{E}_{\text{reg}} : \mathcal{D} \rightarrow \mathbb{R}$  for the regularizer, the overall relaxed problem becomes

$$\min_{u \in \mathcal{D}} \sum_{i=1}^n \int_{\Omega} c_i(x) u_i(x) dx + \bar{E}_{\text{reg}}(u). \quad (3.19)$$

The requirements for  $\bar{E}_{\text{reg}} : \mathcal{D} \rightarrow \mathbb{R}$  is that it must be convex on  $\mathcal{D}$ , and coincide with the original values on  $\mathcal{D}_0$ :

$$\bar{E}_{\text{reg}}(u) = E_{\text{reg}}(u) \quad \forall u \in \mathcal{D}_0. \quad (3.20)$$

One could also only require inequality “ $\leq$ ” in (3.20), but it turns out that it is well possible to find convex relaxations satisfying (3.20) exactly.

The tighter the relaxation  $\bar{E}_{\text{reg}}$ , i.e. the higher its values, the more likely is it that one recovers a true solution of the original problem (3.16). There are several classical ways to define a convex relaxation and thus extend the regularizer to nonbinary indicator functions, which we will present next. Interestingly, they were all proposed at about the same time at the end of 2008.

## 3.4 Relaxations for the Length Prior

### 3.4.1 Dual Representation of the Convex Relaxation

The original regularizer for *binary* indicator functions  $u \in \mathcal{D}_0$  for the Potts special case (3.14) can be written in a dual representation by applying the definition (2.12) of  $TV$  separately to each term:

$$E_{\text{reg}}(u) = \sup_{p \in \mathcal{C}_0} \sum_{i=1}^n \int_{\Omega} \langle p_i, dDu_i \rangle \quad \forall u \in \mathcal{D}_0 \quad (3.21)$$

with the dual constraint set

$$\mathcal{C}_0 = \left\{ p \in C_c^1(\Omega; \mathbb{R}^{m \times n}) \mid |p_i(x)| \leq \frac{\lambda}{2} \quad \forall i \in \mathcal{L}, x \in \Omega \right\}. \quad (3.22)$$

This motivates to also start with a dual representation for finding convex relaxations  $\bar{E}_{\text{reg}}$  of  $E_{\text{reg}}$ , defined on possibly nonbinary  $u \in \mathcal{D}$ :

$$\bar{E}_{\text{reg}}(u) = \sup_{p \in \mathcal{C}} \sum_{i=1}^n \int_{\Omega} \langle p_i, dDu_i \rangle \quad \forall u \in \mathcal{D} \quad (3.23)$$

with a convex constraint set similar to (3.22):

$$\mathcal{C} = \left\{ p \in C_c^1(\Omega; \mathbb{R}^{m \times n}) \mid p(x) \in \mathcal{C}_{\text{loc}} \quad \forall x \in \Omega \right\}. \quad (3.24)$$

Then  $\bar{E}_{\text{reg}}$  is automatically convex. The local set  $\mathcal{C}_{\text{loc}} \subset \mathbb{R}^{m \times n}$  in (3.24) must be chosen suitably and as large as possible so that (3.20) still holds. A larger  $\mathcal{C}_{\text{loc}}$  means more dual functions  $p_i$  over which the supremum in (3.23) is taken, resulting in a larger value  $\bar{E}_{\text{reg}}(u)$  and thus a tighter relaxation. Different existing relaxations differ only in the choice of  $\mathcal{C}_{\text{loc}}$ . While the first two are only for the Potts model (3.7), the last one applies to the general case (3.9).

### 3.4.2 Relaxation of Zach et al.

The most straightforward relaxation for the special case (3.9) is to take the same set (3.22) as in the original regularizer, i.e. the local set is

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid |p_i| \leq \frac{\lambda}{2} \quad \forall i \in \mathcal{L} \right\}. \quad (3.25)$$

This is the relaxation proposed by Zach et al. [140]. It proposes to use the original  $TV$ -based formulation for all values  $u$ , binary or nonbinary: The relaxed regularizer is given by the same expression (3.14) even for nonbinary  $u$ :

$$E_{\text{reg}}(u) = \frac{\lambda}{2} \sum_{i=1}^n TV(u_i). \quad (3.26)$$

It is straightforward to project onto the set (3.25). This makes this relaxation a popular choice in multilabel optimization problems, as it provides a good length regularization effect and is still efficiently computable.

### 3.4.3 Relaxation of Lellmann et al.

**Potts Metric.** In the relaxation of Lellmann et al. [76, 79], again for the special case (3.9), one chooses the local set as

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid |p| \leq \frac{\lambda}{\sqrt{2}} \right\}. \quad (3.27)$$

Here  $|p| = \sqrt{\sum_{i=1}^n |p_i|^2}$  is the Frobenius-Norm of the matrix  $p$ . This corresponds to replacing the original regularizer by the vectorial  $TV$ :

$$\bar{E}_{\text{reg}}(u) = \frac{\lambda}{\sqrt{2}} \int_{\Omega} \sqrt{\sum_{i=1}^n |\nabla u_i|^2} dx \quad (3.28)$$

(resp. its suitable extension to nonsmooth functions).

It is also easy to project onto this set, so that the resulting multilabel convex relaxation is easy to handle. The results are typically slightly inferior than with the Zach et al. relaxation, as in experiments the relaxed solution  $u$  tends to be nonbinary for more points  $x \in \Omega$  than when using (3.25), see Figure 3.3.

**Euclidean Metrics.** The approach also applies to a more general class of distance functions  $d$ , namely to *Euclidean* ones. These are distances  $d$  such that for each label  $i$  there is a vector  $a_i \in \mathbb{R}^M$ , with an  $M \geq 1$  fixed, such that the distance  $d$  is given by the Euclidean distance of the  $a_i$ 's:

$$d(i, j) = |a_i - a_j|. \quad (3.29)$$

Consider the function  $\hat{l}: \Omega \rightarrow \mathbb{R}^M$  defined by

$$\hat{l}(x) = \sum_{i=1}^n a_i u_i(x). \quad (3.30)$$

If a label  $i$  is assigned at the point  $x \in \Omega$ , then only the label indicator function  $u_i(x) = 1$  is nonzero, and we have  $\hat{l}(x) = a_i$ . For a general possibly non-binary  $u$ ,  $\hat{l}(x)$  is some convex combination of the  $a_i$ 's. The authors of [76, 79] propose to use the convex relaxation

$$\bar{E}_{\text{reg}}(u) = TV(\hat{l}) = TV\left(\sum_{i=1}^n a_i u_i\right) \quad (3.31)$$

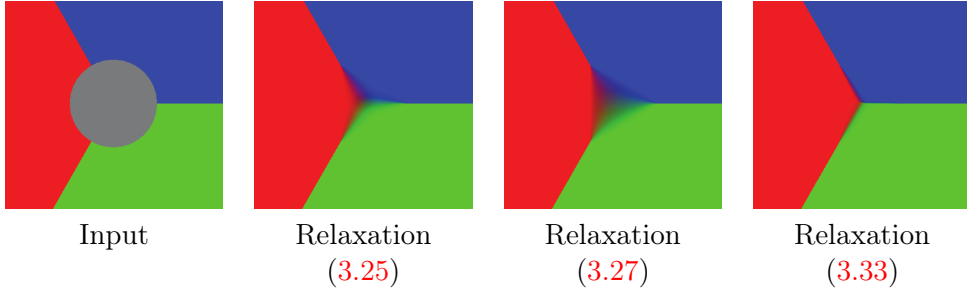
for the length regularizer (3.7). By the dual representation (2.12) of total variation, the local set (3.27) in the generalized case is

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid p_i = \sum_{j=1}^M (a_i)_j q_j \quad \forall i, \quad \text{for a } q \in (\mathbb{R}^m)^M \text{ with } |q| \leq 1 \right\}. \quad (3.32)$$

In other words,  $\mathcal{C}_{\text{loc}}$  is the image of the unit ball in  $\mathbb{R}^{m \times M}$  under the right-multiplication by  $a = (a_1, \dots, a_n) \in \mathbb{R}^{M \times n}$ .

For example, choosing  $M = n$  and  $a_i = \frac{\lambda}{\sqrt{2}} e_i$  with the standard basis  $e_1, \dots, e_n$  of  $\mathbb{R}^n$ , the distance function  $d$  defined by (3.29) is exactly the Potts metric (3.8), and the relaxation (3.31) becomes (3.28).





**Figure 3.3: Comparison of the interface length relaxations.** The values inside the circle are to be inpainted. The relaxation (3.33) yields an almost binary labeling, while the other two are less tight and the labeling is nonbinary over a larger region [30].

#### 3.4.4 Relaxation of Chambolle et al.

The relaxation of Chambolle et al. [29, 99, 30] is the tightest possible relaxation of the form (3.23). In contrast to the previous two relaxations, it also covers the general case (3.7) and not only (3.9) or only Euclidean  $d$ 's. The local constraint set  $\mathcal{C}_{\text{loc}}$  is defined as follows:

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid |p_j - p_i| \leq d(i, j) \quad \forall i < j \right\}. \quad (3.33)$$

For the special case (3.8) this set becomes

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid |p_j - p_i| \leq \lambda \quad \forall i < j \right\}. \quad (3.34)$$

One can see directly that (3.34) is larger than both (3.25) and (3.27). For the first one (3.25) this follows from the estimate

$$|p_j - p_i| \leq |p_j| + |p_i| \leq \frac{\lambda}{2} + \frac{\lambda}{2} = \lambda. \quad (3.35)$$

For the second (3.27), it is implied by the Cauchy-Schwarz-inequality

$$|p_j - p_i| \leq |p_j| \cdot 1 + |p_i| \cdot 1 \leq \sqrt{|p_j|^2 + |p_i|^2} \sqrt{1^2 + 1^2} \leq |p| \sqrt{2} \leq \frac{\lambda}{\sqrt{2}} \sqrt{2} = \lambda. \quad (3.36)$$

The optimization with this relaxation provides the best results of the three considered relaxations, yielding relaxed solutions  $u$  which are binary almost everywhere. This is demonstrated in the comparison Figure 3.3. Here the labeling values are prescribed outside of the gray circle area, while they are missing inside and are to be inpainted so that the interface length between the three regions is minimal. While the three relaxations yield solutions which are nonbinary near the center resulting in a mixture of the colors, the nonbinary region is minimal for the relaxation (3.34). The downside is that it is hard to project onto (3.33) or (3.34), as there is no simple formula. Since the projection must be computed in every iteration of the primal-dual algorithm, this results in a high computational demand for the relaxation.

Let us briefly explain how the constraints (3.33) are derived. Every binary  $u \in \mathcal{D}_0$  is piecewise constant, so that its distributional gradient  $Du$  consists only of the jump part:  $Du = \nu_u \otimes (u^+ - u^-) \mathcal{H}^{m-1} \llcorner S_u$ , respectively for each  $i$ :

$$Du_i = \nu_u(u_i^+ - u_i^-) \mathcal{H}^{m-1} \llcorner S_u. \quad (3.37)$$

Furthermore, the values of  $u$  are constrained to  $\Delta_0$ , so that for points  $x \in S_u$  on the jump set we have  $u^-(x) = e_{l^-(x)}$  and  $u^+(x) = e_{l^+(x)}$  for some indices  $l^-(x), l^+(x) \in \mathcal{L}$ , with the standard basis  $e_1, \dots, e_n$  of  $\mathbb{R}^n$ . Thus, the measure density in (3.37) is nonzero for exactly two  $i$ 's:

$$\nu_u(x)(u_i^+(x) - u_i^-(x)) = \begin{cases} \nu_u(x) & \text{if } i = l^+(x), \\ -\nu_u(x) & \text{if } i = l^-(x), \\ 0 & \text{else.} \end{cases} \quad (3.38)$$

Plugging this into (3.23), this means

$$\bar{E}_{\text{reg}}(u) = \sup_{p \in \mathcal{C}} \int_{S_u} \langle p_{l^+(x)}(x) - p_{l^-(x)}(x), \nu_u(x) \rangle d\mathcal{H}^{m-1}(x). \quad (3.39)$$

In order to fulfill (3.20), this must be at least less than or equal to the original regularizer value (3.7) on  $u$ :

$$\begin{aligned} \int_{S_u} \langle p_{l^+(x)}(x) - p_{l^-(x)}(x), \nu_u(x) \rangle d\mathcal{H}^{m-1}(x) \\ \leq \int_{S_u} d(l^-(x), l^+(x)) d\mathcal{H}^{m-1}(x) = E_{\text{reg}}(u) \end{aligned} \quad (3.40)$$

for every  $p \in \mathcal{C}$ . Demanding the inequality (3.40) pointwise at all  $x \in S_u$  on the jump set then leads to (3.33).

### 3.4.5 Optimality

The relaxed overall problem (3.19) is convex and thus amenable to efficient optimization. Having found a relaxed minimizer  $u^* \in \mathcal{D}$ , the crucial question is how it is related to minimizers of the original problem (3.16).

In the case that the computed solution  $u^*$  is binary, i.e.  $u^* \in \mathcal{D}_0$ , the original problem is solved optimally. This is because on  $\mathcal{D}_0$  the relaxed energy  $\bar{E}$  coincides with the original one due to (3.20). Indeed, then  $E(u^*) = \bar{E}(u^*) \leq \bar{E}(u) = E(u)$  for all  $u \in \mathcal{D}_0$ , so that  $u^*$  is optimal for  $E$ .

**A-posteriori Energy Bounds.** However, it might happen that  $u^* \in \mathcal{D}$  is actually nonbinary at some points  $x \in \Omega$ . Then one needs to apply a suitable *binarization scheme* to find an approximate solution  $u_{\text{bin}} \in \mathcal{D}_0$ , projecting the values  $u^*(x) \in \Delta$  back to  $\Delta_0$  at each  $x$ . A common approach is to take the label with the largest indicator function value at each pixel:

$$u_{\text{bin},i}(x) = \chi_{i=i(x)} \quad \text{with} \quad i(x) = \underset{i \in \mathcal{L}}{\operatorname{argmax}} u_i(x). \quad (3.41)$$

Independently of the binarization choice one can always assess the quality of the solution candidate  $u_{\text{bin}}$ , by comparing its energy  $E(u_{\text{bin}})$  with the optimal energy  $E(u_{\text{bin}}^*)$  of an unknown actual minimizer  $u_{\text{bin}}^* \in \mathcal{D}_0$  of  $E$ . This can be achieved through the energy bound (1.6):

$$\overline{E}(u^*) \leq E(u_{\text{bin}}^*) \leq E(u_{\text{bin}}), \quad (3.42)$$

where both the left and right hand sides are explicitly computable having obtained  $u^*$  and then  $u_{\text{bin}}$  from  $u^*$ .

**A-priori Energy Bounds.** A disadvantage of (3.42) is its dependency on the solution  $u^*$ . Though these bounds are usually very tight in practice, no optimality guarantees can be given ahead of actually computing a solution. However, for the multilabel problem with length regularization one can also prove *a-priori* bounds, provided that one constructs the solution candidate  $u_{\text{bin}} \in \mathcal{D}_0$  through a special *probabilistic* binarization scheme. This means that the binarization process in order to arrive from  $u_{\text{bin}}$  to  $u_{\text{bin}}^*$  consists of infinitely many consecutive steps, each involving a random decision as to which step to take next. Repetition of this random process may well lead to different binarizations  $u_{\text{bin}}^*$ .

In [78] the following energy bound is shown:

$$E(u_{\text{bin}}) \leq CE(u_{\text{bin}}^*) \quad (3.43)$$

where  $C \geq 1$  depends only on the used regularizer relaxation  $\overline{E}_{\text{reg}}$  and not on the data term values  $c_i$ . For example, for the relaxation in Section 3.4.3 one can prove (3.43) with  $C = 2$  [78]. The bound (3.43) is a *probabilistic* one, just as the binarization process itself. This means that the bound (3.43) is satisfied *almost surely*, i.e. for almost all outcomes  $u_{\text{bin}}^*$  of the probabilistic binarization.

Similar a-priori bounds also exist for the MRF domain formulation of the multilabel problem, i.e. when the multilabel energy is defined on a discretized pixel grid and the regularizer measures the length in the ‘‘Manhattan’’  $l^1$ -metric. Bounds with compatible constants as in [78] can then be proved for a number of discrete optimization approaches, for instance for the graph cuts based  $\alpha$ -expansion [19] and for Linear Programming (LP) relaxation [66].

## 3.5 Implementation

In the following we will give some implementation details for the multilabel problem with length regularity (3.19).

### 3.5.1 Discretization

The continuous image domain  $\Omega$  is discretized into a finite rectangular pixel grid, which we again denote by  $\Omega$ . For the corresponding discretization of the gradient  $D$  we use forward differences with Neumann boundary conditions, writing  $\nabla^+$  for the resulting discretized operator. The divergence  $\text{div}$  is then discretized

such that it is the negative adjoint for  $\nabla^+$ , i.e. using backward differences with Dirichlet conditions  $\text{div}^-$  as in (2.35). The discretized energy (3.19) becomes:

$$\min_{u \in \mathcal{D}_d} \max_{p \in \mathcal{C}_d} \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u_i(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle. \quad (3.44)$$

The set  $\mathcal{D}$  of relaxed indicator functions (3.18) is discretized to

$$\mathcal{D}_d = \left\{ u : \Omega \rightarrow \mathbb{R}^n \mid u_i(x) \in [0, 1] \ \forall x \in \Omega, i \in \mathcal{L}, \sum_{i=1}^n u_i(x) = 1 \ \forall x \in \Omega \right\}. \quad (3.45)$$

The constraint set  $\mathcal{C}$  for the duals (3.24) becomes

$$\mathcal{C}_d = \left\{ p : \Omega \rightarrow \mathbb{R}^{m \times n} \mid p(x) \in \mathcal{C}_{\text{loc}} \ \forall x \in \Omega \right\}. \quad (3.46)$$

The local set  $\mathcal{C}_{\text{loc}} \subset \mathbb{R}^{m \times n}$  depends on the used relaxation for the length regularizer, and is one of (3.25), (3.27), or (3.33).

### 3.5.2 The Primal-Dual Algorithm

We use the preconditioned primal-dual Algorithm 3 to solve the min-max problem (3.44). The functions  $D$  and  $F$  in the general energy form (2.38) are in our case hard constraints together with the linear data term:

$$D(u) = \delta_{\mathcal{D}_d}(u) + \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u_i(x) \quad \text{and} \quad F(p) = \delta_{\mathcal{C}_d}(p). \quad (3.47)$$

The corresponding proximal operators are essentially projections onto the constraint sets:

$$\text{prox}_{\tau, D}(\hat{u}) = \pi_{\mathcal{D}_d}(\hat{u} - \tau c) \quad \text{and} \quad \text{prox}_{\sigma, F}(\hat{p}) = \pi_{\mathcal{C}_d}(\hat{p}). \quad (3.48)$$

For the primal prox this follows due to the linearity of the data term:

$$\begin{aligned} \text{prox}_{\tau, D}(\hat{u}) &= \underset{u \in \mathcal{D}_d}{\text{argmin}} \sum_{x \in \Omega} \sum_{i=1}^n \frac{(u_i(x) - \hat{u}_i(x))^2}{2\tau} + \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u_i(x) \\ &= \underset{u \in \mathcal{D}_d}{\text{argmin}} \sum_{x \in \Omega} \sum_{i=1}^n \frac{\left( u_i(x) - (\hat{u}_i(x) - \tau c_i(x)) \right)^2}{2\tau}. \end{aligned} \quad (3.49)$$

To compute the adaptive time steps (2.49), observe that in the bilinear product of (3.44) the factor corresponding to the dual scalar  $p_i(x)_j$  (the  $j$ -th component of the vector  $p_i(x) \in \mathbb{R}^m$ ) is  $\partial_{x_j}^+ u_i(x)$ , consisting of at most 2 terms with coefficients  $\pm 1$  according to (2.34). Conversely, the factor corresponding to the primal scalar  $u(x)$  is  $-(\text{div}^- p)(x)$ , having at most  $2m$  terms with coefficients  $\pm 1$ . Thus, we can set

$$\tau_u = \frac{\tau_0}{2m} \quad \text{and} \quad \sigma_p = \frac{\sigma_0}{2}. \quad (3.50)$$

### 3.5.3 Projection for $u$

For the primal projection, one needs to project the values  $u(x) = (u_i(x))_{1 \leq i \leq n} \in \mathbb{R}^n$  onto the simplex  $\Delta$  in (3.17), independently for each  $x \in \Omega$ . To this end, one can use the simple and explicit algorithm [84], which takes at most  $n$  steps. The direct projection is preferable for small numbers  $n$  of labels, say  $n \leq 8$  or  $n \leq 16$ . However, it is not parallelizable and requires many memory accesses, so that the linear projection run time becomes suboptimal when employing GPUs for a massively parallel computation of the solution  $u$ .

A different way to enforce the simplex constraint is through *dualization*, which is a better choice for GPU parallelization when the number of labels is high. In the primal constraint set (3.45), we can dualize the sum constraint using the relation (2.71) (i.e. by Lagrange multipliers): Introducing a new dual variable  $\mu(x) \in \mathbb{R}$  at each pixel  $x \in \Omega$  and adding a new energy term according to (2.71), we get the following equivalent formulation of (3.44):

$$\min_{u \in \mathcal{D}_d^0} \max_{p \in \mathcal{C}_d, \mu} \sum_{x \in \Omega} \sum_{i=1}^n \left( c_i(x)u(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle \right) + \sum_{x \in \Omega} \mu(x) \left( \sum_{i=1}^n u_i(x) - 1 \right). \quad (3.51)$$

The constraint set (3.44) for  $u$  simplifies to

$$\mathcal{D}_d^0 = \left\{ u : \Omega \rightarrow \mathbb{R}^n \mid u_i(x) \in [0, 1] \quad \forall x \in \Omega, i \in \mathcal{L} \right\} \quad (3.52)$$

and is straightforward to project onto as all constraints are independent of each other. For the new dual variable  $\mu$  there are no hard constraints. The corresponding function  $F_\mu$  in (2.38) is linear  $F_\mu(\mu) = -\sum_{x \in \Omega} \mu(x)$ , so that the corresponding prox is trivial to compute.

The adaptive primal time step for  $u$ , which is now a bit different due to the altered bilinear term, and the adaptive dual time step for  $\mu$  are

$$\tau_u = \frac{\tau_0}{2m+1} \quad \text{and} \quad \sigma_\mu = \frac{\sigma_0}{n}, \quad (3.53)$$

while the dual time step for  $p$  is still as in (3.50).

### 3.5.4 Projection for $p$

The constraints in (3.46) are pointwise, thus the projection for  $p$  decomposes into many independent ones: For each  $x \in \Omega$  we need to project  $p(x) \in \mathbb{R}^{m \times n}$  onto the local set  $\mathcal{C}_{\text{loc}}$ . The concrete implementation depends on  $\mathcal{C}_{\text{loc}}$ , i.e. the employed relaxation of the length regularization term.

**Relaxation (3.25) of Zach et al.** The relaxation of Section 3.4.2 is the most straightforward and quick to implement. The projection of a  $p \in \mathbb{R}^{m \times n} = (\mathbb{R}^m)^n$  onto (3.25) is computed by clipping the absolute value for each  $i$ :

$$\pi_{\mathcal{C}_{\text{loc}}}(p) = \widehat{p}, \quad \text{with} \quad \widehat{p}_i = \frac{p_i}{\max(1, |p_i|/(\lambda/2))} \quad \forall i. \quad (3.54)$$

**Relaxation (3.27) of Lellmann et al.** This relaxation is also easy to implement, although the projection is slightly more costly since the constraint in (3.27) couples all variables  $p_1, \dots, p_n$ :

$$\pi_{\mathcal{C}_{\text{loc}}}(p) = \frac{p}{\max(1, |p|/(\lambda/\sqrt{2}))}. \quad (3.55)$$

**Relaxation (3.33) of Chambolle et al.** Due to the quadratically many coupling constraints, there is no simple explicit projection formula for the set (3.33). Initially, [29] proposed to use Dijkstra's iterative algorithm. However, it gives only an approximation of the projection, and an acceptable accuracy is achieved only after a rather high number of iterations (commonly 100 or higher). In [81] the authors showed that such an approximate projection actually alters the initial multilabel problem, and proposed to enforce the constraints exactly by using additional variables in a Douglas-Rachford splitting approach.

Here we present a different approach to exactly enforce the constraints by using the *dualization* techniques of Section 2.3.6. The idea is to write them equivalently in the form of additional energy terms through the dualization (2.75). Thus, we introduce new primal variables  $a_{ij}(x) \in \mathbb{R}^m$  for each  $x \in \Omega$  and each pair  $i, j$  with  $1 \leq i < j \leq n$ , and write the hard constraints equivalently by adding the terms

$$\inf_{a_{ij}: \Omega \rightarrow \mathbb{R}^m} \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle + d(i, j) |a_{ij}(x)| \quad (3.56)$$

to the energy (3.44), or respectively (3.51). Overall, the energy becomes

$$\begin{aligned} \min_{u \in \mathcal{D}_a^0, a} \max_{p, \mu} & \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle \\ & + \sum_{x \in \Omega} \mu(x) \left( \sum_{i=1}^n u_i(x) - 1 \right) \\ & + \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle + d(i, j) |a_{ij}(x)|. \end{aligned} \quad (3.57)$$

The second term with the dual  $\mu$  is only to be included if one uses the dualization approach (3.51) for the simplex constraint on  $u$ , otherwise this term and  $\mu$  are omitted from the optimization and  $u$  is optimized over the original set (3.45).

There are now no constraints on the dual  $p$ , i.e.  $p(x) \in \mathbb{R}^{m \times n}$  is allowed to vary freely for each  $x \in \Omega$ , and the corresponding proximal operator can be omitted. The prox for  $a$  decomposes into independent prox operators for each  $a_{ij}(x)$ , which are given by a soft-thresholding operation:

$$\text{prox}_{\tau_a, d(i,j)|\cdot|}(a) = a \max(0, 1 - \tau d(i, j)/|a|). \quad (3.58)$$

The adaptive time steps for the primal  $a$  and the dual  $p$ , which for  $p$  are now different from (3.50), are

$$\tau_a = \frac{\tau_0}{2} \quad \text{and} \quad \sigma_p = \frac{\sigma_0}{n+1}. \quad (3.59)$$

Since there are quadratically many primals  $a_{ij}$ , it is better to use the preconditioned primal-dual Algorithm 4 with “bar”-copies for the duals.

Overall, we see that the tight relaxation (3.33) leads to a quadratic complexity in terms of memory, and also of computation time. This is also the case for the special case (3.34). Recently, [143, 141] proposed an alternative way to solve the Potts model with this tight relaxation. Essentially, the idea is to use the lightweight constraints of Section 3.4.2 everywhere, and switch to the pairwise ones (3.34) only in those pixels in which the labeling potentially has a triple junction. This is done by solving a sequence of multilabel problems, checking after each subproblem which pixels need to be switched to the pairwise relaxation. The advantage of this method is that it typically requires less memory and run time than the full dualization described above. However, there is no guarantee as to how many pixels will need to be changed to the more costly relaxation, and also how many overall steps are needed.





## Chapter 4

# Multilabel Segmentation with Nonmetric Priors

In this chapter we will consider the first of the three multilabel priors, namely the extension of the length regularization to general nonmetric distance functions. This chapter is based on joint work with Claudia Nieuwenhuis and Daniel Cremers [125].

### 4.1 Introduction

#### 4.1.1 Nonmetric Priors in Image Segmentation

In image segmentation the task is to divide the image into a set of non-overlapping regions, which are homogeneous in a specific way. Respective algorithms usually define color models for each object or region, e.g. by estimating probability distributions in the color space. Adding *prior* constraints such as minimal boundary or curvature of the segmented objects, impressive results can be obtained. Yet, for a large number of objects additional information is indispensable to resolve the ambiguity between objects of similar or mixed colors. For example, cows, sheep and horses all contain the colors white, black and brown. Depending on the color distributions, this often leads to mixed animals, having e.g. 'cow' labeled bodies and 'sheep' labeled heads. To avoid such problems, label transitions can be punished depending on the probability of co-occurrence of two different objects next to each other. This, however, in general leads to *nonmetric* arbitrary label distance functions, which cannot be handled by common algorithms such as  $\alpha$ -expansion [19] or primal dual schemes [29, 79], which assume metrical distance functions.

Nonmetric priors are encountered in many practical problems, since the triangle inequality is usually not preserved and infinite distances appear between entirely unrelated objects. For example, while sheep and tigers are both frequently encountered in grass, nevertheless they usually do not appear next to each other. A challenge which has largely been neglected is to devise algorithms which allow to impose such nonmetric distance priors in multilabel optimization to be able to apply any learned co-occurrence relations.

### 4.1.2 Related Work

For an overview of common discrete and continuous approaches for the special case of metric distance functions we refer to the general discussion in Section 3.2.

As for the general nonmetric case, in [76] it was shown how metric approximations to nonmetric distance functions can be obtained. In practice, however, these approximations can be arbitrarily far from the original distance function, e.g. in the case of learned distance functions. Guaranteed integrality gaps between the relaxed and the integer solutions for different approximations of the labeling problem have been proven by Chekuri et al. [36].

In the specific field of multilabel segmentation, nonmetric distance functions have been introduced before. Geodesic distances have been formulated by Bai et al. [8]. Co-occurrence probabilities, which penalize the simultaneous appearance of label sets within an image and thus implicitly influence neighboring labels, have been modeled by Ladicky et al. [75].

**Discrete Nonmetric Labeling.** Most closely related to our approach is the approach by Chekuri et al. [36] for handling general nonmetric distance functions in the MRF domain. It is able to handle arbitrary *label distance functions*

$$d : \{1, \dots, n\}^2 \rightarrow \mathbb{R} \quad (4.1)$$

where  $n \geq 1$  is the number of labels. In particular, neither symmetry nor the triangle inequality are assumed on the label distances. The distance  $d(i, j)$  gives the penalization if the multilabel assignment changes from label  $i$  to label  $j$ .

In the discrete case of MRF energies defined on a graph  $(V, E)$ , it gives an LP-relaxation for the general potentials of second order. The regularizer is defined separately on each edge of the graph and gives a penalization if the labeling  $u : V \rightarrow \mathcal{L} := \{1, \dots, n\}$  is different on the two endpoints of the edge:

$$R(u) = \sum_{(a,b) \in E} d(u(a), u(b)). \quad (4.2)$$

For each label  $i$ , a label indicator function  $u_i(a) \in \{0, 1\}$  is introduced, with  $u_i(a) = 1$  if the label  $i$  is set in pixel  $a$  and  $u_i(a) = 0$  otherwise. The regularizer part of the LP-relaxation in [36] is

$$R(u) = \inf_{u_{ij} \geq 0} \sum_{(a,b) \in E} \sum_{i,j \in \mathcal{L}} d(i, j) u_{ij}(a, b) \quad (4.3)$$

where the new variables  $u_{ij}(a, b)$  are constrained by

$$\begin{aligned} \sum_j u_{ij}(a, b) &= u_i(a) \quad \forall i \in \mathcal{L}, \\ \sum_i u_{ij}(a, b) &= u_j(b) \quad \forall j \in \mathcal{L}. \end{aligned} \quad (4.4)$$

However, because of the MRF domain the regularizer (4.3) is defined directly on the underlying pixel grid. Therefore, the interface length between two label

regions is measured in the  $l^1$ - instead of the  $l^2$ -norm leading to grid bias, i.e. grid aligned interfaces between labels are favored.

In contrast, in the continuous setting, variational multilabel approaches have so far been limited to *metric* distance measures, leading to crude approximations of the actual label distances.

### 4.1.3 Contributions

In this chapter, we present a novel spatially continuous approach to the multilabel problem, which allows for arbitrary label distances. We formulate an efficient primal-dual algorithm and compare results to previous approaches, which are restricted to metric label distances, on the MSRC segmentation benchmark.

More specifically, our contributions are as follows:

- We propose a novel regularizer for multilabel optimization which can handle arbitrary explicitly specified label distances.
- The regularizer is spatially continuous and therefore is rotationally invariant, avoiding the grid bias.
- The model is easy to implement and the results are comparable and often superior to the metric and grid based approaches.

## 4.2 Continuous Multilabel Optimization with Non-metric Priors

We consider the multilabel problem (3.16) with  $n \geq 1$  labels and with the general length regularizer in (3.7). It measures the total length of all label interfaces weighted by the distances  $d(i, j)$  of the corresponding labels  $i$  and  $j$  at the two interface sides. As discussed in Section 3.3.2, the original multilabel problem over binary-only labelings from the set  $\mathcal{D}_0$  in (3.15) is nonconvex and therefore very hard to optimize. For efficient optimization we need to revert to *convex relaxation*, replacing the binary constraints  $u_i(x) \in \{0, 1\}$  with  $u_i(x) \in [0, 1]$  and arriving at the relaxed constraint set  $\mathcal{D}$  in (3.18) for  $u$ .

Thus, our goal is to solve the relaxed multilabel problem (3.19), i.e.

$$\min_{u \in \mathcal{D}} \sum_{i=1}^n \int_{\Omega} c_i(x) u_i(x) dx + R(u), \quad (4.5)$$

with a suitable convex regularizer  $R$  which acts as a convexification of the original length regularizer  $E_{\text{reg}}$  in (3.7). Ideally, we want to find a  $R$  which can handle arbitrary nonmetric distance functions  $d$ , i.e. so that it coincides with  $E_{\text{reg}}$  on binary labelings  $u$  according to (3.20). In the following, we propose a definition of such a relaxation and show in Theorem 4.1 that it indeed satisfies this requirement.

### 4.2.1 The Novel Regularizer

We propose the following regularizer:

$$R(u) = \sup_{(p,q) \in \mathcal{C}} \sum_{i=1}^n \int_{\Omega} \langle p_i, dD u_i \rangle + \int_{\Omega} q_i u_i dx \quad (4.6)$$

with the convex set

$$\mathcal{C} = \left\{ (p, q) : \Omega \rightarrow (\mathbb{R}^m)^n \times \mathbb{R}^n \mid |p_j(x) - p_i(x)| + q_i(x) \leq d(i, j) \quad \forall x, i, j \right\}. \quad (4.7)$$

The dual variables consist of  $n$  vector fields  $p_1, \dots, p_n$  and  $n$  scalar fields  $q_1, \dots, q_n$ , corresponding to the  $n$  label indicator functions  $u_1, \dots, u_n$ . The distance  $d$  may be *arbitrary*. In particular we do not require it to be a metric. As usual, the distance measure notion implies symmetry  $d(i, j) = d(j, i)$ , nonnegativity  $d(i, j) \geq 0$ , and reflexivity  $d(i, i) = 0$ , for all  $i, j$ . These are the only conditions we impose on  $d$ .

### 4.2.2 Motivation for the Definition

Introducing Lagrange multipliers  $v_i(a, b), w_j(a, b) \in \mathbb{R}$  for the constraints (4.4), the expression (4.3) can be written as

$$\begin{aligned} R(u) = \inf_{u_{ij} \geq 0} & \sum_{(a,b) \in E} \sum_{i,j \in \mathcal{L}} d(i, j) u_{ij}(a, b) \\ & + \sup_v \sum_{i \in \mathcal{L}} v_i(a, b) \left( u_i(a) - \sum_j u_{ij}(a, b) \right) \\ & + \sup_w \sum_{j \in \mathcal{L}} w_j(a, b) \left( u_j(a) - \sum_i u_{ij}(a, b) \right). \end{aligned} \quad (4.8)$$

Evaluating the infimum over  $u_{ij}$ , we get

$$R(u) = \sup_{v,w} \sum_{(a,b) \in E} \left( \sum_i v_i(a, b) u_i(a) + \sum_j w_j(a, b) u_j(b) \right) \quad (4.9)$$

with  $v, w$  such that

$$v_i(a, b) + w_j(a, b) \leq d(i, j) \quad \forall i, j. \quad (4.10)$$

Writing  $\hat{p}_i(a, b) := w_i(a, b)$  and  $\hat{q}_i(a, b) := v_i(a, b) + w_i(a, b)$ , so that  $v_i(a, b) = \hat{q}_i(a, b) - \hat{p}_i(a, b)$  and  $w_j(a, b) = \hat{p}_j(a, b)$ , (4.9) becomes

$$R(u) = \sup_{\hat{p}, \hat{q}} \sum_{(a,b) \in E} \sum_i \left( \hat{p}_i(a, b) (u_i(b) - u_i(a)) + \hat{q}_i(a, b) u_i(a) \right) \quad (4.11)$$

with the constraints

$$(\hat{p}_j(a, b) - \hat{p}_i(a, b)) + \hat{q}_i(a, b) \leq d(i, j) \quad \forall i, j. \quad (4.12)$$

This expression is already very similar to the proposed one (4.6). We can replace the sum over the edges in (4.11) by the sum over each vertex  $a \in V$ , considering its right and upper neighbors  $b_h$  and  $b_v$ , respectively:

$$R(u) = \sup_{\widehat{p}, \widehat{q}} \sum_{a \in V} \sum_i \left( \begin{aligned} & \left( \widehat{p}_i(a, b_h) \right) \left( u_i(b_h) - u_i(a) \right) \\ & \left( \widehat{p}_i(a, b_v) \right) \left( u_i(b_v) - u_i(a) \right) \\ & + \left( \widehat{q}_i(a, b_h) + \widehat{q}_i(a, b_v) \right) u_i(a) \end{aligned} \right). \quad (4.13)$$

Note that the vectorial expression in  $u_i$  can be regarded as the discretization of the gradient  $Du_i$  at  $a \in V$ . The crucial step to arrive at (4.6) is now to define

$$p_i(a) := \begin{pmatrix} \widehat{p}_i(a, b_h) \\ \widehat{p}_i(a, b_v) \end{pmatrix} \quad \text{and} \quad q_i(a) := \widehat{q}_i(a, b_h) + \widehat{q}_i(a, b_v) \quad (4.14)$$

and to replace the two constraints (4.12) for  $b = b_h$  and  $b = b_v$  by one:

$$|p_j(a) - p_i(a)| + q_i(a) \leq d(i, j) \quad \forall i, j. \quad (4.15)$$

### 4.2.3 Properties of the Regularizer

We prove the following main theorem of this chapter. It shows that while the constraints (4.15) are not equivalent to (4.12) they give rise to a rotationally invariant regularizer having the desired penalization properties.

**Theorem 4.1.** *Let  $u = e_i \chi_A + e_j \chi_{\bar{A}}$  with a subset  $A \subset \Omega$  and some fixed  $i, j \in \{1, \dots, n\}$ . Then*

$$R(u) = d(i, j) \text{Per}(A; \Omega) \quad (4.16)$$

where  $\text{Per}(A; \Omega) = TV(\chi_A)$  is the perimeter of  $A$  in  $\Omega$ .

*Proof.* See appendix Section 4.6. □

In other words, a labeling change from label  $i$  to label  $j$  will be penalized by the label distance  $d(i, j)$  weighted by the length of the interface between the regions where these labels are attained. Furthermore, the regularizer  $R(u)$  has the favorable property of being convex, rendering global optimization possible.

**Proposition 4.2.**  *$R(u)$  is convex.*

*Proof.* The inequality for the convexity property can be easily checked directly similar to (2.7). □

### 4.2.4 Special Case: Regularizer for Metrics

For the special case  $q_i(x) \equiv 0$ , the proposed regularizer reduces to the known relaxation (3.33) for *metric* distances  $d$ , namely

$$R_{\text{metric}}(u) = \sup_{p \in \mathcal{C}_{\text{metric}}} \sum_{i=1}^n \int_{\Omega} \langle p_i, dDu_i \rangle \quad (4.17)$$

with the convex set

$$\mathcal{C}_{\text{metric}} = \left\{ p : \Omega \rightarrow (\mathbb{R}^m)^n \mid |p_j(x) - p_i(x)| \leq d(i, j) \quad \forall x, i, j \right\}. \quad (4.18)$$

Note that  $R_{\text{metric}}$  is still applicable also for nonmetric distances  $d$ . In that case, due to the definition of the set  $\mathcal{C}_{\text{metric}}$  it will implicitly work with a “truncated” version of  $d$ , namely the metric

$$\widehat{d}(i, j) := \sup_{p \in \mathbb{R}^n: |p_l - p_k| \leq d(k, l) \forall k, l} |p_j - p_i|. \quad (4.19)$$

Lellmann et al. [79] proposed a regularizer in the continuous domain for metric distances  $d$  only, and showed in Proposition 3.1 of [79] that  $d$  must necessarily be a metric if the regularizer  $R$  satisfies certain simple conditions. Since our proposed regularizer (4.6) can handle arbitrary nonmetric distances by Theorem 4.1, it is interesting to see which of these conditions are *not* satisfied in our case. It turns out to be only the second condition (P2) which states that the regularizer must be zero for any constant  $u : \Omega \rightarrow \mathbb{R}^n$ . In fact, we have the following result:

**Proposition 4.3.** *Let  $u : \Omega \rightarrow \mathbb{R}^n$  be constant,  $u_i(x) = z_i$  for all  $i$  and  $x \in \Omega$  with some  $z \in \mathbb{R}^n$ . Then  $R(u) = 0$  if  $z_i \geq 0$  for all  $i$ , and  $R(u) = \infty$  otherwise.*

*Proof.* See appendix Section 4.6. □

Because in the optimization we have  $u_i(x) \in \{0, 1\}$  (respectively  $u_i(x) \in [0, 1]$  after relaxation) the proposed regularizer *is* zero for any constant  $u$  which represents a valid labeling. Thus, dropping the condition (P2) of [79] for non-meaningful labelings allows us to handle arbitrary distances, and not only the metric ones.

Recently, [143, 141], which appeared at around the same time as our original work [125] on which this chapter is based, introduced another kind of convex relaxation based on a “discrete-continuous” viewpoint of the image domain. As this relaxation is rather involved thus not allowing an immediate intuitive interpretation, it would be interesting to investigate the relation to our relaxation and possible advantages and drawbacks as future work.

### 4.3 Implementation

After image domain discretization, the overall optimization problem (4.5) with (4.6) discretizes as

$$\min_{u \in \mathcal{D}_d} \max_{p \in \mathcal{C}_d} \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u_i(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle + q_i(x) u_i(x) \quad (4.20)$$

with the primal constraint set  $\mathcal{D}_d$  given by (3.45) as already in Chapter 3. For the duals  $p, q$ , the discretized constraint set  $\mathcal{C}_d$  is given by the same expression (4.7):

$$\mathcal{C}_d = \left\{ (p, q) : \Omega \rightarrow (\mathbb{R}^m)^n \times \mathbb{R}^n \mid |p_j(x) - p_i(x)| + q_i(x) \leq d(i, j) \quad \forall x, i, j \right\}. \quad (4.21)$$

We solve the overall optimization problem (4.20) using the general primal-dual Algorithm 4. For the projection on the primal set  $\mathcal{D}_d$  the same strategies apply as described in Section 3.5.3, i.e. either using a direct projection formula or using Lagrange multipliers for the sum constraint. In this chapter we use the latter variant.

For the dual constraint set  $\mathcal{C}_d$ , we utilize a similar dualization approach as described for the metric case (3.56). The individual constraints for each pair  $i, j$  are first rewritten as

$$|p_j(x) - p_i(x)| \leq d(i, j) - q_i(x) \quad (4.22)$$

and then dualized using the relation (2.76) in the form of additional energy terms:

$$\inf_{(a,b) \in A} \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \left( \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle - b_{ij}(x)q_i(x) + d(i, j)b_{ij}(x) \right). \quad (4.23)$$

This introduces new primal variables  $a_{ij}(x) \in \mathbb{R}^m$ ,  $b_{ij}(x) \in \mathbb{R}$  for all  $x \in \Omega$  and all pairs  $i < j$ , constrained to the set

$$A = \left\{ (a, b) : \Omega \rightarrow (\mathbb{R}^m \times \mathbb{R})^{n \times n} \mid |a_{ij}(x)| \leq b_{ij}(x) \quad \forall x \in \Omega, 1 \leq i < j \leq n \right\}. \quad (4.24)$$

Overall, the final energy becomes

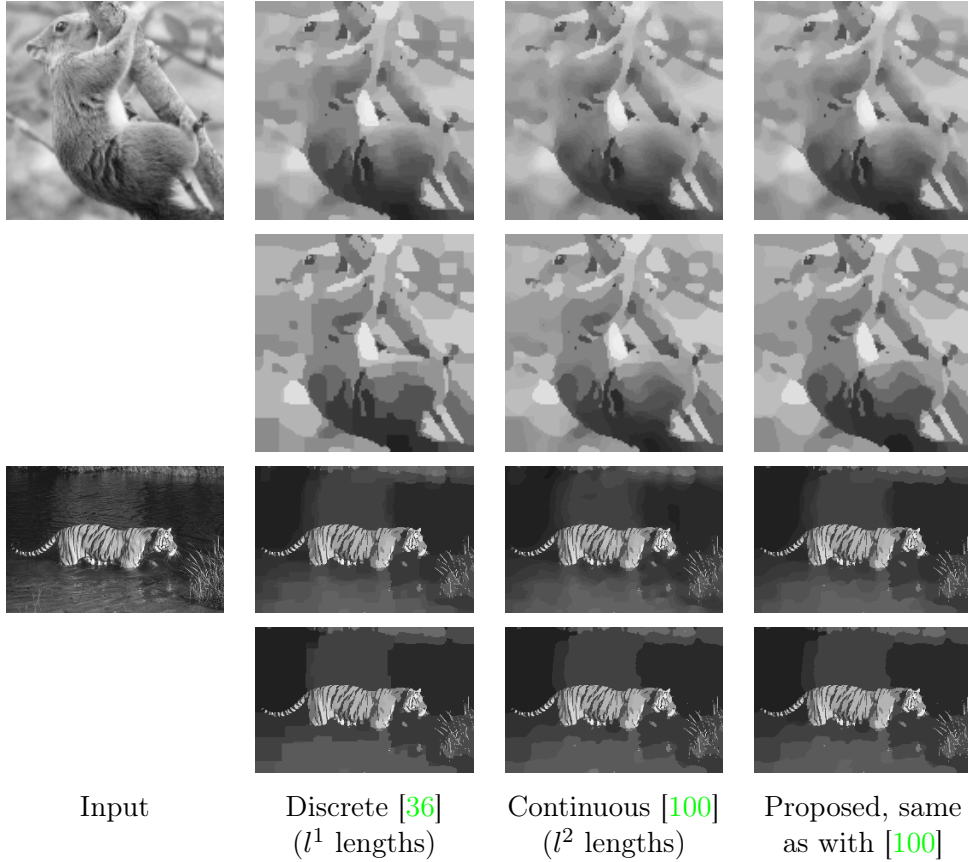
$$\begin{aligned} \min_{u \in \mathcal{D}_d^0} \max_{(a,b) \in A} \max_{(p,q), \mu} & \sum_{x \in \Omega} \sum_{i=1}^n c_i(x)u(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle + q_i(x)u_i(x) \\ & + \sum_{x \in \Omega} \mu(x) \left( \sum_{i=1}^n u_i(x) - 1 \right) \\ & + \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \left( \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle \right. \\ & \quad \left. - b_{ij}(x)q_i(x) + d(i, j)b_{ij}(x) \right) \end{aligned} \quad (4.25)$$

with the reduced primal constraint set  $\mathcal{D}_d^0$  in (3.52) and the set  $A$  in (4.24). The duals  $p, q$  now do not have any constraints. The Lagrange multipliers  $\mu(x) \in \mathbb{R}$  are also unconstrained. The proximal operator for the additional primals  $a, b$  is essentially a projection onto  $A$ , which decomposes into simple pointwise projections.

To implement the discrete model [36] as described in Section 4.1.2 we used the same general primal-dual algorithm, based on the formulation (4.8) of the regularizer.

## 4.4 Experimental Results

We used 5000 iterations for each experiment after which the solutions become visually stable. We used a parallel CUDA implementation on NVIDIA GTX 480.



**Figure 4.1: Piecewise smooth approximations.** Although [36] can handle the nonmetric label distance function induced by the Mumford-Shah functional, it favors grid aligned edges producing block artifacts. In contrast, the proposed approach is based in the continuous setting and produces results visually indistinguishable from [100]. While the nonmetric is implicit in [100], we can specify any nonmetric in a direct way. *First and third rows:* A piecewise smooth approximation. *Second and fourth rows:* A piecewise constant approximation.

Usual run times for  $320 \times 240$  images and 21 labels are around 70 seconds. We observed that the computed relaxed solutions  $u$  are binary almost everywhere except at region boundaries, with more or less sharp transitions. We binarize the result at each pixel  $x$  by taking the label  $i$  with the maximal value  $u_i(x)$ , as in (3.41).

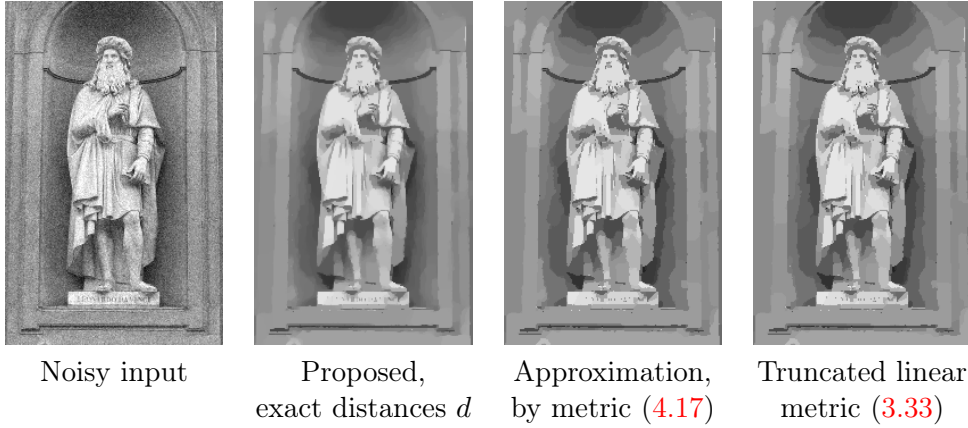
#### 4.4.1 Piecewise Smooth Mumford-Shah Functional

First we demonstrate the application of our approach on the celebrated Mumford-Shah functional. In the continuous domain it is given by

$$E(u, K) = \int_{\Omega} (u - f)^2 dx + \alpha \int_{\Omega \setminus K} |\nabla u|^2 dx + \nu \mathcal{H}^{m-1}(K). \quad (4.26)$$

Given a possibly noisy input image  $f : \Omega \rightarrow \mathbb{R}$ , this yields *piecewise smooth* approximations  $u : \Omega \rightarrow \mathbb{R}$  of  $f$ . The function  $u$  will be smooth, except possibly





**Figure 4.2: Nonmetric versus metric approaches.** *From left to right:* A noisy input image. Piecewise smooth approximation using our approach by specifying the nonmetric Mumford-Shah label distances. Ignoring the nonmetric character of the distance function and treating it as a metric by (4.17) effectively imposes a “truncated linear” metric, which leads to staircasing effects. This is further confirmed by solving for this metric explicitly, plugging (4.29) into (3.33).

for a one-dimensional edge set  $K$  where jumps occur. The parameter  $\nu$  controls the length of the jump set  $K$ . Bigger values of  $\nu$  lead to a smaller jump set, i.e. the solution will be smooth on wider subregions of  $\Omega$ . We will discuss this functional in detail and in the general setting of a vectorial  $u$  in Chapter 10.

Discretizing the range  $[0, 1]$  of  $u$  into  $n$  levels, this leads to a multilabel problem with the well-known *truncated quadratic* label distances:

$$d_{MS}(i, j) = \min\left(\nu, \alpha\left(\frac{i-j}{n}\right)^2\right) \quad \forall 1 \leq i, j \leq n. \quad (4.27)$$

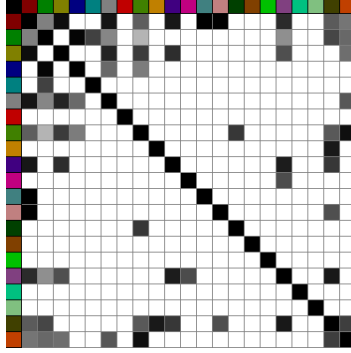
This distance function is not a metric when  $\nu > \frac{2\alpha}{n^2}$ . For example,

$$d_{MS}(0, 2) = \min\left(\nu, \frac{4\alpha}{n^2}\right) > 2 \min\left(\nu, \frac{\alpha}{n^2}\right) = d_{MS}(0, 1) + d_{MS}(1, 2). \quad (4.28)$$

Therefore, our approach applies naturally here. Some results for different parameters  $\nu$  and  $\alpha$  are shown in Figure 4.1. As expected, our approach produces visually the same results as Pock et al. [100] (which handles exclusively the Mumford-Shah model) since both work in the continuous setting. While [100] uses advanced tools such as functional lifting to arrive at the convex relaxation, the proposed approach is more basic as it specifies the label distances explicitly. The discrete grid based approach [36] can also handle the distances (4.27) but measures the interface length in the  $l^1$ -norm. As seen in Figure 4.1, our model evidently visually improves over [36], eliminating its block artifacts.

The experiment in Figure 4.2 shows the importance of the ability of the proposed approach to handle nonmetric distances exactly. Trying to solve the Mumford-Shah problem using the metric approximation (4.17) effectively imposes the *truncated linear* prior instead of the truncated quadratic one, which leads to staircasing effects. In fact, one can easily show that the truncated metric in (4.19) is given by

$$\widehat{d}_{MS}(i, j) = \min\left(\nu, \alpha \frac{|i-j|}{n^2}\right). \quad (4.29)$$



**Figure 4.3: Label distance matrix for the MSRC database in Section 4.4.2.** The distance function is estimated on the training set and is an example of a *nonmetric*. The first row and column are the color coded labels and the other entries depict the values  $d(i, j)$  with 0 as black and 10 as white.

#### 4.4.2 MSRC Segmentation Benchmark

To evaluate the proposed segmentation algorithm we apply it to the task of object segmentation and recognition on the MSRC benchmark. This benchmark comprises around 600 images which contain 23 different labels such as ‘cow’, ‘book’, ‘building’ or ‘grass’. To conduct experiments on this benchmark, we follow Ladicky et al. [75] and divide the image set randomly into 60% training images and 40% test images.

The label distance matrix is learned on the training set. For each pair of labels  $(i, j)$ ,  $i < j$ , we compute the relative frequency that  $i$  and  $j$  are neighbors in the following way. Let  $A_{ij}$  and  $B_{ij}$  denote the number of pairs of pixels where labels  $i$  and  $j$  are direct or diagonal neighbors, respectively:

$$\begin{aligned} A_{ij} &:= \#\{(x, y) \in \Omega \times \Omega \mid l(x) = i, l(y) = j, |x - y| = 1\}, \\ B_{ij} &:= \#\{(x, y) \in \Omega \times \Omega \mid l(x) = i, l(y) = j, |x - y| = \sqrt{2}\}. \end{aligned} \quad (4.30)$$

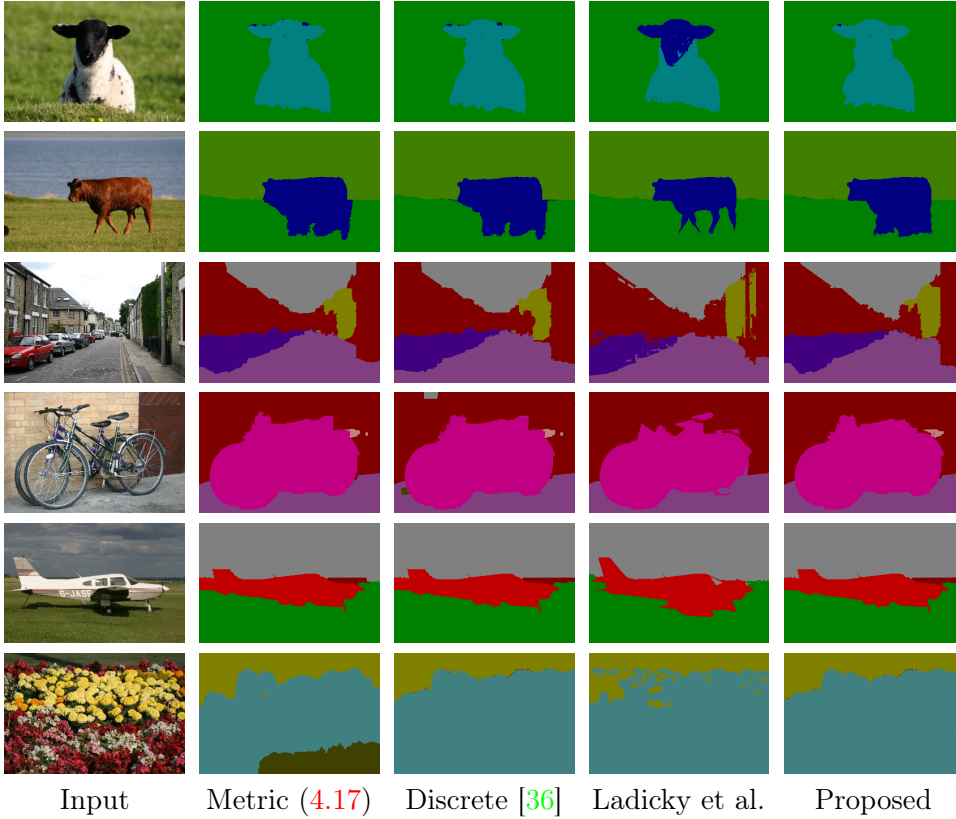
Then the label distance is calculated as the negative logarithm of the relative weighted frequency:

$$d(i, j) = -\log \frac{A_{ij} + \frac{B_{ij}}{\sqrt{2}}}{\sum_k (A_{ik} + \frac{B_{ik}}{\sqrt{2}})}. \quad (4.31)$$

For non-adjacent labels  $i, j$  with  $A_{ij} = B_{ij} = 0$ , we truncate  $d(i, j) = \infty$  to  $d(i, j) = M$  for some  $M > 0$  (we use  $M = 10$ ). Figure 4.3 indicates the distance function obtained from the MSRC training set. The brightness of the entry corresponds to the distance of the coinciding labels.

To evaluate the segmentation accuracy of the proposed method, we compare the scores and the overall accuracies for the following approaches:

- the proposed nonmetric regularization (4.6),
- the “truncation to metric” regularization (4.17),



**Figure 4.4: Results on the MSRC segmentation benchmark.** *From left to right:* Input image, segmentation by (4.17) truncating the label distance to the nearest metric, segmentation using the grid based approach (4.3), segmentation using the proposed regularizer (4.6). The regularizer weighting is optimized separately for each approach.

- the discrete  $l^1$ -nonmetric regularization (4.3),
- the co-occurrence statistics based approach by Ladicky et al. [75].

“Truncation to metric” means that the learned distance function  $d$  is approximated by the closest metrical distance function in the continuous domain, given by (4.19). The  $l^1$  nonmetric regularization penalizes distances in horizontal and vertical direction separately leading to a direction dependent distance function. In the discrete setting, this formulation corresponds to Chekuri et al. [36]. Finally, we compare the obtained benchmark results to those reported by Ladicky et al. [75]. In contrast to the proposed method which defines the distances based on neighboring pixel labels (second order potentials), the authors of [75] use information on the general co-occurrence of two labels in one image to derive label distances (potentials of the highest order  $|\Omega|$ ).

Results for the different approaches are shown in Figure 4.4. A quantitative comparison for each label as well as the overall accuracies can be seen in Table 4.1. We set the regularizer weighting to  $\nu = 0.2$  for every image in the database. The proposed approach leads to best overall accuracies comparing with the metric and grid based approaches.

	Accuracy	Average	Building	Grass	Tree	Cow	Sheep	Sky	Aeroplane	Water	Face	Car	Bicycle	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat
data term	83.99	77.18	67	97	91	85	86	95	88	81	90	82	94	81	62	42	91	66	86	79	54	72	31
metric	84.72	77.54	70	97	91	88	86	96	83	82	90	82	93	83	65	46	92	64	87	81	51	73	31
discrete $l^1$	84.79	<b>77.62</b>	69	97	91	88	86	96	83	83	90	83	93	83	64	46	92	63	86	80	51	73	31
proposed	<b>84.82</b>	<b>77.62</b>	69	97	92	88	86	96	83	83	90	82	93	83	64	46	92	64	86	80	51	73	32
Ladicky et al. [75]	86.76	77.78	76	99	90	77	84	99	82	88	88	80	90	90	71	47	94	68	90	73	55	77	15

**Table 4.1: Segmentation accuracies for the different approaches.** The scores for each label are defined as  $\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$ . Also shown is the average over the scores and the accuracy for each approach, i.e. the overall number of pixels labeled correctly as in the ground truth, divided by the overall number of pixels. The proposed continuous nonmetric approach provides the best overall accuracy. Results of [75] are only for comparison, since they use potentials of the highest order  $|\Omega|$ , instead of order two as in our approach.

## 4.5 Conclusion

For the multilabel optimization problem we introduced a novel regularizer which can handle arbitrary label distances. In contrast to previous discrete approaches, it is based in the continuous setting and does not suffer from metrication artifacts. Being convex it allows to find globally optimal solutions of the relaxation. The proposed model leads to consistently better results than the discrete setting model. Experimental results show competitiveness to state-of-the-art discrete approaches. On the MSRC segmentation database we obtain higher overall accuracy, and the Mumford-Shah experiments evidently show visual improvements over the discrete model, eliminating its block artifacts. Future applications, with a more distinct nonmetric structure of the distance functions based on our approach may lead to substantial improvements.

## 4.6 Appendix: Proofs of Propositions and Theorems

*Proof of Theorem 4.1.* We have, using the divergence theorem,

$$\begin{aligned}
R(u) &= \int_A (-\operatorname{div} p_i + q_i) \, dx + \int_{\bar{A}} (-\operatorname{div} p_j + q_j) \, dx \\
&= \int_{\partial A} (-p_i) \nu_{\partial A} \, d\mathcal{H}^{m-1} + \int_{\partial \bar{A}} (-p_j) \nu_{\partial \bar{A}} \, d\mathcal{H}^{m-1} + \int_A q_i \, dx + \int_{\bar{A}} q_j \, dx \\
&= \int_{\partial A} (p_j - p_i) \nu_{\partial A} \, d\mathcal{H}^{m-1} + \int_A q_i \, dx + \int_{\bar{A}} q_j \, dx \tag{4.32} \\
&\leq \int_{\partial A} (p_j - p_i) \nu_{\partial A} \, d\mathcal{H}^{m-1} + \int_A q_i \, dx.
\end{aligned}$$

For the last inequality we used  $q_j \leq 0$ , which follows from (4.7) by setting  $i = j$ . Using the constraints in (4.7), from this we obtain

$$\begin{aligned} R(u) &\leq \int_{\partial A} (d(i, j) - q_i) \, d\mathcal{H}^{m-1} + \int_A q_i \, dx \\ &= d(i, j) \operatorname{Per}(A; \Omega) + \int_A q_i \, dx - \int_{\partial A} q_i \, d\mathcal{H}^{m-1}. \end{aligned} \tag{4.33}$$

Observe that in the discretized setting it holds

$$\int_{\partial A} q_i \, d\mathcal{H}^{m-1} = \int_{A_1} q_i \, dx \tag{4.34}$$

where  $A_1 := \{x \in A \mid \operatorname{dist}(x, \partial A) \leq 1\}$  are the points in  $A$  near its boundary. Hence, we have

$$\int_A q_i \, dx - \int_{\partial A} q_i \, d\mathcal{H}^{m-1} = \int_{A \setminus A_1} q_i \, dx \leq 0 \tag{4.35}$$

and it follows  $R(u) \leq d(i, j) \operatorname{Per}(A; \Omega)$ . It is also possible to show the equality here, i.e. that the supremum over  $p$  and  $q$  is reached. However, this requires a rather technical argument.  $\square$

*Proposition 4.3.* Since the constant function  $u$  has zero gradient, the representation (4.6) of  $R$  reduces to

$$R(u) = \sup_{(p, q) \in \mathcal{C}} \sum_{i=1}^n z_i \int_{\Omega} q_i(x) \, dx \tag{4.36}$$

with the set  $\mathcal{C}$  in (4.7).

First, assume that  $z_i \geq 0$  for every  $i$ . The constraints in  $\mathcal{C}$  for  $i = j$  specifically yield  $q_i(x) \leq 0$  for all  $i$  and  $x$ . This way we get  $R(u) \leq 0$ . By choosing  $p, q \equiv 0$  we also obtain  $R(u) \geq 0$ , so that overall  $R(u) = 0$ .

Assume now that  $z_{i_0} < 0$  for some  $i_0$ . Choosing  $p \equiv 0$ ,  $q_i(x) := 0$  for  $i \neq i_0$  and  $q_{i_0}(x) := M$  with any  $M \leq 0$ , we obviously have  $(p, q) \in \mathcal{C}$ . Therefore,  $R(u) \geq \sup_{M \leq 0} c_{i_0} M |\Omega| = \infty$ .  $\square$



## Chapter 5

# Multilabel Segmentation with Ordering Constraints

Here we introduce our second multilabel prior, namely the geometric ordering constraints on label jumps for the modeling of general layouts of label regions. This chapter is based on joint work with Daniel Cremers [121].

### 5.1 Introduction

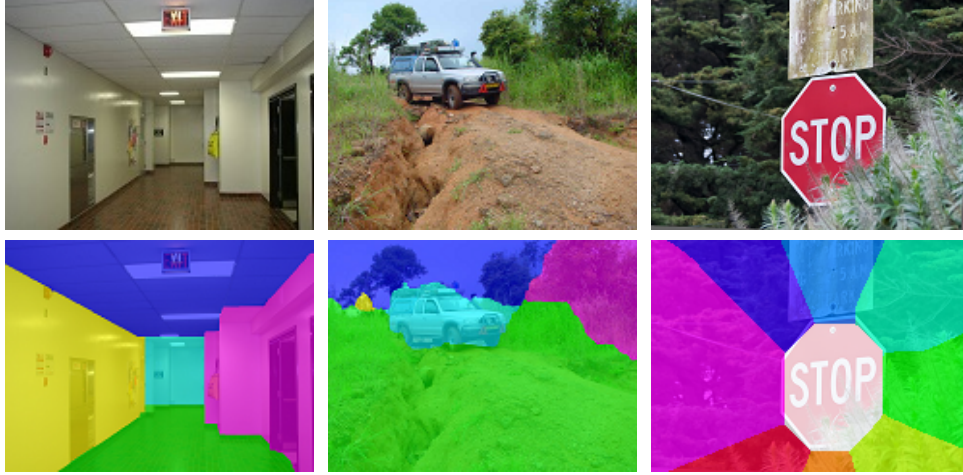
#### 5.1.1 Multilabeling with Ordering Constraints

As discussed in Chapter 3, the first convexification approaches for the multilabel problem focused on the standard length regularization. Its basic property is that it is isotropic, i.e. the local penalization at each fixed point  $x \in \Omega$  is the same regardless of the direction of the jump, or equivalently of the orientation of the jump interface.

A substantial generalization of penalty functions came about with the introduction of ordering constraints into the multilabel optimization. Penalizing label jumps differently depending on the jump *direction* allows to model specific label *layouts*. Liu et al. [82] showed how certain multilabel problems with ordering constraints could be solved using graph cuts. The five regions layout to segment indoor and outdoor images was introduced. Felzenszwalb and Veksler [48] introduced the tiered layout — a generalization of the five regions layout — and showed that it was solvable via dynamic programming. An entirely separate ordering constraint was introduced by the star shape prior of Veksler [131].

#### 5.1.2 Contributions

We propose a novel general framework to incorporate ordering constraints. In contrast to the discrete graph cut or dynamic programming approaches of Liu, Veksler, Felzenszwalb and coworkers, the proposed approach comes from a totally different viewpoint of continuous optimization. We provide an exact characterization of the penalty functions expressible with our approach. In particular, the proposed method exhibits several favorable properties:



**Figure 5.1: Label ordering constraints.** We propose a spatially continuous framework for label ordering constraints which unifies existing approaches such as the five regions layout (*left*) and the tiered layout (*middle*). It generalizes to novel applications such as a convex shape prior (*right*).

- We show in Section 5.5 that the three mentioned layout approaches are special cases of the proposed framework.
- We show that this framework allows applications beyond the above approaches, including tiered layout with four and more tiers, tiered layout with independent floating occlusions and shape priors for arbitrary *convex shapes* — see Figure 5.2.
- In contrast to existing approaches to label ordering constraints the proposed framework naturally extends to three and higher dimensions of the image domain.
- Despite its generality, the model is easily adjusted to various label layouts, it is easily implemented and provides results which are comparable and often superior to those of existing approaches.

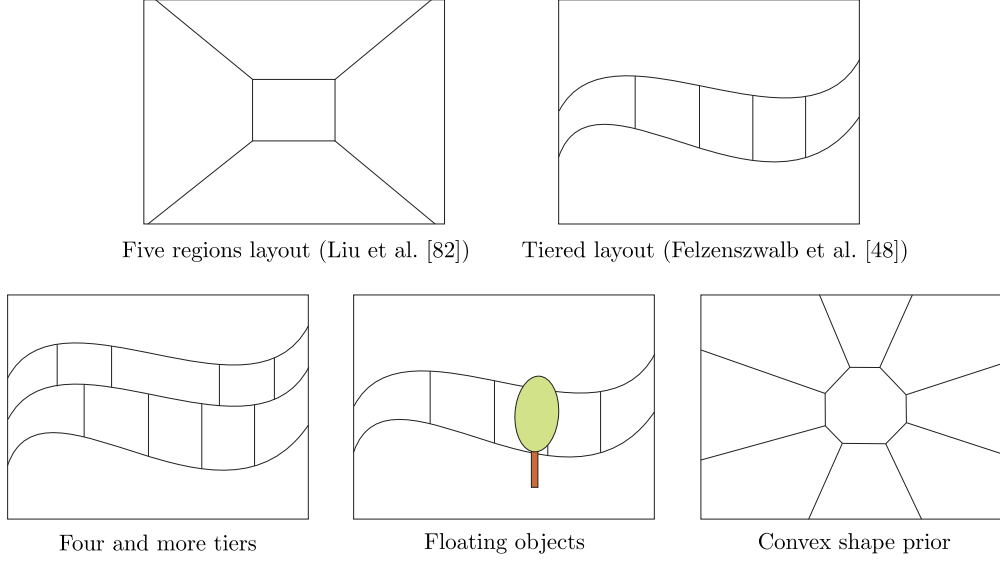
## 5.2 General Framework

We consider solving the multilabel problem (3.19) in image domain  $\Omega \subset \mathbb{R}^m$ ,  $m \geq 1$ , with  $n \geq 1$  labels. The task is to find  $n$  label indicator functions  $u_1, \dots, u_n : \Omega \rightarrow [0, 1]$  minimizing the energy

$$\min_{u \in \mathcal{D}} \sum_{i=1}^n \int_{\Omega} c_i(x) u_i(x) dx + R(u). \quad (5.1)$$

To render efficient optimization possible, we work with the already relaxed constraint set  $\mathcal{D}$  in (3.18). Our goal is to find a regularizer  $R$  which essentially penalizes the length of the interfaces between labels, and also explicitly allows to take the *jump direction* into account.





**Figure 5.2: Schematic summary of our main contributions.**

### 5.2.1 The Novel Regularizer

We propose the following regularizer:

$$R(u) = \sup_{p \in \mathcal{C}} \sum_{i=1}^n \int_{\Omega} \langle p_i, dDu_i \rangle \quad (5.2)$$

with the convex set

$$\mathcal{C} = \left\{ p \in C_c^1(\Omega; \mathbb{R}^{m \times n}) \mid p(x) \in \mathcal{C}_{\text{loc}} \quad \forall x \in \Omega \right\} \quad (5.3)$$

and

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid \langle p_j - p_i, \nu \rangle \leq d(i, j, \nu) \quad \forall 1 \leq i < j \leq n, \nu \in \mathbb{S}^{m-1} \right\}. \quad (5.4)$$

The *label distance function*

$$d : \{1, \dots, n\}^2 \times \mathbb{S}^{m-1} \rightarrow \mathbb{R} \cup \{\infty\} \quad (5.5)$$

gives the penalization if the multilabel assignment  $u$  changes from label  $i$  to label  $j$  *in direction*  $\nu$ . The distance  $d$  and with it the local set  $\mathcal{C}_{\text{loc}}$  may also depend on the position  $x \in \Omega$ , thus enabling different regularizer weights at different image locations. The standard scalar product on  $\mathbb{R}^m$  is denoted by  $\langle \cdot, \cdot \rangle$ , and as usual we use the notation  $p(x) = (p_1(x), \dots, p_n(x)) \in (\mathbb{R}^m)^n = \mathbb{R}^{m \times n}$ . Finally,

$$\mathbb{S}^{m-1} = \{z \in \mathbb{R}^m \mid |z| = 1\} \quad (5.6)$$

denotes the  $(m - 1)$ -sphere.

**Intuitive Explanation.** The derivation (3.40) of the tight convex relaxation for the isotropic regularization in Section 3.4.4 already hints at the origin of proposed constraints (5.4) for our more general case. Instead of repeating this derivation, let us give here a more simple and intuitive motivation which provides a direct insight into the definition of the regularizer (5.2) with the constraint set (5.4). Suppose that at some point  $x_0 \in \Omega$  the labeling changes from label  $i$  to label  $j$  in direction  $\nu$ . Locally, the jump interface is the plane  $\langle x - x_0, \nu \rangle = 0$ . Locally on the  $i$ -side of the interface, i.e. for  $x$  near  $x_0$  with  $\langle x - x_0, \nu \rangle \leq 0$ , only the label  $i$  is set, while on the other side of the interface only the label  $j$  is set. Thus, locally

$$\begin{aligned} u_i(x) &= \begin{cases} 1 & \text{if } \langle x - x_0, \nu \rangle \leq 0, \\ 0 & \text{else,} \end{cases} \\ \text{and } u_j(x) &= \begin{cases} 0 & \text{if } \langle x - x_0, \nu \rangle \leq 0, \\ 1 & \text{else.} \end{cases} \end{aligned} \quad (5.7)$$

All other label indicator functions are zero locally around  $x_0$ . So passing through  $x_0$  in direction  $\nu$ ,  $u_i(x)$  decreases from 1 to 0, and  $u_j$  increases from 0 to 1. Thus,

$$\nabla u_i(x_0) = -\nu \quad \text{and} \quad \nabla u_j(x_0) = \nu \quad (5.8)$$

up to the delta function factor  $\delta(\langle x - x_0, \nu \rangle)$ . The integral (5.2) for a fixed  $p$  is therefore locally equal to

$$\langle p_i(x_0), \nabla u_i(x_0) \rangle + \langle p_j(x_0), \nabla u_j(x_0) \rangle = \langle p_j(x_0) - p_i(x_0), \nu \rangle, \quad (5.9)$$

times the local boundary measure  $|\partial B_{\text{local}}|$  of the jump interface since the delta function factor concentrates the integral on the plane  $\langle x - x_0, \nu \rangle = 0$ . We want this local contribution to be

$$R_{\text{local}}(u) = d(i, j, \nu) |\partial B_{\text{local}}|. \quad (5.10)$$

Therefore, in order to have at least “ $\leq$ ” here we assume the constraints on  $p$  in (5.4), and to obtain equality we take the supremum over all such  $p$ .

**Main Requirement for  $d$ .** This intuitive argument reveals that for the desired correct penalization with  $d(i, j, \nu)$ , when labels jump from  $i$  to  $j$  in direction  $\nu$ , we need the equality

$$\sup_{p \in \mathcal{C}_{\text{loc}}} \langle p_j - p_i, \nu \rangle = d(i, j, \nu). \quad (5.11)$$

From the definition (5.4) of the constraint set  $\mathcal{C}_{\text{loc}}$ , we can immediately conclude that at least “ $\leq$ ” holds. The question is now, what conditions must be imposed on  $d$  to assure (5.11).

From the representation (5.11) a number of necessary conditions on  $d$  follows. In the following we will first derive such conditions, building up on [5, Prop. 5.19], and afterwards show that they are also sufficient.

### 5.2.2 Assumptions on the Distance Function $d$

For a more convenient formulation of the constraints, in the following we extend  $d(i, j, \cdot)$  positive homogeneously from  $\mathbb{S}^{m-1}$  to whole  $\mathbb{R}^m$ , i.e. for every  $i, j \in \mathcal{L} = \{1, \dots, n\}$  we set

$$d(i, j, t\nu) := td(i, j, \nu) \quad \forall t \geq 0, \nu \in \mathbb{S}^{m-1}. \quad (5.12)$$

The representation (5.11) then extends to all  $\xi \in \mathbb{R}^m$  in place of  $\nu$ . The following is an obvious consequence of (5.11):

**Assumption 5.1** (Lower-semicontinuity). We assume that  $d(i, j, \cdot)$  is lower-semicontinuous for every fixed  $i, j$ .

*Proof of Necessity.* This necessary condition follows directly from (5.11) since the left hand side is lower-semicontinuous in  $\nu$ , which is a basic property of support functionals [6, Theorem 9.1.2]. Another way to see this is by observing that the left hand side of (5.11) is the convex conjugate of  $\delta_C$  and recalling that the dual function is always lower-semicontinuous.  $\square$

To formulate the next assumption, recall that for vectors  $\xi \in \mathbb{R}^m$  and  $z \in \mathbb{R}^n$ , the Kronecker product  $\xi \otimes z \in \mathbb{R}^{m \times n}$  is defined similarly to (2.22) as

$$\xi \otimes z = \xi z^T = (z_1 \xi \quad \dots \quad z_n \xi). \quad (5.13)$$

We will consider matrices of the form

$$\xi \otimes (e_j - e_i) \quad (5.14)$$

where  $e_1, \dots, e_n \in \mathbb{R}^n$  is the standard basis of  $\mathbb{R}^n$ . That is,  $\xi \otimes (e_j - e_i)$  has only two non-zero columns, the  $i$ -th and the  $j$ -th, being respectively  $-\xi$  and  $+\xi$ .

**Assumption 5.2** (Generalized Triangle Inequality). We assume that  $d$  is such that

$$d(i, j, \xi) \leq \sum_{\kappa=1}^K d(i_\kappa, j_\kappa, \xi_\kappa) \quad (5.15)$$

whenever

$$\xi \otimes (e_j - e_i) = \sum_{\kappa=1}^K \xi_\kappa \otimes (e_{j_\kappa} - e_{i_\kappa}) \quad (5.16)$$

for vectors  $\xi, \xi_1, \dots, \xi_K \in \mathbb{R}^m$  and label pairs  $(i, j), (i_1, j_1), \dots, (i_K, j_K) \in \mathcal{L}^2$  with a  $K \geq 1$ .

*Proof of Necessity.* This is a necessary consequence if the equality in (5.11) is to hold. To see this, note that we can write

$$\langle p_j - p_i, \xi \rangle = \langle p, \xi \otimes (e_j - e_i) \rangle \quad (5.17)$$

with the standard scalar product  $\langle \cdot, \cdot \rangle$  on  $\mathbb{R}^{m \times n}$ . Therefore, if the equality (5.11) holds, with (5.16) we get

$$\begin{aligned} d(i, j, \xi) &= \sup_{p \in \mathcal{C}_{\text{loc}}} \langle p, \xi \otimes (e_j - e_i) \rangle = \sup_{p \in \mathcal{C}_{\text{loc}}} \left\langle p, \sum_{\kappa=1}^K \xi_{\kappa} \otimes (e_{j_{\kappa}} - e_{i_{\kappa}}) \right\rangle \quad (5.18) \\ &= \sup_{p \in \mathcal{C}_{\text{loc}}} \sum_{\kappa=1}^K \langle p, \xi_{\kappa} \otimes (e_{j_{\kappa}} - e_{i_{\kappa}}) \rangle \leq \sum_{\kappa=1}^K \sup_{p \in \mathcal{C}} \langle p, \xi_{\kappa} \otimes (e_{j_{\kappa}} - e_{i_{\kappa}}) \rangle \\ &= \sum_{\kappa=1}^K d(i_{\kappa}, j_{\kappa}, \xi_{\kappa}). \quad \square \end{aligned}$$

The assumption (5.2) is a kind of generalized *triangle inequality* simultaneously in the labels and the direction. For illustration, let us mention two important special cases of (5.15), corresponding to special decompositions (5.16)<sup>1</sup>.

### 5.2.3 Corollary Assumption for Direction Dependency

Fixing labels  $i, j \in \mathcal{L}$ , for any vectors  $\xi_1, \xi_2 \in \mathbb{R}^m$  we have

$$(\xi_1 + \xi_2) \otimes (e_j - e_i) = \xi_1 \otimes (e_j - e_i) + \xi_2 \otimes (e_j - e_i). \quad (5.19)$$

By (5.15) this means that  $d$  must satisfy

$$d(i, j, \xi_1 + \xi_2) \leq d(i, j, \xi_1) + d(i, j, \xi_2). \quad (5.20)$$

This is the *triangle inequality* for direction dependency for fixed  $(i, j) \in \mathcal{L}^2$ . Since  $d(i, j, \cdot)$  is positive homogeneous, an equivalent statement of (5.20) is that  $d(i, j, \cdot)$  is *convex*.

This, in turn, together with Assumption 5.1 implies an important and useful representation for  $d(i, j, \cdot)$ . Define the set

$$\mathcal{C}_{ij} := \{z \in \mathbb{R}^m \mid \langle z, \nu \rangle \leq d(i, j, \nu) \quad \forall \nu \in \mathbb{S}^{m-1}\}. \quad (5.21)$$

This is a convex and closed set, since the constraint for each fixed  $\nu$  defines a convex and closed halfspace.

**Proposition 5.3.** *Under the Assumption 5.1, the inequality (5.20), or equivalently the convexity of  $d(i, j, \cdot)$ , is equivalent to*

$$d(i, j, \nu) = \sup_{z \in \mathcal{C}_{ij}} \langle z, \nu \rangle \quad \forall \nu \in \mathbb{S}^{m-1}. \quad (5.22)$$

*Proof.* If (5.22) holds then  $d(i, j, \cdot)$  is obviously convex. The converse is a consequence of convex duality (2.8). Namely, using the homogeneity of  $d(i, j, \cdot)$  we can compute its convex dual as follows:

$$\begin{aligned} d^*(i, j, y) &= \sup_{\xi \in \mathbb{R}^m} \langle \xi, y \rangle - d(i, j, \xi) = \sup_{\nu \in \mathbb{S}^{m-1}, t \geq 0} \langle t\nu, y \rangle - d(i, j, t\nu) \\ &= \sup_{\nu \in \mathbb{S}^{m-1}} \sup_{t \geq 0} t (\langle \nu, y \rangle - d(i, j, \nu)) = \sup_{\nu \in \mathbb{S}^{m-1}} \delta_{\langle \nu, y \rangle \leq d(i, j, \nu)} \quad (5.23) \\ &= \delta_{\langle \nu, y \rangle \leq d(i, j, \nu) \leq 0 \quad \forall \nu \in \mathbb{S}^{m-1}} = \delta_{\mathcal{C}_{ij}}(y). \end{aligned}$$

<sup>1</sup>The presentation in the conference paper [121] is incorrect in that it focuses on these corollary assumptions, missing the more general ones (5.15).

Since  $d(i, j, \cdot)$  is assumed to be convex and lower-semicontinuous, (5.22) now follows from (2.8).  $\square$

The inequality constraints of the set  $\mathcal{C}_{\text{loc}}$  in (5.4) can be written more concisely using the sets (5.21) in the form  $p_j - p_i \in \mathcal{C}_{ij}$ , i.e.

$$\mathcal{C}_{\text{loc}} = \left\{ p \in \mathbb{R}^{m \times n} \mid p_j - p_i \in \mathcal{C}_{ij} \quad \forall i, j \right\}. \quad (5.24)$$

The representation (5.22) can also be shown directly from (5.11): If (5.11) holds then

$$d(i, j, \nu) = \sup_{p \in \mathcal{C}_{\text{loc}}} \langle p_j - p_i, \nu \rangle \leq \sup_{p: p_j - p_i \in \mathcal{C}_{ij}} \langle p_j - p_i, \nu \rangle \leq d(i, j, \nu), \quad (5.25)$$

yielding (5.22).

#### 5.2.4 Corollary Assumption on Label Dependency

For the case that all jump directions are handled equally, i.e.  $d(i, j, \nu) = d(i, j)$  for all  $\nu \in \mathbb{S}^{m-1}$ , the condition for (5.11) is that  $d$  must satisfy the triangle inequality

$$d(i, j) \leq d(i, k) + d(k, j) \quad \forall i, j, k$$

together with  $d(i, i) = 0$  and  $d(i, j) = d(j, i)$  [76]. For the general case, the assumption (5.2) includes as a special case that these conditions must hold for every fixed  $\nu$ . This can be seen as follows.

First, since  $\nu \otimes (e_i - e_i) = 0 = 0 \otimes (e_i - e_i)$  and  $d(i, i, 0) = 0$  by homogeneity of  $d(i, i, \cdot)$ , from (5.15) we get

$$d(i, i, \nu) = 0. \quad (5.26)$$

Next, since for vectors  $\nu \in \mathbb{S}^{m-1}$  and labels  $i, j \in \mathcal{L}$  we have  $\nu \otimes (e_j - e_i) = (-\nu) \otimes (e_i - e_j)$ , the inequality (5.15) gives  $d(i, j, \nu) \leq d(j, i, -\nu)$ . Replacing  $\nu$  by  $-\nu$  and  $(i, j)$  by  $(j, i)$  we also get the reverse inequality  $d(j, i, -\nu) \leq d(i, j, \nu)$ . Combined we obtain

$$d(i, j, \nu) = d(j, i, -\nu). \quad (5.27)$$

This necessary condition has an intuitive interpretation: When the labeling changes from label  $i$  to label  $j$  in direction  $\nu$ , this is the same as saying that the labeling changes from  $j$  to  $i$  in the opposite direction  $-\nu$ , so that the penalization by  $d$  should be the same in both cases.

Finally, using

$$\nu \otimes (e_j - e_i) = \nu \otimes (e_k - e_i) + \nu \otimes (e_j - e_k) \quad (5.28)$$

for any  $i, j, k \in \mathcal{L}$ , (5.15) implies that  $d$  must satisfy

$$d(i, j, \nu) \leq d(i, k, \nu) + d(k, j, \nu). \quad (5.29)$$

This is the *triangle inequality* for  $d(\cdot, \cdot, \nu)$  for fixed  $\nu \in \mathbb{R}^m$ .

### 5.2.5 Properties of the Regularizer

The following theorem states that with Assumptions 5.1 and 5.2 we indeed have found a *necessary and sufficient* condition for (5.11).

**Theorem 5.4.** *Equality (5.11) holds if and only if  $d$  satisfies the Assumptions 5.1 and 5.2.*

*Proof.* See appendix Section 5.7.  $\square$

This theorem gives an *exact characterization* of the distance functions expressible with our approach. The restrictions imposed are quite natural for distance functions. Note that the penalization is even allowed to be *negative* for some directions, meaning an endorsement of certain jumps.

For each fixed  $u \in \mathcal{D}_0$ , i.e. a labeling in the original binary constraint set, let  $S_u$  be its jump set. At each  $x \in S_u$  let the labeling change from label  $l^-(x)$  to label  $l^+(x)$  in direction  $\nu_u(x) \in \mathbb{S}^{m-1}$ . That is,  $u^-(x) = e_{l^-(x)}$  and  $u^+(x) = e_{l^+(x)}$ , and  $\nu_u(x)$  is a normal to  $S_u$  pointing to the  $l^+(x)$  side.

Based on Proposition 5.4 we can prove the following main theorem of this chapter. It shows that, assuming the necessary conditions, the regularizer (5.2) does indeed what it promises, namely measuring the total length over all jump interfaces weighted by the local penalizations  $d(i, j, \nu)$ .

**Theorem 5.5.** *Let  $d$  satisfy the Assumptions 5.1 and 5.2. Then*

$$R(u) = \int_{S_u} d(l^-(x), l^+(x), \nu_u(x)) \, d\mathcal{H}^{m-1}(x). \quad (5.30)$$

*Proof.* See appendix Section 5.7.  $\square$

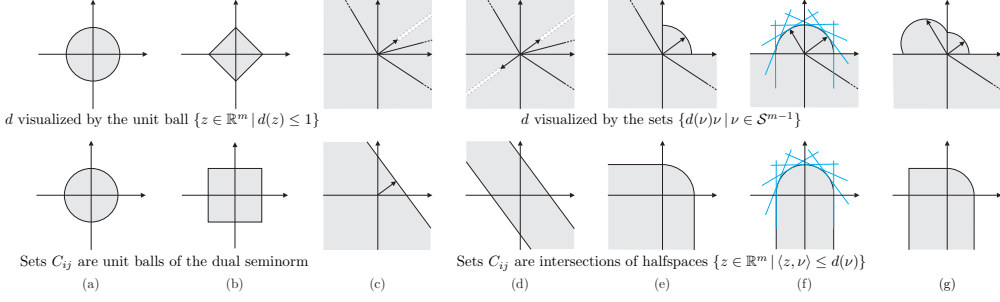
Furthermore,  $R(u)$  is of course convex so that global optimization is possible.

**Proposition 5.6.**  *$R(u)$  is convex.*

*Proof.* For  $0 < \alpha < 1$  and  $u = \alpha u^1 + (1 - \alpha)u^2$  we have

$$\begin{aligned} R(u) &= \sup_{p \in \mathcal{C}} \int_{\Omega} \langle p, \alpha \, dDu^1 + (1 - \alpha) \, dDu^2 \rangle \\ &= \sup_{p \in \mathcal{C}} \left( \alpha \int_{\Omega} \langle p, dDu^1 \rangle + (1 - \alpha) \int_{\Omega} \langle p, dDu^2 \rangle \right) \\ &\leq \alpha \sup_{p \in \mathcal{C}} \int_{\Omega} \langle p, dDu^1 \rangle + (1 - \alpha) \sup_{p \in \mathcal{C}} \int_{\Omega} \langle p, dDu^2 \rangle \\ &= \alpha R(u^1) + (1 - \alpha) R(u^2). \quad \square \end{aligned} \quad (5.31)$$

For the *continuous* label space, Alberti et al. [1, Lemma 3.7] give a relaxation of a general regularizer, which allows penalizations depending on the jump direction. Our regularizer (5.2) can be considered as its discretization to a finite number of labels. However, it is not clear if the direction dependency of [1] transfers correctly to the discrete setting. With Theorem 5.5, we establish an exact result for arbitrary dimensions, showing that the specified penalties are indeed attained at given jump directions. In [5, Sect. 5.4] Ambrosio et al. study the lower-semicontinuity properties of functionals of the form (5.30) defined on piecewise constant functions  $u$ . They show that the same conditions (5.15) on  $d$  play a crucial role in order to be able to prove lower-semicontinuity [5, Th. 5.22].



**Figure 5.3: Distance functions and the corresponding constraint sets.** Different penalization functions  $d$  (top) and the corresponding sets  $C_{ij}$  in (5.21) (bottom). (a) and (b) show  $d$ 's arising by picking a specific seminorm, here the isotropic  $\|\cdot\|_{l_2}$  and anisotropic  $\|\cdot\|_{l_1}$ , the  $l^1$ -norm as used in graph cuts. (c–g) show  $d$ 's arising by an explicit construction, following the arrows until the boundary gives the values  $d(\nu)$ . In (c) only one direction is allowed, in (d) only two, in (e) only right and up, in (f) only from bottom to top and in (g) just as in (f) but the left directions are penalized more.

### 5.3 Constraint Sets

In this section we will give some examples of the constraint sets (5.21) for image domain dimension  $m = 2$ , arising by allowing labels to jump only in particular directions. The sets for common directions are depicted in Figure 5.3. In general, the set  $C_{ij}$  is the intersection of halfspaces, see Figure 5.3 (c–g), and can be found geometrically as in (f). In the case that  $d$  is a seminorm, i.e. satisfies  $d(\nu) \geq 0$  in addition to (5.20), this set is easily found to be the unit ball

$$C_{ij} = \{w \in \mathbb{R}^m \mid d^*(w) \leq 1\} \tag{5.32}$$

of the dual seminorm

$$d^*(w) := \sup_{z \in \mathbb{R}^m: d(z) \leq 1} \langle z, w \rangle, \tag{5.33}$$

see Figure 5.3 (a, b).

An example of a general distance function is

$$d_{\lambda, A}(\nu) = \begin{cases} \lambda & \text{if } \angle \nu \in A, \\ \infty & \text{else} \end{cases} \tag{5.34}$$

for some fixed  $\lambda \geq 0$ , which allows only jump directions with angles from a set  $A$  and aside from that penalizes the interface length isotropically. That is, using a distance function of this kind only imposes additional *hard constraints* in the optimization, restricting possible jump directions. But once a partitioning satisfies these hard constraints, the value of the regularizer is the overall interface length as in the classical partitioning problem. Special cases are

$$d_{\lambda, [0, 2\pi]}, \quad d_{\lambda, \{\alpha\}}, \quad d_{\lambda, \{\alpha, \alpha+\pi\}} \quad \text{and} \quad d_{\lambda, [\alpha, \beta]} \tag{5.35}$$

for some angles  $\alpha$  and  $\beta$  with  $\alpha \leq \beta \leq \alpha + \pi$ , see Figure 5.3 (a), (c), (d), resp. (e-f). These are already all possible cases due to the convexity restriction (5.20). One can easily derive formulas for the projections onto the corresponding sets (5.21).

In discrete graph cut and dynamic programming approaches one only specifies  $d(\nu)$  for the axis directions  $\nu = \pm e_1$  and  $\nu = \pm e_2$ . The resulting set  $\mathcal{C}_{ij}$  is then a rectangle, see Figure 5.3 (b). Computing  $d$  back by (5.22), e.g. for the square case we see that this produces the penalization  $d(\nu) = \|\nu\|_{l_1}$ , which is not rotationally invariant as opposed to  $\|\nu\|_{l_2}$ , see Figure 5.3 (a). Thus, the discrete approaches form a subset of our framework, and we can actually see that they necessarily give rise to a metrication error.

## 5.4 Implementation

### 5.4.1 Discretization

The discretization is done exactly as in the basic multilabel case in Section 3.5, except that the local dual set  $\mathcal{C}_{\text{loc}}$  in (3.46) is given by (5.24). The overall optimization problem (3.44) is solved using the preconditioned primal-dual Algorithm 3, and the proximal operators are here again projections onto the constraint sets. The projection for the primal variable  $u$  can be implemented as described in Section 3.5, either through a direct simplex projection or through auxiliary variables to dualize the sum constraints resulting in energy (3.51). For the experiments in this chapter we used the dualization approach for  $u$ .

Here we will focus on the projection for the local dual set.

### 5.4.2 Projection For the Dual $p$

We need to project  $p \in \mathbb{R}^{m \times n}$  onto (5.24). The constraints couple all  $p_i$ 's similarly to the tight relaxation (3.33) for the direction independent case, so that there is no simple projection formula. In the following we will present two approaches of how the projection can be implemented.

**Approach 1: Variables for the Differences.** The first approach is the one originally proposed in the conference paper [121] on which this chapter is based. The idea is to introduce auxiliary dual variables  $q_{ij}(x) \in \mathbb{R}^m$  for the  $p$  differences:

$$q_{ij}(x) = p_j(x) - p_i(x) \in \mathbb{R}^m \quad \forall x \in \Omega, (i, j) \in \mathcal{L}^2. \quad (5.36)$$

These linear equalities can be enforced through the dualization (2.74), introducing new primal variables  $a_{ij}(x) \in \mathbb{R}^m$  (Lagrange multipliers) for all  $x$  and all pairs  $i, j$  and adding the terms

$$\inf_{a_{ij}: \Omega \rightarrow \mathbb{R}^m} \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \langle -a_{ij}(x), p_j(x) - p_i(x) - q_{ij}(x) \rangle \quad (5.37)$$

to the energy. The projection for  $p$  now translates into projections of each  $q_{ij}(x) \in \mathbb{R}^m$  *independently* onto  $\mathcal{C}_{ij} \subset \mathbb{R}^m$ .



In common cases such as depicted in Figure 5.3, there is a simple formula for this. Specifically, this is the case for all experiments in this chapter. Otherwise, one can always approximate the projection by using some discrete range of directions  $\nu$  and Lagrange multipliers for the constraints in  $\mathcal{C}_{ij}$ . More precisely, discretizing  $\nu$  into  $r \geq 2$  unit vectors  $\nu_1, \dots, \nu_r \in \mathbb{S}^{m-1}$  uniformly covering  $\mathbb{S}^{m-1}$  in some way, we can replace the set (5.21) by the corresponding discretized variant:

$$\mathcal{C}_{ij}^r = \left\{ z \in \mathbb{R}^m \mid \langle z, \nu \rangle \leq d(i, j, \nu) \quad \forall \nu \in \{\nu_1, \dots, \nu_r\} \right\}. \quad (5.38)$$

The  $r$  linear elementary constraints for  $q_{ij}(x) \in \mathcal{C}_{ij}^r$  can now be dualized using (2.72): We introduce new variables  $b_{ij}^k(x) \in \mathbb{R}^m$  for all pairs  $i, j$ , each  $1 \leq k \leq r$  and  $x \in \Omega$ , and add to the energy the new terms

$$\inf_b \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \sum_{1 \leq k \leq r} (-b_{ij}^k(x)) \left( \langle q_{ij}(x), \nu_k \rangle - d(i, j, \nu_k) \right). \quad (5.39)$$

$b_{ij}^k(x) \geq 0 \quad \forall i, j, k, x$

**Approach 2: Direct Dualization.** Another way to enforce the constraints (5.24) on  $p$  is to separately dualize each of them, similarly as was done in (3.57) in Section 3.5.4. With new additional primal variables  $a_{ij}(x) \in \mathbb{R}^m$  for all  $x \in \Omega$  and  $i, j \in \mathcal{L}$ , we add the terms

$$\inf_{a_{ij}: \Omega \rightarrow \mathbb{R}^m} \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle + \delta_{\mathcal{C}_{ij}}^*(a_{ij}(x)) \quad (5.40)$$

to the energy. The overall energy is then

$$\begin{aligned} \min_{u \in \mathcal{D}_{d,a}^0} \max_{p, \mu} & \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle \\ & + \sum_{x \in \Omega} \mu(x) \left( \sum_{i=1}^n u_i(x) - 1 \right) \\ & + \sum_{x \in \Omega} \sum_{1 \leq i < j \leq n} \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle + \delta_{\mathcal{C}_{ij}}^*(a_{ij}(x)). \end{aligned} \quad (5.41)$$

optimizing over the reduced primal set (3.52) for  $u$ , without any constraints on each  $a_{ij}(x) \in \mathbb{R}^m$ , and also without any constraints on the duals  $p_i(x) \in \mathbb{R}^m$  and  $\mu(x) \in \mathbb{R}$  for all  $x \in \Omega$  and  $i, j \in \mathcal{L}$ .

The proximal operator for  $a$  decomposes into independent proximal operators for each  $a_{ij}(x)$ . We can reduce them directly to projections onto  $\mathcal{C}_{ij}$  by using Moreau's identity (2.40):

$$\text{prox}_{\tau_a, \delta_{\mathcal{C}_{ij}}^*}(a_{ij}(x)) = a - \tau_a \pi_{\mathcal{C}_{ij}}(a/\tau_a). \quad (5.42)$$

Thus, what remains is again only to implement the individual projections onto the sets  $\mathcal{C}_{ij}$  as in the first approach above. However, the currently presented implementation method is more efficient w.r.t. both memory and run time, since there is no need to use the auxilliary dual variables  $q_{ij}$  for the differences.

### 5.4.3 Advantage of Our Method

A prominent advantage of our method is that different ordering constraints are encoded only in the projections onto the constraint sets  $\mathcal{C}_{ij}$ . The optimization algorithm itself remains the same. This stands in contrast to [82] and [48], where the algorithms must be devised anew when the layout changes.

With a parallel CUDA implementation on NVIDIA GTX 480, usual run times for  $640 \times 480$  images and 5 labels are around 90 seconds, and for  $320 \times 240$  images 7 seconds. This compares favorably to the reported 9 seconds in [48].

## 5.5 Experimental Results

### 5.5.1 Geometric Class Labeling

In geometric class labeling, one seeks a rough labeling into geometric classes such as ‘left wall’ or ‘sky’. This can be useful in providing geometric context to more elaborate tasks such as 3d reconstruction or robot navigation [58].

**Five regions layout.** Especially for indoor images, Liu et al. [82] used a simple layout consisting of five geometric parts: ‘center’  $C$ , ‘ceiling’  $T$ , ‘floor’  $B$ , ‘left wall’  $L$  and ‘right wall’  $R$ . Natural ordering constraints on these labels result in a layout as in Figure 5.2. With the notation (5.34) we set

$$d(B, L) = d(R, T) = d_{\lambda, [\frac{\pi}{2}, \pi]}, \quad (5.43)$$

$$d(B, C) = d(C, T) = d_{\lambda, \{\frac{\pi}{2}\}}, \quad (5.44)$$

$$d(B, R) = d(L, T) = d_{\lambda, [0, \frac{\pi}{2}]}, \quad (5.45)$$

$$d(L, C) = d(C, R) = d_{\lambda, \{0\}}. \quad (5.46)$$

The distances for the remaining label pairs are then defined implicitly by the triangle inequality (5.29), and are found to be

$$d(L, R) = d(L, C) + d(C, R) = 2d_{\lambda, \{0\}} \quad (5.47)$$

$$\text{and } d(B, T) = d(B, C) + d(C, T) = 2d_{\lambda, \{\frac{\pi}{2}\}}. \quad (5.48)$$

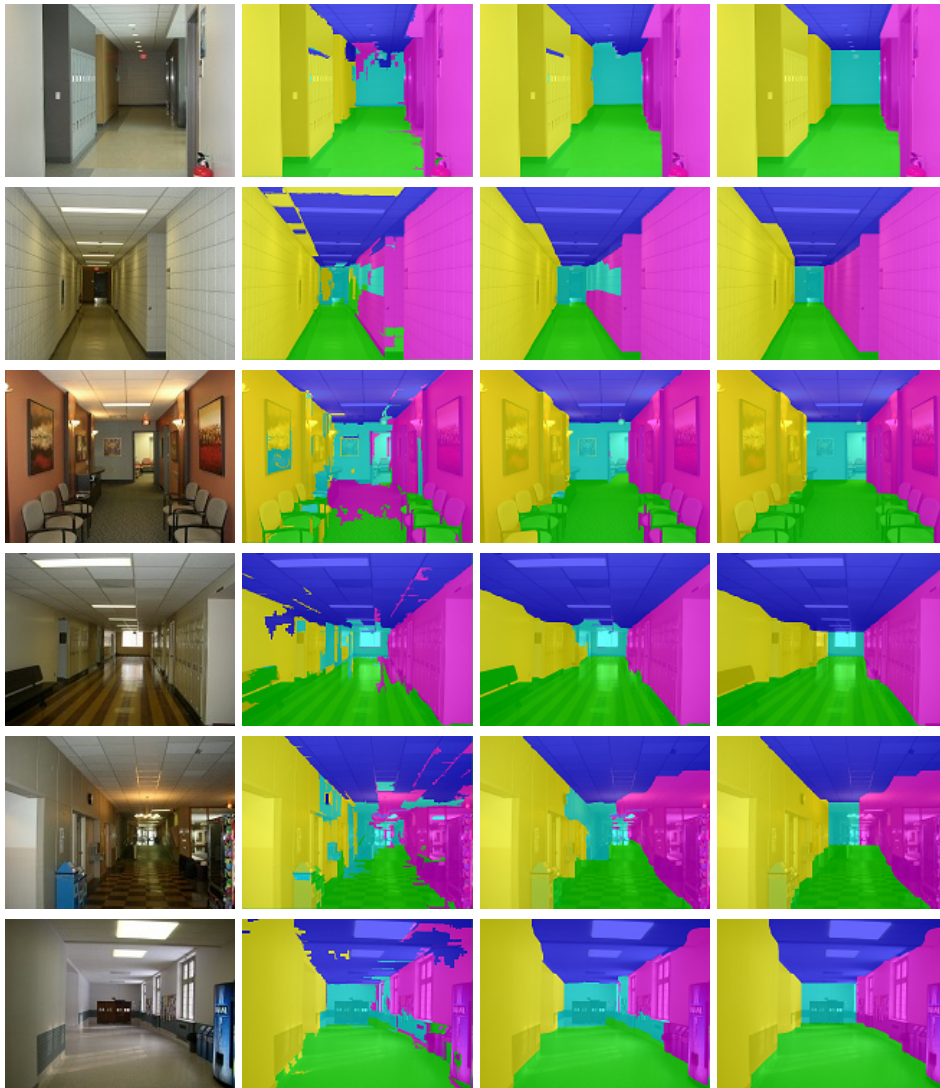
We applied our regularizer on the dataset of 300 indoor images from [82] using their data terms. We set  $\lambda = 20$  in (5.34) and weight all distances  $d$  by

$$w(x) = e^{-|\nabla I(x)|^2/2\sigma^2} \quad \text{with } \sigma^2 = \text{mean of } |\nabla I|^2 \quad (5.49)$$

for each input image  $I$ .

Results for six different images are shown in Figure 5.4. We achieved an overall accuracy of 85.3%, which is comparable to the 85% of [82].

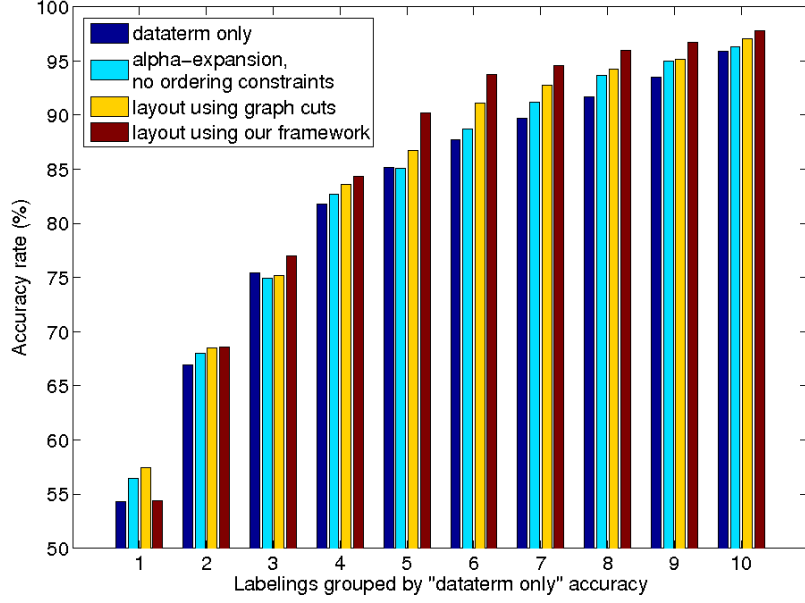
The better result despite the fact that our method is a generalization of [82] may be due to several reasons: Our spatially continuous framework does not suffer from metrication errors. Moreover, convex optimization possibly finds better minima than iterative schemes like alpha-expansion and order-preserving moves.



**Figure 5.4: Five regions layout.** Ordering constraints arise naturally for indoor images and improve the segmentation. *From left to right:* Indoor images, data term only labeling, result with Potts partitioning, result with five regions layout (see Figure 5.2).



**Figure 5.5: Failure case for the five regions layout.** Since the data term (*center*) is very misleading, the ordering constraints do not allow to recover the desired solution. *From left to right:* input image, data term, result with ordering constraints.



**Figure 5.6: Accuracy for the five regions layout.** For the five regions model we ordered the 300 images by “data term only” accuracy in 10 equal groups. The proposed method provides a slight improvement over the existing methods.

**Tiered layout.** Tiered layout [48] is a generalization of the five region layout [82]. There are the ‘top’  $T$ , ‘bottom’  $B$  and in between a number of labels, here  $L$ ,  $C$ ,  $R$ , in any sequence and multiplicity having only vertical borders between each other, see Figure 5.2. While the five regions layout assumes the center  $C$  to be a rectangle, tiered layout is well adapted for outdoor images as well, where the boundary between  $T$  and  $L$ ,  $C$ ,  $R$  is less predictable, see Figure 5.7. We set

$$d(B, L, \nu) = d(R, T, \nu) = \begin{cases} d_{\lambda, [\frac{\pi}{2}, \pi]} & \text{if } \angle \nu \notin [0, \frac{\pi}{2}], \\ \lambda(2|\nu_1| + |\nu_2|) & \text{else,} \end{cases} \quad (5.50)$$

$$d(B, C) = d(C, T) = d_{\lambda, [0, \pi]}, \quad (5.51)$$

$$d(B, R, \nu) = d(L, T, \nu) = \begin{cases} d_{\lambda, [0, \frac{\pi}{2}]}(\nu) & \text{if } \angle \nu \notin [\frac{\pi}{2}, \pi], \\ \lambda(2|\nu_1| + |\nu_2|) & \text{else,} \end{cases} \quad (5.52)$$

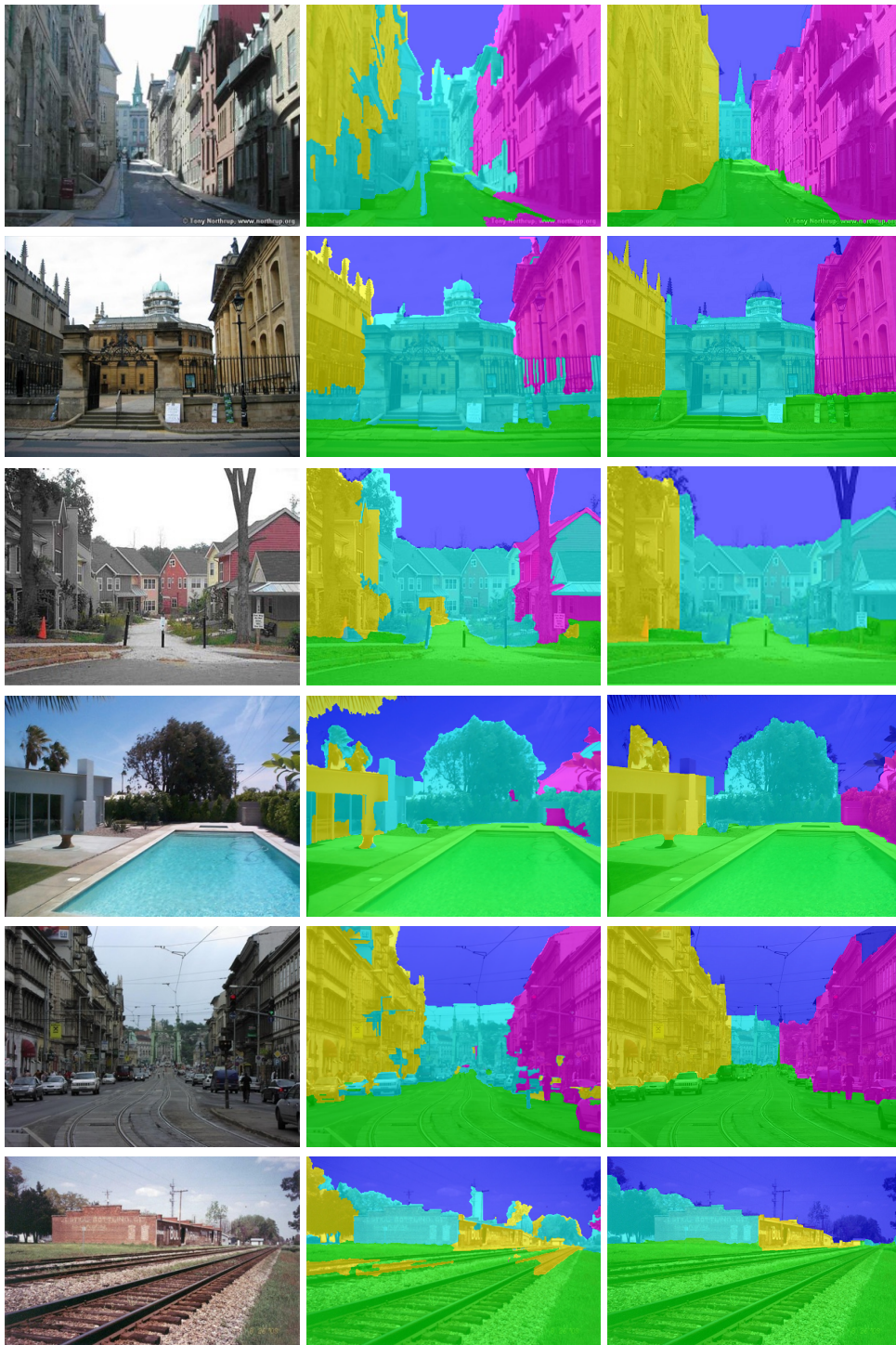
$$d(L, C) = d(C, R) = d_{\lambda, \{0, \pi\}}, \quad (5.53)$$

Other distances follow implicitly by the triangle inequality (5.29):

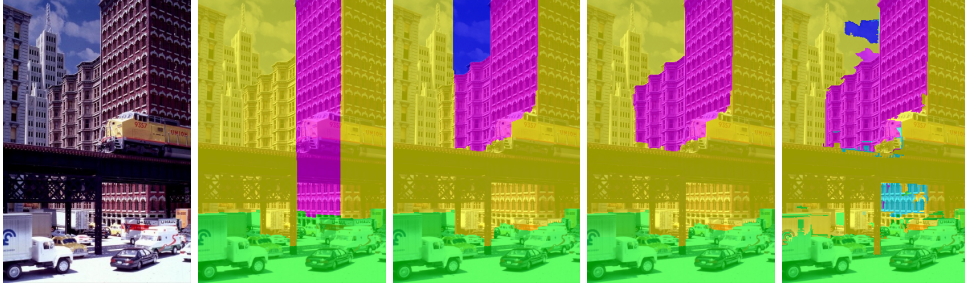
$$d(L, R) = d(L, C) + d(C, R) = 2d_{\lambda, \{0, \pi\}}, \quad (5.54)$$

$$d(B, T) = d(B, C) + d(C, T) = 2d_{\lambda, [0, \pi]}. \quad (5.55)$$

In the optimization, only the projections for explicitly specified  $d$ ’s must be taken into account. The main difference to the five regions layout from the previous section is that (5.44) becomes (5.51).



**Figure 5.7: Tiered layout results.** Ordering constraints improve the segmentation of outdoor images. *From left to right:* input images, Potts partitioning, partitioning with ordering constraints.



**Figure 5.8: Four and more tiers.** We can easily model the four and more tiered layout, thus allowing more than one label  $L$ ,  $C$  or  $R$  to come one after another in one column. *From left to right:* input image, result with 3, 4 and 5 tiers, and data term.

The expression for  $d(B, R)$  in (5.52) results by starting with

$$d(B, R) = d_{\lambda, [0, \frac{\pi}{2}]} =: d_1 \quad (5.56)$$

as in (5.43) and then restricting the values down due to the triangle inequality (5.29):

$$d(B, R) \leq d(B, C) + d(C, R) = 2d_{\lambda, \{0, \pi\}} =: d_2. \quad (5.57)$$

Therefore, we define the actually used distance function  $d(B, R)$  as the largest possible function of the form (5.22) with  $d(B, R) \leq d_1, d_2$ . Due to these inequalities, by definition (5.21) the corresponding set  $\mathcal{C}_{BR}$  of  $d(B, R)$  must be contained in the intersection of the sets  $\mathcal{C}_{d_1}$ , as in Figure 5.3 (e), and  $\mathcal{C}_{d_2}$ , as in Figure 5.3 (d) but with a vertical stripe. The largest possible  $d(B, R)$  is obtained if we set  $\mathcal{C}_{BR}$  to be equal to this intersection:

$$\mathcal{C}_{BR} = \mathcal{C}_{d_1} \cap \mathcal{C}_{d_2}, \quad (5.58)$$

see Figure 5.3 (g). The value in (5.52) is then computed from (5.22).

Six results on the dataset from [58], consisting of 300 outdoor images, are shown in Figure 5.7. We used the confidence estimates provided in [58] for the data term and same parameters as for the five regions layout. Our overall accuracy 86.3% compares favorably to the 81.4% reported in [48].

Our method generalizes [48], in that we can handle all cases where the triangle inequalities are satisfied. While [48] provides optimal solutions, it only applies to a very specific layout. Our method is capable of far more general layouts. This class is NP-complete containing the Potts model [19], so no globally optimal solutions can be expected. Nevertheless, our method provides tight a-posteriori optimality bounds (3.42). The binary solution through the maximum-value binarization (3.41) is usually within 5% of the global optimum energy-wise.

**Four and More Tiers.** To demonstrate the flexibility of our framework, we can easily model more than three tiers, see Figures 5.2 and 5.8.

For  $t \geq 3$  tiers, we place  $B$  in the lowest tier, and  $T$  in the highest tier. The middle tier between  $B$  and  $T$  is split into  $t - 2$  individual tiers, the first consisting of regions  $L_1, C_1, R_1$ , the second of regions  $L_2, C_2, R_2$ , etc.

The bottom label  $B$  is allowed to change to  $L_1, C_1, R_1$  just as before to  $L, C, R$  in (5.50)–(5.52), and labels  $L_{t-2}, C_{t-2}, R_{t-2}$  can change to  $T$  as  $L, C, R$  before:

$$d_{t \text{ tiers}}(B, L_1) = d_{t \text{ tiers}}(R_{t-2}, T) = d_{3 \text{ tiers}}(B, L), \quad (5.59)$$

$$d_{t \text{ tiers}}(B, C_1) = d_{t \text{ tiers}}(C_{t-2}, T) = d_{3 \text{ tiers}}(B, C), \quad (5.60)$$

$$d_{t \text{ tiers}}(B, R_1) = d_{t \text{ tiers}}(L_{t-2}, T) = d_{3 \text{ tiers}}(B, R), \quad (5.61)$$

Within one of the middle tiers  $1 \leq i \leq t - 2$ , labels  $L_i, C_i, R_i$  change as  $L, C, R$  in (5.50):

$$d_{t \text{ tiers}}(L_i, C_i) = d_{t \text{ tiers}}(C_i, R_i) = d_{3 \text{ tiers}}(L, C). \quad (5.62)$$

Finally, for each  $1 \leq i \leq t - 3$  the labels  $L_i, C_i, R_i$  can change “one level up” to  $L_{i+1}, C_{i+1}, R_{i+1}$  in any direction heading from bottom to top:

$$d_{t \text{ tiers}}(X, Y) = d_{\lambda, [0, \pi]} \quad \forall X \in \{L_i, C_i, R_i\}, \quad Y \in \{L_{i+1}, C_{i+1}, R_{i+1}\}. \quad (5.63)$$

Other distances follow implicitly by the triangle inequalities.

In contrast, the dynamic programming approach [48] does not allow an easy extension to four tiers, yielding a significantly more complex algorithm.

The complexity is quadratic in the number of labels *per tier*, here 3 ( $L, C$  and  $R$ ), but is *linear* in the number  $t$  of tiers. Only the interactions between two *consecutive* tiers must be explicitly modeled and implemented. The interactions between all other tiers do not need to be implemented as they follow implicitly through the underlying convexification model, which automatically “fills in” the missing penalization definitions  $d$ .

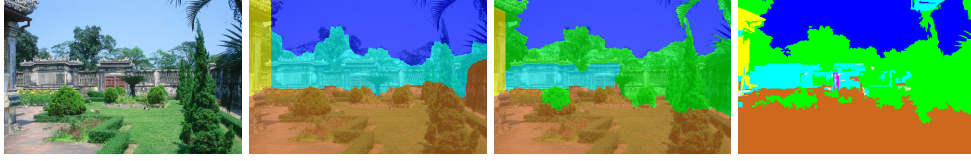
**Floating Objects.** We can also allow arbitrary “floating” objects on top of the layout, see Figures 5.2 and 5.9. The dataset [58] contains also the classes ‘porous’ and ‘solid’. However, they do not fit into the tiered layout, since their relation to other objects is less predictable. These classes are therefore simply ignored in [48]. In contrast, our framework allows to include these extra classes.

To include the ‘solid’ class  $S$ , we regard it as being “on top” of the tiered layout. We then split up the  $L$  region into  $L_S$  and  $L_0$ , meaning the parts that do or do not contain  $S$ , and similarly for other labels of the tiered layout. Jumps within non- $S$  labels, e.g. from  $L_0$  to  $C_0$  are then defined in the same way as previously from  $L$  to  $C$  by (5.50):

$$d_{\text{extra object}}(X_0, Y_0) = d_{\text{tiered}}(X, Y) \quad \forall X, Y \in \{B, L, C, R, T\}. \quad (5.64)$$

Jumps within  $S$ -labels, e.g. from  $L_S$  to  $C_S$  only inherit the direction restrictions of  $L$  and  $C$ , and the penalization is set to zero, since these are actually two parts of the same label  $S$ . Here we use (5.50) with the parameter  $\lambda = 0$  in (5.34):

$$d_{\text{extra object}}(X_S, Y_S) = d_{\text{tiered, but with } \lambda = 0}(X, Y) \quad \forall X, Y \in \{B, L, C, R, T\}. \quad (5.65)$$



**Figure 5.9: Floating objects.** Our framework extends the tiered labeling to allow floating objects, arbitrarily located upon the layout. *From left to right:* input image, tiered layout result, result with extra objects ‘solid’ and ‘porous’, data term.

Finally, jumps from  $S$ -labels to non- $S$  labels just mean a jump from  $S$  to some other label, so we can introduce an arbitrary new penalization here, e.g. the Potts model which penalizes all directions equally:

$$d_{\text{extra object}}(X_S, Y_0) = d_{\lambda, [0, 2\pi]} \quad \forall X, Y \in \{B, L, C, R, T\}. \quad (5.66)$$

Inclusion of  $m$  extra classes  $S_1, \dots, S_m$  can be done analogously, splitting each tiered layout label into  $m + 1$  parts, e.g.  $L_0, L_{S_1}, \dots, L_{S_m}$ :

$$d_{m \text{ extra objects}}(X_0, Y_0) = d_{\text{tiered}}(X, Y) \quad \forall X, Y, \quad (5.67)$$

$$d_{m \text{ extra objects}}(X_{S_i}, Y_{S_i}) = d_{\text{tiered, but with } \lambda = 0}(X, Y) \quad \forall X, Y, i, \quad (5.68)$$

$$d_{m \text{ extra objects}}(X_{S_i}, Y_0) = d_{\lambda, [0, 2\pi]} \quad \forall X, Y, i, \quad (5.69)$$

$$d_{m \text{ extra objects}}(X_{S_i}, Y_{S_j}) = d_{\lambda, [0, 2\pi]} \quad \forall X, Y, i < j. \quad (5.70)$$

Fixing a layout on top of which one wishes to put  $m$  extra objects, the complexity is quadratic in  $m$  due to (5.70).

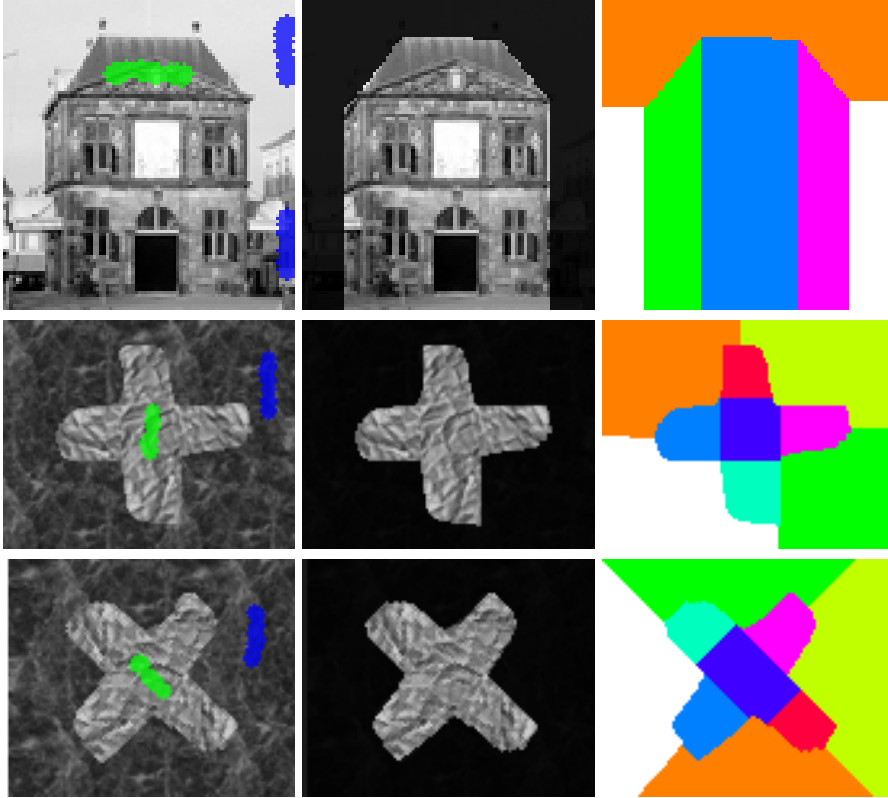
### 5.5.2 Shape Priors

Using our approach we can model certain shape priors. The object of interest is divided into a number of sublabels, and some background labels are introduced surrounding this object. The idea is that changes from object labels to surrounding labels are constrained to certain jump directions only. We can model different shape priors expressible by the tiered labeling approach, such as a rectangle, trapezoid, ‘house’ or ‘cross’ [48]. The latter two are depicted in Figure 5.10. However, our approach is more general, as it allows interfaces in arbitrary directions, not only vertical and horizontal. A simple example is the rotated cross prior in Figure 5.10 which is not expressible using tiered layout. The distance functions  $d$  can be easily derived from the layouts shown in Figure 5.10.

**Star Shape Prior.** We note that the *star shape* prior [131] is expressible with our approach. This is an example where the distance function  $d(x, i, j, \nu)$  also depends on the position  $x \in \Omega$ . Object and background are represented by the labels fg and bg, respectively. We fix an arbitrary point  $x_0 \in \Omega$ , which acts as the star center, and set

$$d(x, \text{fg}, \text{bg}, \nu) := \begin{cases} \lambda & \text{if } \langle x - x_0, \nu \rangle \geq 0, \\ \infty & \text{otherwise} \end{cases} \quad (5.71)$$





**Figure 5.10: Segmentation with different shape priors.** In contrast to [48], the proposed approach also allows rotated crosses. *From left to right:* initial image with user scribbles, segmentation using the prior, prior encoding as a label layout. We use a simple probability estimation from user scribbles to obtain the data term.

for all  $\nu \in \mathbb{S}^{m-1}$  with some  $\lambda \geq 0$ . This yields a  $d$  as in Figure 5.3 (f), with the “up” direction  $x - x_0$ . This way, the labeling is allowed to change from “object” to “background” only in a direction “away” from the point  $x_0$ . Therefore, the object will be star shaped with center  $x_0$ .

**Novel Prior: Convex Shape.** A more advanced novel application of our framework is to model *convex priors*, i.e. to favor objects which are convex. To accomplish this, we explicitly model the boundary of the object. We introduce  $n \geq 3$  auxiliary surrounding regions  $0, \dots, n - 1$ , see Figure 5.11. Let

$$\nu_i := (\cos \alpha_i, \sin \alpha_i) \quad \text{with} \quad \alpha_i := i \cdot \frac{2\pi}{n}. \quad (5.72)$$

The main object label  $\text{fg}$  is then allowed to change to a background label  $0 \leq i \leq n - 1$  only in direction  $\nu_i$ , and a background label  $i$  to the next such label  $i + 1$  only in directions from  $\nu_i^\perp$  to  $\nu_{i+1}^\perp$  (regarding the index  $i + 1$  modulo  $n$ ):

$$d(\text{fg}, i) = d_{\lambda, \{\alpha_i\}}, \quad d(i, i + 1) = d_{0, [\alpha_i + \frac{\pi}{2}, \alpha_{i+1} + \frac{\pi}{2}]}, \quad (5.73)$$

see Figure 5.3 (c, e). Note that only the foreground-background transition penalization  $d(\text{fg}, i)$  is considered with a positive length penalization parameter  $\lambda$ .



**Figure 5.11: Convex shape prior.** Beyond existing approaches, the proposed framework allows to model priors on convex shapes. *From left to right:* input image with user scribbles, Potts partitioning, partitioning using the convex shape prior, prior encoding. The prior is modeled by a  $n$ -gon, here an octagon, with edges pointing in all directions.

The background-background penalizations  $d(i, i + 1)$  are considered with  $\lambda = 0$  in (5.34), so that they only act as hard constraints without altering the actual energy. This way, the  $n$  auxiliary background regions are explicitly oriented around the object, building its boundary. Thus, we obtain a  $n$ -gon shape which is guaranteed to be convex.

The complexity for this convexity prior is *linear* with the number  $n$  of polygon sides, since there are only linearly many constraints to consider. In practice, the choice  $n = 64$  is already sufficient to be able to obtain *any* convex object, because the polygon segmentation is then not visually distinguishable from a “round” boundary.

## 5.6 Conclusion

We introduced a novel framework for a general multilabel regularizer allowing the penalization to depend on the jump direction. Although conceptually entirely different the proposed approach unifies and generalizes existing MRF-based formulations. Despite its generality, the regularizer is easily adapted to various label layouts by merely changing the projections of dual variables. In contrast, existing approaches require entirely different algorithms depending on the choice of the layout. We proved a necessary and sufficient condition on the label distance function to be expressible with our framework. Quantitative experiments show that the proposed method compares favorably to existing approaches.

## 5.7 Appendix: Proofs of Propositions and Theorems

*Proof of Theorem 5.4.* The necessity of the conditions in the assumptions was already shown in Section 5.2.2. Let us now show that the conditions are also sufficient.

First, observe that the representation (5.22) means that  $d(k, l, \cdot)$  is the

Legendre-Fenchel conjugate of the indicator function of the set  $\mathcal{C}_{kl}$ :

$$d(k, l, \cdot) = (\delta_{\mathcal{C}_{kl}})^*. \quad (5.74)$$

The latter function is convex and lower-semicontinuous since  $\mathcal{C}_{kl}$  in (5.21) is convex and closed. By convex duality (2.8) therefore  $\delta_{\mathcal{C}_{kl}} = d(k, l, \cdot)^*$ , i.e.

$$\delta_{\mathcal{C}_{kl}}(q) = \sup_{\xi \in \mathbb{R}^m} \langle \xi, q \rangle - d(k, l, \xi) \quad \forall q \in \mathbb{R}^m. \quad (5.75)$$

We use the representation (5.24) for the constraint set  $\mathcal{C}_{\text{loc}}$  and write the hard constraints as energy terms:

$$\begin{aligned} \sup_{p \in \mathcal{C}_{\text{loc}}} \langle p_j - p_i, \nu \rangle &= \sup_{\substack{p \in \mathbb{R}^{m \times n} \\ p_l - p_k \in \mathcal{C}_{kl} \forall k, l}} \langle p_j - p_i, \nu \rangle \\ &= \sup_{p \in \mathbb{R}^{m \times n}} \langle p_j - p_i, \nu \rangle - \sum_{k, l} \delta_{\mathcal{C}_{kl}}(p_l - p_k). \end{aligned} \quad (5.76)$$

Next, we dualize each individual constraint using the above relation (5.75):

$$\begin{aligned} \sup_{p \in \mathcal{C}_{\text{loc}}} \langle p_j - p_i, \nu \rangle &= \sup_{p \in \mathbb{R}^{m \times n}} \inf_{\xi \in (\mathbb{R}^m)^{n \times n}} \langle p_j - p_i, \nu \rangle \\ &\quad - \sum_{k, l} \langle \xi_{kl}, p_l - p_k \rangle + \sum_{k, l} d(k, l, \xi_{kl}) \\ &= \inf_{\xi \in (\mathbb{R}^m)^{n \times n}} \sum_{k, l} d(k, l, \xi_{kl}) \\ &\quad + \sup_{p \in \mathbb{R}^{m \times n}} \langle p_j - p_i, \nu \rangle - \sum_{k, l} \langle \xi_{kl}, p_l - p_k \rangle. \end{aligned} \quad (5.77)$$

The supremum over  $p$  results in hard constraints on  $\xi$ , namely

$$\nu(\chi_{t=j} - \chi_{t=i}) = \sum_{k, l} \xi_{kl}(\chi_{t=l} - \chi_{t=k}) \quad \forall t \in \mathcal{L}, \quad (5.78)$$

which can be written concisely as

$$\nu \otimes (e_j - e_i) = \sum_{k, l} \xi_{kl} \otimes (e_l - e_k). \quad (5.79)$$

Overall,

$$\sup_{p \in \mathcal{C}_{\text{loc}}} \langle p_j - p_i, \nu \rangle = \inf_{\substack{\xi \in (\mathbb{R}^m)^{n \times n} \\ \text{s.t. (5.79)}}} \sum_{k, l} d(k, l, \xi_{kl}). \quad (5.80)$$

Since the constraint on  $\xi$  is exactly of the form (5.16), we can now employ the Assumption (5.2) to conclude that the infimum in (5.80) is  $\geq d(i, j, \nu)$ . This value is actually achieved, namely by the  $\xi$  defined as  $\xi_{ij} = \nu$  and  $\xi_{kl} = 0$  for all  $(k, l) \neq (i, j)$ , which trivially satisfies the constraint (5.79). Thus the infimum is equal to  $d(i, j, \nu)$  and we get

$$\sup_{p \in \mathcal{C}_{\text{loc}}} \langle p_j - p_i, \nu \rangle = d(i, j, \nu), \quad (5.81)$$

proving the theorem.  $\square$

*Proof of Theorem 5.5.* Since  $u \in \mathcal{D}_0$  is piecewise constant, its distributional gradient  $Du$  consists only of the jump part:  $Du = \nu_u \otimes (e_{l^+} - e_{l^-}) \mathcal{H}^{m-1} \llcorner S_u$ , respectively for label  $i$ :

$$Du_i = \nu_u(\chi_{l^+=i} - \chi_{l^-=i}) \mathcal{H}^{m-1} \llcorner S_u. \quad (5.82)$$

Thus, the integration in (5.2) reduces to the integration on the jump set  $S_u$ :

$$\begin{aligned} R(u) &= \sup_{p \in \mathcal{C}} \int_{S_u} \sum_{i=1}^n \left\langle p_i(x), \nu_u(x)(\chi_{l^+(x)=i} - \chi_{l^-(x)=i}) \right\rangle d\mathcal{H}^{m-1}(x) \\ &= \sup_{p \in \mathcal{C}} \int_{S_u} \left\langle p_{l^+(x)}(x) - p_{l^-(x)}(x), \nu_u(x) \right\rangle d\mathcal{H}^{m-1}(x) \\ &= \int_{S_u} \sup_{p \in \mathcal{C}_{\text{loc}}} \left\langle p_{l^+(x)} - p_{l^-(x)}, \nu_u(x) \right\rangle d\mathcal{H}^{m-1}(x). \end{aligned} \quad (5.83)$$

The theorem now follows from the property (5.11) of  $d$ . □

## Chapter 6

# Multilabel Segmentation with Proportion Priors

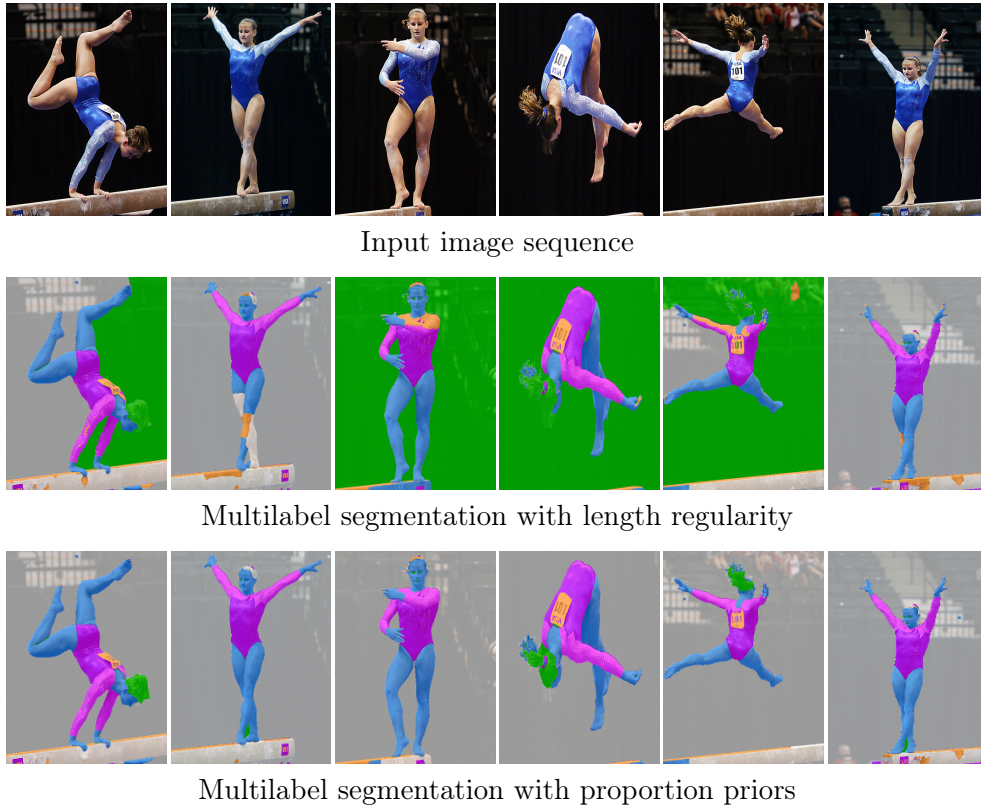
We will introduce here the final of the three multilabel priors, namely different proportion priors to achieve a higher scale-invariance and robustness to shape changes. This chapter is based on joint work with Claudia Nieuwenhuis and Daniel Cremers [94].

### 6.1 Introduction

#### 6.1.1 Image Sequence Segmentation

Automatic image sequence segmentation denotes the task of jointly segmenting one or several objects from a series of images taken under different view points, lighting conditions and background scenes. The difficulty lies in the fact that none of the objects' properties is guaranteed to be preserved over time. Both the geometry and the photometry of the objects of interest may change from one image to the next. Changing illumination affects the observed color model, different viewpoints lead to different object scales and possibly self-occlusions, whole parts of the object can even vanish from the image, and objects may occlude each other in the case of multiple foreground objects. Moreover, articulations and non-rigid deformations give rise to substantial shape changes. Modeling such variable conditions and exploiting information which is shared among the images is a challenging task.

In general, the co-segmentation problem deals with the situation that we have no knowledge on the object of interest besides that it appears in all images. To make the problem tractable many approaches introduce at least some kind of prior knowledge based on training data or user scribbles [11, 91, 62, 132]. The resulting optimization problems are often iterative and hard to optimize [91, 109, 90, 110] leading to run times of several seconds or even minutes. For complex real-world image sequences the task of leveraging relevant shared information for co-segmentation remains an open challenge. In this chapter, we propose a Bayesian framework for multiregion co-segmentation which allows to impose learned *proportion priors*, see Figure 6.1. We show that near-optimal solutions can be efficiently computed by means of convex relaxation techniques.



**Figure 6.1: Proportion priors.** The proposed framework allows to constrain the relative size between different object parts (e.g. hair, skin, body, 101-sign). Proportion priors enable stable segmentations over long image sequences, preventing the seeping out of regions into the background (e.g. the green hair region) or the removal of semantically important small parts (e.g. hands or feet).

### 6.1.2 Related Work

Existing co-segmentation approaches can be loosely grouped into two classes regarding the additional assumptions they make:

**Shape similarity.** One class of methods assumes that the segmentations of each image only differ by a rigid body transformation [139] or — somewhat relaxed — that they are highly similar [107]. Unfortunately, this limits the applicability to rigid objects observed from similar viewpoints. Upon strong viewpoint changes, articulations or non-rigid deformations, the arising segmentations are too different in their shape to be accounted for. Multiple foreground objects cannot be handled, either.

**Color similarity.** On the other extreme are methods which make no assumptions on *shape similarity*, but which strongly rely on *similarity of color or feature distributions* [109, 90, 57, 11, 132, 37]. These methods do not generalize well to certain real-world challenges where firstly object and background may have similar and possibly overlapping color distributions and where secondly these

distributions may vary from image to image due to illumination changes, motion blur or scale-dependent color sampling. Recent approaches relax these assumptions, requiring less color similarity [62] and instead derive high level knowledge on object properties such as object subspaces [91] or region correspondences [110]. However, the resulting optimization problems become more difficult to solve.

To increase the stability of segmentation results, it was suggested [67] to impose shape moment constraints into variational segmentation methods. Since these constraints are *absolute*, they are not invariant to changes in scale, view-point, occlusions or multiple object instances and are thus not applicable to the general co-segmentation problem.

### 6.1.3 Contributions

In this chapter we revisit the problem of image sequence segmentation and reconsider the following question: Can we identify aspects of an object’s *shape* which are shared among the various images of this object and are preserved despite rigid transformation, despite articulation and despite substantial non-rigid deformation?

The central idea is to tackle the problem in a multilabel framework where an object, say an athlete, is made up of multiple components (the various limbs of the body). While the object may undergo substantial changes — rigid body motion, articulation, non-rigid deformation — what is typically preserved is the relative size of object parts (e.g. the head to the entire body), the object part proportions. We formulate image sequence segmentation as a problem of Bayesian inference and introduce *proportion priors* to restrict the relative size of object parts. This approach comes with the following advantages:

- It can handle overlapping color distributions, moderately variable lighting conditions, various object scales and multiple foreground objects. The proposed ratio constraints preserve small or elongated object parts.
- It extends recent convex relaxation techniques [77, 29, 140] described in Chapter 3 from multilabel segmentation to *multilabel sequence segmentation*. We present an efficient optimization scheme which can be parallelized with run times of around one second to compute pixel-accurate segmentations.
- The approach yields state-of-the-art results on the ICoseg benchmark for subdivisible object sequences.

## 6.2 Multilabel Image Sequence Segmentation

We will consider the general multilabel problem with length regularity (3.16) on  $n \geq 1$  labels as the base model and will additionally include a proportion preserving prior into optimization. In order to come up with a suitable prior formulation, let us consider the derivation of the multilabel energy in the framework of *Bayesian inference*.

One can compute a segmentation labeling  $l$  by maximizing the conditional probability

$$\operatorname{argmax}_l \mathcal{P}(l|I) = \operatorname{argmax}_l \mathcal{P}(I|l) \mathcal{P}(l) \quad (6.1)$$

of  $l$  given the input image  $I$ , and applying the Bayes rule (1.3). The right hand side of (6.1) combines the observation probability  $\mathcal{P}(I|l)$  (typically favoring a color-based region association) with prior knowledge  $\mathcal{P}(l)$  regarding what kinds of partitionings are more or less likely.

In the case of image sequence segmentation, commonly used color and boundary length priors are often insufficient to obtain good results. Firstly, the object boundaries are not necessarily short. Secondly, the color distributions of object and background may have substantial overlap and may exhibit strong variations across images. In order to stabilize the segmentation process against color and lighting variations, pose changes and substantial non-rigid deformations, and in order to leverage it to a parsing of objects into their semantic components, we propose to introduce proportion priors into the optimization problem.

**Framework for Proportion Preserving Priors.** In the following, we will introduce proportion priors as a means to impose information on the *relative size* of respective object parts. Whereas the absolute size of parts will vary with viewing angle and distance from the camera, their relative size is typically well preserved — i.e. the size of the head is typically 10% of the size of the entire body.

Let us assume that the object we want to segment can be divided into  $n-1$  subregions (e.g. head, feet, body and hands) with the  $n$ -th region denoting the background of the image. Then in the Bayesian framework (6.1), the prior  $\mathcal{P}(l) = \mathcal{P}(\Omega_1, \dots, \Omega_n)$  can be expressed in the following way:

$$\begin{aligned} \mathcal{P}(l) &= \mathcal{P}(\Omega_1, \dots, \Omega_{n-1} | \Omega_n) \mathcal{P}(\Omega_n) \\ &= \left( \prod_{i=1}^{n-1} \mathcal{P}(\Omega_i | \Omega_n) \right) \mathcal{P}(\Omega_n). \end{aligned} \quad (6.2)$$

Here we assume conditional independence of the segments  $\Omega_1, \dots, \Omega_{n-1}$  given the background  $\Omega_n$ . A ratio constraint relates the size  $|\Omega_i|$  of the  $i$ -th region to the size  $\sum_{j=1}^{n-1} |\Omega_j|$  of the whole object:

$$r_i = \frac{|\Omega_i|}{\sum_{j=1}^{n-1} |\Omega_j|} = \frac{|\Omega_i|}{|\Omega| - |\Omega_n|}, \quad 1 \leq i < n. \quad (6.3)$$

For segmentation we want to impose regions of short (weighted) boundary length  $\operatorname{Per}_g(\Omega_i; \Omega)$ , whose ratios additionally follow a learned (or specified) ratio probability distribution  $\mathcal{P}_{\text{prop}}$

$$\mathcal{P}(\Omega_i | \Omega_n) = \frac{1}{C} \exp \left( -\frac{\lambda}{2} \operatorname{Per}_g(\Omega_i; \Omega) \right) \cdot \mathcal{P}_{\text{prop}}(r_i), \quad 1 \leq i < n, \quad (6.4)$$

where  $C$  is a normalization constant,  $\lambda$  a weighting parameter, and  $g : \Omega \rightarrow \mathbb{R}_{>0}$  is some fixed boundary length weighting function. Finally, the same short



boundary length prior is assumed for the background  $\Omega_n$ :

$$\mathcal{P}(\Omega_n) = \frac{1}{C'} \exp\left(-\frac{\lambda}{2} \text{Per}_g(\Omega_n; \Omega)\right) \quad (6.5)$$

where  $C'$  is some other normalization constant.

Instead of maximizing  $\mathcal{P}(I|l)\mathcal{P}(l)$  in (6.1) we minimize its negative logarithm and obtain the energy

$$\begin{aligned} E(\Omega_1, \dots, \Omega_n, r_1, \dots, r_{n-1}) = \\ \sum_{i=1}^n \left\{ \int_{\Omega_i} c_i(x) dx + \frac{\lambda}{2} \text{Per}(\Omega_i; \Omega) \right\} - \sum_{i=1}^{n-1} \log \mathcal{P}_{\text{prop}}(r_i) \\ \text{s.t. } r_i = \frac{|\Omega_i|}{|\Omega| - |\Omega_n|} \quad 1 \leq i < n, \end{aligned} \quad (6.6)$$

with the data terms  $c_i(x) := -\log \mathcal{P}(I(x)|l(x) = i)$ .

### 6.3 Proportion Preserving Priors

In this section we propose two different proportion preserving priors: the uniform distribution prior and the Laplace distribution prior. Both assume that the ratios of the object parts follow a specific distribution  $\mathcal{P}_{\text{prop}}(r_i)$  whose parameters are estimated from sample data, i.e. sample segmentations from which ratio samples can be obtained.

To convert the energy in (6.6) to a convex optimization problem in Section 6.4.1, the key challenge is to express the terms  $-\log \mathcal{P}_{\text{prop}}(r_i)$  in (6.6) as a convex function of the variables  $a_i := |\Omega_i|/|\Omega|$ , which denote the fraction of the size of region  $\Omega_i$  with respect to the image size. It holds that  $a_i \in [0, 1]$  and  $\sum_{i=1}^n a_i = 1$ . The ratios  $r_i$  in (6.3) can be easily expressed in terms of  $a_i$  by

$$r_i = \frac{a_i}{1 - a_n}. \quad (6.7)$$

#### 6.3.1 Uniform Distribution Prior

As a first case, we assume a uniform distribution of the ratios  $r_i$  over a specific interval  $[l_i, h_i]$ . The left and right boundaries  $l_i$  and  $h_i$  are computed from training data by means of maximum likelihood estimation, which assigns  $l_i$  and  $h_i$  the minimum and maximum values of the sample ratios, respectively. We obtain for the ratio probabilities (or, more precisely, for the probability densities)

$$\mathcal{P}_{\text{prop}}(r_i) = \begin{cases} \frac{1}{h_i - l_i} & \text{if } l_i \leq r_i \leq h_i, \\ 0 & \text{otherwise.} \end{cases} \quad (6.8)$$

Since  $-\log \mathcal{P}_{\text{prop}}$  is either constant or infinity, this prior corresponds to  $2(n-1)$  hard constraints in the optimization problem:

$$l_i(1 - a_n) \leq a_i \leq h_i(1 - a_n) \quad \forall 1 \leq i < n. \quad (6.9)$$

These relative ratio constraints are linear and thus convex in terms of the  $a_i$ . The advantage of this prior is the simple computation and the convexity of the constraints. Yet, in the case of large variations of the ratios  $r_i$  in the sample data (i.e. sample outliers) we can obtain very large intervals  $[l_i, h_i]$ , which hardly limit the resulting segmentations. Therefore, we propose Laplace distribution priors.

### 6.3.2 Laplace Distribution Prior

The Laplace distribution prior penalizes deviations of the ratios  $r_i$  from their median  $\bar{r}_i$ . In this way, the influence of ratio sample outliers on the constraints is limited. We assume the following Laplace distribution

$$\mathcal{P}_{\text{prop}}(r_i) = \frac{1}{2\sigma_i} \exp\left(-\frac{|r_i - \bar{r}_i|}{\sigma_i}\right). \quad (6.10)$$

Given a set of sample ratios  $s_{i1}, \dots, s_{iM}$  for segment  $i$  from  $M$  training segmentations, the parameters  $\bar{r}_i$  are obtained by means of maximum likelihood estimation as

$$\begin{aligned} \bar{r}_i &= \text{median}(s_{i1}, \dots, s_{iM}) \\ \text{and } \sigma_i &= \frac{1}{M} \sum_{j=1}^M |s_{ij} - \bar{r}_i|. \end{aligned} \quad (6.11)$$

In order to have  $\sum_{i=1}^{n-1} \bar{r}_i = 1$  we apply the projection  $\bar{r}_i = \hat{r}_i - \frac{1}{n-1} (\sum_{j=1}^{n-1} \hat{r}_j - 1)$ .

Taking the negative logarithm of (6.10) and multiplying it by a parameter  $\mu > 0$  for balancing the prior with respect to the other terms, we get the energy

$$E_p(r_i) = -\mu \log \mathcal{P}_{\text{prop}}(r_i) = \frac{\mu}{\sigma_i} |r_i - \bar{r}_i| = \frac{\mu}{\sigma_i} \left| \frac{a_i}{1 - a_n} - \bar{r}_i \right|. \quad (6.12)$$

Unfortunately, after replacing  $r_i$  by (6.7), this function is not convex in  $a_i$  and  $a_n$ . For example, for  $r_i < \bar{r}_i$  we obtain  $E_p(r_i) = \frac{\mu}{\sigma_i} (\bar{r}_i - \frac{a_i}{1 - a_n})$ , which is not convex in  $a_n$  for fixed  $a_i$  having the second derivative  $\partial_{a_n}^2 E_p = -\frac{\mu}{\sigma_i} \frac{2a_i}{(1 - a_n)^3} < 0$ . To make global optimization possible, in the following we propose two methods to convexify this prior.

**Convex Relaxation.** First, we consider the convex relaxation of  $E_p$  as a function  $E_p(a_i, a_n)$  of two variables, i.e. the tightest possible convex *lower bound*. The definition domain of  $E_p$  is naturally given by  $a_i, a_n \geq 0$  and  $a_i + a_n \leq 1$ , where the latter inequality follows from  $\sum_{j=1}^n a_j = 1$ . We obtain the following convex relaxation:

**Proposition 6.1.** *The convex relaxation of (6.12) on the domain  $a_i, a_n \geq 0$  and  $a_i + a_n \leq 1$  is given by*

$$E_1(a_i, a_n) := \frac{\mu}{\sigma_i} |a_i - \bar{r}_i(1 - a_n)|. \quad (6.13)$$

*Proof.* See appendix Section 6.7. □

In contrast to the Laplace prior (6.12), this approximation is convex. However,  $E_1$  is minimal not only for  $a_i = \bar{r}_i(1 - a_n)$  but also for  $a_n = 1, a_i = 0$ , i.e. when the segmentation only consists of the background. Thus, this prior is biased towards smaller foreground object areas. This can be understood as an additional compactness prior which removes cluttered background regions, but sometimes also parts of the objects.

**Convex Upper Bound.** As shown in the last paragraph, even the *greatest* convex lower bound  $E_1$  is too small. This suggests to go in the other direction. As a second convexification method we propose a convex *upper bound* on  $E_p$ . Note that  $E_p$  does not have a lowest upper bound, in contrast to the convex relaxation case. To arrive at one possible solution, the idea is to write  $E_p$  in (6.12) by replacing the ratios  $r_i$  by the  $a_i$  in (6.7),

$$E_p = \frac{\mu}{\sigma_i} \frac{|a_i - \bar{r}_i(1 - a_n)|}{\sqrt{1 - a_n}} \cdot \frac{1}{\sqrt{1 - a_n}} \quad (6.14)$$

and apply Young's inequality  $pq \leq \frac{1}{4\varepsilon}p^2 + \varepsilon q^2$ , valid for all  $p, q \in \mathbb{R}$  and arbitrary  $\varepsilon > 0$  (we choose  $\varepsilon = 10$  in the experiments). It is equivalent to  $(\frac{1}{2\sqrt{\varepsilon}}p - \sqrt{\varepsilon}q)^2 \geq 0$ . This leads to  $E_p \leq E_2$  with

$$E_2 := \frac{\mu^2}{4\varepsilon\sigma_i^2} \frac{(a_i - \bar{r}_i(1 - a_n))^2}{1 - a_n} + \frac{\varepsilon}{1 - a_n}. \quad (6.15)$$

This energy is convex: the first addend is a linear transformation of the convex function  $(x, y) \mapsto \frac{x^2}{y}$ ,  $x \in \mathbb{R}, y > 0$ , and the second one is obviously convex in  $a_n \in [0, 1)$ .

The main advantage is that  $E_2$  always favors the relation  $a_i \approx \bar{r}_i(1 - a_n)$  regardless of the size of the object. It also avoids large background regions as  $E_2 \rightarrow \infty$  for  $a_n \rightarrow 1$ , which can be alleviated by using a small  $\varepsilon$ .

We note that a Gaussian distribution instead of a Laplace distribution (6.10) would still yield a nonconvex  $E_p$  in (6.12), while leading to more complex and slower relaxations. In addition, we observed that the Laplace prior better represents the training data than the Gaussian prior.

## 6.4 Implementation

### 6.4.1 Conversion to a Convex Optimization Problem

As described in Section 3.3, the idea to convexify the multilabel problem is to express it in terms of the label indicator functions  $u_1, \dots, u_n$  in (3.10). The energy (6.6) can now be rewritten in a convex way in terms of  $u$  as follows:

First, the (weighted) boundary length  $\text{Per}_g(\Omega_i; \Omega)$  is given by the weighted total variation of  $u_i$  through (2.28):

$$\frac{\lambda}{2} \text{Per}_g(\Omega_i) = \frac{\lambda}{2} \int_{\Omega} g \, d|Du_i| = \sup_{p_i \in \mathcal{C}_i} \int_{\Omega} \langle p_i, dDu_i \rangle \quad (6.16)$$

with

$$\mathcal{C}_i = \left\{ p_i \in C_c^1(\Omega; \mathbb{R}^m) \mid |p_i(x)| \leq \frac{\lambda}{2} g(x) \quad \forall x \in \Omega \right\}. \quad (6.17)$$

The weighting function  $g$  for the length penalization is set to

$$g(x) := e^{-\gamma|\nabla I(x)|}, \quad (6.18)$$

favoring the coincidence of object and image edges. We set  $\gamma = 5$ .

Next, the variables  $a_i = |\Omega_i|/|\Omega|$  used to replace  $r_i$  in (6.7) can be written in terms of the indicator functions  $u_i$  as

$$a_i = \frac{1}{|\Omega|} \int_{\Omega} u_i(x) \, dx. \quad (6.19)$$

These are linear and thus convex constraints in  $a$  and  $u$ .

The final step is to relax the binary constraints  $u_i(x) \in \{0, 1\}$  to the convex ones  $u_i(x) \in [0, 1]$  as was done in Section 3.3. The domain of  $u$  becomes the set  $\mathcal{D}$  in (3.18).

**Convex Energy.** We obtain the following energy

$$\min_{\substack{u \in \mathcal{D}, a \\ \text{s.t. (6.19)}}} \sup_{p_i \in \mathcal{C}_i} \sum_{i=1}^n \int_{\Omega} \left( c_i u_i \, dx + \langle p_i, dDu \rangle \right) - \sum_{i=1}^{n-1} \log \mathcal{P}_{\text{prop}}(a_i, a_n) \quad (6.20)$$

which is to be minimized w.r.t.  $u$  and  $a$  and maximized w.r.t.  $\xi$ . For the uniform prior the constraints (6.9) replace the second term. For the Laplace convex relaxation prior it is given by (6.13), and for the Laplace convex upper bound prior by (6.15). The relaxed convex optimization problem can be minimized globally yielding results which are independent of the initialization.

In order to obtain a binary solution candidate for the original optimization problem, after solving the relaxed problem we use the maximum-value binarization (3.41).

Note that optimizing for the proportions by alternating minimization is not reasonable: Starting with an initial segmentation, the proportions will be first set in accordance with it, so that each subsequently determined segmentation will be driven towards this first (and possibly bad) one.

## 6.4.2 Prior Implementation

After discretizing the image domain into a rectangular pixel grid, the energy (6.20) is discretized in a straightforward way analogous to (3.5) and optimized with the primal-dual Algorithm 3.

The constraints (6.19) become

$$a_i = \frac{1}{|\Omega|} \sum_{x \in \Omega} u_i(x) \quad \forall i, \quad (6.21)$$

where  $|\Omega| = (\sum_{x \in \Omega} 1)$  is the number of pixels overall. They can be easily enforced through the basic Lagrange multiplier dualization (2.71), appending

the terms

$$\sup_{\eta_i \in \mathbb{R}} \sum_{i=1}^n \eta_i \left( a_i - \frac{1}{|\Omega|} \sum_{x \in \Omega} u_i(x) \right) \quad (6.22)$$

to the energy. For the projection of the primal variable  $u$  onto the constraint set  $\mathcal{D}_d$  in (3.45), we use the explicit simplex-projection method because the number of labels is typically rather very small. The projection of each dual  $p_i \in \mathcal{C}_i$  is done by simple clipping of the absolute value.

Let us now give more details about the implementation of the different proportion prior terms.

**Uniform Prior.** Similarly to (6.22), the hard constraints (6.9) for the uniform prior are also implemented through Lagrange multiplier dualizations, but utilizing (2.72) and (2.73). We append the terms

$$\sup_{\alpha_i \in \mathbb{R}, \alpha_i \leq 0} \sum_{i=1}^n \alpha_i (a_i - l_i(1 - a_n)) \quad (6.23)$$

and

$$\sup_{\beta_i \in \mathbb{R}, \beta_i \geq 0} \sum_{i=1}^n \beta_i (a_i - h_i(1 - a_n)) \quad (6.24)$$

to the energy. The proximal operators for  $\alpha$  and  $\beta$  are straightforward to compute, being essentially the projections onto the nonpositive, respectively the nonnegative numbers.

**Laplace Convex Relaxation.** For the convex relaxation (6.13) of the Laplace prior, in order to simplify the resulting proximal operators it is convenient to decouple the  $a_1, \dots, a_{n-1}$  from  $a_n$  by applying the dualization (2.77):

$$E_1 = \frac{\mu}{\sigma_i} |a_i - \bar{r}_i(1 - a_n)| = \sup_{\alpha_i \in \mathbb{R}, |\alpha_i| \leq \mu/\sigma} \alpha_i (a_i - \bar{r}_i(1 - a_n)). \quad (6.25)$$

This introduces  $n$  new dual variables  $\alpha_i \in \mathbb{R}$ ,  $1 \leq i < n$ , into the optimization. The prox for  $\alpha$  is essentially a projection onto the hard constraints, which is a simple absolute value clipping.

**Laplace Upper Bound.** For the Laplace convex upper bound prior (6.15), it is also advisory to decouple the  $a_i$ s to simplify the computations. We apply the dual representation (2.79) to (6.15) for all  $1 \leq i < n$  and obtain the following formulation of the convex upper bound for the Laplace proportion preserving prior:

$$\sup_{(\alpha_i, \beta_i) \in A_i} \sum_{i=1}^{n-1} \left( \alpha_i (a_i - \bar{r}_i(1 - a_n)) - \beta_i (1 - a_n) \right) + \frac{\varepsilon(n-1)}{1 - a_n}. \quad (6.26)$$

with the constraint set

$$A_i := \left\{ (\alpha, \beta) \in \mathbb{R} \times \mathbb{R} \mid \beta_i \geq \frac{\varepsilon \sigma_i^2}{\mu^2} \alpha_i^2 \right\}. \quad (6.27)$$

The overall energy for this prior is

$$\begin{aligned}
\min_{u \in \mathcal{D}_d, a} \max_{p_i \in \mathcal{C}_i, (\alpha_i, \beta_i) \in A_i} & \sum_{x \in \Omega} \sum_{i=1}^n c_i(x) u(x) + \langle p_i(x), \nabla^+ u_i(x) \rangle \\
& + \sum_{i=1}^n \eta_i \left( a_i - \frac{1}{|\Omega|} \sum_{x \in \Omega} u_i(x) \right) \\
& + \sum_{i=1}^{n-1} \left( \alpha_i (a_i - \bar{r}_i (1 - a_n)) - \beta_i (1 - a_n) \right) + \frac{\varepsilon(n-1)}{1 - a_n}.
\end{aligned} \tag{6.28}$$

After a linear adjustment from the linear terms  $-\sum_{i=1}^{n-1} (\alpha_i \bar{r}_i + \beta_i)$  analogously to (3.49), the prox for  $(\alpha, \beta)$  reduces to a projection onto  $A$ . The projection of  $\alpha_i, \beta_i$  onto their individual constraints sets, as well as the proximal operator for  $a_n$  are detailed in the appendix Section 6.7.

## 6.5 Experimental Results

We have developed an approach for image sequence segmentation which preserves object proportions by imposing relative size priors on different object components. In this way we allow for arbitrary scaling of the objects, spatially varying color distributions and the recovery of easily missed object components. To evaluate the proposed prior we selected several different image series from the ICoseg-database [11] and the web, requiring that the foreground objects can be subdivided into several parts, see Figure 6.3 and 6.4.

The image series in the ICoseg dataset consist of up to 40 images in each class. To obtain a small set of proportion samples and separate color models for each object part, we created a small training set for each class consisting of five images partitioned into object part labels. The color models  $c_i$  in (6.6) for each part can then either be learned from these images or derived from user scribbles on one or several of them, e.g. by means of the Parzen density estimator [97, 93]. We used the CIELab color space and a Gaussian kernel with standard deviation 8. The remaining images of the series are then segmented automatically. For most image sequences the parameters  $\lambda = 15$  and  $\mu = 200$  gave optimal results.

### 6.5.1 Benchmark Results

In Table 6.1 we compare the average accuracy (ratio of correctly labeled pixels with respect to total number of pixels) of the proposed proportion priors on the sequences of the ICoseg benchmark, which contain subdivisible objects. First, we compare the proportion prior results to the results of the same algorithm without proportion priors. The table shows drastic improvements of up to 40 or 50%.

We also compare against state-of-the-art image sequence and co-segmentation approaches. Almost all of these approaches are at least partially supervised just as ours since they use training data from a small set of images or scribbles to learn common object properties or classifiers, e.g. Mukherjee et al. [91] learn dictionaries of appearance models, Vicente et al. [132] train a random forest

Dataset	w/o prop.	uniform	relaxed L.	bounded L.	[132]	[62]	[110]	[91]	[37]	[11]
Gymnastics	55.13	98.35	97.86	98.51	91.7	90.9	87.1	92.18	-	-
Ferrari	49.71	97.83	97.72	98.19	89.9	85.0	84.3	89.95	96.4	-
Balloon	83.53	96.46	96.74	96.73	90.1	85.2	89.0	95.17	-	-
KitePanda	78.77	82.41	89.69	92.09	90.2	73.2	78.3	93.37	-	-
Baseball	63.01	95.85	96.20	95.01	90.9	73.0	90.5	95.66	-	-
Skating	93.72	96.42	96.55	96.60	77.5	82.1	76.8	96.64	-	-
Liverpool	70.22	95.44	95.55	95.50	87.5	76.4	82.6	-	93.9	-
Ice-skating	56.05	99.33	99.41	99.41	-	-	-	-	-	-
Mean	68.77	95.26	96.22	<b>96.51</b>	88.26	80.83	85.7	93.83	93.14	92.8

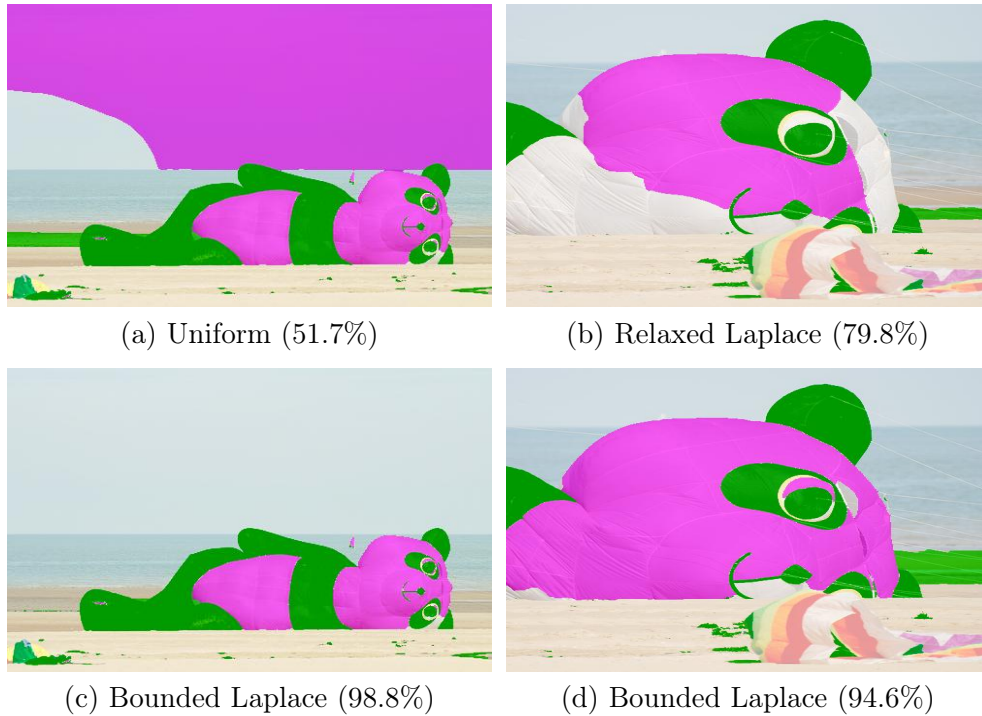
**Table 6.1: Proportion prior accuracy.** Comparison of the accuracy (in %) of the proposed proportion prior formulations (uniform, relaxed Laplace, bounded Laplace prior) to the same algorithm without proportion priors and to state-of-the-art segmentation results on the ICoseg benchmark. The upper bound formulation of the Laplace distribution performs best.

based on appearance model features, [Batra et al. \[11\]](#) learn Gaussian Mixture Models for foreground and background from user scribbles, and [Joulin et al. \[62\]](#) train a supervised classifier for object class separation. Others such as [Rubio et al. \[110\]](#) and [Collins et al. \[37\]](#) do not make use of prior knowledge. However, the former builds on an algorithm to measure ‘objectness’ that again requires learning on general images. Not all of these methods indicate accuracy per sequence or not for all sequences we tested. Missing results are marked by ‘-’. [Collins et al. \[37\]](#) evaluated only on two sequences of our test set, so we indicate the average score reported in [\[37\]](#), and [Batra et al. \[11\]](#) only gave the average accuracy on the whole benchmark. The results show that we outperform all other methods on the given test set.

Finally, we compare the different types of proportion prior formulations we proposed in [Section 6.3](#). The results show that all three types outperform the other approaches. However, two points need to be mentioned: 1) The uniform prior ([6.9](#)) is too weak in case of outliers since all ratios within the maximum and minimum sample range are equally likely. This can be observed in the kite panda series in the benchmark, where the accuracy drops to 82% since most images contain only parts of the kite with strongly varying proportions (for most other scenes the object is usually fully visible). 2) The relaxed Laplace prior ([6.13](#)) is biased towards large background regions and thus comes with a shrinking bias which tries to minimize the area of the whole foreground object, see ([6.13](#)) and the remark thereafter. [Figure 6.2](#) illustrates both points. Hence, we can conclude that the convex upper bound Laplace prior ([6.15](#)) yields the best and most stable performance over all test sequences.

## 6.5.2 Qualitative Results

In this section we show qualitative results for the proportion prior algorithm on the ICoseg dataset, for single objects in [Figure 6.3](#) and for multiple foreground objects in [Figure 6.4](#). For each series, we show the result of the proposed segmentation algorithm without proportion priors in the left column and with proportion priors based on the convex upper bound Laplace relaxation ([6.15](#)) in the right column. From the results we can draw four conclusions.

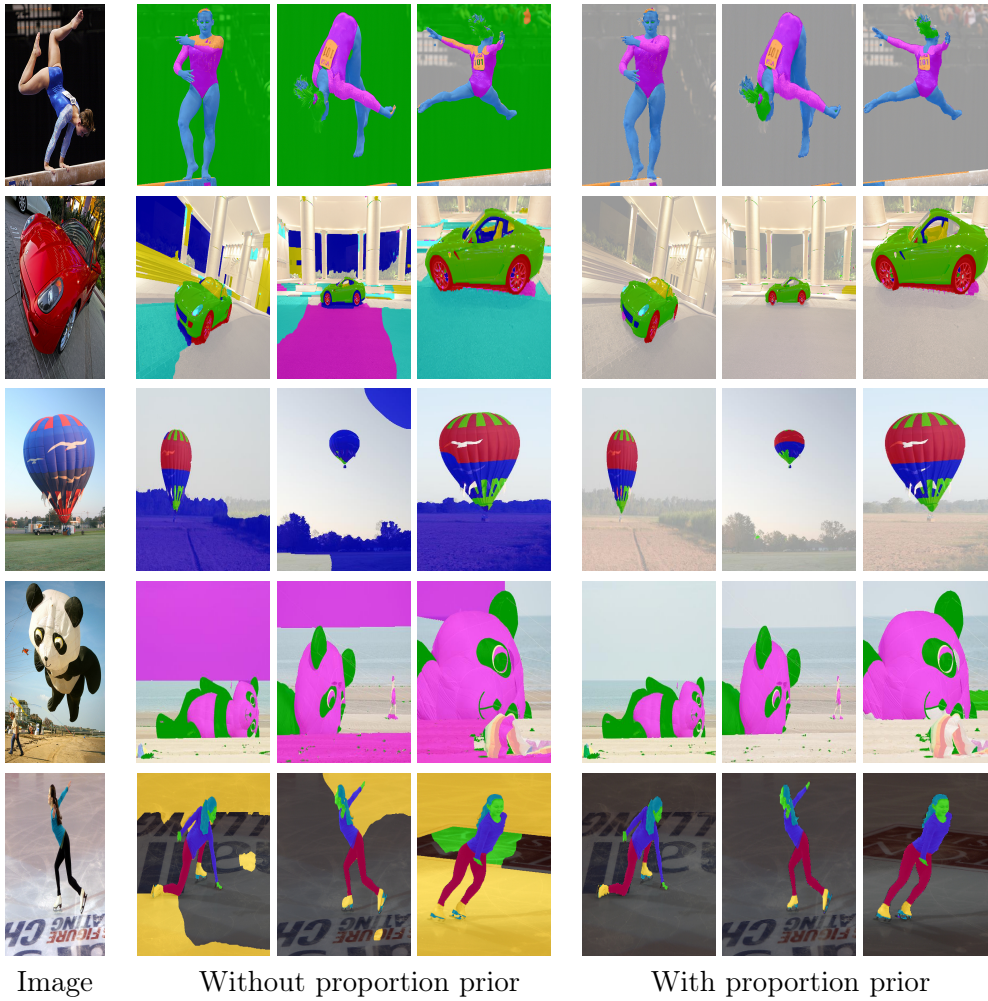


**Figure 6.2: Proportion prior comparison.** (a) The uniform prior (6.9) is weakened in case of strongly varying sample proportions (the panda kite is often only partially visible in the training images). (b) The Laplace convex relaxation prior (6.13) is biased towards larger backgrounds and thus comes with a shrinking bias, which sometimes yields suboptimal results. (c–d) The Laplace convex upper bound prior (6.15) yields stable results.

**Higher Accuracy.** The imposed proportion prior yields segmentation results of much higher accuracy than the original approach without size constraints. In several of the series in Figure 6.3 it is a common problem for object regions to seep out into the background and cover large parts of it, e.g. the background of the gymnast series is sometimes completely assigned to the ‘hair’ label, or the ice background is marked as ‘skates’ in the figure-skating series. Larger parts of the car background are assigned as various car elements as well, and the sky in the panda kite series is often misclassified as kite. Without proportion priors, further problems appear with different regions of similar color, which are easily mixed up due to low energy differences. An example are the shoulders of the gymnast in the leftmost image of Figure 6.3, which are marked as ‘101-sign’.

**Scale Invariance.** Depending on the viewpoint of the camera, the objects have different sizes. Take for example the balloon or the car in Figure 6.3, which appear at different distances to the camera. Absolute size constraints would not allow for a correct segmentation in these cases without changing the size constraints for each image, which would be very tedious for the user. In contrast, relative proportion constraints relating different parts of the object, are naturally invariant with respect to object scale.

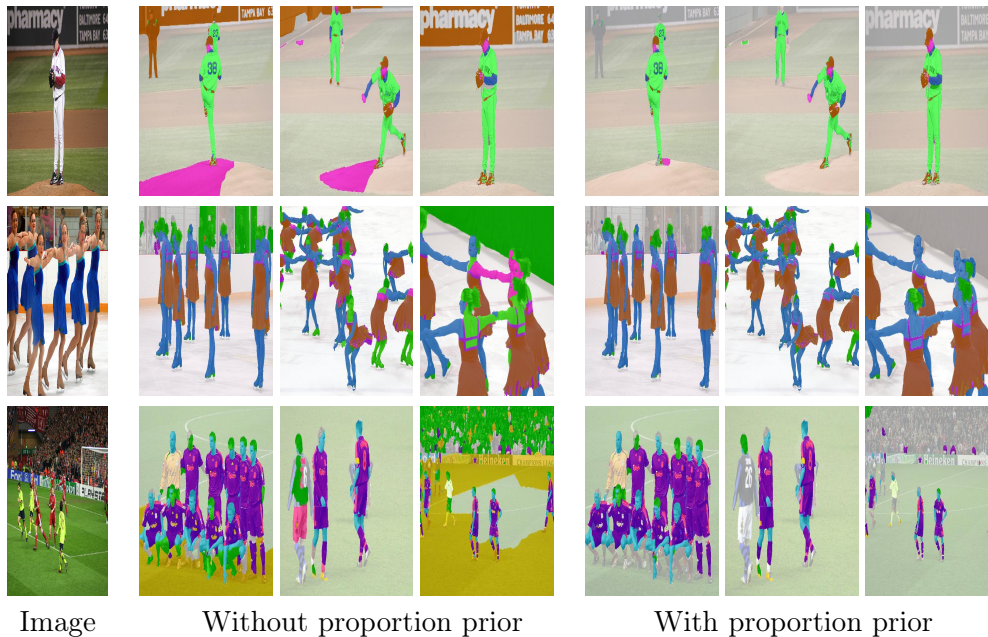




**Figure 6.3: Segmentation results for different image sequences containing a single object.** The left three columns of each sequence show the segmentation results without proportion prior, the right three columns after imposing proportion priors.

**Preservation of Small or Elongated Object Parts.** Another advantage of the proposed approach is that it preserves semantically meaningful small-scale objects. If these parts are either similar to the background in color or small or elongated, the algorithm without size constraints will remove these parts to minimize the segment boundary length. Examples can be seen in the first and third image of the ice-skater series in Figure 6.3, where the shoes are assigned to the background. By defining proportion constraints for each object part separately, we can learn and impose ratio likelihoods, which prevent such regions from disappearing.

**Multiple Object Instances.** Most segmentation algorithms are limited to single foreground objects. Since the proportion priors within the multilabel framework naturally apply to multiple object instances in the same image we



**Figure 6.4: Segmentation results for different image sequences containing multiple objects.** See Figure 6.3 for the explanation.

can also apply our approach to image series with many foreground objects such as baseball or soccer scenarios, see Figure 6.4.

In addition, proportion priors are not limited to different parts of the same object but can also relate different objects, e.g. the size of a mouse with respect to a dog or the heart with respect to the lungs in medical imaging.

### 6.5.3 Runtime

An important advantage of the proposed method is its efficiency, especially for large sequences of images. Previous and current state-of-the-art approaches demand run times per image of 45 seconds [62], 10 seconds [37] or 25–100 seconds [91]. Due to the inherent parallel structure of the algorithm, each pixel can be updated independently. Hence, the proposed method can be easily parallelized. Using a single NVIDIA GTX 680 GPU we obtained an average run time of 2.2 seconds per image. Using three GPUs in parallel we reduced the average run time to one second.

## 6.6 Conclusion

We introduced the concepts of part decompositions and proportion priors into a framework for multilabel co-segmentation. The problem is formulated as a variational approach together with a convex relaxation which can be globally optimized. Extensive evaluations on various image series show that proportion priors provide accurate segmentations despite changing illumination, despite viewpoint or background changes, and despite substantial articulations and non-

rigid deformations. The relativity of the proportion constraints allows for stable, scale-invariant segmentations over long ranges of images containing single or multiple foreground objects and helps to recover small-scale semantically important object parts which are frequently suppressed in existing approaches. With an average run time of about one second on graphics hardware the algorithm is more than an order of magnitude faster than existing techniques.

## 6.7 Appendix: Proofs and Proximal Operators

### 6.7.1 Proof of Proposition 6.1

*Proof.* W.l.o.g. let  $\frac{\mu}{\sigma_i} = 1$ . First,  $E_1$  is a convex lower bound on  $E_p$  in (6.12) since it is convex with  $E_1 = E_p \cdot (1 - a_n) \leq E_p$ . For any other such bound  $\widehat{E}_1$ , by  $\widehat{E}_1 \leq E_p$  it follows

$$\widehat{E}_1(0, a_n) \leq \bar{r}_i, \quad (6.29)$$

$$\widehat{E}_1(\bar{r}_i(1 - a_n), a_n) \leq 0. \quad (6.30)$$

From this,  $\widehat{E}_1(0, 0) \leq \bar{r}_i$  and  $\widehat{E}_1(0, 1) \leq 0$ , and therefore

$$\begin{aligned} \widehat{E}_1(0, a_n) &\leq a_n \widehat{E}_1(0, 1) + (1 - a_n) \widehat{E}_1(0, 0) \\ &\leq \bar{r}_i(1 - a_n). \end{aligned} \quad (6.31)$$

For  $a_i \leq \bar{r}_i(1 - a_n)$  we can define  $\alpha := \frac{a_i}{\bar{r}_i(1 - a_n)} \in [0, 1]$ . By convexity of  $\widehat{E}_1$ , and from (6.31) and (6.30) we get

$$\begin{aligned} \widehat{E}_1(a_i, a_n) &= \widehat{E}_1((1 - \alpha) \cdot 0 + \alpha \cdot \bar{r}_i(1 - a_n), a_n) \\ &\leq (1 - \alpha) \widehat{E}_1(0, a_n) + \alpha \widehat{E}_1(\bar{r}_i(1 - a_n), a_n) \\ &\leq (1 - \alpha) \cdot \bar{r}_i(1 - a_n) + \alpha \cdot 0 \\ &= \bar{r}_i(1 - a_n) - a_i = E_1(a_i, a_n). \end{aligned}$$

Similarly, one can show  $\widehat{E}_1 \leq E_1$  also for  $a_i \geq \bar{r}_i(1 - a_n)$ . Thus,  $E_1$  is the greatest convex lower bound on  $E_p$ .  $\square$

### 6.7.2 Proximal Operator for $a_n$

For the primal-dual algorithm, the proximal operator for  $a_n$  is the minimization problem

$$\operatorname{argmin}_{a_n} \left\{ \frac{(a_n - a_n^0)^2}{2\tau} + \frac{\varepsilon(n-1)}{1 - a_n} \right\}, \quad (6.32)$$

where  $a_n^0 \in \mathbb{R}$  and  $\tau > 0$  are constants. Setting the derivative w.r.t.  $a_n$  to zero, one has to solve a cubic equation. We use the method of [83] for this. Define  $c := \tau\varepsilon(n-1)$ ,  $v := \frac{1 - a_n^0}{3}$ ,  $w := v^3$  and  $D := \frac{c}{4} + w$ .

**The case  $D \geq 0$ .** In this case the solution is given by

$$a_n = 1 - v - z - \frac{v^2}{z} \quad (6.33)$$

with  $z := \sqrt[3]{\frac{c}{2} + w + \sqrt{cD}} > 0$ .

**The case  $D < 0$ .** Otherwise, the solution is

$$a_n = 1 - v + 2v \cos\left(\frac{1}{3} \arccos\left(1 - \frac{2D}{w}\right)\right). \quad (6.34)$$

### 6.7.3 Proximal Operator for $\alpha, \beta$

The proximal operator is here computed independently for each  $1 \leq i < n$ :

$$\operatorname{argmin}_{(\alpha_i, \beta_i) \in A_i} \frac{(\alpha - \alpha_i^0)^2}{2\tau} + \frac{(\beta - \beta_i^0)^2}{2\tau} + (-\alpha_i \bar{r}_i - \beta_i) \quad (6.35)$$

for some  $\alpha_i^0, \beta_i^0 \in \mathbb{R}$  and  $\tau > 0$  with the set  $A_i$  in (6.27). Define  $\hat{\alpha}_i := \alpha_i^0 + \tau \bar{r}_i$  and  $\hat{\beta}_i := \beta_i^0 + \tau$ . Then the solution is given by the projection onto a parabola:

$$(\alpha_i, \beta_i) = \pi_{\beta_i \geq \hat{\varepsilon}_i \alpha_i^2}(\hat{\alpha}_i, \hat{\beta}_i) \quad (6.36)$$

with  $\hat{\varepsilon}_i := \frac{\varepsilon \sigma_i^2}{\mu^2}$ . We use the explicit formula of Section 9.9.2 to compute this.

Part II

Vectorial Functionals



## Chapter 7

# Functional Lifting: Convex Relaxation for Scalar Problems

In this second part, consisting of Chapters 7, 8, 9 and 10, we will consider minimizing nonconvex energies

$$\min_u E(u) \tag{7.1}$$

for functions  $u$  of the form

$$u : \Omega \rightarrow \mathbb{R}^k, \quad k \geq 1, \tag{7.2}$$

which are *vector-valued* and have a *continuous range* space. This is opposed to the first part, which dealt with the multilabeling problem and where the range was discrete and finite.

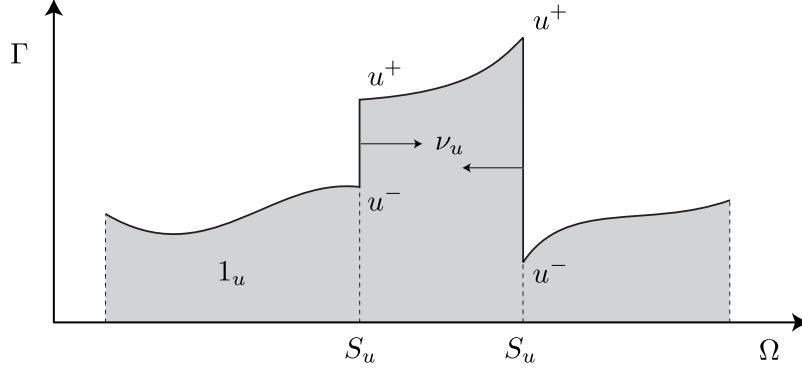
In the current Chapter 7 we will briefly describe the basic and important *functional lifting* approach. It is devised for the *scalar case*  $k = 1$  only and yields a general method to convexify and efficiently optimize energies  $E$ . In the subsequent three chapters we will use the idea of functional lifting to devise novel convex relaxations for the *vectorial case*,  $k \geq 2$ . Each chapter concentrates on a different special case of vectorial functionals. Chapter 8 considers separable regularizers, without any coupling between the channels except for the data term. Chapter 9 considers special coupling regularizers such as the  $l^2$ -coupled total variation and its Huber-regularized variant. Finally, in Chapter 10 we will consider the problem of convexifying the Mumford-Shah regularizer, which is a special and very important instance of vectorial and coupling regularizers.

Let us first review and explain the general idea of the state-of-the-art convex relaxation for the scalar case  $k = 1$ , the functional lifting approach.

### 7.1 Introduction

#### 7.1.1 The Class of Functionals

Recall that scalar functions  $u \in \text{SBV}(\Omega, \mathbb{R})$ , or rather their distributional gradient  $Du$ , can be decomposed as into a smooth and a jump part as in (2.23). We



**Figure 7.1: Graph of a function  $u \in \text{SBV}(\Omega, \mathbb{R})$ .** A special function of bounded variation  $u$  has an approximate gradient  $\nabla u$  everywhere except for a nullset  $S_u$ , where the values jump from  $u^-$  to  $u^+$  in direction  $\nu_u$ . The graph function  $1_u$  is defined as 1 in the shaded area under the graph and 0 otherwise.

are dealing here with the scalar case  $k = 1$ , so that this decomposition becomes

$$Du = \nabla u \mathcal{L}^m + \nu_u(u^+ - u^-) \mathcal{H}^{m-1} \llcorner S_u. \quad (7.3)$$

The set  $S_u$  is the  $(m - 1)$ -dimensional jump set of  $u$  (a hypersurface), where it assumes two distinct values  $u^-, u^+ \in \mathbb{R}$  with  $u^- < u^+$  on the two sides of the jump interface, and  $\nu \in \mathbb{S}^{m-1}$  is the direction of the jump from the  $u^-$  to the  $u^+$  side, see Figure 7.1. The function  $\nabla u$  is not the classical, but rather the “approximate gradient” which is defined a.e. through an appropriate averaging limit, see Section 2.2.

The functional lifting framework is applicable to functionals of the following form:

$$E(u) = \int_{\Omega \setminus S_u} h(x, u(x), \nabla u(x)) \, dx + \int_{S_u} d(x, u^-(x), u^+(x)) \, d\mathcal{H}^{m-1}(x). \quad (7.4)$$

These energies explicitly use the fact that  $u$  can be decomposed into a smooth and a jump part and assign different penalizations to each parts. The function  $h$  usually has a special form, namely the sum of a data term  $c$  only depending on  $t = u$ , and a pure regularizer part  $f$  only depending on  $p = \nabla u$ :

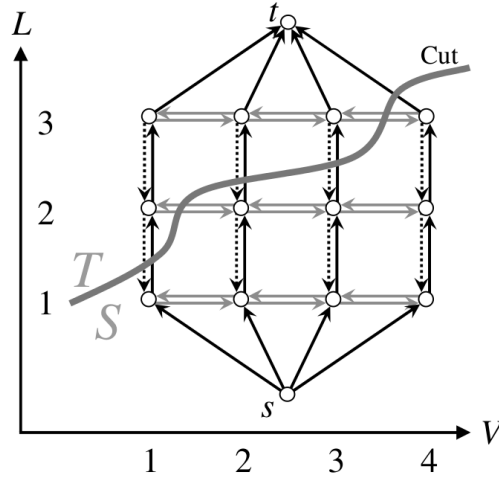
$$h(x, t, p) = c(x, t) + f(x, p). \quad (7.5)$$

For instance, this will be the case in all of our application in this thesis. The above energy  $E$  then becomes

$$E(u) = \int_{\Omega} c(x, u(x)) \, dx + \int_{\Omega \setminus S_u} f(x, \nabla u(x)) \, dx + \int_{S_u} d(x, u^-(x), u^+(x)) \, d\mathcal{H}^{m-1}(x). \quad (7.6)$$

Note that the data term integral is taken over whole  $\Omega$  instead of  $\Omega \setminus S_u$ , which we can do since  $u(x)$  is defined a.e.; we could have done so already in (7.4), but when dealing with  $\nabla u$  it is preferable to keep  $\Omega \setminus S_u$  for clarity.





**Figure 7.2: Graph construction for the Ishikawa approach [60].** The graph contains as many copies of the image grid (indexed by  $V$ ) as there are labels  $L$  in order to represent any convex penalization function. This is related to range discretization in the functional lifting approach.

Functionals (7.6) form a very broad class. Minimizers are allowed to have jump discontinuities and one can explicitly control the jump behavior through the function  $d$ . Undesired discontinuities can be discouraged by making  $d$  large, while still being able to have a general behavior in the smoothness region  $\Omega \setminus S_u$  between the jumps by choosing  $f$  appropriately. For example, choosing  $f$  to be quadratic ensures smoothness between jumps, while making  $f$  large overall will lead to almost piecewise constant solutions. The key advantage of functionals (7.6) is that they allow almost *arbitrarily complicated* and *possibly nonconvex* data terms  $c$ . This generality (resp. the multi-channel variant of (7.6)) encompasses many frequently arising problems of computer vision, including optical flow, image denoising and 3D reconstruction.

However, there are also some limitations. For one,  $E$  depends *pointwise* on  $u$ , i.e. at each image point  $x \in \Omega$  the penalization functions  $c$ ,  $f$ ,  $d$  depend on the values of  $u(x)$ ,  $\nabla u(x)$ , and  $u^\pm(x)$  *only at this point*. This means that data terms involving *convolutions*, such as in deconvolution and deblurring problems [43, 35, 136, 117] and superresolution [9, 25, 117, 130], as well as nonlocal variants of regularization such as nonlocal total variation [73, 137], are not covered by (7.6). Another one is that (7.6) is of first-order, i.e. only depends on the gradient and not higher-order derivatives.

## 7.2 Functional Lifting

The energy  $E$  in (7.6) is highly nonconvex in general. This is due to the possibly nonconvex data term  $c$ , as well as the possibly nonconvex regularization depending on the jump set  $S_u$  (though there are special cases with convex regularization, see Section 7.2.2). Consequently, minimization of such functionals is a challenge and local methods are likely to fail and will need an appropriate

initialization.

Surprisingly, provided suitable regularity properties are fulfilled, for instance convexity of  $f$  in  $\nabla u$ , it turns out that functionals of the form (7.6) can always be represented in a convex way via functional lifting. The key idea is to consider the space of values as an additional dimension and to show that the optimal solution is a minimal hypersurface, i.e. a co-dimension-one structure in this higher-dimensional space. Namely, the energy is reformulated in terms of the *graph function* of  $u$  and one considers Cartesian currents associated with the graph [1, 46, 47, 51].

This provides a general approach to make efficient minimization possible and allows to compute optimal [102, 101] or near-optimal [1, 100, 30] solutions. The important contribution of the works [61, 102, 101] is to properly introduce the interaction terms  $f$  and  $d$  in this program and to suggest a practical implementation.

The main advantage of the lifting approach is that it allows to use *non-convex data terms* and also, to some extent, *nonconvex regularizers* within a convex optimization framework. On the other side, such a general convexification method does not come “free of charge”: The problem dimensionality is increased by one and for the implementation one needs to *discretize the range* of  $u$ . In dimension  $m = 0$  (an image consisting of a single pixel), it is like replacing the problem  $\min_{u \in [0,1]} c(u)$  with  $\min_{1 \leq i \leq n} c(u_i)$ ,  $u_i = \frac{i}{n}$ . It is therefore quite memory intensive and requires suitable hardware to be computationally tractable.

Functional lifting is related to the Ishikawa approach [60] in the discrete setting, which uses a graph construction consisting of as many copies of the image pixel grid as there are labels, see Figure 7.2. Functional lifting is more general, as it also allows for non-convex regularizers and a continuous range of functions.

### 7.2.1 General Functionals

We will first present here the convexification for the general case. Afterwards, in Section 7.2.2 we will consider an important specialization of the approach.

**Graph Functions.** Let us assume that the range of  $u$  is contained in some bounded open interval  $\Gamma = (t_{\min}, t_{\max}) \subset \mathbb{R}$ , i.e.  $u : \Omega \rightarrow \Gamma$ . This will always be the case in practice and so does not pose a real restriction.

The idea is to restate  $E$  in terms of the *graph function*  $1_u : \Omega \times \Gamma \rightarrow \mathbb{R}$  of  $u$ , defined by

$$1_u(x, t) = \begin{cases} 1 & \text{if } t < u(x), \\ 0 & \text{else.} \end{cases} \quad (7.7)$$

This is the characteristic function of the *subgraph* of  $u$ , being equal to 1 below the graph, and to 0 above it, see Figure 7.1. Note that  $1_u$  is defined on the higher-dimensional space  $\Omega \times \Gamma$ , i.e. the range space  $\Gamma$  of  $u$  becomes part of the domain  $\Omega \times \Gamma$  of  $1_u$ . We could of course equally well regard it as a function  $1_u : \Omega \times \Gamma \rightarrow \{0, 1\}$  with the smaller range  $\{0, 1\}$  instead of  $\mathbb{R}$ . For fixed  $x$ ,

$1_u(x, t)$  is a nonincreasing function of  $t$ , starting with value 1 for small  $t$  and jumping to 0 at  $t = u(x)$ . It holds  $1_u \in \text{BV}(\Omega \times \Gamma, \mathbb{R})$  for  $u \in \text{SBV}(\Omega, \Gamma)$ .

**The Lifting Lemma.** The following lemma [1, Lemma 3.7] gives a convex reformulation of  $E$  in terms of the graph function:

**Lemma 7.1** (Functional Lifting). *For functions  $v \in \text{BV}(\Omega \times \Gamma, \mathbb{R})$  define*

$$\mathcal{E}(v) = \sup_{\varphi \in \mathcal{C}} \int_{\Omega \times \Gamma} \langle \varphi, dDv \rangle \quad (7.8)$$

where the set  $\mathcal{C}$  is defined as

$$\begin{aligned} \mathcal{C} = \left\{ \varphi = (\varphi^x, \varphi^t) \in C_c^0(\Omega \times \Gamma; \mathbb{R}^m \times \mathbb{R}) \mid \right. \\ \left. \begin{aligned} &\varphi^t(x, t) \geq -c(x, t) + f^*(x, t, \varphi^x(x, t)) \quad \forall x \in \Omega, t \in \Gamma, \\ &\left| \int_t^{t'} \varphi^x(x, s) ds \right| \leq d(x, t, t') \quad \forall x \in \Omega, t, t' \in \Gamma \end{aligned} \right\}. \end{aligned} \quad (7.9)$$

Then, for all  $u \in \text{SBV}(\Omega, \Gamma)$ ,

$$\mathcal{E}(1_u) = E(u). \quad (7.10)$$

Here,  $f^*(x, t, q) = \sup_{p \in \mathbb{R}^k} \langle p, q \rangle - f(x, t, p)$  is the dual of  $f(x, t, p)$  w.r.t.  $p$ . In the set  $C_c^0(\Omega \times \Gamma; \mathbb{R}^m \times \mathbb{R})$  the subscript  $c$  indicates that  $\varphi$  must have compact support, but only w.r.t. the spatial variable  $x$ . Note that the dual variable  $\varphi$  has two components, one vectorial  $\varphi^x$  with values in  $\mathbb{R}^m$  and one scalar  $\varphi^t$ , and the integral (7.8) can be written as

$$\int_{\Omega \times \Gamma} \langle \varphi, dDv \rangle = \int_{\Omega \times \Gamma} \langle \varphi^x, dD_x v \rangle + \varphi^t dD_t v, \quad (7.11)$$

where  $D = (D_x, D_t)$ , with distributional gradients  $D_x$  and  $D_t$  w.r.t.  $x$  and  $t$ .

The functional  $\mathcal{E}$  defined by (7.8) is *convex*. This can be used to efficiently find minimizers of  $E$  as was done e.g. in [100].

The integral constraint in (7.9) is responsible for correctly representing the jump part of  $E$ . Intuitively, it means that, when at  $x \in \Omega$  the value of  $u$  jumps from  $t$  to  $t'$ , this jump gets penalized by at most  $d(x, t, t')$ . There is one constraint for each possible pair of values  $u^-, u^+$  at the two sides of the jump interface.

In order for the equality in (7.10) to hold, the functions  $f$  and  $d$  in (7.6) must satisfy certain properties as well as compatibility conditions. Specifically, we must have:

- $f(x, \cdot)$  is convex and lower-semicontinuous for fixed  $x \in \Omega$ .
- $d(x, \cdot, \cdot)$  is a metric on  $\Gamma$  for fixed  $x \in \Omega$ , and  $d$  is continuous.
- For all  $x \in \Omega, t, t' \in \Gamma$  and  $\nu \in \mathbb{S}^{m-1}$ :

$$d(x, t, t') \leq f_\infty(x, \nu(t' - t)), \quad (7.12)$$

where  $f_\infty(x, p)$  denotes the recession function of  $f(x, p)$  w.r.t.  $p$ , see (2.4).

Even without these restrictions, at least the inequality  $\mathcal{E}(1_u) \leq E(u)$  will always hold, i.e. in any case  $\mathcal{E}$  is a convex lower bound for  $E$  in terms of  $1_u$ .

**Constraint Derivation.** The dual vector fields  $\varphi$  in (7.8) do not seem to have a direct intuitive interpretation and are simply chosen in a special way, through the constraint  $\varphi \in \mathcal{C}$ , such that the equality (7.10) is satisfied.

Let us briefly explain how the constraint set (7.9) is derived. For fixed  $u$  and  $\varphi$  we can use the decomposition (7.3) to write the integral in (7.8) for  $v = 1_u$  as [1, Lemma 3.7]:

$$\begin{aligned} \int_{\Omega \times \mathbb{R}} \langle \varphi, dD1_u \rangle &= \int_{\Omega \setminus S_u} \left( \langle \varphi^x(x, u(x)), \nabla u(x) \rangle - \varphi^t(x, u(x)) \right) dx \\ &\quad + \int_{S_u} \left\langle \int_{u^-(x)}^{u^+(x)} \varphi^x(x, s) ds, \nu_u(x) \right\rangle d\mathcal{H}^{m-1}(x). \end{aligned} \quad (7.13)$$

In order for  $\mathcal{E}$  to be a convex lower bound for the energy  $E$ , the expression (7.13) must be less than or equal to  $E$ . Comparing with the analogous representation (7.6) of  $E$ , for this it is sufficient to have

$$\langle \varphi^x(x, u(x)), \nabla u(x) \rangle - \varphi^t(x, u(x)) \leq c(x, u(x)) + f(x, \nabla u(x)) \quad (7.14)$$

$$\left\langle \int_{u^-(x)}^{u^+(x)} \varphi^x(x, s) ds, \nu_u(x) \right\rangle \leq d(x, u^-(x), u^+(x)) \quad (7.15)$$

on  $\Omega \setminus S_u$  resp. on  $S_u$ . Setting  $t = u(x) \in \Gamma$  and  $p = \nabla u(x) \in \mathbb{R}^m$ , resp.  $t = u^-(x)$  and  $t' = u^+(x)$ , this in turn is implied if

$$\langle \varphi^x(x, t), p \rangle - \varphi^t(x, t) \leq c(x, t) + f(x, p) \quad \forall x \in \Omega, t \in \Gamma, p \in \mathbb{R}^m, \quad (7.16)$$

$$\left| \int_t^{t'} \varphi^x(x, s) ds \right| \leq d(x, t, t') \quad \forall x \in \Omega, t, t' \in \Gamma. \quad (7.17)$$

These are already the constraints in (7.9). Namely, (7.16) is equivalent to

$$\varphi^t(x, t) \geq \langle \varphi^x(x, t), p \rangle - c(x, t) - f(x, t, p) \quad \forall x \in \Omega, t \in \Gamma, p \in \mathbb{R}^m, \quad (7.18)$$

and taking the supremum over all  $p$  this is the same as

$$\begin{aligned} \varphi^t(x, t) &\geq -c(x, t) + \sup_{p \in \mathbb{R}^m} \left( \langle \varphi^x(x, t), p \rangle - f(x, t, p) \right) \\ &= -c(x, t) + f^*(x, \varphi^x(x, t)) \quad \forall x \in \Omega, t \in \Gamma, \end{aligned} \quad (7.19)$$

which is the first constraint of (7.9).

## 7.2.2 Functionals with Convexity in the Gradient

For a special and important kind of energies (7.6) the constraint set (7.9) can be significantly simplified, making the application of functional lifting much more efficient in practice in both memory and run time. Basically, if the regularizer part  $E_{\text{reg}}$  (the last two terms in (7.6)) of  $E$  is convex in the gradient  $\nabla u$  then the integral constraints of (7.9) can be omitted.

This is the class of regularizers which are already fully defined by their values on “smooth” functions  $u \in W^{1,1}(\Omega; \Gamma)$  (which do not have any jumps,

i.e. the jump set  $S_u$  has  $\mathcal{H}^{m-1}$ -measure zero). Specifically,  $E_{\text{reg}}$  is first defined for  $u \in W^{1,1}(\Omega; \Gamma)$  by

$$E_{\text{reg}}(u) = \int_{\Omega} f(x, \nabla u(x)) \, dx \quad (7.20)$$

with a  $f$  convex in  $\nabla u$ , then for general  $u \in \text{SBV}(\Omega, \Gamma)$  through lower-semicontinuity considering approximations  $u_n \rightarrow u$  by “smooth”  $u_n \in W^{1,1}(\Omega; \Gamma)$ .

It turns out, see [5, Th. 5.54] and [15], that under suitable assumptions on  $f$ ,  $E_{\text{reg}}$  can be expressed for a general  $u \in \text{SBV}(\Omega, \Gamma)$  by

$$E_{\text{reg}}(u) = \int_{\Omega \setminus S_u} f(x, \nabla u(x)) \, dx + \int_{S_u} f_{\infty}(x, \nu_u(u^+ - u^-)) \, d\mathcal{H}^{m-1}(x). \quad (7.21)$$

Here,  $f_{\infty}(x, p)$  is the recession function of  $f$  w.r.t. the last variable defined in (2.4). This is connected to the reason why the compatibility condition (7.12) is needed in the case of general functionals  $E$ .

Important concrete examples of regularizers falling under this class are total variation, its Huber-regularized variant, and quadratic regularization. Their corresponding functions  $f$  in (7.20) are respectively  $f(\nabla u) = |\nabla u|$ ,  $f(\nabla u) = \max(|\nabla u| - \frac{\varepsilon}{2}, |\nabla u|^2 / 2\varepsilon)$  for some  $\varepsilon > 0$ , and  $f(\nabla u) = |\nabla u|^2$ .

**Specialized Lifting Lemma.** This case was studied extensively in [101]. The authors show the equality (7.10), and that one can leave out the integral constraints in (7.9), so that the constraint set  $\mathcal{C}$  simplifies to

$$\mathcal{C} = \left\{ \varphi = (\varphi^x, \varphi^t) \in C_c^0(\Omega \times \Gamma; \mathbb{R}^m \times \mathbb{R}) \mid \varphi^t(x, t) \geq -c(x, t) + f^*(x, \varphi^x(x, t)) \quad \forall x \in \Omega, t \in \Gamma \right\}. \quad (7.22)$$

This makes energies of the form (7.20) an important special case in terms of the *complexity* of the convex relaxation. For numerical minimization of (7.8), the range  $\Gamma$  of  $u$  must be discretized into finitely many values, say  $n$ . This then translates into *linearly* many constraints in the corresponding discretization of the set  $\mathcal{C}$  (in terms of  $n$ , at each pixel  $x \in \Omega$ ). In contrast, the general set (7.9) results in *quadratically* many constraints per pixel.

### 7.3 Convex Relaxation

Through (7.10), the minimization of the original possibly nonconvex energy

$$\min_{u \in \text{SBV}(\Omega, \Gamma)} E(u) \quad (7.23)$$

over the space  $\text{SBV}(\Omega, \Gamma)$  is transformed into the minimization

$$\min_{v \in \mathcal{D}_0} \mathcal{E}(v) \quad (7.24)$$

of the *convex* energy  $\mathcal{E}$  over the set of all graph functions

$$\mathcal{D}_0 = \left\{ 1_u \mid u \in \text{SBV}(\Omega, \mathbb{R}) \right\}. \quad (7.25)$$

One can now try to optimize directly in terms of graph functions  $1_u$  to solve the original problem (7.23). However, this poses a challenge because, while the energy  $\mathcal{E}$  in (7.24) is convex, the new optimization domain  $\mathcal{D}_0$  is *not convex*. This is because graph functions have a *binary range*,  $1_u : \Omega \times \Gamma \rightarrow \{0, 1\}$ .

### 7.3.1 Domain Relaxation

To overcome this difficulty the key idea to efficient minimization is to *convexify the domain*  $\mathcal{D}_0$  by relaxing the binary constraint. We consider *relaxed* graph functions, replacing the range  $\{0, 1\}$  by  $[0, 1]$ :

$$\mathcal{D}' = \left\{ v \in \text{BV}(\Omega \times \Gamma; [0, 1]) \mid \begin{aligned} &v(x, \cdot) \text{ nonincreasing for a.e. } x \in \Omega, \\ &\lim_{t \rightarrow t_{\min}} v(x, t) = 1 \quad \text{for a.e. } x \in \Omega, \\ &\lim_{t \rightarrow t_{\max}} v(x, t) = 0 \quad \text{for a.e. } x \in \Omega \end{aligned} \right\}.$$

Here  $t_{\min}$  and  $t_{\max}$  are the two endpoints of the range  $\Gamma = (t_{\min}, t_{\max})$  of  $u$ .

In other words,  $v$  must still be nonincreasing, starting with value 1 for small  $t$  and ending with value 0 for large  $t$ , but is allowed to have smooth transitions from 1 to 0 instead of only a jump from 1 to 0. We now consider the minimization

$$\min_{v \in \mathcal{D}'} \mathcal{E}(v) \tag{7.26}$$

of  $\mathcal{E}$  over the relaxed domain  $\mathcal{D}'$ .

**Eliminating the Monotonicity Constraint.** We note that the monotonicity constraint in  $\mathcal{D}'$  is actually not necessary and could be omitted without affecting the set of minimizers  $v$ . In fact, we can easily see that  $\mathcal{E}(v) = \infty$  whenever  $v$  is increasing somewhere. Namely, since  $\varphi^t(x, t)$  can be chosen arbitrarily large in (7.9), the supremum in (7.8) will be finite only if  $D_t v \leq 0$  (the distributional gradient w.r.t.  $t$ ), i.e. if  $v(x, \cdot)$  is nonincreasing. Thus, instead of (7.26), equivalently we will consider the minimization

$$\min_{v \in \mathcal{D}} \mathcal{E}(v) \tag{7.27}$$

over the set

$$\mathcal{D} = \left\{ v \in \text{BV}(\Omega \times \Gamma; [0, 1]) \mid \begin{aligned} &\lim_{t \rightarrow t_{\min}} v(x, t) = 1 \quad \text{for a.e. } x \in \Omega, \\ &\lim_{t \rightarrow t_{\max}} v(x, t) = 0 \quad \text{for a.e. } x \in \Omega \end{aligned} \right\}. \tag{7.28}$$

### 7.3.2 Optimality

Having obtained a minimizer  $v^* : \Omega \times \Gamma \rightarrow [0, 1]$  of (7.27), because of the range relaxation it may happen that  $v^*(x, \cdot)$  is *nonbinary* at some points  $x \in \Omega$ . Then  $v^*$  is not in the original set (7.25), i.e. not the graph function of any  $u$ . Thus, the question is how to recover from  $v^*$  an optimal solution of the original unrelaxed problem (7.23), respectively (7.24).

**Optimal Solutions for Special Functionals.** For the special case of functionals with convexity in the gradient as described in Section 7.2.2, it turns out [101] that the original problem (7.24) can be solved optimally via functional lifting. A binary graph function  $v_{\text{bin}} \in \mathcal{D}_0$  minimizing (7.24) can be constructed from  $v^*$  by simple *thresholding*:

$$v_{\text{bin}}(x, t) = \begin{cases} 1 & \text{if } v^*(x, t) > s, \\ 0 & \text{else} \end{cases} \quad (7.29)$$

with an arbitrary  $s \in [0, 1)$ . Then  $v_{\text{bin}} = 1_u$  for some  $u$ , namely

$$u(x) = \sup \left\{ t \in \Gamma \mid v_{\text{bin}}(x, t) \geq \frac{1}{2} \right\}, \quad (7.30)$$

and this  $u$  is a global minimizer of (7.23), regardless for the choice of  $s$  in (7.29) (although it may be that  $u$  is in  $\text{BV}(\Omega, \Gamma)$  and not in  $\text{SBV}(\Omega, \Gamma)$ , which is not an issue).

**Energy Bound in the General Case.** In the general case, there is unfortunately no thresholding theorem as for the above special case. If the obtained relaxed minimizer  $v^*$  is actually binary, then  $v^* \in \mathcal{D}_0$  and it is already an optimal solution of (7.24).

Otherwise one can try to find at least an approximate solution  $v_{\text{bin}} \in \mathcal{D}_0$  by projecting, in some way,  $v^* \in \mathcal{D} \supset \mathcal{D}_0$  back to the set  $\mathcal{D}_0$  of binary graph functions. An obvious choice is to construct  $v_{\text{bin}}$  by the same thresholding binarization (7.29) as above, e.g. with  $s = \frac{1}{2}$ .

In general one cannot expect that the obtained candidate solution  $v_{\text{bin}}$  is a global minimizer of (7.24), because discretized versions of functionals of the form (7.6) are known to lead to NP-hard problems. Nonetheless, we can give an estimate how far  $v_{\text{bin}}$  is from a true global minimizer  $v_{\text{bin}}^*$  by means of the energy bound (1.6). In our case, this bound is

$$\mathcal{E}(v^*) \leq \mathcal{E}(v_{\text{bin}}^*) \leq \mathcal{E}(v_{\text{bin}}). \quad (7.31)$$

The lower and upper values are explicitly computable after having computed  $v^*$  and then  $v_{\text{bin}}$  from  $v^*$ .

## 7.4 Implementation

Let us give some notes about the numerical implementation of the lifted optimization problem (7.27).

### 7.4.1 Discretization

**Image Domain and Variable Discretization.** We discretize the image domain  $\Omega$  into a rectangular pixel grid, again denoted by  $\Omega$ . The range space  $\Gamma = (t_{\min}, t_{\max})$  of  $u$  must also be discretized since the lifted energy (7.8) lives

on the space  $\Omega \times \Gamma$ . For simplicity we assume  $\Gamma = (0, 1)$  and discretize it into  $n \geq 2$  uniformly spaced levels

$$\frac{0}{n-1}, \dots, \frac{n-1}{n-1}, \quad \text{with spacing } \Delta t = \frac{1}{n-1}. \quad (7.32)$$

The discretized variables  $v$  and  $\varphi$ , as well as the data term  $c$  are represented by their node values at the pixels  $x \in \Omega$  and levels  $0 \leq i < n$ :

$$v(x, \frac{i}{n-1}) = v_i(x) \in \mathbb{R}, \quad (7.33)$$

$$\varphi^x(x, \frac{i}{n-1}) = \frac{1}{\Delta t} \varphi_i^x(x) \in \mathbb{R}^m, \quad (7.34)$$

$$\varphi^t(x, \frac{i}{n-1}) = \varphi_i^t(x) \in \mathbb{R}, \quad (7.35)$$

$$c(x, \frac{i}{n-1}) = c_i(x) \in \mathbb{R}. \quad (7.36)$$

In (7.34) we include the factor  $\frac{1}{\Delta t}$  to simplify the subsequent energy discretization.

**Differential Operator Discretization.** The spatial (distributive) gradient  $D_x$  is discretized as in (2.33) using forward differences  $\nabla_x^+$  with Neumann boundary conditions, and the spatial divergence  $\text{div}_x$  correspondingly as in (2.35) by backward differences  $\text{div}_x^-$  with Dirichlet conditions.

The (distributive) derivative  $(D_t v)(x, t)$  of  $v$  in the range direction is discretized by forward differences  $\partial_t^+$  with the Dirichlet boundary condition:

$$(D_t v)(x, \frac{i}{n-1}) = \frac{1}{\Delta t} (\partial_t^+ v_i)(x), \quad (7.37)$$

$$(\partial_t^+ v_i)(x) = \begin{cases} v_{i+1}(x) - v_i(x) & \text{if } i < n-1, \\ -v_i(x) & \text{if } i = n-1. \end{cases} \quad (7.38)$$

The case  $i = n-1$  implicitly enforces  $v_{i+1}(x) = v_n(x) = 0$ , i.e. one of the constraints in (7.28). The corresponding negative adjoint  $\partial_t^- = -(\partial_t^+)^T$  is given by backward differences:  $(\partial_t^- p_i)(x) = p_i(x) - p_{i-1}(x) \chi_{i \geq 1}$ .

**Energy and Constraint Set Discretization.** Integrals  $\int_{\Omega} dx$  over the image domain are discretized by  $\sum_{x \in \Omega}$ , and integrals  $\int_{\Gamma} dt$  over the range by  $\sum_{0 \leq i < n} \Delta t$ , i.e. with the scaling factor  $\Delta t$  in (7.32).

The discretized energy (7.8), written out as (7.11), becomes:

$$\min_{v \in \mathcal{D}_d} \max_{\varphi \in \mathcal{C}_d} \sum_{x \in \Omega} \sum_{0 \leq i < n} \langle \varphi_i^x(x), \nabla^+ v_i(x) \rangle + \varphi_i^t(x) \partial_t^+ v_i(x). \quad (7.39)$$

Note that the scaling factor  $\Delta t$  cancels out completely, for the first term because we used the scaling (7.34) for  $\varphi^x$  and for the second term because of the range derivative discretization (7.37). The discretized primal constraint set (7.28) is

$$\mathcal{D}_d = \left\{ (v_i)_{0 \leq i < n} \mid v_i : \Omega \rightarrow [0, 1], \quad v_0(x) = 1 \quad \forall x \in \Omega \right\}. \quad (7.40)$$

We keep only one of the two boundary value constraints of (7.28) since the other one  $v_n(x) = 0$  is already implicitly ensured through the  $\partial_t^+$  discretization



(7.38). The fixed values  $v_0(x) = 1$  could be kept implicit as well, i.e. there is actually no need to store them explicitly in the implementation. However, this would only save a negligible fraction of needed memory (one image layer out of  $n$  total) and it is more convenient to work with an explicit  $v_0(x)$ .

The constraint set for the duals (7.9) becomes

$$\begin{aligned} \mathcal{C}_d = \{ & (\varphi_i)_{0 \leq i < n} \mid \varphi_i = (\varphi_i^x, \varphi_i^t) \text{ with } \varphi_i^x : \Omega \rightarrow \mathbb{R}^m, \varphi_i^t : \Omega \rightarrow \mathbb{R}, \\ & \varphi_i^t(x) \geq -c_i(x) + f^*\left(x, \frac{1}{\Delta t} \varphi_i^x(x)\right) \quad \forall x \in \Omega, 0 \leq i < n, \\ & \left| \sum_{i < k \leq j} \varphi_k^x(x) \right| \leq \widehat{d}(x, i, j) \quad \forall x \in \Omega, 0 \leq i < j < n \}. \end{aligned} \quad (7.41)$$

The scaling factor  $\Delta t$  appears in the first constraint due to (7.34), while it is not present in the integral constraint due to the cancellation after the discretization  $\int_{\Gamma} dt \rightarrow \sum_{0 \leq i < n} \Delta t$ . The discrete distance function is defined analogously to (7.33)–(7.36) as  $\widehat{d}(x, i, j) = d(x, \frac{i}{n-1}, \frac{j}{n-1})$  for all  $0 \leq i < j < n$ . For the special kind of energies  $E$  considered in Section 7.2.2, only the first constraint in (7.41) must be kept:

$$\begin{aligned} \mathcal{C}_d^{\text{local}} = \{ & (\varphi_i)_{0 \leq i < n} \mid \varphi_i = (\varphi_i^x, \varphi_i^t) \text{ with } \varphi_i^x : \Omega \rightarrow \mathbb{R}^m, \varphi_i^t : \Omega \rightarrow \mathbb{R}, \\ & \varphi_i^t(x) \geq -c_i(x) + f^*\left(x, \frac{1}{\Delta t} \varphi_i^x(x)\right) \quad \forall x \in \Omega, 0 \leq i < n \}. \end{aligned} \quad (7.42)$$

### 7.4.2 The Primal-Dual Algorithm

To solve the saddle-point problem (7.39) we can use the preconditioned primal-dual Algorithm 3. The energy is of the general form (2.38), and the functions  $D$  and  $F$  of (2.38) are present here only in the form of hard constraints:  $D(v) = \delta_{\mathcal{D}_d}(v)$  and  $F(\varphi) = \delta_{\mathcal{C}_d}(\varphi)$ . Therefore, the proximal operators in the update equations of Algorithm 3 reduce to Euclidean projections  $\pi_{\mathcal{D}_d}$  and  $\pi_{\mathcal{C}_d}$  onto the constraints sets, see (2.41).

The projection  $\pi_{\mathcal{D}_d}$  for the main primal variable  $v$  is straightforward. It can be computed independently and in parallel for each  $x$  and  $i$  by simple clipping of the values  $v_i(x)$  to  $[0, 1]$  for  $i > 0$  and setting  $v_0(x) = 1$ .

The projection for  $\varphi$  is more involved, and there are basically two different strategies depending on which regularizer is used.

**Implementation for the Special Case (7.42).** In this case all constraints are local and the projection  $\pi_{\mathcal{C}_d^{\text{local}}}$  is very easy to handle and to compute explicitly. It reduces to independent projections for each fixed  $x$  and  $i$ , needing to project onto  $\{(p, q) \in \mathbb{R}^m \times \mathbb{R} \mid q \geq -c_i(x) + f^*(x, p)\}$ . This can be done in closed form for commonly used regularizers, such as  $TV$ , Huber- $TV$ , and quadratic regularization, see [101] for more details. For the quadratic regularizer one has to project onto a parabola, for which we suggest to use the explicit formula in Section 9.9.2 instead of the iterative scheme suggested in [101].

**Implementation for the General Case (7.41).** This case is more complicated because of the quadratically many *nonlocal* sum-constraints in (7.41), so that there is no simple projection formula. In [100, 30] it was suggested to use iterative Dijkstra’s algorithm, but it is rather slow requiring many iterations and can compute only approximate projections.

Here we propose a *dualization* approach to exactly enforce the constraints, analogously as was done in Section 3.5.4 for the pairwise constraints (3.33). We use the dualization (2.75) to rewrite the hard constraints equivalently as new energy terms. With new primal variables  $a_{ij}(x) \in \mathbb{R}^m$  for all  $x \in \Omega$  and all pairs  $i, j$  with  $0 \leq i < j < n$ , we add the terms

$$\inf_{a_{ij}: \Omega \rightarrow \mathbb{R}^m} \sum_{x \in \Omega} \sum_{0 \leq i < j < n} \left\langle -a_{ij}(x), \sum_{i < k \leq j} \varphi_k^x(x) \right\rangle + \widehat{d}(x, i, j) |a_{ij}(x)| \quad (7.43)$$

to the energy (7.39). To simplify the inner sums of  $\varphi$ ’s, we also introduce auxiliary dual variables  $p_i(x) \in \mathbb{R}^m$  for all  $x \in \Omega$  and  $0 \leq i < n$  by requiring

$$\partial_t^- p_i(x) = \varphi_i^x(x) \quad (7.44)$$

for all  $x$  and  $i$ . Then the sums reduce to

$$\sum_{i < k \leq j} \varphi_k^x(x) = p_j(x) - p_i(x). \quad (7.45)$$

The equalities (7.44) can be enforced using the dualization (2.74), adding

$$\inf_{\mu_i: \Omega \rightarrow \mathbb{R}^m} \sum_{x \in \Omega} \sum_{0 \leq i < n} \langle \mu_i(x), \partial_t^- p_i(x) - \varphi_i^x \rangle \quad (7.46)$$

to the energy with new primal variables  $\mu_i(x) \in \mathbb{R}^m$  (Lagrange multipliers) for all  $x \in \Omega$ ,  $0 \leq i < n$ . Overall, the initial energy (7.39) now becomes

$$\begin{aligned} \min_{v \in \mathcal{D}_d, \mu, a} \max_{\varphi \in \mathcal{C}_d^{\text{local}}, p} & \sum_{x \in \Omega} \sum_{0 \leq i < n} \langle \varphi_i^x(x), \nabla^+ v_i(x) \rangle + \varphi_i^t(x) \partial_t^+ v_i(x) \\ & + \sum_{x \in \Omega} \sum_{0 \leq i < n} \langle \mu_i(x), \partial_t^- p_i(x) - \varphi_i^x \rangle \\ & + \sum_{x \in \Omega} \sum_{0 \leq i < j < n} \langle -a_{ij}(x), p_j(x) - p_i(x) \rangle + \widehat{d}(x, i, j) |a_{ij}(x)|. \end{aligned} \quad (7.47)$$

One now optimizes only over the simplified constraint set (7.42), and there are no constraints on  $\mu$ ,  $a$  and  $p$ . The proximal operator for  $a$  decomposes into individual prox operators for each  $a_{ij}(x)$ , which are given by soft-thresholding:

$$\text{prox}_{\tau, \widehat{d}(x, i, j) \cdot |\cdot|}(a) = a \max(0, 1 - \tau \widehat{d}(x, i, j) / |a|). \quad (7.48)$$

We suggest to use the variant of the primal-dual Algorithm 4 where the “bar”-copies are introduced for the duals rather than for the primals, since there are quadratically many  $a_{ij}$ ’s.

## Chapter 8

# Vectorial Problems with Separable Regularization

As the first special case of vectorial functionals we consider separable regularizers and apply (a reformulation of) the functional lifting approach separately to each channel, resulting in a significant speedup in comparison to previous methods. A difficulty will be that the data term becomes nonconvex, for which we succeed to find the tightest possible convex envelope relaxation. This chapter is based on joint work with Bastian Goldlücke and Daniel Cremers [124, 55].

### 8.1 Introduction

#### 8.1.1 Vectorial Energies and Multilabel Problems

We consider the minimization of energies of the form

$$\min_{u:\Omega\rightarrow\Gamma} E(u), \quad E(u) = \int_{\Omega} c(x, u(x)) \, dx + R(u) \quad (8.1)$$

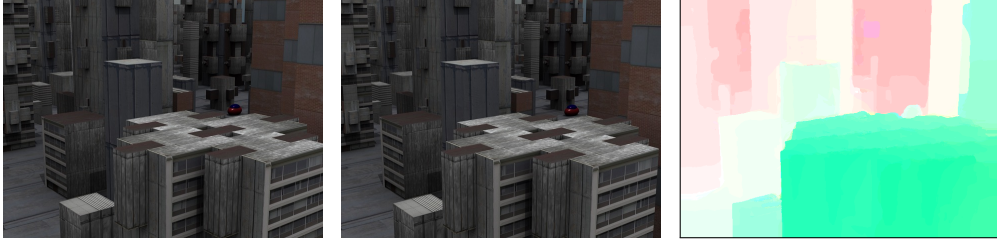
with a data term  $c$  and a regularizer  $R$ , where the unknown functions  $u : \Omega \rightarrow \Gamma$  have a *vectorial*,  $k$ -dimensional range:

$$\Gamma \subset \mathbb{R}^k \quad \text{with a } k \geq 1. \quad (8.2)$$

Such energies arise in a multitude of applications because of the naturally frequent occurrence of vectorial data. Some of the more prominent examples are color image denoising where  $k = 3$ , and optical flow estimation where  $k = 2$ , see Figure 8.1. While the general functional lifting convexification method can be employed to find a convexification in the scalar case  $k = 1$ , it is essentially limited to  $k = 1$  and does not generalize to the vectorial case. Thus, optimization of vectorial energies (8.1) still remains a challenge.

Our proposed convexification approach in this chapter will be based on functional lifting, essentially applying it separately for each of the  $k$  channels of  $u$ . Thus, we will need to *discretize* the range of  $u$  in the end in order to practically compute a solution.

When the range  $\Gamma$  is discretized, vectorial energies (8.1) can be equivalently regarded as being *multilabel problems*, with the additional information that the



**Figure 8.1: The proposed approach for large multilabel problems.** The proposed relaxation method can approximate the solution to multilabeling problems with a huge number of possible labels by globally solving a convex relaxation model. This example shows two images and the optic flow field between them, where flow vectors were assigned from a possible set of  $50 \times 50$  vectors, with truncated linear distance as a regularizer. The problem has so many different labels that a solution cannot be computed by alternative relaxation methods on current hardware.

finite label space  $\Gamma$  has a special *multidimensional* structure. Conversely, every such multilabel problem can be regarded as a special vectorial energy (8.1), namely where only finitely many values for  $u$  are allowed. Current convexification approaches for (8.1) all work with the finite multilabel viewpoint of the problem, of which they only use the fact that the range is finite, disregarding the vectorial label structure.

In this chapter we propose a convex relaxation which crucially utilizes the vectorial nature of the labels to achieve a drastic reduction in memory and run time requirements. Furthermore, our approach works with a *continuous* label space, thus allowing to formulate more general regularizers such as piecewise-smooth Mumford-Shah.

### 8.1.2 Contribution: Relaxation for Product Label Spaces

We consider multidimensional label spaces which can be written as a product of a finite number  $k \geq 1$  of scalar spaces  $\Gamma_i \subset \mathbb{R}$ ,

$$\Gamma = \Gamma_1 \times \cdots \times \Gamma_k. \quad (8.3)$$

The central idea is as follows. For illustration, assume that the individual spaces  $\Gamma_i$  are discrete or have been discretized, and let

$$n_i = |\Gamma_i| \quad (8.4)$$

be the number of elements in  $\Gamma_i$ . Then the total number of labels is

$$n = |\Gamma| = \prod_i |\Gamma_i| = \prod_i n_i. \quad (8.5)$$

In previous relaxations for the multilabel problem, as seen in Part I, this means that we need to optimize over  $n$  binary indicator functions, which can easily amount to thousands of scalar functions in practical problems. In order to

make problems of this form feasible to solve, we present a *reduction method* which only requires

$$s = \sum_i n_i \quad (8.6)$$

binary functions. As a consequence, memory grows linearly (rather than exponentially) in the number of range dimensions  $k$ , while the computation time is greatly reduced.

An important limitation, however, is that we only consider *separable* regularizers of the form

$$R(u) = \sum_{i=1}^k R_i(u_i), \quad (8.7)$$

which means that  $R$  acts on the label components  $u_i$  of  $u$  independently.

We will show that with the novel reduction technique it is possible to efficiently solve convex relaxations to multilabel problems which are far too large to approach with existing techniques. A prototypical example is optic flow, where the total number of labels is typically around  $32 \times 32$  in practice. In that case, for example, we only require  $s = 64 = 2 \cdot 32$  indicator functions instead of  $n = 1024 = 32^2$ . However, the proposed method applies to a much larger class of labeling problems. This reduction in variable size not only allows a substantially higher resolution of the label space, but it also gives rise to a drastic speedup.

Complete source code to reproduce the experiments is publicly available on Sourceforge under a GPL3 license as part of our CUDA library for continuous convex optimization in image processing <sup>1</sup>.

### 8.1.3 Related Work

**Discrete Approaches.** In [113, 114] the problem of image registration is formulated as an MRF labeling problem, which is minimized via LP relaxation. The authors present a decoupling strategy for the displacement components which is related to ours, albeit only applicable in the discrete case. It allows a simplification of the graph construction and consequently a larger numbers of labels. Another discrete method which is related to ours is [105]. The authors present a compact encoding scheme for the multilabel problem called a log-transformation which makes the unary term non-submodular. This is in analogy to our transformation, which makes the previously convex data term nonconvex. The problem of large label spaces is also tackled in [52], where the authors compute optical flow from an MRF labeling problem using a lower dimensional parametric description for the displacements.

General approaches for multilabel problems were discussed in detail in Part I, especially in Chapter 3. For instance, optimal solutions can be found for convex regularizers and a linearly ordered label space. Otherwise, approximations methods like  $\alpha$ -expansion can be used. However, in many important scenarios

---

<sup>1</sup><https://sourceforge.net/p/cocolib>

the label space cannot be ordered, which is a typical situation in the vectorial case. Moreover, a nonconvex regularizer is often more desirable to better preserve discontinuities in the solution.

**Continuous Approaches.** The framework presented in this paper is based on the functional lifting idea described in depth in Chapter 7, which however is limited to the scalar case  $k = 1$ .

Viewing (8.1) as a multilabel problem, general continuous multilabel approaches described in Chapter 3 can be applied. However, none of them is known to lead to provably optimal solutions and they all have in common that they are very memory intensive if the number of labels becomes larger. This makes it impossible to use them for scenarios with thousands of labels, like for example optic flow.

This chapter is based on our previous conference [53, 124] and journal publications [55].

## 8.2 Multidimensional Label Spaces

### 8.2.1 Discrete Label Spaces and Dimensionality Reduction

To keep the notion as clear as possible while working with a multidimensional label space, throughout this chapter we keep the following conventions. The index  $i = 1, \dots, k$  enumerating the factors of the product space (8.3), respectively the channels of the vectorial mapping  $u$ , is always written as a subscript. For functions taking labels as one of the arguments, the label argument can also be written as a superscript, so that one can consider functions for a fixed label. E.g. we will write  $v_i^s(x) = v_i(x, s)$ , and  $v_i^s$  denotes the function  $v_i(\cdot, s)$ .

In order to give a more visual explanation of the main idea behind our approach, we first discuss the *discrete* case, i.e. that each factor  $\Gamma_i$  consists of finitely many labels. In the following, we will assume that the cost functions  $c^t(x) = c(x, t)$  lie in the Hilbert space of square integrable functions:  $c^t \in L^2(\Omega; \mathbb{R})$  for each  $t \in \Gamma$ .

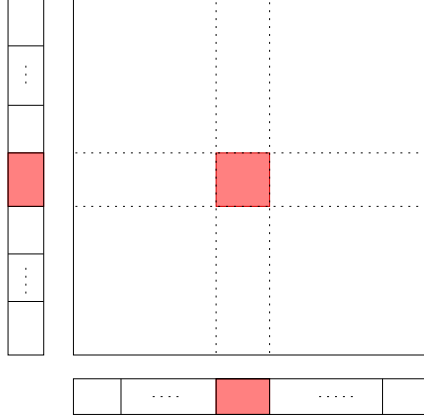
**General Multilabel Approaches.** As described in Chapter 3, the existing general multilabel relaxations work by introducing label indicator functions  $u^t : \Omega \rightarrow \{0, 1\}$  for each  $t \in \Gamma$  through the relation

$$u^t(x) = \begin{cases} 1 & \text{if } u(x) = t, \\ 0 & \text{else.} \end{cases} \quad (8.8)$$

The collection of all  $n$  indicator functions  $u^t$ , denoted again by  $u$ , is a mapping  $u : \Omega \rightarrow \Delta_0$  with the binary simplex  $\Delta_0$  in (3.12), or  $u \in L^2(\Omega, \Delta_0)$ . The problem (8.1) is then written in the equivalent form

$$\min_{u \in L^2(\Omega, \Delta_0)} \sum_{t \in \Gamma} \int_{\Omega} c^t(x) u^t(x) dx + R(u). \quad (8.9)$$

As can be immediately seen, this does not use any structure properties of the range  $\Gamma$  and always requires  $n = |\Gamma| = \prod_i n_i$  indicator functions  $u^t$ .



**Figure 8.2: The central idea of the reduction technique.** If a single indicator function in the product space  $\Gamma$  takes the value 1, then this is equivalent to setting an indicator function in each of the factors  $\Gamma_i$ . The memory reduction stems from the fact that there are much more labels in  $\Gamma$  than in all the factors  $\Gamma_i$  combined.

**Novel Reduction Approach.** The central idea of the chapter is the following. The full discrete label space  $\Gamma$  has  $n = \prod_i n_i$  elements, which means that it requires  $n$  indicator functions to represent a labeling, one for each label. We will show that it suffices to use  $s = \sum_i n_i$  indicator functions, which is a considerable reduction in problem dimensionality, thus also in computation time and memory requirements. We achieve this by replacing the indicator functions on the product  $\Gamma$  by indicator functions *on the components*  $\Gamma_i$ . Intuitively, a label in  $\Gamma \subset \mathbb{R}^k$  is uniquely determined by its  $k$  coordinates, so we only need to consider indicator functions for each coordinate space  $\Gamma_i$  separately.

To this end, we consider indicator functions defined similarly to (8.8), but *separately* for each channel  $u_i$ ,  $1 \leq i \leq k$ . For fixed channel  $1 \leq i \leq k$  and each channel value  $s \in \Gamma_i$ , we associate an indicator function  $v_i^s : \Omega \rightarrow \{0, 1\}$  by

$$v_i^s(x) = \begin{cases} 1 & \text{if } u_i(x) = s, \\ 0 & \text{else.} \end{cases} \quad (8.10)$$

The collection  $v_i(x) = (v_i^s(x))_{s \in \Gamma_i}$  of the  $n_i = |\Gamma_i|$  indicator functions is then a mapping  $v_i : \Omega \rightarrow \Delta_i$  into the simplex

$$\Delta_i = \left\{ x \in \{0, 1\}^{n_i} \mid \sum_{j=1}^{n_i} x_j = 1 \right\} \subset \mathbb{R}^{n_i}, \quad (8.11)$$

or also  $v_i \in L^2(\Omega, \Delta_i)$ . Collecting the mappings for all  $k$  channels, we get the functions  $v = (v_i)_{1 \leq i \leq k} = (v_i^s)_{1 \leq i \leq k, s \in \Gamma_i} : \Omega \rightarrow \Delta_\times$  with range

$$\Delta_\times = \Delta_1 \times \dots \times \Delta_k \subset \mathbb{R}^{\sum_i n_i}, \quad (8.12)$$

or  $v \in L^2(\Omega, \Delta_\times)$ . As each  $v_i$  consists of  $n_i = |\Gamma_i|$  indicator functions, the overall reduced mapping  $v$  consists of exactly  $\sum_i n_i$  such functions.

The following proposition illuminates the relationship between the original space of indicator functions  $L^2(\Omega, \Delta)$  and the reduced indicator function space  $L^2(\Omega, \Delta_\times)$ , which is easy to understand visually, see Figure 8.2.

**Proposition 8.1.** *A bijection  $v \mapsto u$  from  $L^2(\Omega, \Delta_\times)$  onto  $L^2(\Omega, \Delta)$  is defined by setting*

$$u^t(x) := v_1^{t_1}(x) \cdot \dots \cdot v_k^{t_k}(x), \quad \forall t = (t_1, \dots, t_k) \in \Gamma, \quad x \in \Omega. \quad (8.13)$$

The proof of this proposition as well as of the other following results is given in the appendix Section 8.8.

Using this reduced function space and plugging (8.13) into (8.9), another equivalent formulation to (8.1) can be given as

$$\min_{v \in L^2(\Omega, \Delta_\times)} \sum_{t \in \Gamma} \int_{\Omega} c^t(x) v_1^{t_1}(x) \cdot \dots \cdot v_k^{t_k}(x) dx + R(v). \quad (8.14)$$

We use the same symbol  $R$  to also denote the regularizer on the reduced space. Its definition requires careful consideration, and will be discussed in detail later in Section 8.4.1.

While we have reduced the dimensionality of the problem considerably from (8.5) to (8.6), we have introduced another difficulty: the data term is *not convex anymore*, since it contains a product of indicator functions. Thus, in the relaxation, we need to take additional care to make the final problem convex again.

## 8.2.2 Continuous Label Spaces and Relaxation Framework

We now turn to the more general case that each factor  $\Gamma_i$  is an interval in  $\mathbb{R}$ , which means that we deal with a continuous label space with an infinite number of labels. In this situation, one is also interested in a number of continuous regularizers, which cannot be modeled satisfyingly on a discrete label space.

As in the discrete case, the regularizers are usually not convex and require a relaxation. Our relaxation will be based on the central functional lifting idea from Chapter 7. Functional lifting uses the representation in terms of the *graph functions*

$$1_{u_i}(x, s) = \chi_{s < u_i(x)} = \begin{cases} 1 & \text{if } s < u_i(x), \\ 0 & \text{else} \end{cases} \quad (8.15)$$

and can be used to convexify regularizers on *scalar* functions such as each of the individual channels  $u_i$ . For the regularizer, we can restrict ourselves to the case that it can be decomposed into the sum of regularizers on each component, so that the lifting approach could be applied channel-by-channel. This motivates our assumption (8.7).

However, in order to be able to also simultaneously formulate a convex relaxation for the data term with arbitrary costs  $c$  we need a different representation, namely in terms of *indicator functions*  $v_i^s$  as in (8.10). Therefore, for the regularizer relaxation we are going to “translate” the functional lifting framework to our representation, which will be done later in Section 8.4.



The indicator functions  $v_i$  for each channel  $i$  denote whether a specific label  $s$  is set at a point  $x \in \Omega$ , i.e.  $u_i(x) = s$ . While in the discrete case they could be defined by (8.10), in the general case of a *continuous* label space  $\Gamma_i$ , the relationship is

$$v_i(x, s) = \delta(u_i(x) - s), \quad (8.16)$$

where  $\delta$  is the Dirac distribution. Note that  $v_i(x, \cdot)$  are actually distributions on the higher dimensional space  $\Omega \times \Gamma_i$ , which however will reduce to regular functions after discretization. They serve as a generalization of the discrete label indicator functions  $v_i \in L^2(\Omega, \Delta_i)$  in (8.10) to the continuous case, in particular they satisfy the relations

$$\int_{\Gamma_i} v_i(x, s) \, ds = 1, \quad \int_{\Gamma_i} s v_i(x, s) \, ds = u_i(x), \quad (8.17)$$

which mimic the discrete case with sums replaced by integrals. Intuitively, this means that for each fixed  $x \in \Omega$ ,  $v_i(x, \cdot)$  has a total mass of 1 and is concentrated on the label  $u_i(x) \in \Gamma_i$ .

The minimization problem (8.9) which we want to solve is not convex: neither is the energy a convex function nor is the domain of minimization a convex set. Thus, the task of finding a global minimizer is in general computationally infeasible. We therefore propose a convex relaxation. We will formulate the data term in a convex way in terms of the new variables  $v$  in Section 8.3, and will introduce the class of regularizers possible with our approach and their corresponding convexifications in Section 8.4.

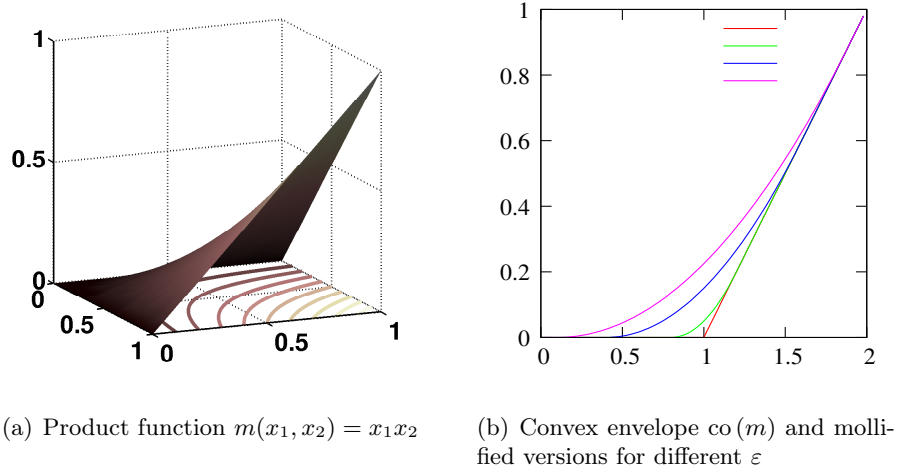
Some things have to be kept in mind, however. Since the new variables are distributions in the continuous case, we cannot formulate a well-defined minimization problem without first reducing them to  $L^2$ -functions. This means that before writing down the actual minimization problem we want to solve in the new variables, we have to introduce a discretization of the label space. Despite the necessary discretization, we follow other previous works which employ the lifting idea [99, 100, 101, 102], and still insist that we correctly deal with a continuous label space. This is justified since the definition of the continuous regularizers in Section 8.4 does not make use of the discretization, in contrast to e.g. (3.32), where the label space is discretized from the beginning. One thing which remains to be discussed, however, is whether the discrete solutions converge to the continuous one when the label space discretization is refined, in the spirit of [27]. This is a possible avenue for future work.

### 8.3 Convex Relaxation of the Data Term

In this section, we deal with the nonconvexity of the data term in (8.14),

$$E_{\text{data}}(v) = \sum_{t \in \Gamma} \int_{\Omega} c^t(x) v_1^{t_1}(x) \cdot \dots \cdot v_k^{t_k}(x) \, dx. \quad (8.18)$$

Specifically, we show two different ways how it can be replaced with a convex function which coincides with the original data term for binary functions.



**Figure 8.3: Product function relaxation.** Product function and its smoothed convex envelope for the case of two channels  $k = 2$ .

We first describe the convexification idea from the original conference paper [53] in the discrete case with a label space of dimension  $k = 2$ . While it leads to a working relaxation, it has certain shortcomings, the main problem being that an unwanted constant solution has to be avoided by additional smoothing when moving on from binary to continuous functions. These shortcomings will be remedied by a new relaxation technique which we explain thereafter. We will show that this relaxation is actually the best possible one, i.e. the convex envelope of the data term. Note that for the data term, we already work in the setting of a discretized label space. While it is possible to give a well-defined theoretical justification of the relaxation for the continuous case, the associated trouble and loss of clarity is not worth the small theoretical gain.

### 8.3.1 Previous Relaxation

In [53], it was suggested to replace the multiplication function  $m(v_1^{t_1}, \dots, v_k^{t_k}) := v_1^{t_1} \cdot \dots \cdot v_k^{t_k}$  with its convex envelope  $\text{co}(m)$ . Analyzing the epigraph of  $m$ , see Figure 8.3(a), shows that

$$\text{co}(m)(v_1^{t_1}, \dots, v_k^{t_k}) = \begin{cases} 1 & \text{if } v_1^{t_1} = \dots = v_k^{t_k} = 1, \\ 0 & \text{if any } v_i^{t_i} = 0. \end{cases} \quad (8.19)$$

This means that if in the functional,  $m$  is replaced by the convex function  $\text{co}(m)$ , we retain the same binary solutions, since the function values on binary inputs are the same.

We lose nothing on first glance, but on second glance, we forfeited differentiability of the data term, since  $\text{co}(m)$  is not a smooth function anymore. Furthermore, the new function we obtain is not the correct convex envelope of the full data term, but only for the constituting addends, i.e. for each  $t \in \Gamma$  separately in (8.18). The particular problem this leads to is that for the constant

function  $v$  defined by

$$v_i^s(x) := 1/n_i \quad (8.20)$$

the energy of the data term and hence the total energy is zero.

In [53], this problem was circumvented by an additional mollification of the convex envelope: One replaces  $\text{co}(m)$  by a mollified function  $\text{co}(m)_\varepsilon$ , where  $\varepsilon > 0$  is a small constant. We illustrate this for the case  $k = 2$ , where one can easily write down the functions explicitly. In this case, the convex envelope of multiplication is

$$\text{co}(m)(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 + x_2 \leq 1, \\ x_1 + x_2 - 1 & \text{otherwise.} \end{cases} \quad (8.21)$$

This is a piecewise linear function of the sum of the arguments, i.e symmetric in  $x_1$  and  $x_2$ , see Figure 8.3(b). We smoothen the kink by replacing  $\text{co}(m)$  with a smoothed version  $\text{co}(m)_\varepsilon$ , see [53]. This function does not satisfy the envelope condition (8.19) exactly, but only fulfills the less tight

$$\text{co}(m)_\varepsilon(x_1, \dots, x_k) \begin{cases} = 1 & \text{if } x_1 = \dots = x_k = 1, \\ \leq \varepsilon & \text{if any } x_j = 0. \end{cases} \quad (8.22)$$

Notably, the data term energy of the constant trivial minimizer (8.20) becomes now  $\varepsilon \sum_t c^t(x)$  at each point  $x \in \Omega$ , which means that the relaxation of the data term leads to the correct pointwise solution with energy  $\min_{t \in \Gamma} c^t(x)$  if  $\varepsilon > \min_{t \in \Gamma} c^t(x) / \sum_{t \in \Gamma} c^t(x)$ . Since the condition must be satisfied for each point  $x \in \Omega$ , it is best to let  $\varepsilon = \varepsilon(x)$  depend on  $x \in \Omega$  and set it pointwise to the minimal possible value. However, the choice of the smoothed envelope is suboptimal since it is just an approximation to the correct envelope and distorts the original problem. Thus, we are now going to propose a novel relaxation of the data term which avoids this problem altogether and is easier to deal with in higher dimensional label spaces.

### 8.3.2 Novel Tightest Convex Envelope Relaxation

In this section, we describe our new relaxation of the data term. It is the tightest possible relaxation and does not suffer from the described drawbacks of the relaxation in [53]. The new relaxation of  $E_{\text{data}}(v)$  is one of the main contributions of this chapter. It is defined as

$$\bar{E}_{\text{data}}(v) := \sup_{z \in \mathcal{Z}} \int_{\Omega} \left( \sum_{t_1 \in \Gamma_1} z_1^{t_1}(x) v_1^{t_1}(x) + \dots + \sum_{t_k \in \Gamma_k} z_k^{t_k}(x) v_k^{t_k}(x) \right) dx. \quad (8.23)$$

The additional dual variables  $z = (z_i)_{1 \leq i \leq k}$  range over the convex set

$$\mathcal{Z} := \left\{ z \in L^2(\Omega, \mathbb{R}^{\sum_i n_i}) \text{ such that for all } x \in \Omega \text{ and } t \in \Gamma, \right. \\ \left. z_1^{t_1}(x) + \dots + z_k^{t_k}(x) \leq c^t(x) \right\}. \quad (8.24)$$

We first establish that the relaxation coincides with the original energy for binary functions.

**Proposition 8.2.** *Let  $v \in L^2(\Omega, \Delta_\times)$  be a binary function representing the label  $t(x) \in \Gamma$  at each point  $x \in \Omega$ , i.e.  $v_i^s(x) = \chi_{s=t(x)_i}$  for all  $x \in \Omega$ ,  $1 \leq i \leq k$ ,  $s \in \Gamma_i$ . Then*

$$\overline{E}_{data}(v) = \int_{\Omega} c^{t(x)}(x) dx = E_{data}(v). \quad (8.25)$$

In addition, one can prove the following proposition, which shows that the relaxation of the data term has the correct pointwise minimizers, in contrast to the one proposed in [53]. This means that no smoothing is necessary and an exact minimization algorithm can be employed to obtain solutions.

**Proposition 8.3.** *Let  $\hat{v} \in L^2(\Omega, \Delta_\times)$  be a binary minimizer of  $E_{data}$ . Then  $\hat{v}$  is also a minimizer of the relaxation,*

$$\hat{v} \in \operatorname{argmin}_{v \in L^2(\Omega, \operatorname{co}(\Delta_\times))} \overline{E}_{data}(v). \quad (8.26)$$

*In particular,  $E_{data}(\hat{v}) = \overline{E}_{data}(\hat{v}) = \int_{\Omega} c^*(x) dx$  with  $c^*(x) := \inf_{t \in \Gamma} (c^t(x))$  for  $x \in \Omega$ .*

In fact, it turns out that the proposed data term relaxation is the best possible one, being the convex envelope of the data term as stated in the proposition below. More specifically, this is up to the natural sum equality constraint  $\sum_{s \in \Gamma_i} v_i^s = 1$ , which is the first equation in (8.17) and is also used later in the definition (8.48) of the  $v$  domain in the overall optimization problem. To make this statement precise, we first need a general definition of the data term  $E_{data}$  for *all* indicator functions  $v \in L^2(\Omega, \mathbb{R}^{\sum_i n_i})$ , and not only for binary ones. If  $v$  is binary representing the label  $t(x) \in \Gamma$  at each  $x \in \Omega$ , i.e.  $v_i^s(x) = \chi_{s=t_i(x)}$ ,  $E_{data}$  is already defined by (8.18) through

$$E_{data}(v) = \int_{\Omega} c^{t(x)}(x) dx = \int_{\Omega} \sum_{t \in \Gamma} c^t(x) v_1^{t_1}(x) \cdots v_k^{t_k}(x) dx. \quad (8.27)$$

For all other  $v$  we set  $E_{data}(v) := \infty$ .

**Theorem 8.4.** *The convex envelope of  $E_{data}(v)$  is given by  $\overline{E}_{data}(v) + \delta_{\mathcal{S}}(v)$  with  $\mathcal{S} := \{v \mid \sum_{s \in \Gamma_i} v_i^s(x) = 1 \quad \forall 1 \leq i \leq k, x \in \Omega\}$ .*

## 8.4 Convex Relaxation of the Regularizer

### 8.4.1 Regularizer Class and its Convex Relaxation

**Regularizer Class.** Making use of the decomposition (7.3) of each scalar channel  $u_i$  into a smooth part and a jump part, we can introduce the framework for the regularization. We consider regularizers of the *separable* form (8.7), and choose each individual regularizer  $R_i$  to be of the form (7.6) (without the data term part), in order for the functional lifting relaxation to be applicable. Thus, we consider  $R_i$  of the form

$$R_i(u_i) = \int_{\Omega \setminus S_{u_i}} h_i(x, \nabla u_i(x)) dx + \int_{S_{u_i}} d_i(x, u_i^-(x), u_i^+(x)) d\mathcal{H}^{m-1}(x). \quad (8.28)$$

The functions  $h_i : \Omega \times \Gamma_i \times \mathbb{R}^m \rightarrow \mathbb{R}$  and  $d_i : \Omega \times \Gamma_i \times \Gamma_i \rightarrow \mathbb{R}$  have to satisfy the conditions stated after Lemma 7.1, i.e.

- $h_i(x, \cdot)$  is convex and lower-semicontinuous for fixed  $x \in \Omega$
- $d_i(x, \cdot, \cdot)$  is a metric on  $\Gamma_i$  for fixed  $x \in \Omega$ , and  $d_i$  is continuous.
- For all  $x \in \Omega$ ,  $t, t' \in \Gamma$  and  $\nu \in \mathbb{S}^{m-1}$ :

$$d_i(x, t, t') \leq (h_i)_\infty(x, \nu(t' - t)), \quad (8.29)$$

where  $(h_i)_\infty(x, p)$  denotes the recession function of  $h_i(x, p)$  w.r.t.  $p$ , as defined in (2.4).

The interesting task, of course, is to identify suitable choices of  $h_i$  and  $d_i$ , and to interpret what the choice means in practice. We will turn to this task later in this section beginning from Section 8.4.2. Before this, we will first introduce a convex relaxation of the general regularizer (8.28).

**Convex Relaxation.** The general lifting Lemma 7.1 gives a convex representation of the regularizer (8.28) in terms of the graph function (8.15) of the channel  $u_i$ . Since we work with the indicator functions  $v_i$  as defined in (8.16) instead, our goal is to arrive at an equivalent representation in terms these functions. We give this new reformulation in the following theorem.

**Theorem 8.5.** *Let  $R_i$  be of the form (8.28), and the indicator functions  $v_i$  defined as in (8.16). Then*

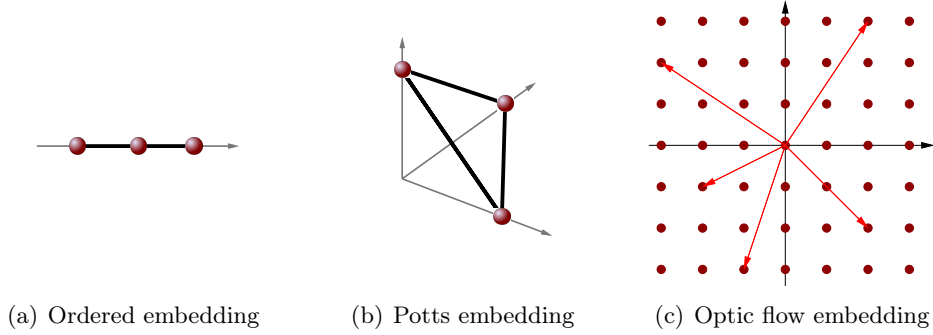
$$R_i(u_i) = \sup_{(p,q) \in \mathcal{C}_i} \int_{\Omega \times \Gamma_i} (-\operatorname{div}_x p - q) v_i \, d(x, s), \quad (8.30)$$

with the convex set

$$\mathcal{C}_i = \left\{ (p, q) \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R}) \mid \forall x \in \Omega, \, s, s' \in \Gamma_i, \right. \\ \left. \begin{aligned} q(x, s) &\geq h_i^*(x, s, \partial_s p(x, s)), \\ |p(x, s) - p(x, s')| &\leq d_i(x, s, s') \end{aligned} \right\}. \quad (8.31)$$

Note that similarly to the discrete version of the indicator functions, the discrete version of the set  $\mathcal{C}_i$  in (8.31) will consist of tuples  $(p^s, q^s)_{s \in \Gamma_i}$  of functions. Taking a closer look at equation (8.30), we can see that the right hand side is a convex functional in the new variables  $v_i$ .

Thus, we have achieved our goal and can turn towards exploring suitable choices of the regularizer, and how they fit within the proposed framework. In particular, we will see how our model can be specialized to the case of discrete label spaces where the label distance has an Euclidean representation. This special case (3.31) was discussed in [76, 53], and we will see that our framework leads to a tighter relaxation for this case. We will also discuss additional continuous regularizers which become possible based on the lifting framework. These were introduced in the previous works [29, 100, 101] when the unknowns were the characteristic functions of the subgraphs of  $u_i$ . We show how we can accommodate them to depend on the indicator variables instead. Notably, in each dimension of the label space its own type of regularization can be chosen, in particular discrete and continuous regularizers can be mixed freely.



**Figure 8.4: Different embeddings for a label space.** In an ordered embedding, all labels are mapped onto a line, while for the Potts model, every label is mapped onto a different unit vector. For optical flow, each label is already a vector in  $\mathbb{R}^2$ , so a sensible embedding is given by the identity.

### 8.4.2 Discrete Label Space: Euclidean Representation

We first consider the special case of a discrete label space  $\Gamma_i$ . Thus, we need to define a regularizer  $R_i : L^2(\Omega, \text{co}(\Delta_i)) \rightarrow \mathbb{R}$  for functions with values in the convex hull of the simplex  $\Delta_i$ . We first present the construction used in [76, 53], and then show in Section 8.4.3 how we can embed it into our more general framework resulting in a tighter relaxation.

We assume that the metric  $d_i$  has an *Euclidean* representation in the sense of (3.29) in Section 3.4.3, i.e. that there are  $M_i$ -dimensional vectors  $a_i^s \in \mathbb{R}^{M_i}$  with a  $M_i \geq 1$  such that the distance function  $d_i$  in (8.28) is given by

$$d_i(s, s') = |a_i^s - a_i^{s'}| \quad \forall s, s' \in \Delta_i. \quad (8.32)$$

We can define a regularizer with desirable properties by (3.31), i.e.

$$R_i^A(v_i) := \lambda TV \left( \sum_{i=1}^{n_i} a_i^s v_i \right) \quad (8.33)$$

with a scaling factor  $\lambda > 0$ . The following theorem has been proven in [76] and shows why the above definition makes sense.

**Theorem 8.6.** *The regularizer  $R_i^A$  defined in (8.33) has the following properties:*

1.  $R_i^A$  is convex and positive homogeneous on  $L^2(\Omega, \text{co}(\Delta_i))$ .
2.  $R_i^A(v_i) = 0$  for any constant labeling  $v_i$ .
3. If  $S \subset \Omega$  has finite perimeter  $\text{Per}(S; \Omega)$ , then for all labels  $s, s' \in \Gamma_i$ ,

$$R_i^A(s 1_S + s' 1_{S^c}) = \lambda d_i(s, s') \text{Per}(S; \Omega), \quad (8.34)$$

*i.e. a change in labels is penalized proportionally to the distance between the labels and the perimeter of the interface.*

For the sake of simplicity, we only give the main examples for distances with Euclidean representations. More general classes of distances on the labels can also be used, see [76].

- The case of ordered labels, where the embedding follows the natural ordering of  $s, s' \in \mathbb{R}$ , Figure 8.4(a), for example by setting simply  $a_i^s = s$ . If  $k = 1$ , then this case can be solved in a globally optimal way using the lifting method [101].
- The Potts or uniform distance, where  $d_i(s, s') = 1$  if and only if  $s = s'$ , and zero otherwise. This distance function can be achieved by setting  $a_i^s = \frac{1}{\sqrt{2}}e^s$ , where  $(e^s)_{s \in \Gamma_i}$  is an orthonormal basis in  $\mathbb{R}^{n_i}$ , see Figure 8.4(b). All changes between labels are penalized equally.
- Another typical case is that the  $a_i^s$  denote feature vectors or actual geometric points, for which  $|\cdot|$  is a natural distance. For example, in the case of optic flow, each label corresponds to a flow vector in  $\mathbb{R}^2$ , see Figure 8.4(c). The representations  $a_1^s, a_2^{s'}$  are just real numbers, denoting the possible components of the flow vectors in  $x$ - and  $y$ -direction, respectively. The Euclidean distance is a sensible distance on the components to regularize the flow field, corresponding to the regularizer of the  $TV$ - $L^1$  functional in [142]. Optic flow (and other geometric kinds of labels) would however more naturally be modeled with a continuous label space using one of the continuous regularizers in the later subsections.

### 8.4.3 Discrete Label Space: New Relaxation

We will now show how to formulate the regularizer  $R_i^A$  defined above in the new more general framework. While the previous formulation (8.33) already yields a relaxation to nonbinary functions  $v$ , we will see that our framework results in a provably tighter one.

Taking a look at Theorem 8.6, we see that the regularizer must penalize the length of the jump set weighted by the label distance. Thus, our general regularizer in (8.28) must reduce to

$$R_i(u_i) = \lambda \int_{S_{u_i}} d_i(u_i^-, u_i^+) d\mathcal{H}^{m-1}, \quad (8.35)$$

where  $d_i$  is the same metric as used above in the representation (8.32), and  $\lambda > 0$  is a weighting constant. We can see that in order to reduce the general form to the one above, we must enforce a piecewise constant labeling, since the approximate gradient  $\nabla u_i$  must be constantly zero outside the jump set. Applying Theorem 8.5 we can find a convex representation of  $R_i$  in terms of the variables  $v$ , which we formulate in the following proposition in its discretized form.

**Proposition 8.7.** *A convex representation of (8.35) in terms of the variables  $v$  is given by*

$$R_i(u_i) = \sup_{p \in \mathcal{C}_i} \sum_{s \in \Gamma_i} \int_{\Omega} (-\operatorname{div}_x p^s) v_i^s dx, \quad (8.36)$$

with

$$\begin{aligned} \mathcal{C}_i = \left\{ p : \Omega \times \Gamma_i \rightarrow \mathbb{R}^m \mid p^s \in C_c^1(\Omega; \mathbb{R}^m) \quad \forall s \in \Gamma_i, \right. \\ \left. |p^s - p^{s'}| \leq \lambda d_i(s, s') \quad \forall s, s' \in \Gamma_i \right\}. \end{aligned} \quad (8.37)$$

We note that this is the version for *continuous* label spaces of the analogous finite multilabel relaxation (3.23) with the constraint set (3.33).

We can now establish the relationship between our framework and the regularizer  $R_i^A$  in (8.33) which was used in [76, 53], and show that ours is tighter and thus leads to solutions closer to the global optimum.

**Proposition 8.8.** *Let the regularizer  $R_i$  be defined by the relaxation on the right hand side in equation (8.36). Then for all  $v_i \in L^2(\Omega, \text{co}(\Delta_i))$ ,*

$$R_i(v_i) \geq R_i^A(v_i). \quad (8.38)$$

*Equality holds if  $v_i$  is binary.*

We will show in the remainder of the section that in addition to handling the discrete case better, our method also can handle continuous regularizers which penalize a *smooth variation* of the labels. This is not possible with the piecewise constant approach of [76, 53] which uses vectorial total variation. For instance, our formulation is capable of representing more sophisticated regularizers such as Huber-TV and the piecewise smooth Mumford-Shah functional, as we will show in the following section. For the regularizers presented in the remainder of this section, relaxations have previously been proposed for the case of a one-dimensional label space in [29, 100, 101, 122]. However, the framework presented here is more general and allows to combine them freely in the different label dimensions.

#### 8.4.4 Linear (TV) and Truncated Linear

For many applications, it is useful to penalize the difference between two label values  $s$  and  $s'$  only up to a certain threshold, reasoning that once they are that different, it does not matter anymore how different they are exactly. This means that if  $|s - s'|$  becomes greater than a certain value  $T$ , jumps from  $s$  to  $s'$  are still penalized, but only by the constant  $T$ . Using linear penalization for small values this leads to the robust *truncated linear* regularizer [29]

$$R_i(u_i) = \lambda \int_{\Omega \setminus S_{u_i}} |\nabla u_i| \, dx + \int_{S_{u_i}} \min(T, \lambda |u_i^+ - u_i^-|) \, d\mathcal{H}^{m-1} \quad (8.39)$$

with a  $\lambda > 0$ . The constraint set (8.31) for this case is

$$\begin{aligned} \mathcal{C}_i = \left\{ (p, q) \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R}) \text{ such that for all } s, s' \in \Gamma_i, \right. \\ \left. |\partial_s p^s| \leq \lambda, \quad |p^s - p^{s'}| \leq T, \quad q \equiv 0 \right\}. \end{aligned} \quad (8.40)$$

The second constraint needs to be imposed only if  $|s - s'| \geq T$ , since otherwise it is already implied by the first constraint. In particular, the standard linear (TV) penalizer can be implemented by letting  $T \rightarrow \infty$  and only using the first constraint.



### 8.4.5 Cyclic Penalizer, $TV\text{-}\mathcal{S}^1$

Some applications have a cyclic or circular set of labels, for example regularization in the hue component in HSV or HSL color space. In this case, the distance between the last and first label is the same as between any other subsequent pair of labels. This form of regularization was discussed in the functional lifting setting in the recent work [122], and can be expressed in our framework by setting

$$R_i(u_i) = \lambda \int_{\Omega \setminus S_{u_i}} |\nabla u_i| \, dx + \lambda \int_{S_{u_i}} \min(u_i^+ - u_i^-, 1 - (u_i^+ - u_i^-)) \, d\mathcal{H}^{m-1} \quad (8.41)$$

for functions  $u_i$  with range  $\Gamma_i := [0, 1]$ . The corresponding constraint set is given by

$$\mathcal{C}_i = \left\{ (p, q) \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R}) \text{ such that for all } s \in \Gamma_i, \right. \\ \left. |\partial_s p^s| \leq \lambda, \quad p(x, 0) = p(x, 1), \quad q \equiv 0 \right\}. \quad (8.42)$$

Note that the constraints are the same as for the usual  $TV$  regularizer in Section 8.4.4 (with  $T = \infty$ ), except for the additional periodicity constraint on  $p$ .

### 8.4.6 Huber- $TV$

The  $TV$  regularization is known to produce staircasing effects in the reconstruction, i.e. the solution will be piecewise constant. While this is natural in the case of a discrete label space, for continuous label spaces it impedes smooth variations of the solution. A remedy for this is replacing the norm  $|\nabla u_i|$  of the gradient by  $h_\alpha(\nabla u_i)$  with the Huber function

$$h_\alpha(z) := \begin{cases} \frac{1}{2\alpha} |z|^2 & \text{if } |z| < \alpha, \\ |z| - \frac{\alpha}{2} & \text{else} \end{cases} \quad (8.43)$$

for some  $\alpha > 0$ , which smooths out the kink at the origin. The Huber- $TV$  regularizer is then defined by

$$R_i(u_i) = \int_{\Omega} h_\alpha(\nabla u_i) \, dx + \int_{S_{u_i}} |u_i^+ - u_i^-| \, d\mathcal{H}^{m-1}. \quad (8.44)$$

The limiting case  $\alpha = 0$  leads to the usual  $TV$  regularization. Theorem (8.5) gives a convex representation for  $R_i$ , see also [101]. The constraint set in (8.31) is found to be

$$\mathcal{C}_i = \left\{ (p, q) \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R}) \text{ such that for all } s \in \Gamma_i, \right. \\ \left. q^s \geq \frac{\alpha}{2} |\partial_s p^s|^2, \quad |\partial_s p^s| \leq 1 \right\}. \quad (8.45)$$

### 8.4.7 Piecewise Smooth Mumford-Shah Model

The celebrated Mumford-Shah regularizer

$$R_i(u_i) = \int_{\Omega \setminus S_{u_i}} \frac{1}{2\alpha} |\nabla u_i|^2 dx + \nu \mathcal{H}^{m-1}(S_{u_i}) \quad (8.46)$$

with parameters  $\alpha, \nu > 0$  allows to estimate a denoised image  $u_i$  which is *piecewise smooth*. Parameter  $\nu$  can be used to easily control the total length of the jump set  $S_{u_i}$ . Bigger values of  $\nu$  lead to a smaller jump set, i.e. the solution will be smooth on wider subregions of  $\Omega$ . We will consider this functional in detail for the general vectorial case in Chapter 10.

The constraint set in the convex representation of Theorem 8.5 becomes

$$\mathcal{C}_i = \left\{ (p, q) \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R}) \text{ such that for all } s, s' \in \Gamma_i, \right. \\ \left. q^s \geq \frac{\alpha}{2} |\partial_s p^s|^2, \quad |p^s - p^{s'}| \leq \nu \right\}. \quad (8.47)$$

The limiting case  $\alpha = 0$  gives the piecewise constant Mumford-Shah regularizer (also called Potts regularizer), which can also be obtained from Proposition 8.7 by setting  $d_i(s, s') = \nu$  for all  $s \neq s'$ . Compared to (8.33), this alternative yields a tighter, but more memory intensive relaxation for the Potts regularizer.

## 8.5 Implementation

### 8.5.1 Final Relaxation to a Convex Problem

**Domain Relaxation.** In order to transform the multilabel problem into the final form which we are going to solve, we formulate it in terms of the indicator functions  $v_i^s$  on the discretized label space using the relaxation (8.23) of the data term and the relaxation (8.30) for the regularizer. Discretization of the label space is necessary now to arrive at a well-posed problem. Although it is possible to formulate the optimization problem without discretization, this requires to work in general measure spaces, which would significantly complicate the notation and clarity at only marginal gain in generality.

Let us briefly summarize and review the objects we are dealing with in the final problem. The minimizer we are looking for is a vector  $v = (v_i)_{1 \leq i \leq k}$  of functions  $v_i \in L^2(\Omega, \text{co}(\Delta_i))$ . Thus we optimize for  $v$  in the convex set

$$\mathcal{D} := \left\{ v \in L^2(\Omega, \mathbb{R}^{\sum_i n_i}) \text{ such that for all } x \in \Omega, v(x) \in \text{co}(\Delta_x), \right. \\ \left. \text{with } \Delta_x = \Delta_1 \times \dots \times \Delta_k \right\}. \quad (8.48)$$

In the convex hull

$$\text{co}(\Delta_x) = \text{co}(\Delta_1) \times \dots \times \text{co}(\Delta_k) \subset \mathbb{R}^{\sum_i n_i}, \quad (8.49)$$

each  $\text{co}(\Delta_i)$  is given by the same expression as in (8.11) but with the set  $\{0, 1\}$  replaced by  $[0, 1]$ :

$$\text{co}(\Delta_i) = \left\{ x \in [0, 1]^{n_i} \mid \sum_{j=1}^{n_i} x_j = 1 \right\} \subset \mathbb{R}^{n_i}. \quad (8.50)$$

Let us now turn to the regularizer, which is defined via the relaxation in Theorem 8.5. The key ingredients are the convex sets  $\mathcal{C}_i$  which depend on the kind of regularization we want to use. Possible options were detailed in the last section.

**Final Problem.** Let

$$\mathcal{C} := \mathcal{C}_1 \times \dots \times \mathcal{C}_k \quad (8.51)$$

denote the convex set of all regularizer dual variables, where the individual constraint sets  $\mathcal{C}_i$  in (8.31) are defined according to the used regularizers as described in Section 8.4. Then Theorem 8.5 in fact shows that the regularizer can be written in terms of  $v$  in the simple form

$$R(v) = \sup_{(p,q) \in \mathcal{C}} \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} (-\text{div}_x p_i^s - q_i^s) v_i^s \, dx. \quad (8.52)$$

The fully relaxed problem we are going to solve can now be written as

$$\underset{v \in \mathcal{D}}{\text{argmin}} \mathcal{E}(v), \quad \mathcal{E}(v) := \overline{E}_{\text{data}}(v) + R(v), \quad (8.53)$$

using the relaxation  $\overline{E}_{\text{data}}$  of the data term defined in (8.23). It is straightforward to prove the existence of solutions:

**Proposition 8.9.** *Problem (8.53) always has a minimizer  $v^* \in \mathcal{D}$ .*

**Optimality and Energy Bounds.** After obtaining a solution  $v^*$  of the final relaxation (8.53), the question remains of whether it corresponds to a function  $u^*$  which solves the original problem (8.1). Note that because of the relaxation,  $v^*$  might not be binary, i.e. the values  $v_i(x)$  might not lie in  $\Delta_i$ .

If  $v^*$  is already binary, we have found the global optimum of the original problem.

Otherwise we have to project the result back to the smaller set of binary valued functions. Using the relation (8.13), the function  $u^* \in L^2(\Omega, \Gamma)$  closest to  $v^*$  is given by setting

$$u_{\text{bin}}(x) = \underset{t \in \Gamma}{\text{argmax}} (v^*)_1^{t_1}(x) \cdot \dots \cdot (v^*)_k^{t_k}(x). \quad (8.54)$$

In other words, at each image point  $x \in \Omega$  we choose the label  $t$  where the combined indicator functions have the highest value. This is the same as choosing the label by maximizing each component  $v_k$  separately:

$$u_{\text{bin}}(x) = t, \quad \text{with } t_i = \underset{s \in \Gamma_i}{\text{argmax}} (v^*)_i^s(x) \quad \text{for all } 1 \leq i \leq k. \quad (8.55)$$

We cannot guarantee that the solution candidate  $u_{\text{bin}}$  is indeed a global optimum of the original problem (8.1), since there is nothing equivalent to the thresholding theorem [33] known for this kind of relaxation. However, we still can give an energy bound (1.6) on how close we are to the global optimum:

$$\mathcal{E}(v^*) \leq E(u_{\text{bin}}^*) \leq E(u_{\text{bin}}), \quad (8.56)$$

where  $u_{\text{bin}}^*$  is the unknown optimal solution of (8.1).

### 8.5.2 Numerical Method

**Discretized Energy.** We discretize the image domain  $\Omega$  into a finite pixel grid, again denoted by  $\Omega$ . Using the representation (8.52) for  $R$ , and the definition (8.23) for the relaxation  $\bar{E}_{\text{data}}$ , we can transform the final formulation (8.53) of the multilabel problem into the saddle point problem

$$\min_{v \in \mathcal{D}} \max_{\substack{(p,q) \in \mathcal{C} \\ z \in \mathcal{Z}}} \sum_{x \in \Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} \left( -\text{div}_x p_i^s(x) - q_i^s(x) + z_i^s(x) \right) v_i^s(x), \quad (8.57)$$

with the straightforwardly discretized versions of the constraint sets  $\mathcal{D}$  in (8.48),  $\mathcal{C}$  in (8.51), and  $\mathcal{Z}$  in (8.24).

The spatial gradient  $\nabla_x$  is discretized using forward differences with Neumann boundary conditions. The discrete divergence  $\text{div}_x$  is then the corresponding negative adjoint operator, which is given by backward differences with Dirichlet boundary conditions. The discretization of the constraint sets  $\mathcal{D}$  in (8.48),  $\mathcal{Z}$  in (8.24) are straightforward as all constraints are defined per pixel.

The individual regularizer constraint sets  $\mathcal{C}_i$  in (8.31) are initially defined for the case of a continuous range space  $\Gamma_i$ . For optimization, we discretize the range  $\Gamma_i = [s_i^{\min}, s_i^{\max}]$  into  $n_i$  equidistant labels

$$s_i^j = s_i^{\min} + j \Delta s_i, \quad 0 \leq j \leq n_i - 1, \quad \text{where } \Delta s_i = \frac{s_i^{\max} - s_i^{\min}}{n_i - 1}. \quad (8.58)$$

The derivative  $(\partial_s p)(x, s)$  w.r.t. the labels in (8.31) is discretized using backward differences with Dirichlet boundary conditions:

$$(\partial_s p)(x, s_i^j) = \frac{1}{\Delta s_i} \left( p(x, s_i^j) - p(x, s_i^{j-1}) \right), \quad \text{with } p(x, s_i^{-1}) = 0. \quad (8.59)$$

If for a channel  $i$  one uses the cyclic regularizer from Section 8.4.5, the end points of  $\Gamma_i$  actually represent the same value. Therefore in this case  $\Gamma_i$  is discretized slightly differently, namely using  $\Delta s_i = \frac{s_i^{\max} - s_i^{\min}}{n_i}$ . The derivative  $(\partial_s p)(x, s)$  is then discretized as above, but with periodic boundary conditions, i.e. setting  $p(x, s_i^{-1}) = p(x, s_i^{n_i-1})$ .

We minimize the energy (8.57) using the primal-dual Algorithm 3. The proximal operators in our case are just the usual orthogonal projections onto the respective constraint sets  $\mathcal{D}$ ,  $\mathcal{C}$  and  $\mathcal{Z}$ .

**Decouplings of Hard Constraints.** Since these sets are defined by numerous nonlocal constraints, a direct projection is quite costly. Therefore, we suggest to implement as many constraints as possible using Lagrange multipliers by adding specific additional terms to the energy. This comes at the cost of having more terms in the final overall energy and that the optimization is done also over additional variables, the Lagrange multipliers. However, in the end less of the explicit constraints remain, so that the projections become easier to calculate. Also, the algorithm complexity remains the same since the update equations are still straightforward.

First, the simplex constraint  $v \in \mathcal{D}$ , i.e.  $v_i \in \text{co}(\Delta_i)$  with  $\Delta_i$  in (8.11) for  $1 \leq i \leq k$ , is enforced by adding the Lagrange multiplier terms

$$\sup_{\sigma} \sum_{x \in \Omega} \sum_{i=1}^k \sigma_i(x) \left( \sum_{s \in \Gamma_i} v_i^s(x) - 1 \right) \quad (8.60)$$

to the energy (8.57), optimizing over  $\sigma : \Omega \rightarrow \mathbb{R}^k$  in addition to the other variables. This leaves just the simple condition  $v \geq 0$  for the indicator variables  $v$ . We note that it is also possible to implement the simplex constraint explicitly by the iterative algorithm [84]. However, in the end this increases the computation time per iteration many times over since the projection then requires  $\mathcal{O}(\sum_i n_i)$  steps in the worst case. Also, the explicit projection only slightly reduces the number of iterations needed to compute a minimizer of (8.57) to a certain precision. Therefore, overall the Lagrange multiplier approach turns out to be faster and is also easier to implement.

Next, we enforce the constraints  $(p, q) \in \mathcal{C}$  on the dual variables of the regularizer by introducing new variables

$$d_i^s = \partial_s p_i^s \quad \text{or} \quad d_i^{s,s'} = p_i^s - p_i^{s'}, \quad (8.61)$$

depending on the kind of constraints in  $\mathcal{C}_i$ . To enforce these equalities, we add the corresponding Lagrange multiplier terms through the use of (2.74):

$$\inf_{\eta} \sum_{x \in \Omega} \left\langle \eta_i^s(x), \partial_s p_i^s(x) - d_i^s(x) \right\rangle \text{ or } \inf_{\eta} \sum_{x \in \Omega} \left\langle \eta_i^{s,s'}(x), p_i^s(x) - p_i^{s'}(x) - d_i^{s,s'}(x) \right\rangle \quad (8.62)$$

for each  $1 \leq i \leq k$  and  $s \in \Gamma_i$ , respectively  $s, s' \in \Gamma_i$  to the energy. Instead of computing the projection of  $(p, q)$  in each step, we can then perform the projection of the new variables  $(d_i, q_i)$  on a corresponding constraint set. The advantage is that the overall projection decouples into independent projections of  $d_i^s$  or  $d_i^{s,s'}$  and  $q_i^s$  onto simple convex sets, which are easy to implement. Alternatively, constraints of the form  $|p_i^s - p_i^{s'}| \leq M$  as used in (8.37), (8.40) and (8.47) can be enforced using convex duality, by adding the terms

$$\inf_{\eta} \sum_{x \in \Omega} \left\langle \eta_i^{s,s'}(x), p_i^s(x) - p_i^{s'}(x) \right\rangle + M |\eta_i^{s,s'}(x)| \quad (8.63)$$

to the energy instead of (8.62). We used this way in our implementation, as it turns out to be much faster in practice. The optimization (8.57) is now performed over the primals  $v, d, \eta$  and the duals  $p, q, z, \sigma$ .

Finally, the projection of a  $\tilde{z}$  onto  $\mathcal{Z}$  consists of solving

$$\operatorname{argmin}_{z \in \mathcal{Z}} \sum_{x \in \Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} (z_i^s(x) - \tilde{z}_i^s(x))^2. \quad (8.64)$$

Since the constraints in  $\mathcal{Z}$  are pointwise in  $x \in \Omega$ , (8.64) can be solved separately and independently for each  $x \in \Omega$ . The number of constraints in  $\mathcal{Z}$ , as defined in (8.24), equals the total number of labels in the product space  $\Gamma = \Gamma_1 \times \dots \times \Gamma_k$ . Unfortunately, implementing these constraints by adding Lagrange multiplier terms

$$\inf_{\mu \geq 0} - \sum_{x \in \Omega} \sum_{t \in \Gamma} \mu^t(x) \left( z_1^{t_1}(x) + \dots + z_k^{t_k}(x) - c^t(x) \right) \quad (8.65)$$

to the *global* problem (8.57), i.e. for each  $x \in \Omega$ , is not possible for larger problems since it requires too many additional variables  $\mu$  to be memory efficient.

Thus, for larger problems, the projection needs to be computed explicitly after each outer iteration as a *subproblem* by solving (8.64), which increases the run time, see Table 8.3. To make sure that  $z$  lies in  $\mathcal{Z}$  we add the corresponding Lagrange multiplier terms to the *local* energy (8.64). This results in another saddle point problem to be optimized over now unconstrained  $z$  and  $\mu \geq 0$ . For this we can use the accelerated primal-dual Algorithm 5, since the  $z$ -only terms are uniformly convex. Since there is only a small change in the variables  $z$  per outer iteration, only a small number of inner iterations is required. In our experiments, we used 10 inner iterations.

### 8.5.3 Memory Requirements

When the domain  $\Omega$  is discretized into  $|\Omega|$  pixels, the primal and dual variables are represented as matrices. There are essentially two types of data. The first type is relatively cheap to store, since memory requirements scale with the sum  $\sum_i n_i$  of the independent dimensions. The second type is expensive, since it scales with  $\prod_i n_i$ . The variables and constants appearing in the energy are classified in Table 8.1, where  $m = \dim \Omega$  is the dimension of the image domain  $\Omega$  and  $k$  is the number of channels.

The relaxation of the regularizer incurs additional costs per label space dimension, depending on the type of regularizer. This is summarized in Table 8.2. Obviously, truncated linear and piecewise smooth regularization can require a lot of additional memory if the dimensions of the factors are large. They seem therefore practically feasible only when the label space consists of many small factors. However, the projections onto the regularizer constraints can also be solved locally in each iteration for each dimension separately. This removes the need to store these variables globally at the cost of additional computation time.

The most expensive variable is  $\mu$ , appearing in the global problem (8.65) or local problem (8.64), respectively. If we have enough memory to store it, it is more efficient to solve the global problem. However, it is also possible here to trade off computation time for a reduction in memory requirements by solving the local problem (8.64) in each iteration. For this, note that (8.64) can be separated into *independent* subproblems for each  $x \in \Omega$  and can be solved in

Variable or constant	Floating point numbers
$\sigma$	$ \Omega  \cdot k$
$v, z$	$ \Omega  \cdot \sum_i n_i$
$p$	$m \Omega  \cdot \sum_i n_i$
$\mu, c$	$ \Omega  \cdot \prod_i n_i$

**Table 8.1: Number of floating point numbers for the energy.** Shown are the numbers for the basic variables and constants always appearing in the primal-dual model. Different regularizers will need additional variables as shown in Table 8.2.

Regularizer	Additional variables	Additional floating point numbers
Potts by (8.33)	–	–
$TV$ or cyclic $TV$	$\eta_i^s, d_i^s$	$2m \Omega  \cdot n_i$
Huber- $TV$	$\eta_i^s, d_i^s, q_i^s$	$(2m + 1) \Omega  \cdot n_i$
Truncated linear	$\eta_i^{s,s'}$	$m \Omega  \cdot n_i(n_i - 1)/2$
Potts by (8.47), $\alpha = 0$	$\eta_i^{s,s'}$	$m \Omega  \cdot n_i(n_i - 1)/2$
Piecewise smooth	$\eta_i^{s,s'}, \eta_i^s, d_i^s, q_i^s$	$m \Omega  \cdot n_i(n_i - 1)/2$ $+ (2m + 1) \Omega  \cdot n_i$

**Table 8.2: Additional number of floating point numbers depending on the regularizer.** These are numbers per channel since each regularizer works on one specific channel only.

chunks of points  $x$  in parallel. The size of the chunks can be chosen to fit into the available memory, ideally we choose it as large as possible for maximum parallelization.

Finally, note that the data term  $c$  is an expensive constant to store. If there is not enough GPU memory for it, it is possible to e.g. hold it in main memory and transfer separate layers of it to the GPU during computation of the primal prox operator. This increases computation time by a factor of 5–10, so it is usually much more efficient to compute the data term on the fly on the GPU if it is of a simple form.

In summary, with the above reduction techniques it is possible to get rid of all memory expensive variables and constants at the cost of more computation time. To give an idea about the final requirements, they are compared for the case of a 2D label space and  $TV$  penalization in Table 8.3. Note that the memory requirements for the original method without using the reduction are  $(m + 2)|\Omega| \cdot (n_1 \cdots n_k)$  to store all primal and dual variables even for the most simple regularizer, while all regularizer costs scale with  $n = n_1 \cdots n_k$  according to the table on the previous page. Statistics for a 3D label space with different regularizers can be found in Table 8.4. Clearly, large scale problems can only be solved using the proposed reduction technique.

# of Pixels $P = P_x \times P_y$	# Labels $N_1 \times N_2$	Memory [MB]		Run time [s]	
		Previous	Proposed (g/p)	Previous	Proposed (g/p)
$320 \times 240$	$8 \times 8$	112	112 / 102	31	26 / 140
$320 \times 240$	$16 \times 16$	450	337 / 168	125	80 / 488
$320 \times 240$	$32 \times 32$	1800	1124 / 330	503	215 / 1953
$320 \times 240$	$50 \times 50$	4394	2548 / 504	2110	950 / 5188
$320 \times 240$	$64 \times 64$	7200	4050 / 657	-	1100 / 8090
$640 \times 480$	$8 \times 8$	448	521 / 413	127	102 / 560
$640 \times 480$	$16 \times 16$	1800	1351 / 676	539	295 / 1945
$640 \times 480$	$32 \times 32$	7200	4502 / 1327	-	1290 / 7795
$640 \times 480$	$50 \times 50$	17578	10197 / 2017	-	- / 32887
$640 \times 480$	$64 \times 64$	28800	16202 / 2627	-	- / 48583

**Table 8.3: Memory requirements.** The table shows the total amount of memory required for the implementations of the previous and proposed methods depending on the size of the problem (using  $TV$  regularization). For the proposed method, the projection (8.64) of the data term dual variables can be implemented either globally ( $g$ ), or slower but more memory efficient as a sub-problem of the proximation operator ( $p$ ), here using  $N_1/5$  chunks. Also shown is the total run time for 5000 iterations, which usually suffices for convergence. Numbers in red indicate a memory requirement larger than what fits on the largest currently available CUDA capable devices (6 GB). Note that the proposed framework can still handle all problem sizes above.

## 8.6 Experimental Results

We demonstrate the correctness and usability of our method on several examples. Different regularizers are used. In the cases where the regularizer can be simulated with the previous relaxation [53], we compared the resulting optimality bounds. On average, our bounds were approximately three times better (3–5% with the proposed framework compared to 10–15% with the previous relaxation). All experiments were performed with a parallel CUDA implementation running on a NVIDIA GTX 680 GPU for Section 8.6.1, respectively on a TESLA C2070 for all other experiments. The number of iterations in each experiment is chosen appropriately so that visually the solution remains stable and does not change anymore (usually 1000–5000 depending on problem size).

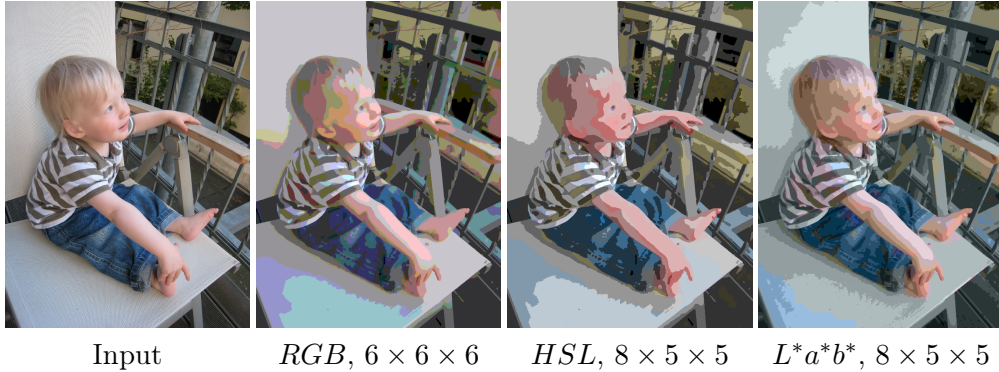
### 8.6.1 Segmentation

Multidimensional label spaces occur naturally in the problem of image segmentation, where a multi-channel input image  $f$  is segmented according to a local cost equal to the squared distance to the labels,

$$c(x, t) = \sum_{i=1}^k (f_i(x) - t_i)^2. \quad (8.66)$$

Typical multi-channel images are of course color images with various color spaces which are usually three-dimensional, see Figure 8.5 for some segmentation examples.





**Figure 8.5: Segmentation using different three-dimensional spaces of equidistant color labels.** The perceptually uniform color space CIELAB gives the visually most compelling results with the most natural looking color reproduction. Linear penalizer in all channels, except for the  $H$  channel, which requires cyclic regularization. Less than two minutes run time and less than 3% from global optimum for all results.

We choose this archetypical problem for an extensive comparison of our method to different labeling approaches. We first compare our proposed scheme for vectorial multilabel problems (VML) to the similar continuous method for a scalar label space (SML) [76]. For comparisons with discrete approaches, we used the MRF software accompanying the comparative study in [128]<sup>2</sup>. We compare to the  $\alpha$ -expansion ( $\alpha$ -EXP) and  $\alpha$ - $\beta$ -swap (SWAP) algorithms based on the maxflow library [19, 18, 71] using the newest updated implementation for [41]<sup>3</sup>. We also compare to two message-passing methods, max-product belief propagation (BP) [129] and sequential tree-reweighted message passing (TRW-S) [134, 69].

Table 8.4 shows detailed statistics of memory requirements, run time and error bounds. Throughout all experiments, we used a fixed number of iterations to keep results comparable, tuned to be sufficient for convergence on intermediate-sized label spaces. For VML and SML, we used 1000 iterations, 50 iterations for TRW-S and BP, and 5 iterations for  $\alpha$ -EXP and SWAP.

**Memory requirements and largest problem size.** Our method can deal with much larger problems than any of the other algorithms. On a high-end workstation with 64 GB of main memory, the generic implementation of TRW-S already stops working at  $12^3$  labels for the Potts model. However, the implementation is extremely general and requirements could probably be reduced by more specialized code. About the largest problem which can be solved with  $\alpha$ -EXP has  $15^3$  labels, after which the reference implementation segfaults — this hints at a hidden limitation of the implementation, memory-wise it should be able to cope with about  $17^3$ . In contrast, the GPU has only 4 GB of memory, and our method can still handle problems up to  $31^3$  (almost 30000 labels),

<sup>2</sup>MRF 2.1, <http://vision.middlebury.edu/MRF/code/>

<sup>3</sup>gco-v3 library, <http://vision.csd.uwo.ca/code/>

**Potts regularizer (using (8.33) for the continuous methods)**

Algorithm	$4 \times 4 \times 4$			$6 \times 6 \times 6$		
	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>
VML	367	15.25	0.46	998	42.83	0.72
SML	607	16.81	1.43	> 4 GB	–	–
$\alpha$ -EXP	677	25.13	0.20 <sup>4</sup>	1369	88.90	0.54 <sup>4</sup>
SWAP	677	29.84	0.28 <sup>5</sup>	1369	137.57	0.81 <sup>5</sup>
BP	1368	36.63	6.94 <sup>5</sup>	4256	119.43	11.56 <sup>5</sup>
TRW-S	1368	40.06	0.16	4256	129.52	0.80
	$8 \times 8 \times 8$			$10 \times 10 \times 10$		
	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>
VML	2173	94.59	1.03	2185 <sup>6</sup>	209.34 <sup>6</sup>	0.80
$\alpha$ -EXP	2746	173.64	0.90 <sup>4</sup>	5020	343.67	1.01 <sup>4</sup>
SWAP	2746	461.48	1.34 <sup>5</sup>	5020	1472.45	1.62 <sup>5</sup>
BP	8667	254.08	16.29 <sup>5</sup>	16610	496.12	17.73 <sup>5</sup>
TRW-S	8667	287.30	1.95	16610	539.51	2.70

**Linear (TV) regularizer**

Algorithm	$4 \times 4 \times 4$			$6 \times 6 \times 6$		
	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>
VML	401	18.67	1.50	1055	48.09	1.67
$\alpha$ -EXP	677	18.04	0.06 <sup>4</sup>	1369	82.70	– <sup>4</sup>
SWAP	677	23.67	0.16 <sup>5</sup>	1369	139.90	– <sup>5</sup>
BP	51000 <sup>8</sup>	740.49 <sup>8</sup>	5.87 <sup>5</sup>	> 64 GB	–	–
TRW-S	51000 <sup>8</sup>	746.00 <sup>8</sup>	0.04	> 64 GB	–	–
	$8 \times 8 \times 8$			$10 \times 10 \times 10$		
	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>
VML	2253	101.75	2.27	2287 <sup>6</sup>	217.66 <sup>6</sup>	3.10
$\alpha$ -EXP	2746	201.05	– <sup>4</sup>	5020	408.93	– <sup>4</sup>
SWAP	2746	504.47	– <sup>5</sup>	5020	1576.32	– <sup>5</sup>

**Truncated linear regularizer**

Algorithm	$4 \times 4 \times 4$			$6 \times 6 \times 6$		
	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>
VML	435	20.77	0.95	1168	55.01	1.83
$\alpha$ -EXP	677	18.13	0.09 <sup>4</sup>	1369	82.78	– <sup>4</sup>
SWAP	677	23.68	0.18 <sup>5</sup>	1369	139.30	– <sup>5</sup>
BP	51000 <sup>8</sup>	750.36 <sup>8</sup>	4.49 <sup>5</sup>	> 64 GB	–	–
TRW-S	51000 <sup>8</sup>	741.80 <sup>8</sup>	0.04	> 64 GB	–	–
	$8 \times 8 \times 8$			$10 \times 10 \times 10$		
	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>	mem [MB] <sup>1</sup>	time [s] <sup>2</sup>	bound [%] <sup>3</sup>
VML	1523 <sup>6</sup>	132.33 <sup>6</sup>	3.24	805 <sup>6,7</sup>	1304 <sup>6,7</sup>	3.50
$\alpha$ -EXP	2746	200.11	– <sup>4</sup>	5020	408.73	– <sup>4</sup>
SWAP	2746	507	– <sup>5</sup>	5020	1578.29	– <sup>5</sup>

- 1: GPU memory for VML and SML, otherwise CPU memory (measured with `valgrind` memory profiler).
- 2: Intel Core i7-3820 CPU @ 3.8 GHz with 64 GB of RAM, NVIDIA GTX 680 GPU with 4 GB of RAM.
- 3: Optimality bound in percent of the lower bound to solution provided by the algorithm.
- 4: No lower bound provided by algorithm, a theoretical (far from tight) a-priori bound is known to be  $\geq 100\%$  [19, 41]. Bound was computed with lower bound returned by TRW-S, if any is given.
- 5: No lower bound provided by algorithm, no theoretical bound known as far as we know. See also 4.
- 6: Data term computed on the fly (saves GPU storage, minimal run time increase).
- 7: Data term relaxation reprojection computed in each iteration, reduces GPU memory usage by 2 GB, but increases computation time by a factor of 5.
- 8: Requires general regularizer implementation, which is inefficient according to the author of the code. Unfortunately, no specialized implementation is available for 3D regularizers.

**Table 8.4: Performance comparison.** Performance comparison of several continuous and discrete multilabel algorithms on the segmentation problem using a 3D label space and different regularizers. Results are averaged over 10 different images with average resolution of 0.7 Megapixels. See Section 8.6.1 for a discussion.



**Figure 8.6: Comparison of the algorithms.** Closeups of  $4 \times 4 \times 4$  RGB label segmentations (*top*) using different algorithms reveal a typical problem of discrete approaches: they exhibit a preference for horizontal and vertical region boundaries since they penalize an  $l^1$ -distance instead of the correct Euclidean length [68, 143]. The error can be reduced by increasing the neighborhood connectivity, but only at extremely high costs of memory and computation time. Overall, these metrication errors lead to blocky and visually less pleasing segmentation results (*bottom*).

albeit with high run time requirements. Table 8.5 shows limit cases at an image resolution of  $640 \times 950$  — note that GPU memory shown is the theoretical minimum amount required, but we leverage all the remaining GPU memory to minimize the number of chunks for the local projections.

**Performance.** For smaller problems when we can use the global implementation of the constraints, our method outperforms the others in terms of run time, while attaining comparable optimality bounds. When the problems become larger, we need to switch to local projections per iteration, which increases run time five-fold and makes the other methods faster across a certain range of problem sizes. However, note that the run time of our algorithm scales better, so that at problem size  $23^3$  the algorithm is about to break even with the estimated computation times of  $\alpha$ -EXP again, the latter approximately doubling its computation time every time the dimension of each factor is increased by 2, see Table 8.4.

From a theoretical point of view, the move-making schemes  $\alpha$ -EXP and SWAP solve the problem by iterating binary decisions instead of dealing with the whole problem at once, which explains their efficiency. However, as a result they cannot give reasonable optimality bounds, see remarks in Table 8.4. As an additional drawback, all discrete methods suffer from metrication errors [68,

label space	VML		$\alpha$ -EXP	
	GPU mem [MB]	run time [min]	CPU mem [MB]	run time [min]
$15 \times 15 \times 15$	550	79	18120	31
$23 \times 23 \times 23$	836	485	> 64 GB	(est. 480)
$31 \times 31 \times 31$	1123	1179	> 64 GB	(est. 7680)

**Table 8.5: Memory and run time for segmentation of a color  $640 \times 950$  image.** Our method can deal with much larger problems than other algorithms.

143], as detailed in Figure 8.6.

### 8.6.2 Adaptive Denoising

As a novel application of a multidimensional label space, we present adaptive denoising, where we *jointly* estimate a noise level and a denoised image by solving a single minimization problem. Note that here we require the *continuous* label space to represent the image intensity range.

The Mumford-Shah energy can be interpreted as a denoising model which yields the maximum a-posteriori estimate for the original image under the assumption that the input image  $f$  was distorted with Gaussian noise of standard deviation  $\sigma$ . An interesting generalization of this model is when the standard deviation of the noise is not constant but rather varies over the image. Viewing it as an additional unknown, the label space becomes two-dimensional, with one dimension representing the unknown intensity  $u$  of the original image, and the second dimension representing the unknown standard deviation  $\sigma$  of the noise. The data term of the energy can then be written as [24]

$$\int_{\Omega} \frac{(u - f)^2}{2\sigma^2} + \frac{1}{2} \log(2\pi\sigma^2) dx. \quad (8.67)$$

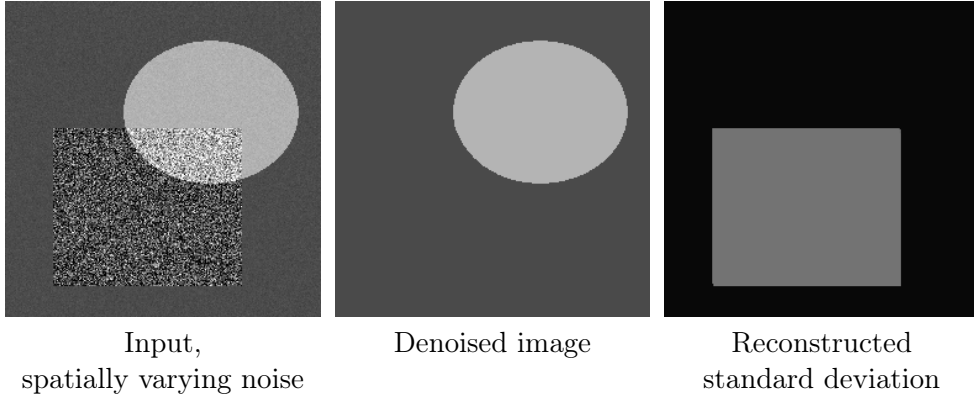
In our framework, the label space is two-dimensional comprising the intensities  $f$  and standard deviations  $\sigma$ . The cost function is given as the integrand of (8.67):

$$c(x, (u, \sigma)) = \frac{(u - f(x))^2}{2\sigma^2} + \frac{1}{2} \log(2\pi\sigma^2). \quad (8.68)$$

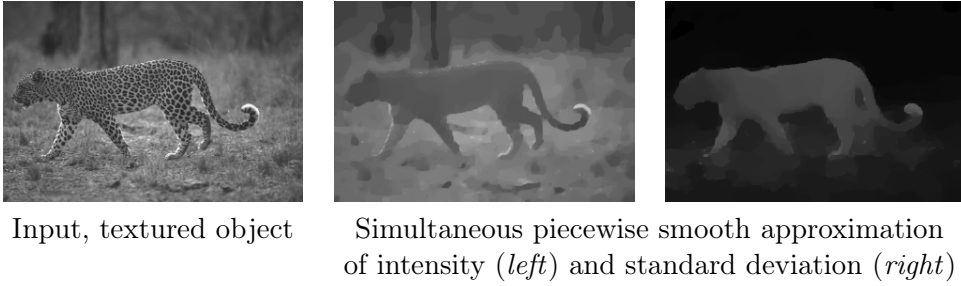
Results of the optimization can be observed in Figures 8.7 and 8.8. For the regularizer, we used piecewise constant Mumford-Shah for both  $\sigma$  and  $u$  in Figure 8.7, and piecewise smooth Mumford-Shah in Figure 8.8. In the real world example Figure 8.8, the solution can be interpreted as a uniformly smooth approximation, where all regions attain a similar smoothness level regardless of the amount of texture in the input.

### 8.6.3 Depth and Occlusion Map

In this test, we simultaneously compute a depth map and an occlusion map for a stereo pair of two color input images  $I_L, I_R : \Omega \rightarrow \mathbb{R}^3$ . The occlusion map shall be a binary map denoting whether a pixel in the left image has a matching



**Figure 8.7: Adaptive denoising on a synthetic example.** The algorithm allows to jointly recover the unknown standard deviation  $\sigma$  of the noise as well as the intensity of a denoised image by solving a single optimization problem. Ground truth: Within rectangle Gaussian noise with standard deviation  $\sigma = 0.25$ , outside  $\sigma = 0.02$ ; image intensity within ellipsoid  $u = 0.7$ , outside  $u = 0.3$ . Image resolution is  $256 \times 256$  using  $32 \times 32$  labels. Computation time is 4.4 minutes.



**Figure 8.8: Adaptive denoising for texture separation.** A piecewise smooth image approximation of both intensity and noise standard deviation using (8.67) and the Mumford-Shah regularizer for both  $u$  and  $\sigma$ . This model allows to separate textured objects in a natural way by jointly estimating the mean and standard deviation of image intensities. The amount of smoothing is stronger in regions of larger standard deviation. Image resolution is  $320 \times 214$  using  $32 \times 32$  labels, leading to a run time of 10.3 minutes.

pixel in the right image. Thus, the space of labels is two-dimensional with  $\Gamma_1$  consisting of the disparity values and a binary  $\Gamma_2 = \{0, 1\}$  for the occlusion map. We use the  $TV$  smoothness penalty on the disparity values. The Potts regularizer is imposed for the occlusion map. The distance on the label space thus becomes

$$d(t, t') = w_1 |t_1 - t'_1| + w_2 |t_2 - t'_2|, \quad (8.69)$$

with suitable weights  $w_1, w_2 > 0$ . We penalize an occluded pixel with a constant cost  $c_{occ} > 0$ , which corresponds to a threshold for the similarity measure above which we believe that a pixel is not matched correctly anymore. The cost



**Figure 8.9: Displacement with occlusion maps.** The proposed method can be employed to simultaneously optimize for a displacement and an occlusion map. This problem is also too large to be solved by alternative relaxation methods on current GPUs. *From left to right:* Left and right input image  $I_L$  and  $I_R$ , and computed disparity and occlusion map; red areas denote occluded pixels.

associated with a label  $t$  at  $(x, y) \in \Omega$  is then defined as

$$c((x, y), t) = \begin{cases} c_{occ} & \text{if } t_2 = 1, \\ |I_L(x, y) - I_R(x - t_1, y)| & \text{otherwise.} \end{cases} \quad (8.70)$$

The result for the “Moebius” test pair from the Middlebury benchmark is shown in Figure 8.9. The input image resolution was scaled to  $640 \times 512$ , requiring 128 disparity labels, which resulted in a total memory consumption which was slightly too big for previous methods, but still in reach of the proposed algorithm. Total computation time required was 1170 seconds.

#### 8.6.4 Optic Flow

In this experiment, we compute optic flow between two color input images  $I_0, I_1 : \Omega \rightarrow \mathbb{R}^3$  taken at two different time instants. The space of labels is again two-dimensional, with  $\Gamma_1 = \Gamma_2$  denoting the possible components of flow vectors in  $x$  and  $y$ -direction, respectively. We regularize both directions with either  $TV$  or a truncated linear penalty on the component distance, i.e.

$$d(t, t') = w \min(T, |t_1 - t'_1|) + w \min(T, |t_2 - t'_2|), \quad (8.71)$$

with a suitable manually chosen weight  $w > 0$  and threshold  $T > 0$ . Note that we can provide a tight relaxation of the exact penalizer, which was only coarsely approximated in the previous approaches [53, 76]. The cost function just compares pointwise pixel colors in the images, i.e.

$$c((x, y), t) = |I_0(x, y) - I_1(x + t_1, y + t_2)|. \quad (8.72)$$

Results can be observed in Figures 8.1, 8.10, 8.11 and 8.12. See Figure 8.11 for the color code of the flow vectors. In all examples, the number of labels is so high that this problem is currently impossible to solve with previous convex relaxation techniques by a large margin, see Table 8.3.

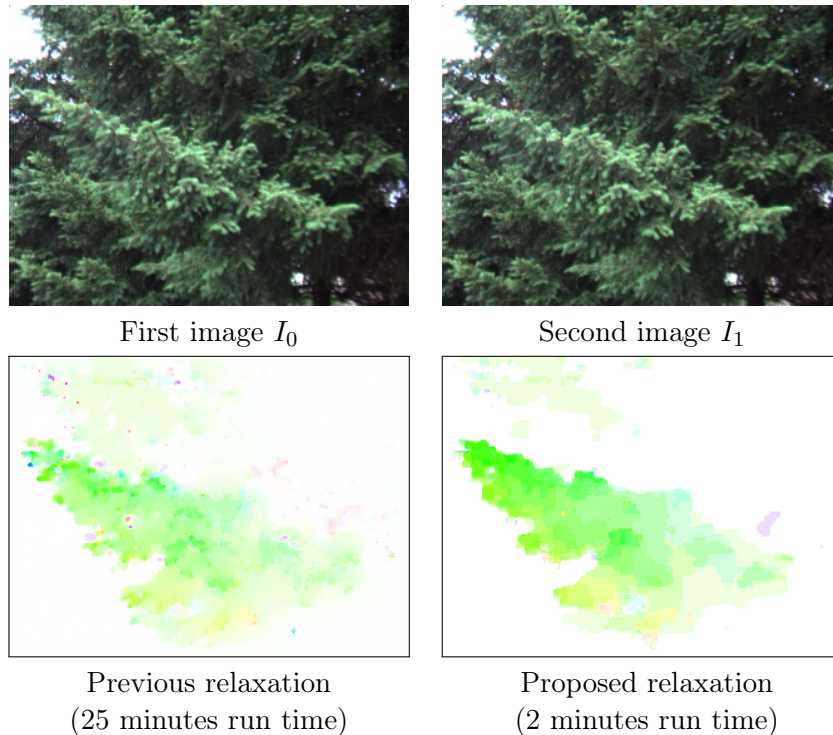
Compared to the relaxation proposed in the original conference publication [53], total computation time was reduced dramatically, see Figure 8.10.

Dataset size $\Delta$	VML			VML-ECCV [53]			$TV-L^1$ [142]	
	aep	aan	bound [%]	aep	aan	bound [%]	aep	aan
<b>Venus</b> 420 × 380 10	0.39	4.25	0.21	0.81	5.44	5.88	0.44	7.74
<b>Dimetrodon</b> 584 × 388 5	0.22	4.58	6.12	0.62	6.38	7.87	0.22	3.94
<b>Hydrangea</b> 584 × 388 12	0.42	4.06	2.64	0.81	5.60	9.26	0.22	2.64
<b>RubberWhale</b> 584 × 388 5	0.18	5.73	2.36	0.29	6.12	8.60	0.20	6.29
<b>Grove2</b> 640 × 480 5	0.34	4.54	5.01	0.55	6.16	13.25	0.22	3.12
<b>Grove3</b> 640 × 480 15	1.06	12.02	9.22	2.01	14.49	10.50	0.76	7.41
<b>Urban2</b> 640 × 480 22	0.81	9.31	1.07	0.97	8.15	6.32	0.47	3.51
<b>Urban3</b> 640 × 480 18	1.38	8.95	1.05	1.65	10.82	4.99	0.90	8.02

**Table 8.6: Accuracy comparison on Middlebury data sets.** Maximum displacement ( $\Delta$ ) and average endpoint error (*aep*) are measured in pixels, average angular error (*aan*) in degrees. Not surprisingly, accuracy for the vectorial multilabel (*VML*) method is strongly correlated with the amount of labels per pixel, and thus decreases with larger maximum displacement. On data sets with small maximum displacement, the accuracy using  $35 \times 35$  labels is comparable to  $TV-L^1$  optical flow, while other data sets would require either coarse-to-fine schemes or a greater number of labels for the method to remain competitive. The proposed new relaxation outperforms the previous one [53] in all respects.

The large computation time for the  $TV$  relaxation of [53] is caused by an overly restrictive constraint on the time steps due to the structure of the embedding matrix  $A_i$  in (8.33). Using  $n_1 = n_2 := N$  labels in each direction of the two-dimensional optic flow label space, the time steps can be seen to be proportional to  $N^{-3/2}$ . In contrast, the proposed  $TV$  relaxation in Section 8.4.4 allows larger time steps proportional to  $N^{-1/2}$  and thus leads to a substantially lower number of iterations. We suggest to use the preconditioning Algorithm 3 where the time steps are chosen adaptively.

Due to the global optimization of a convex energy, we can successfully capture large displacements without having to implement a coarse-to-fine scheme, see Figure 8.11. Table 8.6 shows detailed numeric results of our method on the data sets of the Middlebury benchmark with public ground truth available. We compare our current method with a linear regularizer using  $35 \times 35$  labels to our old relaxation [53] and  $TV-L^1$  optical flow [142], which utilizes a very similar energy which is optimized with a coarse-to-fine scheme and quadratic relaxation of the linearized functional. Results show that we get reasonable optimality bounds for the energy and are in most cases within 5% of the global optimum, while accuracy of the actual optical flow results depends on how fine the discretization is compared to the maximum displacement. The new method is obviously superior to the old relaxation in all respects — the previous one is only provided for reference, and we strongly recommend to use the new one proposed in this chapter. For a method which is competitive on the Middlebury benchmark, we would need to further increase the amount of labels by



**Figure 8.10: Comparison of total variation relaxations.** The figure shows optical flow fields with  $32 \times 32$  labels computed on an image with resolution  $320 \times 240$  using  $TV$  regularization. With the new relaxation of the regularizers, we achieve optimality bounds which are on average three times lower than with previous relaxations from [53, 76], using the proposed data term relaxation (8.23) for both cases. The result in the lower left is computed with the  $TV$  relaxation from [76]. Since the scaling of the regularity term is not directly comparable, we chose optimal parameters for both algorithms manually. The large time difference results from a narrow constraint on the time step, see Section 8.6.4.

e.g. implementing a coarse-to-fine scheme, and fine-tune our data terms. This is however out of the scope of this thesis, since our focus is to provide an efficient optimization framework.

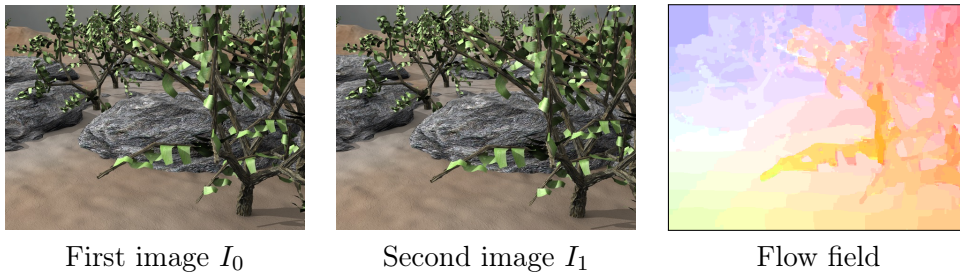
## 8.7 Conclusion

We have introduced a continuous convex relaxation for multilabel problems where the label space has a product structure and the regularizer is separable. Such labeling problems are plentiful in computer vision. The proposed reduction method improves on previous methods in that it requires orders of magnitude less memory and computation time, while retaining the advantages: a very flexible choice of regularizer on the label space, a globally optimal solution of the relaxed problem and an efficient parallel GPU implementation with guaranteed convergence.





**Figure 8.11: Optical flow with large displacements.** When employed for optic flow, the proposed method can successfully capture large displacements without the need for coarse-to-fine approaches, since a global optimization is performed over all labels. In contrast to existing methods, our solution is within a known bound of the global optimum.



**Figure 8.12: Optical flow on larger images.** Example with a larger image resolution of  $640 \times 480$  pixels, which requires  $32 \times 32$  labels. Regularizer is the total variation in each component. Computation time is 21.6 minutes.

The proposed framework combines the advantages of the efficient multidimensional data term relaxation [124] with the tight relaxation of the regularizers in [29]. It allows a very general class of continuous regularizers on multidimensional label spaces and can thus solve a significant range of problems efficiently. For example, we can explicitly encourage the solution to be smooth in certain regions, and can represent Huber- $TV$  and truncated linear regularization by an exact and tight relaxation. The regularizers can be arbitrarily mixed, in the sense that each dimension of the label space can have its own type of regularity. Because of the reduced memory requirements, we can successfully handle specific problems with very large number of labels, which could not be done with previous labeling methods. A systematic experimental comparison with respective discrete algorithms ( $\alpha$ -EXP, SWAP, BP, TRW-S) shows a good performance and often improved results.

## 8.8 Appendix: Proofs of Propositions and Theorems

### 8.8.1 Proof of Proposition 8.1

*Proof.* In order to prove the proposition, we show that the mapping induces a pointwise bijection from  $\Delta_\times$  onto  $\Delta$ . We first show it is onto: for  $u(x)$  in  $\Delta$ ,

there exists exactly one  $t \in \Gamma$  with  $u^t(x) = 1$ . Set  $v_i^s(x) = 1$  if  $s = t_i$ , and  $v_i^s(x) = 0$  otherwise. Then equation (8.13) is satisfied as desired, see Figure 8.2. To see that the map is one-to-one, we just count the elements in  $\Delta_\times$ . Since  $\Delta_i$  contains  $n_i$  elements, the number of elements in  $\Delta_\times$  is  $\prod_i n_i = n$ , the same as in  $\Delta$ .  $\square$

### 8.8.2 Proof of Proposition 8.2

*Proof.* Since at each point  $x \in \Omega$ ,  $t(x)$  is the label indicated by  $v(x)$ , by the defining property (8.16) we have  $v_i^s(x) = 1$  for  $s = t_i(x)$  and  $v_i^s(x) = 0$  for all  $s \in \Gamma_i$  with  $s \neq t_i(x)$ . Thus, for all  $z \in \mathcal{Z}$ ,

$$\sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s v_i^s(x) = \sum_{i=1}^k z_i^{t_i(x)} \leq c^{t(x)}(x). \quad (8.73)$$

This shows that at least  $\bar{E}_{\text{data}}(v) \leq \int_{\Omega} c^{t(x)}(x) dx = E_{\text{data}}(v)$ . To prove equality, first observe that we can safely interchange the supremum over  $z \in \mathcal{Z}$  with the integration over  $\Omega$  since the constraints in  $\mathcal{Z}$  on  $z$  are pointwise in  $x$ . This means that we only need to show the pointwise equality, i.e. that for each fixed  $x \in \Omega$  the integrand in (8.23) yields  $c^t(x)$  when taking the supremum over  $z \in \mathcal{Z}$ . We use Lagrange multipliers  $\mu$  to write the constraints in (8.24) as additional energy terms:

$$\begin{aligned} \bar{E}_{\text{data}}(v) &= \sup_{z \in \mathcal{Z}} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s v_i^s \\ &= \sup_z \inf_{\mu^{\hat{\gamma}} \geq 0} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s v_i^s - \sum_{\hat{\gamma} \in \Gamma} \mu^{\hat{\gamma}} (z_1^{\hat{\gamma}1} + \dots + z_k^{\hat{\gamma}k} - c^{\hat{\gamma}}) \\ &= \inf_{\mu^{\hat{\gamma}} \geq 0} \sum_{\hat{\gamma} \in \Gamma} \mu^{\hat{\gamma}} c^{\hat{\gamma}} + \sup_z \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s \left( v_i^s - \sum_{\hat{\gamma} \in \Gamma: \hat{\gamma}_i = s} \mu^{\hat{\gamma}} \right), \end{aligned} \quad (8.74)$$

interchanging the ordering of  $\sup_z$  and  $\inf_{\mu}$ . Evaluating the supremum over  $z$  leads to constraints on the variables  $\mu^{\hat{\gamma}}$  and we obtain

$$\bar{E}_{\text{data}}(v) = \inf_{\mu^{\hat{\gamma}} \geq 0} \sum_{\hat{\gamma} \in \Gamma} \mu^{\hat{\gamma}} c^{\hat{\gamma}} \quad (8.75)$$

with  $\mu^{\hat{\gamma}}$  such that additionally

$$\sum_{\hat{\gamma} \in \Gamma: \hat{\gamma}_i = s} \mu^{\hat{\gamma}} = v_i^s \quad \text{for all } 1 \leq i \leq k \text{ and } s \in \Gamma_i. \quad (8.76)$$

First, for any fixed  $1 \leq i \leq k$  and any  $s \in \Gamma_i$  with  $s \neq t_i$ , by assumption we have  $v_i^s = 0$ . Since  $\mu^{\hat{\gamma}} \geq 0$ , (8.76) then gives  $\mu^{\hat{\gamma}} = 0$  for all  $\hat{\gamma} \in \Gamma$  with  $\hat{\gamma}_i \neq t_i$ . Combining this for all  $1 \leq i \leq k$  we get  $\mu^{\hat{\gamma}} = 0$  for all  $\hat{\gamma} \neq t$ . Next, plug  $s = t_i$  for some  $i$  into (8.76). Since any other addend  $\mu^{\hat{\gamma}}$  is zero, the sum is just  $\mu^t$ , while the right hand side is  $v_i^{t_i} = 1$ .

Therefore, the constraints (8.76) ensure that  $\mu^{\hat{\gamma}} = 0$  for all  $\hat{\gamma} \neq t$  and  $\mu^t = 1$ , so that (8.75) gives  $\bar{E}_{\text{data}}(v) = c^t$ .  $\square$

### 8.8.3 Proof of Proposition 8.3

*Proof.* Let  $v \in L^2(\Omega, \text{co}(\Delta_\times))$  be arbitrary, and set  $z_i^s(x) := c^*(x)/k$ . Then

$$\sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) = \sum_{i=1}^k \frac{c^*(x)}{k} \sum_{s \in \Gamma_i} v_i^s(x) = \sum_{i=1}^k \frac{c^*(x)}{k} = c^*(x), \quad (8.77)$$

and  $\sum_i z_i^{t_i}(x) = c^*(x) \leq c^t(x)$  for all  $t \in \Gamma$  and  $x \in \Omega$ , so  $z \in \mathcal{Z}$ . This shows that  $\bar{E}_{\text{data}}(v) \geq \int_\Omega c^*(x) dx$ , which is the minimum of  $E_{\text{data}}$  for binary functions.  $\square$

### 8.8.4 Proof of Theorem 8.4

*Proof.* By convex duality [108], the convex envelope of  $E_{\text{data}}$  is given by the Legendre-Fenchel bi-conjugate  $E_{\text{data}}^{**}$ . The first convex conjugate  $E_{\text{data}}^*$  of  $E_{\text{data}}$  is given as

$$E_{\text{data}}^*(z) = \sup_v \int_\Omega \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx - E_{\text{data}}(v). \quad (8.78)$$

Since  $E_{\text{data}}$  is finite only for binary  $v$ , i.e. if  $v_i^s(x) = \chi_{s=t_i(x)}$  for some  $t : \Omega \rightarrow \Gamma$ , this reduces to

$$F(z) := E_{\text{data}}^*(z) = \sup_{t: \Omega \rightarrow \Gamma} \int_\Omega \left( \sum_{i=1}^k z_i^{t_i(x)}(x) - c^{t(x)}(x) \right) dx. \quad (8.79)$$

The bi-conjugate is then

$$E_{\text{data}}^{**}(v) = \sup_z \int_\Omega \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx - F(z). \quad (8.80)$$

For functions  $z \in L^2(\Omega, \mathbb{R}^{\sum_i n_i})$  and  $a \in L^2(\Omega, \mathbb{R}^k)$  define  $z_a \in L^2(\Omega, \mathbb{R}^{\sum_i n_i})$  by  $(z_a)_i^s(x) := z_i^s(x) + a_i(x)$ . Then obviously

$$F(z_a) = F(z) + \int_\Omega \sum_{i=1}^k a_i(x) dx. \quad (8.81)$$

Inserting  $z_a$  for  $z$  in (8.80) we obtain

$$\begin{aligned} E_{\text{data}}^{**}(v) &= \sup_{z, a} \int_\Omega \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx - F(z) \\ &\quad + \int_\Omega \sum_{i=1}^k a_i(x) \left( \sum_{s \in \Gamma_i} v_i^s(x) - 1 \right) dx. \end{aligned} \quad (8.82)$$

Holding a  $z$  fixed and taking the supremum over  $a$  we see that in order for  $E_{\text{data}}^{**}(v)$  to be finite, we necessarily must have

$$\sum_{s \in \Gamma_i} v_i^s(x) = 1 \quad \text{for all } 1 \leq i \leq k, \text{ for a.e. } x \in \Omega. \quad (8.83)$$

We assume these equalities from now on, i.e. that  $v \in \mathcal{S}$  with  $\mathcal{S}$  as defined in the theorem.

By the same way as we arrived at (8.82) we see that given (8.83) the expression in (8.80), over which the supremum is taken, does not change if we replace  $z$  by  $z_a$  for some  $a$ . Also, (8.81) shows that for each  $z$  and any fixed  $\alpha \in \mathbb{R}$  we can find an  $a$  with  $F(z_a) = \alpha$ . This combined, we obtain

$$E_{\text{data}}^{**}(v) = \sup_{z: F(z)=\alpha} \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx - \alpha. \quad (8.84)$$

Bringing  $\alpha$  on the left hand side and taking the supremum over all  $\alpha \leq 0$  we get

$$E_{\text{data}}^{**}(v) = \sup_{z: F(z) \leq 0} \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx. \quad (8.85)$$

This is almost the expression (8.23) for  $\overline{E}_{\text{data}}(v)$ . We only need to replace the integral constraints on  $z$  in  $F(z) \leq 0$ , with  $F$  in (8.79), with *pointwise* constraints. For this, note that in (8.79) the supremum over  $t$  may be safely put inside the integral over  $\Omega$  since the label space  $\Gamma$  is finite (after the assumed discretization). Therefore,  $F(z) \leq 0$  is equivalent to

$$\int_{\Omega} \sup_{t \in \Gamma} \left( \sum_{i=1}^k z_i^t(x) - c^t(x) \right) dx \leq 0. \quad (8.86)$$

Denoting the integrand by  $a(x)$ , this becomes equivalent to

$$\exists a : \Omega \rightarrow \mathbb{R} : \int_{\Omega} a(x) dx \leq 0, \quad \sum_{i=1}^k z_i^t(x) - c^t(x) \leq a(x) \quad \forall t \in \Gamma, x \in \Omega. \quad (8.87)$$

Given a  $z$  satisfying these constraints for some  $a$ , define  $\widehat{z}$  by  $\widehat{z}_i^s(x) := z_i^s(x) - a(x)/k$ . Then  $\widehat{z}$  satisfies (8.87) with  $a \equiv 0$ , i.e.  $\widehat{z} \in \mathcal{Z}$  with the constraint set  $\mathcal{Z}$  in (8.24). Furthermore,

$$\begin{aligned} \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} \widehat{z}_i^s(x) v_i^s(x) dx &= \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} \left( z_i^s(x) - \frac{a(x)}{k} \right) v_i^s(x) dx \\ &= \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx - \int_{\Omega} a(x) dx \\ &\geq \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx \end{aligned} \quad (8.88)$$

using first (8.83) and then  $\int_{\Omega} a(x) dx \leq 0$ . Thus, among all  $z$  with  $F(z) \leq 0$  or, equivalently, with (8.87) the expression in the supremum (8.85) will be largest if we choose  $a \equiv 0$  in (8.87). Hence,

$$E_{\text{data}}^{**}(v) = \sup_{z \in \mathcal{Z}} \int_{\Omega} \sum_{i=1}^k \sum_{s \in \Gamma_i} z_i^s(x) v_i^s(x) dx = \overline{E}_{\text{data}}(v) \quad (8.89)$$

for  $v \in \mathcal{S}$ . □

### 8.8.5 Proof of Theorem 8.5

*Proof.* Integrating by parts, the integral in (7.11) for fixed  $\varphi$  can be written as

$$\begin{aligned}
 J_\varphi &:= \int_{\Omega \times \Gamma_i} (\varphi^1) \cdot dD_x 1_{u_i} + \varphi^2 dD_s 1_{u_i} \\
 &= \int_{\Omega \times \Gamma_i} (-\operatorname{div}_x \varphi^1 - \partial_s \varphi^2) 1_{u_i} d(x, s) \\
 &= \int_{\Omega \setminus S_{u_i}} \int_{-\infty}^{u_i(x)} (-\operatorname{div}_x \varphi^1 - \partial_s \varphi^2) d(x, s) \\
 &= \int_{\Omega \setminus S_{u_i}} (-P(x, u_i(x)) - q(x, u_i(x))) d(x, s)
 \end{aligned} \tag{8.90}$$

where

$$P(x, s) := \int_{-\infty}^s (\operatorname{div}_x \varphi^1)(x, s') ds' = \operatorname{div}_x \int_{-\infty}^s \varphi^1(x, s') ds' = (\operatorname{div}_x p)(x, s) \tag{8.91}$$

with

$$p(x, s) := \int_{-\infty}^s \varphi^1(x, s') ds', \tag{8.92}$$

and

$$q(x, s) := \int_{-\infty}^s \partial_s \varphi^2(x, s') ds' = \varphi^2(x, s). \tag{8.93}$$

Since  $\varphi \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R})$ , also  $(p, q) \in C_c^1(\Omega \times \Gamma_i; \mathbb{R}^m \times \mathbb{R})$ . Therefore, from (8.90),

$$\begin{aligned}
 J_\varphi &= \int_{\Omega \setminus S_{u_i}} \left( -(\operatorname{div}_x p)(x, u_i(x)) - q(x, u_i(x)) \right) d(x, s) \\
 &= \int_{\Omega \times \Gamma_i} (-\operatorname{div}_x p - q) v_i d(x, s).
 \end{aligned} \tag{8.94}$$

The last equality is simply the definition of how the distribution  $v_i(x, s) = \delta(u_i - s)$ , defined for  $u_i \in \operatorname{SBV}(\Omega, \mathbb{R})$ , acts on functions. Now, the claim of the proposition follows directly from (7.8) and (8.94).  $\square$

### 8.8.6 Proof of Proposition 8.7

*Proof.* We can enforce a piecewise constant labeling  $u_i$ , if we enforce the approximate gradient  $\nabla u_i$  to be constant zero. In (8.28), this can be achieved by setting  $h_i(x, u_i(x), \nabla u_i(x)) = c |\nabla u_i|$  with a constant  $c > 0$ , and then letting  $c \rightarrow \infty$  to enforce  $\nabla u_i \equiv 0$  on  $\Omega \setminus S_{u_i}$ . Inserting the convex conjugate  $h_i^*(x, s, q) = \delta_{\{|q| \leq c\}}$ , we find that the conditions in (8.31) now reduce to

$$q^s \geq 0, \quad |\partial_s p^s| \leq c, \quad |p^s - p^{s'}| \leq d_i(s, s'). \tag{8.95}$$

The supremum over  $q^s \geq 0$  is easily eliminated from (8.30) since  $v_i^s \geq 0$ , i.e.  $-q^s v_i^s \leq 0$  with 0 being the maximum possible value. The second constraint in (8.95) follows from the third if we choose  $c \geq \max_{s > s'} \frac{d_i(s, s')}{|s - s'|}$ . Thus we arrive at (8.36) with the set  $\mathcal{C}_i$  as claimed in the proposition.  $\square$

### 8.8.7 Proof of Proposition 8.8

*Proof.* The claim follows from our general formulation (8.36) with a special choice of the dual variables  $p$  together with additional relaxations of the equations in  $\mathcal{C}_i$ . As the special form for  $p^s$  we choose

$$p^s = \sum_{j=1}^{M_i} (a_i^s)_j z_j, \quad (8.96)$$

with  $z : \Omega \times \{1, \dots, M_i\} \rightarrow \mathbb{R}^m$  such that  $|z| \leq 1$  and the vectors  $a_i^s \in \mathbb{R}^{M_i}$  which define the Euclidean representation of  $d_i$ , see equation (8.32). The constraint on  $p$  in (8.37) is satisfied, since by the Cauchy-Schwarz inequality and the definition of the representation,

$$\begin{aligned} |p^s - p^{s'}| &= \left| \sum_{j=1}^{M_i} ((a_i^s)_j - (a_i^{s'})_j) z_j \right| \leq \sqrt{\sum_{j=1}^{M_i} ((a_i^s)_j - (a_i^{s'})_j)^2} \cdot \sqrt{\sum_{j=1}^{M_i} |z_j|^2} \\ &= |a_i^s - a_i^{s'}| |z| \leq |a_i^s - a_i^{s'}| = d_i(s, s'). \end{aligned} \quad (8.97)$$

Thus,  $p$  is feasible for the supremum in (8.36) and we obtain the desired result

$$\begin{aligned} R_i(v_i) &\geq \sup_{|z| \leq 1} \sum_{s \in \Gamma_i} \int_{\Omega} -\operatorname{div}_x \left( \sum_{j=1}^{M_i} (a_i^s)_j z_j \right) v_i^s \, dx \\ &= \sup_{|z| \leq 1} \int_{\Omega} \sum_{j=1}^{M_i} (-\operatorname{div}_x z_j) \left( \sum_{s \in \Gamma_i} (a_i^s)_j v_i^s \right) \, dx \\ &= TV \left( \sum_{s \in \Gamma_i} a_i^s v_i^s \right) = R_i^A(v_i). \end{aligned} \quad (8.98)$$

□

### 8.8.8 Proof of Proposition 8.9

*Proof.* Both  $R$  and  $\overline{E}_{\text{data}}$  are support functionals of convex sets in the Hilbert space  $\mathcal{L} := L^2(\Omega, \mathbb{R}^{\sum_i n_i})$ : equation (8.52) shows that the regularizer  $R$  is the support functional of  $K(\mathcal{C})$ , while we can see from definition (8.23) that the data term  $\overline{E}_{\text{data}}$  is the support functional of  $\mathcal{Z}$ . It follows that both  $R$  and  $\overline{E}_{\text{data}}$  are lower-semicontinuous and convex on  $\mathcal{L}$ . The set  $\mathcal{D}$  is closed, thus its indicator function  $\delta_{\mathcal{D}}$  is also convex and closed, furthermore  $\delta_{\mathcal{D}}$  is coercive since  $\mathcal{D}$  is bounded. From the above, it follows that the functional

$$v \mapsto R(v) + \overline{E}_{\text{data}}(v) + \delta_{\mathcal{D}}(v) \quad (8.99)$$

is closed and coercive. Since being closed is equivalent to being lower-semicontinuous in the Hilbert space topology of  $\mathcal{L}$ , these properties imply the existence of a minimizer in  $\mathcal{L}$ , see theorems 3.2.5 and 3.3.3 in [6], which must necessarily lie in  $\mathcal{D}$ . Since neither functional is strictly convex, the solution is in general not unique. □

## Chapter 9

# Vectorial Problems with Coupled Regularization

Here we consider special coupling regularizers for vectorial functions such as quadratic penalization and total variation. It turns out that for such regularizers an efficiently optimizable convex relaxation can still be established. This chapter is based on joint work with Antonin Chambolle and Daniel Cremers [120].

### 9.1 Introduction

#### 9.1.1 Special Vectorial Problems

In the previous Chapter 8, we considered convexifications of special vectorial functionals by proposing to represent the vectorial solutions  $u$  by  $k$  separate graph functions, one for each channel  $u_i$ , or respectively by  $k$  independent collections of indicator functions. This allowed to reduce the number of variables from  $N^k$  to  $kN$ , if  $N$  is the number of samples in each channel range, and we also introduced an appropriate convexification of the data term in this representation.

However, the regularizer part was handled by a channel-wise application of the lifting method. This limited the approach to *separable* interaction terms, which regularize each coordinate  $u_i$  independently. Our contribution in this chapter is to try to consider more general interaction terms in this setting.

In this chapter, we propose a novel convex relaxation for the estimation of vector-valued functions with nonconvex data terms and *convex* regularizers. This can be interpreted as a generalization to the vectorial case of the lifting relaxation for the special class of scalar energies in Section 7.2.2. What makes this class special is that the lifting relaxation yields an especially simple form in this case and furthermore allows to obtain globally optimal binary solutions. This motivates to consider the *vectorial* analogon of this special class.

Let  $\Omega \subset \mathbb{R}^m$  be a bounded open set, with, in practice,  $m = 2, 3$ . We want to find an appropriate convex representation for functionals of the form

$$E(u) = \int_{\Omega} h(x, u, \nabla u) \, dx \tag{9.1}$$

where  $u \in W^{1,1}(\Omega; \mathbb{R}^k)$ ,  $k \geq 2$ . We will also consider the case of BV-fields  $u$ , with a suitable definition on the jump set.

Our proposed convexification approach enables to handle the combination of a nonconvex data term with *coupled* convex regularizers such as isotropic  $l^2$ -total variation  $TV_{l^2}$ . This generalizes the approach of the previous chapter, which is limited to separable and thus non-coupling regularizers. It also generalizes [54] which allows for convex coupling regularizers but assumes convex data terms. The key idea of the convexification is to consider a collection of hypersurfaces with a relaxation which takes into account the *entire functional at once*, rather than separately treating the data term and the regularizers of each component as in the previous chapter.

### 9.1.2 Contributions

We focus particularly on convex representations which

- are computationally tractable,
- are as tight as possible, that is, as close as possible to the convex envelope of the energy  $E(u)$  (in an appropriate representation).

To stay computationally realistic, we therefore choose the representation by separate graph functions (and thus discretizations) for each channel as in the previous Chapter 8. There are restrictions on the type of energies in order for the relaxation to still be exact, i.e. that it coincides with the initial energy for binary graph functions. In particular, we will focus on problems of the form

$$\min_u E(u) := \int_{\Omega} c(x, u(x)) dx + \int_{\Omega} f(x, \nabla u(x)) dx \quad (9.2)$$

for  $u \in W^{1,1}(\Omega; \mathbb{R}^k)$ , where  $f(x, p) : \Omega \times \mathbb{R}^{m \times k} \rightarrow \mathbb{R}_+$  is continuous in  $x$  and convex in  $p$  (with possibly linear growth, in which case  $u \in \text{BV}(\Omega, \mathbb{R}^k)$ ). For the definition of the Sobolev space  $W^{1,1}(\Omega; \mathbb{R}^k)$  and the space  $\text{BV}(\Omega, \mathbb{R}^k)$  of bounded variation functions we refer to [5], see also Section 2.2.

Specifically, we make the following contributions:

- We propose a general convexification strategy for vectorial problems of the form (9.2). The main novelty of our approach is the applicability to possibly non-separable convex regularizers such as the isotropic  $TV_{l^2}$ . For separable regularizers it reduces to the relaxation of Chapter 8. Since the functional is treated as a whole, this provides a natural derivation of the data term relaxation given in Chapter 8. We prove the non-trivial fact that the proposed relaxation is exact.
- Our framework also allows nonconvex data terms  $c(x, \cdot)$  in (9.2), which arise in many useful applications such as stereo reconstruction and optical flow. Furthermore, nonconvex data terms can be used together with non-separable regularizers which further improves the results e.g. in denoising applications. This is not possible with previous approach in [54] and that in Chapter 8.



- For the important special cases of isotropic total variation  $TV_{l_2}$  and its Huber-regularized variant we give a reformulation of the constraint set which allows one to minimize energies with these regularizers as efficient as in the scalar case.
- We provide extensive implementation details, including memory analysis and notes on the GPU usage. Several experiments demonstrate the advantage of coupled regularizers over the separable ones in applications such as optical flow, denoising and inpainting. The case of nonconvex data terms with coupling regularizers, which is only possible with our approach, leads to superior results than with previous approaches.
- We give an explicit formula for the projection onto a parabola, which is needed to implement Huber- $TV_{l_2}$ . This is more robust than using Newton's iterative method as in [101] since the number of iterations depends on the data and increases with increasing regularizer weight  $\lambda$ . We found the explicit projection to be also faster by a factor of 2 and more.

## 9.2 Convex Relaxation

### 9.2.1 Convexification Framework

Following the functional lifting framework in Chapter 7 and the approach of the previous Chapter 8, given  $u \in L^1_{\text{loc}}(\Omega, \mathbb{R}^k)$  we consider the collection of graph functions

$$1_u := (1_{u_1}, \dots, 1_{u_k}) \in L^1_{\text{loc}}(\Omega \times \mathbb{R}, \{0, 1\}^k) \quad (9.3)$$

where for each  $i = 1, \dots, k$ , the individual graph functions are defined as

$$1_{u_i}(x, t) = \begin{cases} 1 & \text{if } t < u_i(x), \\ 0 & \text{else.} \end{cases} \quad (9.4)$$

Then, we define a convex relaxation of (9.2), on the set  $L^1_{\text{loc}}(\Omega \times \mathbb{R}, [0, 1]^k)$ . As in Section 7.2.2 and Chapter 8 it takes the form, if  $v \in \text{BV}(\Omega \times \mathbb{R}; [0, 1]^k)$ ,

$$\mathcal{E}(v) = \sup \left\{ \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i \cdot dDv_i \mid \varphi \in C^1(\Omega \times \mathbb{R}; \mathbb{R}^{m \times k}) \cap \mathcal{C} \right\} \quad (9.5)$$

for some convex set  $\mathcal{C}$ . We need this relaxation

- to be exact on characteristics of subgraphs, that is, we want

$$\mathcal{E}(1_u) = E(u) \quad (9.6)$$

for any  $u$ ,

- to be as “tight” as possible, that is, as close as possible to the convex envelope of the function  $v \mapsto E(u)$  if  $v = 1_u$  for some  $u$ ,  $+\infty$  else.

If, to simplify,  $u \in W^{1,1}(\Omega; \mathbb{R}^k)$ , then the terms in (9.5) can be written as [1]

$$\sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i \cdot dD1_{u_i} = \sum_{i=1}^k \int_{\Omega} \varphi_i^x(x, u_i(x)) \cdot \nabla u_i(x) - \varphi_i^t(x, u_i(x)) dx. \quad (9.7)$$

As a consequence, a sufficient condition in order to have

$$\mathcal{E}(1_u) \leq E(u) \quad (9.8)$$

is that for any  $x \in \Omega$ ,  $t = (t_1, \dots, t_k) \in \mathbb{R}^k$  and  $p \in \mathbb{R}^{m \times k}$ , the fields  $\varphi$  in  $\mathcal{C}$  satisfy

$$\sum_{i=1}^k \varphi_i^x(x, t_i) \cdot p_i - \varphi_i^t(x, t_i) \leq h(x, t, p).$$

Rearranging and taking the supremum over  $p$ , this is equivalent to

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq h^*(x, t, (\varphi_i^x(x, t_i))_{i=1}^k), \quad (9.9)$$

where  $h^*$  is the Legendre-Fenchel conjugate of  $h$  with respect to the variable  $p$ . For  $h(x, t, p) = f(x, p) + c(x, t)$  as in (9.2), it boils down to

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq f^*\left(x, (\varphi_i^x(x, t_i))_{i=1}^k\right) - c(x, t) \quad (9.10)$$

for all  $x \in \Omega$  and  $t \in \mathbb{R}^k$ .

### 9.2.2 Concrete Example: The Vectorial ROF Model

Before we continue, let us make this more explicit for a concrete example, namely the vectorial case of the classical Rudin-Osher-Fatemi total variation denoising problem (ROF model) [111]. For  $u \in C^1(\Omega; [0, 1]^k)$ , it is given by

$$E_{ROF}(u) = \int_{\Omega} (u - u_0)^2 + \lambda |\nabla u| dx. \quad (9.11)$$

We want to rewrite this energy as the supremum of the expression on the left hand side of (9.7) with the dual variable  $\varphi$  constrained to some appropriately chosen convex set  $\mathcal{C}$ . This set should be as large as possible in order for the convex relaxation to be as tight as possible. Yet, how should one choose this set? A necessary condition is that (9.7) should be less than or equal to the ROF-energy (9.11) for all  $\varphi \in \mathcal{C}$  and all  $u$ . This amounts to the inequality:

$$\begin{aligned} \int_{\Omega} \sum_{i=1}^k \varphi_i^x(x, u_i(x)) \cdot \nabla u_i - \lambda |\nabla u| - \sum_{i=1}^k \varphi_i^t(x, \nabla u_i) dx \\ \leq \int_{\Omega} (u - u_0(x))^2 dx \quad \forall \varphi \in \mathcal{C}, \forall u. \end{aligned} \quad (9.12)$$

A sufficient condition is the *local* version of this constraint:

$$\sum_{i=1}^k \varphi_i^x(x, t_i) \cdot p_i - \lambda |p| - \sum_{i=1}^k \varphi_i^t(x, p_i) \leq (t - u_0(x))^2, \quad \forall \varphi \in \mathcal{C}, \forall t, x, p. \quad (9.13)$$

Taking the supremum over  $p$ , we observe that the first two terms on the left-hand side are equal to:

$$\sup_p \sum_{i=1}^k \varphi_i^x(x, t_i) \cdot p_i - \lambda |p| = \begin{cases} 0 & \text{if } \sqrt{\sum_{i=1}^k |\varphi_i^x(x, t_i)|^2} \leq \lambda, \\ \infty & \text{else.} \end{cases} \quad (9.14)$$

Therefore, the localized version of the constraint (9.12) is equivalent to the two constraints

$$\begin{aligned} \sum_{i=1}^k \varphi_i^t(x, t_i) &\geq -(t - u_0(x))^2 \quad \forall \varphi \in \mathcal{C}, \forall t, x, \\ \sqrt{\sum_{i=1}^k |\varphi_i^x(x, t_i)|^2} &\leq \lambda \quad \forall \varphi \in \mathcal{C}, \forall t, x. \end{aligned} \quad (9.15)$$

### 9.2.3 The Scalar Case

In the *scalar case* and for “reasonable” Lagrangians  $h$  in (9.1) (in particular, continuous and convex in  $p$  with at least linear growth), one can check (see [101]) that condition (9.10) allows to recover tightly the initial energy, in the sense that if  $\mathcal{C}$  is the set of smooth vector fields  $\varphi$  satisfying (9.9), i.e.

$$\varphi^t(x, t) \geq h^*(x, t, \varphi^x(x, t)) \quad \forall x, t, \quad (9.16)$$

then minimizing the energy  $\mathcal{E}$  defined by (9.5) always solves the original problem.

In the simplified case (9.2), one can show that the energy  $\mathcal{E}$  has the form

$$\mathcal{E}(v) = \int_{\Omega \times \mathbb{R}} \widehat{f}(dDv) - c(x, t) dD_t v$$

with  $\widehat{f}(p^x, p^t) : \mathbb{R}^m \times \mathbb{R} \rightarrow (-\infty, +\infty]$  the convex, one-homogeneous integrand defined by:

$$\widehat{f}(p^x, p^t) = \begin{cases} |p^t| f(p^x / |p^t|) & \text{if } p^t < 0, \\ f_\infty(p^x) & \text{if } p^t = 0, \\ +\infty & \text{if } p^t > 0. \end{cases}$$

In this notation,  $f_\infty(p)$  is the convex and positive one-homogeneous recession function of  $f$ , see (2.4)

## 9.3 General Vectorial Case

### 9.3.1 Exactness of the Relaxation

In the vectorial case, though, things are not so simple and as we already observed, one cannot hope to recover in general an equivalent convex formulation, which remains tractable computationally. Let us now fix arbitrary (finite)

bounds  $a$  and  $b > a$  for the values of the functions  $u_i$  (in practice we could choose different intervals  $[a_i, b_i]$  for each variable  $t_i$ , but the analysis would be strictly identical), and let  $\Gamma = [a, b]$ : in what follows we will work in  $\Omega \times \Gamma^k$ . We let:

$$\mathcal{C} = \left\{ \varphi = (\varphi_1, \dots, \varphi_k) \mid \varphi_i = (\varphi_i^x, \varphi_i^t) \in C_c^0(\Omega \times \Gamma; \mathbb{R}^m \times \mathbb{R}), \right. \\ \left. (\varphi_i)_{i=1}^k \text{ satisfies (9.10) for all } x \in \Omega, t \in \Gamma^k \right\}. \quad (9.17)$$

We will show the following main result of this chapter:

**Proposition 9.1.** *For  $v \in \text{BV}(\Omega \times \Gamma; [0, 1]^k)$  let  $\mathcal{E}(v)$  be defined by (9.5) with the set  $\mathcal{C}$  in (9.17). Then if  $u \in W^{1,1}(\Omega; \Gamma^k)$ , one has  $\mathcal{E}(1_u) = E(u)$ .*

This result will be a consequence of the stronger Lemmas 9.2 and 9.3 in the subsections below, which show the result separately for the data and interaction terms.

We now introduce the convex sets (note that the first one corresponds to the data term relaxation constraint set of Section 8.3.2)

$$\mathcal{L} = \left\{ \gamma = (\gamma_1, \dots, \gamma_k) \mid \gamma_i \in C_c^0(\Omega \times \Gamma; \mathbb{R}), \right. \\ \left. \sum_{i=1}^k \gamma_i(x, t_i) \geq -c(x, t) \quad \forall (x, t) \in \Omega \times \Gamma^k \right\} \quad (9.18)$$

and

$$\mathcal{C}_0 = \left\{ \varphi = (\varphi_1, \dots, \varphi_k) \mid \varphi_i = (\varphi_i^x, \varphi_i^t) \in C_c^0(\Omega \times \Gamma; \mathbb{R}^m \times \mathbb{R}), \right. \\ \left. \sum_{i=1}^k \varphi_i^t(x, t_i) \geq f^*\left(x, (\varphi_i^x(x, t_i))_{i=1}^k\right) \quad \forall (x, t) \in \Omega \times \Gamma^k \right\}. \quad (9.19)$$

Observe in particular that  $\mathcal{C}_0 + \mathcal{L} \subseteq \mathcal{C}$ . Thus, to prove Proposition 9.1 it is enough to prove it with  $\mathcal{C}$  replaced by  $\mathcal{C}_0 + \mathcal{L}$  in (9.5).

### 9.3.2 Exactness for the Data Term Part

The following result is an extension of classical results (in particular, relative to the Monge-Kantorovich duality in optimal transportation problems [133]).

**Lemma 9.2.** *Let  $u \in L^1(\Omega, \Gamma^k)$  and assume  $c$  is bounded, l.s.c. in  $(x, u)$ <sup>1</sup>. Then*

$$\int_{\Omega} c(x, u(x)) \, dx = \sup_{\gamma \in \mathcal{L}} \int_{\Omega \times \Gamma} \sum_{i=1}^k \gamma_i \cdot \, dD_{t_i} 1_{u_i}. \quad (9.20)$$

<sup>1</sup>One could consider integrands  $c$  which are merely measurable in  $x$  and continuous in  $u$ , by relaxing the continuity assumption of the fields  $\gamma_i$ .

Here we use the notation  $D_{t_i} 1_{u_i}$ , which is consistent with (9.5) and makes sense since  $1_{u_i}$  is nonincreasing in the  $t_i$ -variable. However observe that  $D_{t_i} 1_{u_i}$  is simply equal to  $-\delta_{u_i(x)}$ . In particular, the integral on the right-hand side can be equivalently rewritten in the simpler form

$$- \int_{\Omega} \sum_{i=1}^k \gamma_i(x, u_i(x)) \, dx .$$

*Proof.* Without loss of generality, we may assume that  $0 \leq c \leq K$  on  $\Omega \times \Gamma^k$ , for some constant  $K \in \mathbb{R}$ . We first assume that  $c$  is uniformly continuous on  $\Omega \times \Gamma$ . Let  $u \in L^1(\Omega, \Gamma^k)$ . Let  $\varepsilon > 0, r > 0$ : by standard covering arguments [44], one can find a disjoint covering  $(B_{\alpha})_{\alpha \in \mathbb{N}}$  of almost all  $\Omega$ , that is, disjoint closed balls  $B_{\alpha} = \overline{B(x_{\alpha}, r_{\alpha})}$  with  $|\Omega \setminus \bigcup_{\alpha} B(x_{\alpha}, r_{\alpha})| = 0$ , with the following properties:

1.  $r_{\alpha} \leq r$  for all  $\alpha \in \mathbb{N}$ ,
2.  $\int_{B_{\alpha}} |u(x) - u(x_{\alpha})| \, dx \leq r\varepsilon |B_{\alpha}|$  for all  $\alpha \in \mathbb{N}$ .
3.  $\int_{B_{\alpha}} |c(x, u(x)) - c(x_{\alpha}, u(x_{\alpha}))| \, dx \leq \varepsilon |B_{\alpha}|$  for all  $\alpha \in \mathbb{N}$ .

We assume  $r > 0$  is chosen in such a way that  $|c(x, t) - c(y, s)| \leq \varepsilon$  if  $|y - x| \leq r$  and  $\max_i |t_i - s_i| \leq r$ . Let us choose  $N \in \mathbb{N}$  such that  $\sum_{\alpha > N} |B_{\alpha}| < \varepsilon$ . Then, we let for  $i = 1, \dots, k$   $\gamma_i(x, t) = K$  if  $x \in B_{\alpha}, \alpha > N$ , while if  $x \in B_{\alpha}$  with  $\alpha \leq N$ ,

$$\begin{aligned} \gamma_i(x, t_i) = & -\frac{c(x_{\alpha}, u(x_{\alpha}))}{k} \varphi\left(\frac{x - x_{\alpha}}{r_{\alpha}}\right) \eta\left(\frac{t_i - u_i(x_{\alpha})}{\varrho}\right) \\ & + K \left(1 - \varphi\left(\frac{x - x_{\alpha}}{r_{\alpha}}\right) \eta\left(\frac{t_i - u_i(x_{\alpha})}{r}\right)\right) + \frac{\varepsilon}{k} \end{aligned} \quad (9.21)$$

where the cut-off functions  $\varphi \in C_c^{\infty}(B(0, 1); [0, 1])$ ,  $\eta \in C_c^{\infty}([-1, 1]; [0, 1])$  will be precised later on.

If  $x \in B_{\alpha}, \alpha \leq N$ , and  $t \in \Gamma^k$  with  $|t_i - u_i(x)| \geq r$  for at least one  $i$ , then  $\sum_i \gamma_i(x, t_i) \geq 0 \geq -c(x, t)$ . This is also clear if  $x \in B_{\alpha}$  with  $\alpha > N$ . Now, if  $x \in B_{\alpha}, \alpha \leq N$ , and  $|t_i - u_i(x)| < r$  for all  $i = 1, \dots, k$ , then

$$\sum_i \gamma_i(x, t_i) \geq -c(x_{\alpha}, u(x_{\alpha})) + \varepsilon \geq -c(x, t)$$

so that  $\gamma \in \mathcal{L}$ . On the other hand, we have

$$\begin{aligned} & - \int_{\Omega} \sum_{i=1}^k \gamma_i(x, u_i(x)) \, dx \\ & \geq -(kK + \varepsilon) \left( \sum_{\alpha > N} |B_{\alpha}| \right) - \sum_{\alpha \leq N} \sum_{i=1}^k \int_{B_{\alpha}} \gamma_i(x, u_i(x)) \, dx . \end{aligned} \quad (9.22)$$

Now, for  $\alpha \leq N$ ,

$$\begin{aligned} - \int_{B_\alpha} \gamma_i(x, u_i(x)) \, dx &\geq \frac{1}{k} |B_\alpha| (c(x_\alpha, u(x_\alpha)) - \varepsilon) \\ &\quad - K \left(1 + \frac{1}{k}\right) \int_{B_\alpha} \left(1 - \varphi\left(\frac{x - x_\alpha}{r_\alpha}\right) \eta\left(\frac{u_i(x) - u_i(x_\alpha)}{r}\right)\right) \, dx \end{aligned} \quad (9.23)$$

which we now estimate. Assume that  $\varphi$  was chosen such that

$$\int_{B_1} (1 - \varphi(x)) \, dx \leq \varepsilon$$

and  $\eta(0) = 1$ ,  $\eta$  has a Lipschitz constant  $L \geq 1$ . Then,

$$\begin{aligned} &\int_{B_\alpha} \left(1 - \varphi\left(\frac{x - x_\alpha}{r_\alpha}\right) \eta\left(\frac{u_i(x) - u_i(x_\alpha)}{r}\right)\right) \, dx \\ &= \int_{B_\alpha} \left(1 - \varphi\left(\frac{x - x_\alpha}{r_\alpha}\right)\right) \, dx \\ &\quad + \int_{B_\alpha} \varphi\left(\frac{x - x_\alpha}{r_\alpha}\right) \left(1 - \eta\left(\frac{u_i(x) - u_i(x_\alpha)}{r}\right)\right) \, dx \\ &\leq \varepsilon |B_\alpha| + \frac{L}{r} \int_{B_\alpha} |u_i(x) - u_i(x_\alpha)| \, dx \leq (1 + L)\varepsilon |B_\alpha|. \end{aligned}$$

Together with (9.23), we deduce that for  $\alpha \leq N$ ,

$$\begin{aligned} - \sum_{i=1}^k \int_{B_\alpha} \gamma_i(x, u_i(x)) \, dx &\geq |B_\alpha| (c(x_\alpha, u(x_\alpha)) - \varepsilon) - \varepsilon K(k+1)(1+L) |B_\alpha| \\ &\geq \int_{B_\alpha} c(x, u(x)) \, dx - \varepsilon(2 + K(k+1)(1+L)) |B_\alpha| \end{aligned}$$

so that, using (9.22)

$$\begin{aligned} - \int_{\Omega} \sum_{i=1}^k \gamma_i(x, u_i(x)) \, dx &\geq \int_{\Omega} c(x, u(x)) \, dx \\ &\quad - ((k+1)K + \varepsilon) \left(\sum_{\alpha > N} |B_\alpha|\right) - \varepsilon(2 + K(k+1)(1+L)) \left(\sum_{\alpha \leq N} |B_\alpha|\right) \\ &\geq \int_{\Omega} c(x, u(x)) \, dx - (((k+1)K + \varepsilon) + (2 + K(k+1)(1+L))|\Omega|)\varepsilon, \end{aligned}$$

which shows that (9.20) holds when  $c$  is uniformly continuous.

Now, if  $c$  is only l.s.c. (and bounded), there exist  $c_n$  bounded, uniformly continuous such that  $\sup_n c_n = c$ . If  $\mathcal{L}_n$  is the corresponding set for  $c_n$ , we have  $\mathcal{L}_n \subset \mathcal{L}$  so that

$$\int_{\Omega} c(x, u(x)) \, dx \geq \sup_{\gamma \in \mathcal{L}} \int_{\Omega \times \Gamma} \sum_{i=1}^k \gamma_i \cdot dD_{t_i} 1_{u_i} \geq \int_{\Omega} c_n(x, u(x)) \, dx$$

and the result follows by sending  $n \rightarrow \infty$ .  $\square$

### 9.3.3 Exactness for the Regularizer Part

Next, we need the following result. To simplify, let  $f$  be minimal and vanishing for  $p = 0$ , i.e.  $f(x, 0) = 0 = \min_p f(x, p)$  for all  $x$ . We assume that  $f$  is continuous in both variables  $(x, p)$ , and that there exists a constant  $C > 0$  such that

$$f(x, p) \geq C(|p| - 1) \tag{9.24}$$

for all  $x$  and  $p$ . This guaranties in particular that if  $\int_{\Omega} f(x, dDu) < +\infty$ , then  $u \in \text{BV}(\Omega, \mathbb{R}^k)$ .

**Lemma 9.3.** *Let  $u \in \text{BV}(\Omega; \Gamma^k)$ . With the set  $\mathcal{C}_0$  in (9.19) we have*

$$\int_{\Omega} f(x, dDu) = \sup_{\varphi \in \mathcal{C}_0} \sum_{i=1}^k \int_{\Omega \times \Gamma} \varphi_i \cdot dD1_{u_i}. \tag{9.25}$$

*Proof.* To simplify, we give an idea of the construction in case where  $u \in W^{1,1}(\Omega; \Gamma^k)$ . A precise proof in the general case is discussed in the appendix Section 9.8. In this case, (9.7) holds, and in particular one deduces that “ $\geq$ ” trivially holds in (9.25). Now, let for all  $x$  and  $(t_i)_{i=1}^k$

$$(\varphi_i^x(x, t_i))_{i=1}^k = \nabla_p f(x, \nabla u(x))$$

and

$$\varphi_i^t(x, t_i) = \frac{1}{k} f^*(x, \nabla_p f(x, \nabla u(x))).$$

This field is only measurable, and observe it does not depend on  $t$ . It can be smoothed by standard mollification: we consider  $\varrho \in C_c^\infty(B(0, 1); \mathbb{R}_+)$  with  $\int_{B(0,1)} \varrho \, dx = 1$  and let  $\varrho_\varepsilon(x) = \varepsilon^{-d} \varrho(x/\varepsilon)$ , and then let  $\varphi^\varepsilon = \varrho_\varepsilon * (\varphi \chi_{\Omega^\varepsilon})$  where  $\Omega^\varepsilon = \{x \in \Omega \mid \text{dist}(x, \partial\Omega) > \varepsilon\}$  and the convolution is only in the  $x$  variable. Here we use the fact that  $f^*(x, 0) = 0$  so that, in particular,  $0 \in \mathcal{C}_0$ . This smooth (and compactly supported in  $x$ ) function might not be in  $\mathcal{C}_0$ , but one can show that it is “close” to  $\mathcal{C}_0$  in some sense. Moreover, using (9.7), we have that

$$\begin{aligned} \lim_{\varepsilon \rightarrow 0} \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i^\varepsilon \cdot D1_{u_i} \, dx \\ = \sum_{i=1}^k \int_{\Omega} \varphi_i^x(x) \cdot \nabla u_i(x) - \varphi_i^t(x) \, dx = \int_{\Omega} f(x, \nabla u(x)) \, dx, \end{aligned} \tag{9.26}$$

as expected. □

### 9.3.4 Existence of Minimizers

For the relaxation (9.5) we seek for a solution  $v \in \text{BV}(\Omega \times \Gamma; [0, 1]^k)$ , with  $\Gamma = [0, 1]$ .

**Theorem 9.4.** *The problem*

$$\min \left\{ \mathcal{E}(v) \mid v \in \text{BV}(\Omega \times \Gamma; [0, 1]^k), v_i(x, 0) \equiv 1, v_i(x, 1) \equiv 0 \right\}, \tag{9.27}$$

where  $\mathcal{E}$  is defined by (9.5) with  $\mathcal{C}$  given by (9.17), has a solution.

Here the trace conditions on  $v_i$  are meant in the following sense: the functions are extended on  $\Omega \times \mathbb{R}$ , with  $v_i(x, t) \equiv 0$  if  $t \geq 1$  and 1 if  $t \leq 0$ , and the derivative  $Dv$  is then restricted to  $\Omega \times \Gamma$ . As a consequence, if the inner trace of  $v$  is different from 0 at  $t = 1$  or 1 at  $t = 0$ ,  $Dv$  carries a measure on the corresponding boundary.

*Proof.* This is straightforward, since by definition (9.5),  $\mathcal{E}$  is lower-semicontinuous on BV (with respect to weak convergence), while (9.24) (and the fact we have assumed that  $v$  is bounded) yields compactness of minimizing sequences for the problem (9.27).  $\square$

## 9.4 Special Cases

In this section we give some examples of regularizers  $R(u) = \int_{\Omega} f(x, \nabla u(x)) \, dx$  in (9.2) which can be handled in our framework. For each case we will give the corresponding constraints (9.10) for the set (9.17). We assume that the range set of each channel  $u_i$  is  $\Gamma := [0, 1]$  for clarity of presentation. The whole theory can of course be formulated with general intervals as range sets.

### 9.4.1 Separable Regularizers

We first consider separable regularizers

$$R(u) = \sum_{i=1}^k R_i(u_i), \quad R_i(u_i) = \int_{\Omega} f_i(x, \nabla u_i(x)) \, dx, \quad (9.28)$$

i.e.  $f(x, p) = \sum_{i=1}^k f_i(x, p_i)$  for all  $(x, p) \in \Omega \times \mathbb{R}^{m \times k}$ , which acts on each  $u_i$  independently. We have previously considered the convex relaxation of the functional (9.2) with the restriction to this kind of regularizers in Chapter 8. The relaxation was obtained by convexifying each term separately, the data term  $\int_{\Omega} c(x, u(x)) \, dx$  and the regularizers  $R_1(u_1), \dots, R_k(u_k)$ . In contrast, our proposed relaxation of (9.2) in the current chapter considers the functional as a whole and uses a single combined constraint set (9.17). However, for separable regularizers it turns out to be equivalent to relaxing each term separately as in Chapter 8, as we will show next.

Define the sets  $\mathcal{C}_0^i$  similarly as  $\mathcal{C}_0$  in (9.19) but for the one-dimensional case ( $k = 1$ ):

$$\mathcal{C}_0^i = \left\{ \varphi = (\varphi^x, \varphi^t) \in C_c^0(\Omega \times \Gamma; \mathbb{R}^m \times \mathbb{R}) \mid \right. \\ \left. \varphi^t(x, t) \geq f_i^*(x, \varphi^x(x, t)) \quad \forall (x, t) \in \Omega \times \Gamma \right\}. \quad (9.29)$$

**Proposition 9.5.** *Let the regularizer be separable as in (9.28). Then the relaxation*

$$\mathcal{E}(v) = \sup_{\varphi \in \mathcal{C}} \sum_{i=1}^k \int_{\Omega \times \Gamma} \varphi_i \cdot dDv_i \, dx = \sup_{\varphi \in \mathcal{C}} \sum_{i=1}^k \int_{\Omega \times \Gamma} (\varphi_i^x \cdot dD_x v_i + \varphi_i^t \, dD_t v_i) \quad (9.30)$$



with the general set  $\mathcal{C}$  in (9.17) is equal to the relaxation of each term separately:

$$\mathcal{E}(v) = \sum_{i=1}^k \left( \sup_{\varphi_i \in \mathcal{C}_0^i} \int_{\Omega \times \Gamma} \varphi_i \, dDv_i \right) + \left( \sup_{\gamma \in \mathcal{L}} \int_{\Omega \times \Gamma} \sum_{i=1}^k \gamma_i \, dD_t v_i \right). \quad (9.31)$$

*Proof.* Let  $(\varphi^x, \varphi^t) \in \mathcal{C}$  be fixed. The Legendre-Fenchel conjugate of  $f(x, p) = \sum_i f_i(x, p_i)$  is given by

$$f^*(x, q) = \sup_{p \in \mathbb{R}^{m \times k}} \sum_{i=1}^k q_i p_i - \sum_{i=1}^k f_i(x, p_i) = \sum_{i=1}^k f_i^*(x, q_i)$$

for all  $q \in \mathbb{R}^{m \times k}$ . The constraints (9.10) thus become

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq -c(x, t) + \sum_{i=1}^k f_i^*(x, \varphi_i^x(x, t_i)). \quad (9.32)$$

Define  $\gamma_i, \bar{\varphi}_i^t \in C^0(\Omega \times \Gamma; \mathbb{R})$  by

$$\begin{aligned} \bar{\varphi}_i^t(x, t_i) &:= f_i^*(x, \varphi_i^x(x, t_i)), \\ \gamma_i &:= \varphi_i^t - \bar{\varphi}_i^t \end{aligned}$$

for all  $(x, t_i) \in \Omega \times \Gamma$ . Obviously we have  $(\varphi_i^x, \bar{\varphi}_i^t) \in \mathcal{C}_0^i$  for all  $i$ , and by (9.32) also  $\gamma \in \mathcal{L}$ . On the other hand, if  $(\varphi_i^x, \bar{\varphi}_i^t) \in \mathcal{C}_0^i$  and  $\gamma \in \mathcal{L}$  then for  $\varphi_i^t := \bar{\varphi}_i^t + \gamma_i$  we have  $(\varphi^x, \varphi^t) \in \mathcal{C}$ . Therefore, (9.31) follows directly from (9.30).  $\square$

As a consequence, in order to arrive at novel relaxations, one has to consider *coupled* regularizers. For this general case, no tractable relaxations have yet been given. In fact, existing relaxations all rely on discretizing the whole  $k$ -dimensional label space  $\Gamma^k$  and are thus by no means tractable.

In the following we will give a brief overview of some interesting special cases for separable regularizers (9.28) which were studied in [101]. We will discuss these regularizers in more detail later in Sections 9.4.3 – 9.4.7, where we introduce the corresponding *coupled* versions.

**Total Variation with  $l^1$ -Coupling.** Setting  $f_i(x, p_i) := \lambda |p_i|$  with a  $\lambda > 0$  we obtain the total variation

$$TV_{l^1}(u) = \lambda \sum_{i=1}^k \int_{\Omega} |\nabla u_i| \, dx. \quad (9.33)$$

Although this regularizer is a simple way to extend the total variation to vector valued signals, there is no coupling of the channels. This generally leads to inferior reconstructions, as is demonstrated in Figure 9.5. The corresponding constraints in (9.29) are

$$\varphi_i^t(x, t) \geq 0, \quad |\varphi_i^x(x, t)| \leq \lambda \quad \forall x \in \Omega, t \in \Gamma, 1 \leq i \leq k. \quad (9.34)$$

**Huber-TV with  $l^1$ -Coupling.** We now set  $f_i(x, p_i) := \lambda h_\varepsilon(|p_i|)$  with the Huber function  $h_\varepsilon(p_i)$ , which basically equals  $|p_i|$  but smoothes out the kink at the origin, see the definition in (8.43). This yields the Huber-TV $_{l^1}$  penalization:

$$\lambda \sum_{i=1}^k \int_{\Omega} h_\varepsilon(|\nabla u_i|) \, dx. \quad (9.35)$$

This alleviates the staircasing effect caused by TV (i.e. the solutions tend to become piecewise constant in regions where  $u$  is almost constant but smooth), however there is no coupling of the channels at all. We will discuss the  $l^2$ -coupled version later in Section 9.4.4. The constraints in (9.29) are

$$\varphi_i^t(x, t) \geq \frac{\varepsilon}{2\lambda} |\varphi_i^x(x, t)|^2, \quad |\varphi_i^x(x, t)| \leq \lambda \quad \forall x \in \Omega, t \in \Gamma, 1 \leq i \leq k. \quad (9.36)$$

**Lipschitz-Constraint with  $l^1$ -Coupling.** Finally, when setting  $f_i(x, p_i) := \delta_{|p_i| \leq \lambda}$ , which is zero for  $|p_i| \leq \lambda$  and  $\infty$  otherwise, with a  $\lambda > 0$  we obtain the Lipschitz constraint on the gradients of the channels:

$$\sum_{i=1}^k \int_{\Omega} \delta_{|\nabla u_i| \leq \lambda} \, dx = \delta_{(|\nabla u_i(x)| \leq \lambda \text{ for a.e. } x \in \Omega, 1 \leq i \leq k)}.$$

The growth rate of each channel is constrained by  $\lambda$  individually without any coupling. The constraints in (9.29) are

$$\varphi_i^t(x, t) \geq \lambda |\varphi_i^x(x, t)| \quad \forall x \in \Omega, t \in \Gamma, 1 \leq i \leq k. \quad (9.37)$$

## 9.4.2 Separable Data Terms

The proposed convex relaxation handles the functional (9.2) as a whole, i.e. both the data and the regularization term are relaxed simultaneously using a unified constraint set. Consider separable data terms  $c$ ,

$$D(u) = \sum_{i=1}^k D_i(u_i), \quad D_i(u_i) = \int_{\Omega} c_i(x, u_i(x)) \, dx, \quad (9.38)$$

i.e.  $c(x, t) = \sum_{i=1}^k c_i(x, t_i)$  for all  $(x, t) \in \Omega \times \mathbb{R}^k$ . In this case we can show that the overall relaxation is equivalent to relaxing the data and regularizer term separately (as also was the case for separable regularizers in the previous section). Furthermore, the data term part decouples into separate relaxations of each channel:

**Proposition 9.6.** *Let the data term be separable as in (9.38). Then*

$$\mathcal{E}(v) = \left( \sup_{\varphi \in \mathcal{L}_0} \sum_{i=1}^k \int_{\Omega \times \Gamma} \varphi_i \cdot dDv_i \right) + \sum_{i=1}^k \left( \sup_{\gamma_i \in \mathcal{L}_i} \int_{\Omega \times \Gamma} \gamma_i \, dD_t v_i \right). \quad (9.39)$$

The sets  $\mathcal{L}_i$  are defined similarly to  $\mathcal{L}$  in (9.18) but for the one-dimensional case:

$$\mathcal{L}_i = \left\{ \gamma \in C_c^0(\Omega \times \Gamma; \mathbb{R}) \mid \gamma(x, t) \geq -c_i(x, t) \quad \forall (x, t) \in \Omega \times \Gamma \right\}.$$

*Proof.* The proof basically uses the same construction as the corresponding proof of (9.31) for the case of separable regularizers. The constraints (9.10) read

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq - \sum_{i=1}^k c_i(x, t_i) + f^*(x, (\varphi_i^x(x, t_i))_{i=1}^k). \quad (9.40)$$

For  $(\varphi^x, \varphi^t) \in \mathcal{C}$  with the set  $\mathcal{C}$  in (9.17) define  $\gamma_i, \bar{\varphi}_i^t \in C_c^0(\Omega \times \Gamma; \mathbb{R})$  by

$$\begin{aligned} \gamma_i(x, t_i) &:= -c_i(x, t_i), \\ \bar{\varphi}_i^t &:= \varphi_i^t - \gamma_i \end{aligned}$$

for all  $(x, t) \in \Omega \times \Gamma^k$  and  $1 \leq i \leq k$ . Then  $(\varphi^x, \bar{\varphi}^t) \in \mathcal{C}_0$  by (9.40) and evidently also  $\gamma_i \in \mathcal{L}_i$  for all  $i$ . On the other hand, if  $(\varphi^x, \bar{\varphi}^t) \in \mathcal{C}_0$  and  $\gamma_i \in \mathcal{L}_i$  for all  $i$ , then for  $\varphi_i^t := \bar{\varphi}_i^t + \gamma_i$  we have  $(\varphi^x, \varphi^t) \in \mathcal{C}$ . Thus, (9.39) follows from (9.30).  $\square$

**Remark.** Another case where the overall relaxation is equivalent to relaxing the data term and the regularizer separately, is when the Legendre-Fenchel dual of  $f$  has the form

$$f^*(x, p) = \sum_{i=1}^k g_i(x, p_i) + \delta_{C(x)}(p) \quad (9.41)$$

with some convex functions  $g_i$  and sets  $C(x) \subset \mathbb{R}^{m \times k}$ . This can be proven analogously as for (9.31) in Section 9.4.1. Specifically, in the case discussed in Section 9.4.1 we have  $g_i = f_i^*$  and  $C(x) = \mathbb{R}^{m \times k}$ . An example of a regularizer which satisfies (9.41) with a non-trivial set  $C(x)$  is given by  $TV_{l^2}$ , which we will discuss next.

### 9.4.3 Total Variation with $l^2$ -Coupling

As a first non-separable regularizer for vectorial signals  $u : \Omega \rightarrow \mathbb{R}^k$  we consider the total variation with the  $l^2$ -coupling of the channels. For smooth functions  $u$  it is given by

$$TV_{l^2}(u) = \lambda \int_{\Omega} |\nabla u| \, dx = \lambda \int_{\Omega} \sqrt{\sum_{i=1}^k |\nabla u_i|^2} \, dx \quad (9.42)$$

with a  $\lambda > 0$ . Basically, it penalizes the Euclidean norm of the gradient. For general  $u \in L^1(\Omega, \mathbb{R}^k)$  it can be defined by its dual representation (2.12). This *coupled* total variation generally leads to higher quality reconstructions in inverse problems than its *separable* counterpart in (9.33), as will be shown in the experiments. Our approach yields the first convex relaxation of this regularizer for vectorial multilabel problems. Furthermore, we will show how to efficiently reformulate the relaxation to obtain roughly the same run time and memory efficiency as in the decoupled case (9.33).

The corresponding function  $f : \Omega \times \mathbb{R}^{m \times k} \rightarrow \mathbb{R}$  in (9.2) is  $f(x, p) := \lambda |p|$  with the Legendre-Fenchel convex dual

$$f^*(x, q) = \sup_{p \in \mathbb{R}^{m \times k}} p q - |p| = \begin{cases} 0 & \text{if } |q| \leq \lambda, \\ \infty & \text{else.} \end{cases}$$

Thus, the constraints (9.10) are given by

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq -c(x, t), \quad (9.43)$$

$$\sqrt{\sum_{i=1}^k |\varphi_i^x(x, t_i)|^2} \leq \lambda \quad (9.44)$$

for all  $(x, t) \in \Omega \times \Gamma^k$ .

**Constraint Decoupling for the Smoothness Part (9.44).** For a practical implementation, the range set  $\Gamma$  of each channel  $u_i$  must be discretized into a number  $n_i \geq 1$  of levels. For each fixed  $x \in \Omega$ , the second constraint (9.44) then poses  $n_1 \cdots n_k$  individual constraints because of  $t \in \Gamma^k$ . Implementing them requires a large amount of memory for  $k \geq 3$  and to some extent for  $k \geq 2$ . Surprisingly, the special form of the  $l^2$ -coupled  $TV_{l^2}$  allows one to decouple (9.44) into only  $n_1 + \dots + n_k$  constraints.

The inequalities (9.44) for all  $t \in \Gamma^k$  are equivalent to

$$\sup_{t \in \Gamma^k} \sqrt{\sum_{i=1}^k |\varphi_i^x(x, t_i)|^2} = \sqrt{\sum_{i=1}^k \left( \sup_{t_i \in \Gamma} |\varphi_i^x(x, t_i)| \right)^2} \leq 1.$$

Introducing a new dual variable  $a : \Omega \rightarrow \mathbb{R}^k$ ,  $a_i(x) := \sup_{t_i \in \Gamma} |\varphi_i^x(x, t_i)|$  for all  $i$ , this shows that the constraints (9.44) can be equivalently written as

$$|a(x)| \leq \lambda \quad \forall x \in \Omega, \quad (9.45)$$

$$|\varphi_i^x(x, t_i)| \leq a_i(x) \quad \forall (x, t_i) \in \Omega \times \Gamma, \quad 1 \leq i \leq k. \quad (9.46)$$

For each  $x \in \Omega$  these are now only linearly many constraints.

**Remark.** There is an interesting interpretation of this decoupling: Considering the constraints (9.46) without (9.45) for each  $i$ , the supremum over  $\varphi_i^x$  gives the total variation of  $u_i$  where the contribution at each point  $x \in \Omega$  is weighted by  $a_i(x)$ :

$$TV_{a_i}(u_i) = \int_{\Omega} a_i(x) |\nabla u_i| \, dx. \quad (9.47)$$

Taking the supremum also over (9.45) means that  $TV(u)$  is represented as a weighted sum of  $TV_{a_i}(u_i)$ :

$$TV_{l^2}(u) = \sup_{\substack{a: \Omega \rightarrow \mathbb{R}^k, a \geq 0 \\ \sum_{i=1}^k a_i(x)^2 \leq \lambda}} \sum_{i=1}^k \int_{\Omega} a_i(x) |\nabla u_i| \, dx. \quad (9.48)$$

**Constraint Decoupling for Data Part (9.43).** Similarly to (9.44), after discretization the first constraint (9.43) in its original form gives  $n_1 \cdots n_k$  individual constraints for each  $x \in \Omega$ . However, if the data term is serapable as discussed in Section 9.4.2, we can also decouple (9.43) into linearly many constraints:

**Proposition 9.7.** *Assume that  $c(x, t) = \sum_{i=1}^k c_i(x, t_i)$  for all  $(x, t) \in \Omega \times \mathbb{R}^k$  with some functions  $c_i : \Omega \times \Gamma \rightarrow \mathbb{R}$ . Then (9.43) can be replaced by*

$$\varphi_i^t(x, t_i) \geq -c_i(x, t_i) \quad \forall (x, t_i) \in \Omega \times \Gamma, 1 \leq i \leq k. \quad (9.49)$$

without altering the supremum in the convex relaxation (9.5).

*Proof.* First, since the constraints on  $\varphi^t$  and on  $\varphi^x$  are independent, the relaxation (9.5) reads

$$\mathcal{E}(v) = \left( \sup_{\varphi^x} \int_{\Omega \times \Gamma} \sum_{i=1}^k \varphi_i^x \cdot dD_x v_i \right) + \left( \sup_{\varphi^t} \int_{\Omega \times \Gamma} \sum_{i=1}^k \varphi_i^t dD_t v_i \right)$$

with  $\varphi^x$  satisfying (9.44) and  $\varphi^t$  satisfying (9.43). For the second term on the right-hand side, regarding the data-term-only constraints (9.43) as the general constraints (9.10) with  $f \equiv 0$  and  $\varphi^x \equiv 0$ , we can use (9.39) (as the data term is separable) to get

$$\begin{aligned} \sup_{\varphi^t} \int_{\Omega \times \Gamma} \sum_{i=1}^k \varphi_i^t dD_t v_i \\ = \sum_{i=1}^k \sup_{\gamma_i \in \mathcal{L}_i} \int_{\Omega \times \Gamma} \gamma_i dD_t v_i + \sup_{\hat{\varphi}^t} \int_{\Omega \times \Gamma} \sum_{i=1}^k \hat{\varphi}_i^t dD_t v_i \end{aligned} \quad (9.50)$$

with  $\hat{\varphi}^t$  such that  $\sum_{i=1}^k \hat{\varphi}_i^t(x, t_i) \geq 0$ . The first term on the right-hand side of (9.50) yields the desired constraints (9.49), after renaming  $\gamma_i$  back to  $\varphi_i^t$ . It remains to show that the supremum in the second term on the right-hand side of (9.50) is actually zero. But this follows directly from Lemma 9.2.  $\square$

### 9.4.4 Huber-TV with $l^2$ -Coupling

Total variation regularization is known to produce solutions exhibiting so called “staircasing” effects. In the regions where the solution  $u$  is almost constant or varies very slowly, it may become piecewise constant instead of having a smooth variation. A common solution is to apply the Huber-TV regularization:

$$R(u) = \lambda \int_{\Omega} h_{\varepsilon}(|\nabla u|) dx \quad (9.51)$$

for some small  $\varepsilon > 0$ , where the Huber function  $h_{\varepsilon} : \mathbb{R} \rightarrow \mathbb{R}$  is defined as in (8.43).

It smooths out the kink at the origin of  $z \mapsto |z|$ . The quadratic penalization for near zero  $\nabla u$  ensures smooth variations in the regions where  $u$  is nearly

constant, thus avoiding the staircasing effect in these regions. The limiting case  $\varepsilon = 0$  yields the usual  $TV$ . The advantage in comparison to applying the channel-wise Huber-regularization (9.35), is that here the quadratic penalization kicks in only if *every* gradient component is small. The coupled Huber- $TV_{l_2}$  is thus nearer to the actual  $TV_{l_2}$  which it is approximating.

The function  $f$  in (9.2) is  $f(x, p) := \lambda h_\varepsilon(|p|)$ , with the dual

$$f^*(x, q) = \sup_{p \in \mathbb{R}^{m \times k}} \sum_{i=1}^k q_i p_i - \lambda h_\varepsilon(|p|) = \begin{cases} \frac{\varepsilon}{2\lambda} |q|^2 & \text{if } |q| \leq \lambda, \\ \infty & \text{else,} \end{cases} \quad (9.52)$$

and the constraints in (9.17) become

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq -c(x, t) + \frac{\varepsilon}{2\lambda} \sum_{i=1}^k |\varphi_i^x(x, t_i)|^2, \quad (9.53)$$

$$\sqrt{\sum_{i=1}^k |\varphi_i^x(x, t_i)|^2} \leq \lambda \quad (9.54)$$

for all  $(x, t) \in \Omega \times \Gamma^k$ .

**Constraint Decoupling for the Smoothness Part (9.54).** Note that the second constraint (9.54) is exactly the same as (9.44) in the case of  $TV$  in Section 9.4.3. The same reduction technique can therefore be applied to decouple this constraint into (9.45) and (9.46).

**Constraint Decoupling for the Data and Smoothness Part (9.53).** As for the first constraint (9.53), for general data terms  $c$  we can always separate it into a data-term-only part, and one responsible for regularization. Namely, define  $\gamma_i, \bar{\varphi}_i^t \in C^0(\Omega \times \Gamma; \mathbb{R})$  by

$$\begin{aligned} \bar{\varphi}_i^t(x, t_i) &:= \frac{\varepsilon}{2\lambda} |\varphi_i^x(x, t_i)|^2, \\ \gamma_i &:= \varphi_i^t - \bar{\varphi}_i^t \end{aligned}$$

for all  $(x, t_i) \in \Omega \times \Gamma$  and  $1 \leq i \leq k$ . Then (9.53) is equivalent to

$$\varphi_i^t = \gamma_i + \bar{\varphi}_i^t, \quad (9.55)$$

$$\sum_{i=1}^k \gamma_i(x, t_i) \geq -c(x, t), \quad (9.56)$$

$$\bar{\varphi}_i^t(x, t_i) \geq \frac{\varepsilon}{2\lambda} |\varphi_i^x(x, t_i)|^2 \quad (9.57)$$

for all  $(x, t) \in \Omega \times \Gamma^k$ . Thus, Huber- $TV_{l_2}$  penalization requires only linearly many new constraints (9.57) in addition to those of  $TV_{l_2}$ . This regularizer is therefore also very efficient, w.r.t. memory and run time.

If the data term is separable,  $c(x, t) = \sum_i g_i(x, t_i)$ , then we can further decouple the data term constraint (9.56) into linearly many constraints (9.49), as has been done for the identical constraints (9.43) in Section 9.4.3. For brevity

and to reduce the overall number of variables, we can use (9.55) to combine these data term constraints with (9.57) into just

$$\varphi_i^t(x, t_i) \geq -g_i(x, t_i) + \frac{\varepsilon}{2\lambda} |\varphi_i^x(x, t_i)|^2 \quad (9.58)$$

for all  $(x, t_i) \in \Omega \times \Gamma$  and  $1 \leq i \leq k$ .

### 9.4.5 Total Variation for General Norms

We can also define more general versions of the total variation by choosing other norms  $\|\cdot\|$  instead of the Euclidean norm in (9.42) in Section 9.4.3 in which to penalize image gradients  $\nabla u$ :

$$TV_{\|\cdot\|}(u) = \lambda \int_{\Omega} \|\nabla u\| \, dx. \quad (9.59)$$

The interaction term here is  $f(x, p) = \lambda \|p\|$ . The convex dual is given by the indicator function of the corresponding dual norm  $\|\cdot\|_*$ :

$$\begin{aligned} f^*(x, q) &= \sup_{p \in \mathbb{R}^{m \times k}} p q - \lambda \|p\| = \sup_{\substack{p \in \mathbb{R}^{m \times k}, t \geq 0 \\ \|p\|=1}} t(p q - \lambda) \\ &= \sup_{\substack{p \in \mathbb{R}^{m \times k} \\ \|p\|=1}} \delta_{p q \leq \lambda} = \delta_{(\sup_{p \in \mathbb{R}^{m \times k}, \|p\|=1} p q \leq \lambda)} \\ &= \delta_{\|q\|_* \leq \lambda} \end{aligned} \quad (9.60)$$

with

$$\|q\|_* := \sup_{\substack{p \in \mathbb{R}^{m \times k} \\ \|p\| \leq 1}} p q = \sup_{\substack{p \in \mathbb{R}^{m \times k} \\ \|p\|=1}} p q. \quad (9.61)$$

Constraints (9.10) thus become

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq -c(x, t), \quad (9.62)$$

$$\|(\varphi_i^x(x, t_i))_{1 \leq i \leq k}\|_* \leq \lambda \quad (9.63)$$

for all  $(x, t) \in \Omega \times \Gamma^k$ . While the first constraint (9.62) is the same as (9.43), the second one (9.63) is a generalization of (9.44). Note that (9.63) reduces to (9.44) for  $\|\cdot\| = |\cdot|$  since the dual norm is again  $|\cdot|$ .

**Constraint Decoupling.** Since the number of constraints in (9.63) is quite large, a practical question is, which norms allow to decouple the constrains. Immediate candidates for the generalization of the Euclidean case technique of Section 9.4.4 are the  $\kappa$ -norms:

$$\|x\|_{\kappa} := \sqrt[\kappa]{\sum_{i=1}^k |x_i|^{\kappa}} \quad \text{for } 1 \leq \kappa < \infty, \quad \|x\|_{\infty} := \max_{1 \leq i \leq k} |x_i|, \quad (9.64)$$

defined for  $x \in \mathbb{R}^{m \times k}$ . The dual norm (9.61) is  $\|\cdot\|_* = \|\cdot\|_\zeta$  where  $1 \leq \zeta \leq \infty$  is defined by  $\frac{1}{\kappa} + \frac{1}{\zeta} = 1$ , i.e.  $\zeta = \frac{\kappa}{\kappa-1}$ . Just as in Section 9.4.4, we can show that (9.63) is equivalent to

$$\|a(x)\|_{\frac{\kappa}{\kappa-1}} \leq 1 \quad \forall x \in \Omega, \quad (9.65)$$

$$|\varphi_i^x(x, t_i)| \leq a_i(x) \quad \forall (x, t_i) \in \Omega \times \Gamma, \quad 1 \leq i \leq k, \quad (9.66)$$

introducing additional dual variables  $a : \Omega \rightarrow \mathbb{R}^k$ .

**Natural Total Variation  $TV_J$  for Color Images.** An interesting special case of (9.73) is the vectorial total variation  $TV_J$  of Goldluecke et al. [54]. They showed that it yields improved results in inverse problems such as denoising, inpainting and superresolution in comparison to other possible total variations such as (9.33) and (9.42), (with norms  $\|\cdot\|_1$ , respectively  $\|\cdot\|_2 = |\cdot|$ ). While the initial approach [54] can only be used for convex data terms  $c$ , our vectorial multilabel convexification framework extends its applicability to arbitrary data terms.

The corresponding norm in (9.59) is defined for  $TV_J$  as the largest singular value of  $\nabla u$ :

$$\|p\| = \|(\sigma_i)_{1 \leq i \leq m}\|_\infty = \max_{1 \leq i \leq m} \sigma_m \quad (9.67)$$

where  $\sigma_1, \dots, \sigma_m \geq 0$  with  $m \leq \min(m, k)$  are the singular values of  $p \in \mathbb{R}^{m \times k}$ . The dual norm is the nuclear norm of  $p$ , which is the sum of the singular values:

$$\|p\|_* = \|(\sigma_i)_{1 \leq i \leq m}\|_1 = \sum_{i=1}^m \sigma_m. \quad (9.68)$$

There is no immediate way to decouple the arising constraints (9.63). Thus this regularizer is more costly in terms of memory and run time for nonconvex data terms than  $TV_{l_2}$  and  $TV_{l_1}$ .

### 9.4.6 Huber-TV for General Norms

Just as for  $TV_{l_2}$  in Section 9.4.4, one can consider Huber-TV regularization with general norms. The staircasing effects are then eliminated while the desired properties of the respective TV are still preserved. The case (9.51) generalizes to

$$R(u) = \lambda \int_{\Omega} h_\varepsilon(\|\nabla u\|) dx \quad (9.69)$$

with a general norm  $\|\cdot\|$  and the Huber-function  $h_\varepsilon$  in (8.43). The convexification is a straightforward generalization of Section 9.4.4: The function  $f$  in (9.2) is  $f(x, p) := \lambda h_\varepsilon(\|p\|)$ , and (9.52) becomes

$$f^*(x, q) = \begin{cases} \frac{\varepsilon}{2\lambda} \|q\|_*^2 & \text{if } \|q\|_* \leq \lambda, \\ \infty & \text{else} \end{cases} \quad (9.70)$$



with the dual norm (9.61). The constraints in (9.17) are now

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq -c(x, t) + \frac{\varepsilon}{2\lambda} \left\| (\varphi_i^x(x, t_i))_{1 \leq i \leq k} \right\|_*^2, \quad (9.71)$$

$$\left\| (\varphi_i^x(x, t_i))_{1 \leq i \leq k} \right\|_* \leq \lambda \quad (9.72)$$

for all  $(x, t) \in \Omega \times \Gamma^k$ . As an example, one can consider the Huber regularization of the natural TV for color images in Section 9.4.5.

### 9.4.7 Lipschitz-Constraint with $l^2$ -Coupling

In some applications the rate of growth of  $u$  is bounded a-priori by a constant. To enforce this, we can consider the following regularizer:

$$R(u) = \int_{\Omega} \delta_{|\nabla u| \leq \lambda} dx. \quad (9.73)$$

The interaction term here is  $f(x, p) = \delta_{|p| \leq \lambda}$  with the dual

$$f^*(x, q) = \sup_{p \in \mathbb{R}^{m \times k}} pq - \delta_{|p| \leq \lambda} = \sup_{|p| \leq \lambda} pq = \lambda |q|. \quad (9.74)$$

The constraints (9.10) become

$$\sum_{i=1}^k \varphi_i^t(x, t_i) \geq -c(x, t) + C \sqrt{\sum_{i=1}^k |\varphi_i^x(x, t_i)|^2} \quad (9.75)$$

for all  $(x, t) \in \Omega \times \Gamma^k$ . The constraints (9.75) cannot be easily decoupled, making the  $l^2$ -coupled Lipschitz constraint a costly regularizer.

## 9.5 Implementation

### 9.5.1 Discretization

**Variable Discretization.** We discretize the image domain into a rectangular pixel grid, which we again denote by  $\Omega$ . For each  $1 \leq i \leq k$  we also discretize the range set  $\Gamma$  of  $u_i : \Omega \rightarrow \Gamma$  into a number  $n_i \geq 1$  of levels

$$\frac{0}{n_i - 1}, \dots, \frac{n_i - 1}{n_i - 1} \quad \text{with spacing } \Delta t_i = \frac{1}{n_i - 1}. \quad (9.76)$$

The range discretization is necessary since the relaxed energy (9.5) is defined on the space  $\Omega \times \Gamma$ . We also consider the geometric average of the spacings

$$\Delta t = \sqrt{\prod_{i=1}^k \Delta t_i}. \quad (9.77)$$

We write  $\Gamma_i := \{0, \dots, n_i - 1\}$ , and set

$$\Lambda := \Gamma_1 \times \dots \times \Gamma_k \quad (9.78)$$

for the set of all labels. The discretized variables  $v$ ,  $\varphi$  and  $c$  are represented by their node values

$$\begin{aligned} v_i(x, \frac{j}{n_i-1}) &= v_i^j(x) \in \mathbb{R}, \\ \varphi_i^x(x, \frac{j}{n_i-1}) &= \frac{1}{\Delta t} \varphi_i^{x,j}(x) \in \mathbb{R}^m, \\ \varphi_i^t(x, \frac{j}{n_i-1}) &= \varphi_i^{t,j}(x) \in \mathbb{R}, \\ c(x, (\frac{j_1}{n_1-1}, \dots, \frac{j_k}{n_k-1})) &= c^{(j_1, \dots, j_k)}(x) \in \mathbb{R} \end{aligned} \quad (9.79)$$

for all pixels  $x \in \Omega$ ,  $0 \leq j < n_i$ ,  $1 \leq i \leq k$  and  $(j_1, \dots, j_k) \in \Lambda$ . Similarly as in the scalar case in Section 7.4.1, the reason to include the factors  $\frac{1}{\Delta t}$  for  $\varphi^x$  is to balance out the coefficients in front of the variables after the range discretizations. One and the same factor is chosen for all channels  $i$  to still enable efficient computation of projections for the prox operators.

**Differential Operator Discretization.** We use forward differences (2.33) with Neumann boundary conditions for the spatial gradient  $\nabla_x^+$ . Divergence  $\text{div} := -\nabla^T$  is then set to the negative adjoint operator, which is computed by (2.35).

The  $t$ -derivative  $(D_t v_i)(x, t)$  along the  $i$ -th range space  $\Gamma_i$  is discretized by forward differences  $\partial_t^+$  with zero boundary condition:

$$(D_t v)(x, \frac{i}{n-1}) = \frac{1}{\Delta t_i} (\partial_t^+ v_i)(x), \quad (9.80)$$

$$(\partial_t^+ v_i^j)(x) = \begin{cases} v_i^{j+1}(x) - v_i^j(x) & \text{if } j < n_i - 1, \\ -v_i^j(x) & \text{if } j = n_i - 1. \end{cases} \quad (9.81)$$

This way, in (9.82) we implicitly use  $v_i^{n_i}(x) = 0$ . The negative adjoint  $t$ -derivative  $\partial_{t,i}^- = -(\partial_{t,i}^+)^T$  is then given by backward differences:  $\partial_{t,i}^- p_t = p_t - p_{t-1}$  if  $t > 0$  and  $p_t$  if  $p = 0$ .

**Energy and Constraint Set Discretization.** Discretizing the spatial and range integrals similarly as in Section 7.4.1, the discretized energy becomes

$$\min_{v \in \mathcal{D}} \mathcal{E}(v), \quad (9.82)$$

$$\mathcal{E}(v) = \max_{\varphi \in \mathcal{C}} \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j < n_i} \left( \frac{\Delta t_i}{\Delta t} \langle \varphi_i^{x,j}(x), \nabla_x^+ v_i^j(x) \rangle + \varphi_i^{t,j}(x) \partial_{t,i}^+ v_i^j(x) \right).$$

We are looking for minimizers  $v$  which lie in the convex set

$$\mathcal{D} = \left\{ v = (v_i)_{1 \leq i \leq k} \mid v_i : \Omega \rightarrow [0, 1]^{n_i}, v_i^0(x) = 1 \quad \forall x \in \Omega, 1 \leq i \leq k \right\}. \quad (9.83)$$

Of the two boundary conditions  $v_i(x, 0) = 1$  and  $v_i(x, 1) = 0$ , only the first one is imposed explicitly in (9.83). The second one is encoded implicitly through the discretization of  $\partial_{t,i}^+$  as described above.

The monotonicity constraint on  $v_i$ , i.e. that  $v_i^j(x)$  is nonincreasing in  $j$ , is not included in  $\mathcal{D}$  since it is already implicitly implied by the constraint set (9.17): The dual variable  $\varphi^t$  may be arbitrarily large, and therefore the supremum in (9.82) is finite only if  $\partial_{t,i}^+ v_i^j \leq 0$ , i.e. if  $v_i$  is nonincreasing. This is analogous to the scalar case in Section 7.3.1.

The discretized set  $\mathcal{C}$  in (9.17) is

$$\mathcal{C} = \left\{ \varphi = (\varphi_i)_{1 \leq i \leq k} \mid \varphi_i = (\varphi_i^x, \varphi_i^t) : \Omega \times \Gamma_i \rightarrow \mathbb{R}^m \times \mathbb{R}, \right. \\ \left. \sum_{i=1}^k \varphi_i^{t,j_i}(x) \geq -c^j(x) + f^* \left( x, \frac{1}{\Delta t} (\varphi_i^{x,j_i}(x))_{i=1}^k \right) \quad \forall x \in \Omega, j \in \Lambda \right\}. \quad (9.84)$$

The set  $\mathcal{C}$  depends on the employed regularizer function  $f$ , as well as on the corresponding strategies to decouple the constraints. Implementation of these constraints is detailed later in Section 9.5.2.

**Optimality of Solutions.** Because of the convex relaxation of the range of graph functions from  $\{0, 1\}$  to  $[0, 1]$ , the computed solution  $v^*$  of (9.82) may be nonbinary. Therefore, at the end we need to project the result back to the space of binary functions. One possible solution is to threshold at  $\frac{1}{2}$ , i.e.

$$(v_{\text{bin}})_i^t(x) = \begin{cases} 1 & \text{if } t \leq t_i, \\ 0 & \text{if } t > t_i \end{cases} \quad \text{with } t_i := \max \left\{ t \in \{0, \dots, n_i - 1\} \mid v_i^t(x) \geq \frac{1}{2} \right\} \quad (9.85)$$

for every channel  $1 \leq i \leq k$ . From this we then construct a solution  $u_{\text{bin}}$  by (9.4).

Though this solution is not necessarily optimal for the initial problem (9.2), we have the energy bound (1.6) to estimate how far  $v_{\text{bin}}$  is from the unknown true solution  $u_{\text{bin}}^*$  of  $E$ :

$$\mathcal{E}(v^*) \leq E(u_{\text{bin}}^*) \leq E(u_{\text{bin}}) = \mathcal{E}(v_{\text{bin}}). \quad (9.86)$$

In our experiments the upper bound

$$\frac{E(u_{\text{bin}}) - E(u_{\text{bin}}^*)}{E(u_{\text{bin}})} \leq \frac{\mathcal{E}(v_{\text{bin}}) - \mathcal{E}(v^*)}{\mathcal{E}(v_{\text{bin}})} \quad (9.87)$$

was around 3% for the separable regularizers such as  $TV_{l1}$ , and around 6% for the coupled ones such as  $TV_{l2}$ . This shows that our approach is able to provide optimal or near-optimal solutions.

For candidate solutions  $u$  of the initial problem (9.2) it is not required that the values lie in a fixed discretized set, as is the case for  $u_{\text{bin}}$  above. Therefore, when computing the actual end result for (9.2) we use interpolated thresholding:

$$u_{\text{res},i}(x) := \frac{j_0 + s - \frac{1}{2}}{n_i - 1} \quad \text{with} \quad s := \frac{v_i^{j_0}(x) - \frac{1}{2}}{v_i^{j_0}(x) - v_i^{j_0+1}(x)} \quad (9.88)$$

and  $j_0$  given by (9.85). We observed that this generally yields a higher quality solution, with  $E(u_{\text{res}}) < E(u_{\text{bin}})$ .

### 9.5.2 Numerical Algorithm

To solve the saddle-point problem (9.82) we use Algorithm 4. We use the version of the algorithm where the “bar”-copies are introduced for the primals rather than for the duals since there will be many more dual variables than primal ones.

The step factors  $\tau_0, \sigma_0$  in (2.49) are chosen as  $\tau_0 = 10$  with  $\sigma_0 = \frac{1}{\tau_0}$  in all our experiments, except for inpainting where  $\tau_0 = 1000$ . Compared to  $\tau_0 = 1$  we observed a speed by around a factor of 2–5, and sometimes even more.

The projection of the main primal variable  $v$  is straightforward and can be done by simple clipping of the values  $v_i^j(x)$  to  $[0, 1]$ . For  $\varphi$  the projection is more involved since  $\mathcal{C}$  contains many nonlocal constraints. Our strategy is therefore to implement these constraints using the method of Lagrange multipliers or of convex dualization, depending on the regularizer.

Note that according to (9.84) we need to include the factor  $\frac{1}{\Delta t}$  for  $\varphi^x$  in every constraint discussed in Section 9.4.

**Total Variation with  $l^2$ -Coupling.** The continuous version has the constraints (9.45) and (9.46) for the smoothness part, as well as (9.43) respectively (9.49) for the data term part. We dualize the constraints (9.46) using the relation (2.76), adding new energy terms

$$\inf_{\alpha, \beta} \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j < n_i} \langle -\alpha_i^j(x), \frac{1}{\Delta t} \varphi_i^{x,j}(x) \rangle + \beta_i^j(x) a_i(x) \quad (9.89)$$

to the energy, with the constraints  $|\alpha_i^j(x)| \leq \beta_i^j(x)$  for all  $x, i$  and  $j$ . The optimization is then performed also over these additional dual variables  $\alpha$  and  $\beta$ , with  $\alpha_i^j(x) \in \mathbb{R}^m$  and  $\beta_i^j(x) \in \mathbb{R}$  for all  $x, i$  and  $j$ . This way, for  $a$  only the constraints (9.45) remain. The corresponding projection is easily done by clipping the absolute value of  $a$ . The projection of  $\alpha, \beta$  can also be easily computed, see appendix Section 9.9.1. One can write (9.89) equivalently without the factor  $\Delta t$  in the energy by replacing the constraint  $|\alpha_i^j(x)| \leq \beta_i^j(x)$  by  $\Delta t |\alpha_i^j(x)| \leq \beta_i^j(x)$ .

For data term part, we have the constraints (9.49) if  $c$  is separable, and (9.43) for the general case. The projection onto (9.49) is straightforward. For general data terms, there are two strategies to handle (9.43), depending on available memory.

As a first approach, we dualize every constraint of (9.43). Using (2.73), we include the new energy terms

$$\inf_{\mu} \sum_{x \in \Omega} \sum_{j \in \Lambda} -\mu^j(x) \left( \sum_{i=1}^k \varphi_i^{t,j_i}(x) + c^j(x) \right) \quad (9.90)$$

with the constraints  $\mu^j(x) \leq 0$  for all  $x \in \Omega$  and  $j \in \Lambda$ . The prox-operator for  $\mu$  is local for every  $x \in \Omega$  and  $j \in \Lambda$ :

$$\operatorname{argmin}_{\mu \leq 0} \frac{(\mu - \mu^0)^2}{2\tau} - \mu c^j(x) \quad (9.91)$$

for some  $\tau > 0$  and  $\mu^0 \in \mathbb{R}$ . The solution can be easily computed giving  $\mu = \min(0, \mu^0 + \tau c^j(x))$ . Dualization (9.90) introduces additional dual variables  $\mu$  into the *global* energy. Since these are  $|\Omega| \prod_{i=1}^k n_i$  individual variables, this approach is quite costly memory-wise. In comparison, the number of all other variables scales only as  $|\Omega| \sum_{i=1}^k n_i$ .

The second approach is to solve the projection

$$\underset{\varphi^t \text{ s.t. (9.43)}}{\operatorname{argmin}} \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j < n_i} \left( \varphi_i^{t,j}(x) - (\varphi_i^{t,j}(x))^0 \right)^2 \quad (9.92)$$

directly as a subproblem. This does not require any additional variables, but the projection must then be performed after each iteration of the primal-dual algorithm. Thus, this trades off a reduction of memory requirements for an increased run time (many times over). The projection (9.92) can be solved by introducing variables  $\mu$  and terms (9.90) to the *local* energy (9.92), with sup instead of inf and with  $\mu$  instead of  $-\mu$ . This can be done sequentially pixel for pixel, requiring only  $\prod_{i=1}^k n_i$  additional variables. To accelerate this process, one can process chunks of  $N_c \geq 1$  pixels in parallel, where  $N_c \leq |\Omega|$  is chosen as large as possible to fit into the available memory.

We employ the accelerated Algorithm 5 for the projection subproblem (9.92), which is possible because of the quadratic terms in  $\varphi^t$ . Since  $\varphi^t$  does not change much between two outer iterations, a small number of inner iterations can be chosen, e.g.  $N_{\text{iter}} = 10$ .

**Huber-TV with  $l^2$ -Coupling.** Here we have the constraints (9.54), as well as (9.58) for separable data terms  $c$ , respectively (9.55)–(9.57) for general  $c$ . Implementation of (9.54) is done exactly as for the *TV* case above. For separable data terms, one has to project onto the remaining constraint (9.58). This is a projection onto a parabola. Its computation leads to a cubic equation which can be solved in closed form. This is detailed in the appendix Section 9.9.2. Note that after the discretization,  $\frac{1}{\Delta t} \varphi^x$  in place of  $\varphi^x$  in (9.58) must be considered.

For general data terms, by (9.55)  $\varphi^t$  is replaced by two independent dual variables  $\gamma_i$  and  $\bar{\varphi}_i$ . The constraints (9.56) are implemented in exactly the same way as for the case of *TV* above. Finally, the projection onto (9.57) is again the projection onto a parabola and can be done quickly and in closed form. Again, note that the factor  $\frac{1}{\Delta t}$  must be included in front of  $\varphi^x$ .

**Total Variation for General Norms.** For special cases of  $\kappa$ -norms  $\|\cdot\|$  in (9.64), we have the simplified constraints (9.65) and (9.66) together with (9.62). The implementation for this case is the same as previously for *TV* with the Euclidean norm, again using the dualization (9.89). The only difference is that (9.45) is now replaced by (9.65), i.e.  $a$  is constrained in the  $\frac{\kappa}{\kappa-1}$ -norm instead of the 2-norm. The most interesting cases for  $\kappa$  are 1, 2 and  $\infty$ . The projection onto the corresponding dual ball (9.65) can be done in a straightforward way for each of these cases. Moreover, because of (9.66) the  $a_i$ s are always nonnegative, so (9.65) can be replaced by the more simple inequality  $\sum_{i=1}^k a_i \leq 1$  for  $\kappa = \infty$ .

For general norms we have the constraints (9.63). In general they cannot be decoupled to linearly many constraints. To implement them we can dualize every constraint using (2.75). We add the following new energy terms:

$$\inf_{\eta} \sum_{x \in \Omega} \sum_{j \in \Lambda} \left\{ \sum_{i=1}^k \left\langle -\eta_i^j(x), \frac{1}{\Delta t} \varphi_i^{x, j_i}(x) \right\rangle + \lambda \|\eta^j(x)\| \right\}. \quad (9.93)$$

This introduces additional primal variables  $\eta^j(x) \in \mathbb{R}^{m \times k}$  for every  $x \in \Omega$  and  $j \in \Lambda$  into the overall optimization. The factor  $\frac{1}{\Delta t}$  in front of  $\varphi^x$  can be eliminated by instead equivalently writing  $\Delta t$  in front of  $\lambda$ . The prox-operator is local for each  $x \in \Omega$  and  $j \in \Lambda$ :

$$\operatorname{argmin}_{\eta \in \mathbb{R}^{m \times k}} \frac{(\eta - \eta^0)^2}{2\tau} + \lambda \|\eta\|. \quad (9.94)$$

Using Moreau's identity (2.40), the solution is given by

$$\eta = \eta^0 - \pi_{\|\cdot\|_* \leq \tau\lambda}(\eta^0) \quad (9.95)$$

where  $\pi_{\|\cdot\|_* \leq \tau\lambda}$  is the projection onto the dual ball  $\{x \in \mathbb{R}^{m \times k} \mid \|x\|_* \leq \tau\lambda\}$ . In the case of  $TV_J$ , the dual norm  $\|\cdot\|_*$  is the nuclear norm (9.68). We refer to [54] for a detailed description of how to perform the corresponding projection.

The dualization (9.93) requires a considerable number of additional variables, namely  $km|\Omega| \prod_{i=1}^k n_i$  for  $\eta$ , where  $m = \dim \Omega$  and  $k$  is the number of channels. The same discussion about memory reduction applies here as previously with (9.90) for the case of non-separable data terms. If there is enough available memory, we can dualize every constraint of (9.63). Otherwise we can handle them locally by a subproblem, computing a projection after every outer iteration of the primal-dual algorithm. Again, chunks of pixels can be processed in parallel to accelerate the process.

**General Regularizers.** In the general case we have the constraints (9.10). These are  $n_1 \cdots n_k$  constraints, after the discretization of the range spaces, for each pixel  $x \in \Omega$ . Beside the  $TV$  and Huber- $TV$  cases, in general they cannot be decoupled into linearly many constraints. To implement the general case, we can dualize every constraint using the convex dualization based on the general relation (2.8) (for  $x \in \mathbb{R}^{m \times k}$ ,  $y \in \mathbb{R}$ ):

$$\delta_{y \geq f^*(x)} = \sup_{\eta \in \mathbb{R}^{m \times k}, \mu \in \mathbb{R}} (\mu y + \eta x) - \delta_{y \geq f^*(x)}^*(\eta, \mu). \quad (9.96)$$

We add the terms  $-\delta_{\sum_i \varphi_i^t + g \geq f^*(\frac{1}{\Delta t} \varphi^x)}$  to the energy, i.e.

$$\inf_{\eta, \mu} \sum_{x \in \Omega} \sum_{j \in \Lambda} \left\{ -\mu^j(x) \left( \sum_{i=1}^k \varphi_i^{t, j_i}(x) + c^j(x) \right) - \sum_{i=1}^k \left\langle \eta_i^j(x), \varphi_i^{x, j_i}(x) \right\rangle + \delta_{y \geq f^*(\frac{1}{\Delta t} x)}^*(\eta^j(x), \mu^j(x)) \right\}. \quad (9.97)$$

This introduces additional primal variables  $\eta^j(x) \in \mathbb{R}^{m \times k}$  and  $\mu^j(x) \in \mathbb{R}$  for every  $x \in \Omega$  and  $j \in \Lambda$ . Note that both (9.90) and (9.93) are special cases of the above general dualization (9.97).

The prox-operator is local for each  $x \in \Omega$  and  $j \in \Lambda$ :

$$\operatorname{argmin}_{\eta \in \mathbb{R}^{m \times k}, \mu \in \mathbb{R}} \frac{(\eta - \eta^0)^2}{2\tau} + \frac{(\mu - \mu^0)^2}{2\tau} - \mu c^j(x) + \delta_{y \geq f^*(\frac{1}{\Delta t}x)}^*(\eta, \mu). \quad (9.98)$$

Define  $\widehat{\mu}^0 := \mu^0 + \tau c^j(x)$ . Using Moreau's identity (2.40) again, the solution is then

$$(\eta, \mu) = (\eta^0, \widehat{\mu}^0) - \tau \pi_{y \geq f^*(\frac{1}{\Delta t}x)} \left( \frac{\eta^0}{\tau}, \frac{\widehat{\mu}^0}{\tau} \right) \quad (9.99)$$

where  $\pi_{y \geq f^*(\frac{1}{\Delta t}x)}(x^0, y^0)$  is the projection onto the set  $\{(x, y) \in \mathbb{R}^{m \times k} \times \mathbb{R} \mid y \geq f^*(\frac{1}{\Delta t}x)\}$ .

Note that the advantage of the implementation framework (9.97) is that it is the same independent of the interaction terms  $f$ , i.e. of the regularizer. The only place where  $f$  enters the computation are the projections (9.99). Once the framework is implemented, it can be easily adapted for different regularizers by merely replacing the projection.

As previously with total variation for general norms, the dualization (9.97) also requires many additional variables. The same memory reduction strategy can be applied, at the expense of a higher run time.

**Huber-TV for the Spectral Norm.** The regularizer (9.69) is implemented using the general scheme presented above. By formula (9.70), in (9.99) we need to project onto

$$\left\{ (x, y) \in \mathbb{R}^{m \times k} \times \mathbb{R} \mid y \geq \frac{\varepsilon}{2\lambda\Delta t^2} \|x\|_*^2, \|x\|_* \leq \lambda\Delta t \right\} \quad (9.100)$$

with the nuclear norm  $\|\cdot\|_*$  in (9.68). Writing out the norm of  $x$  in terms of the singular values, this projection can be done in closed form. The common case of  $2D$  color images, i.e.  $m = 2$  and  $k = 3$  is detailed in the appendix Section 9.9.3.

**Lipschitz Regularizer with  $l^2$ -Coupling.** The Lipschitz regularizer (9.73) is also implemented by the general scheme. By (9.74), one has to project onto

$$\left\{ (x, y) \mid y \geq \frac{1}{\Delta t} \lambda |x| \right\}. \quad (9.101)$$

This projection can be done in closed form as detailed in the appendix Section 9.9.1.

### 9.5.3 Memory Requirements

Here we give summary of the overall memory required to implement the proposed convexification approach with coupling regularizers such as *TV* and *Huber-TV*. Depending on the regularizer and on the separability of the data term

Variable or constant	Floating point numbers
$v_i^j$	$ \Omega  \sum_i n_i$
$\varphi_i^j$	$(m+1)  \Omega  \sum_i n_i$
$c_i^j$ (if $c$ separable)	$ \Omega  \sum_i n_i$
$c^j$ (if $c$ non-sep.)	$ \Omega  \prod_i n_i$

**Table 9.1: Number of floating point numbers for energy (9.82).** The required memory is proportional to these amounts with 4 bytes per float.

Regularizer	Additional variables	Floating point numbers
$TV_{l^1}$ /Huber- $TV_{l^1}$	$\mu^j$ (if $c$ non-sep.)	$ \Omega  \prod_i n_i$
$TV_{l^2}$ /Huber- $TV_{l^2}$ / $TV_{l^\kappa}$ /Huber- $TV_{l^\kappa}$	$\alpha_i^j$ $\beta_i^j$ $\gamma_i^j$ (if $c$ non-sep., for Huber) $\mu^j$ (if $c$ non-sep.)	$km  \Omega  \sum_i n_i$ $ \Omega  \sum_i n_i$ $ \Omega  \sum_i n_i$ $ \Omega  \prod_i n_i$
$TV_J/TV_{\ \cdot\ }$	$\eta_i^j$ $\mu^j$ (if $c$ non-sep.)	$km  \Omega  \prod_i n_i$ $ \Omega  \prod_i n_i$
Huber- $TV_J$ / Lipschitz $_{l^2}$ /general	$\eta_i^j$ $\mu^j$	$km  \Omega  \prod_i n_i$ $ \Omega  \prod_i n_i$

**Table 9.2: Additional float numbers depending on the employed regularizer.** Overall memory for isotropic  $TV_{l^2}$  and Huber- $TV_{l^2}$  scales *linearly* with the range discretization for separable data terms. It is also in the same range as for the case of non-isotropic  $TV_{l^1}$  and Huber- $TV_{l^1}$ .

there are basically two kinds of variables and constants. The first kind, such as the basic variables  $v$  and  $\varphi$ , requires a *linearly* scaling amount of memory in terms of image size and the discretization levels,  $\mathcal{O}(|\Omega| \sum_{i=1}^k n_i)$ , and is thus relatively cheap to store. For the second kind, e.g. for variables  $\eta$  and  $\mu$ , the memory scales *exponentially* as  $\mathcal{O}(|\Omega| \prod_{i=1}^k n_i)$ .

The variables and constants appearing in the energy (9.82) require the amounts of floating point numbers, and thus memory, as shown in Table 9.1. Additional memory is needed to decouple and dualize the nonlocal constraints in the set  $\mathcal{C}$  in (9.17) depending on the type of employed regularizer, as shown in Table 9.2. First, observe that in the case of separable data terms, the overall memory for  $TV$  and Huber- $TV$  with  $l^2$ -coupling scales *linearly*. Also, it is always in the same range as in the uncoupled  $l^1$ -case, no matter if the data term is separable or not. Thus, for  $TV$  and Huber- $TV$  regularized optimization problems with nonconvex data terms, the proposed framework offers the advantage of *channel coupling* at nearly the same costs as without any coupling.

For general data terms, as well as for general regularizers, additional memory of size proportional to  $|\Omega| \prod_i n_i$  is needed, if we use the global dualization in each case. This strategy should be used whenever possible as this gives the best run times. If there is not enough memory, as described in Section 9.5.2 we can revert to *local* projection subproblems instead. The memory is then



$\mathcal{O}(N_C \prod_i n_i)$  and  $N_C$  is chosen appropriately as big as possible for the overall problem to fit on the GPU. This approach however comes at a high penalty in terms of run time (larger by factor 5–10).

Another costly constant is the data term  $c^j$  (if it is non-separable). During local projections it is advisable to store the whole array on the GPU. If this is not possible,  $c$  can be stored on the CPU side, and one can copy the required parts for the currently processed chunk of pixels to the GPU. Since CPU-to-GPU memory copies are rather slow, this increases the run time by a factor of 5–10. Thus, if the data term has a simple structure, one should compute it on the fly as needed.

Finally, the whole approach can also be parallelized on multiple GPUs. For this one subdivides the image domain  $\Omega$  in equal parts (e.g. horizontal stripes), and lets each GPU perform the described computations for one such part. After each iteration, the overlap regions need to be copied between the GPUs, which comes at virtually no cost since the overlap regions are rather small. Beside decreasing the run time, another compelling advantage of a multi-GPU setting is the much higher overall amount of memory available, allowing to solve larger problems. This is especially interesting e.g. for optical flow computation, as will be detailed in Section 9.6.1.

## 9.6 Experimental Results

In the following we demonstrate the usage of the proposed convex relaxations on several vectorial imaging problems. All algorithms were parallelized using the CUDA framework on three NVIDIA GTX 680 GPUs. The number of iterations for the primal-dual algorithm was chosen appropriately so that the solution remained visually stable and did not change anymore, which is usually the case after 1000–5000 iterations.

### 9.6.1 Optical Flow

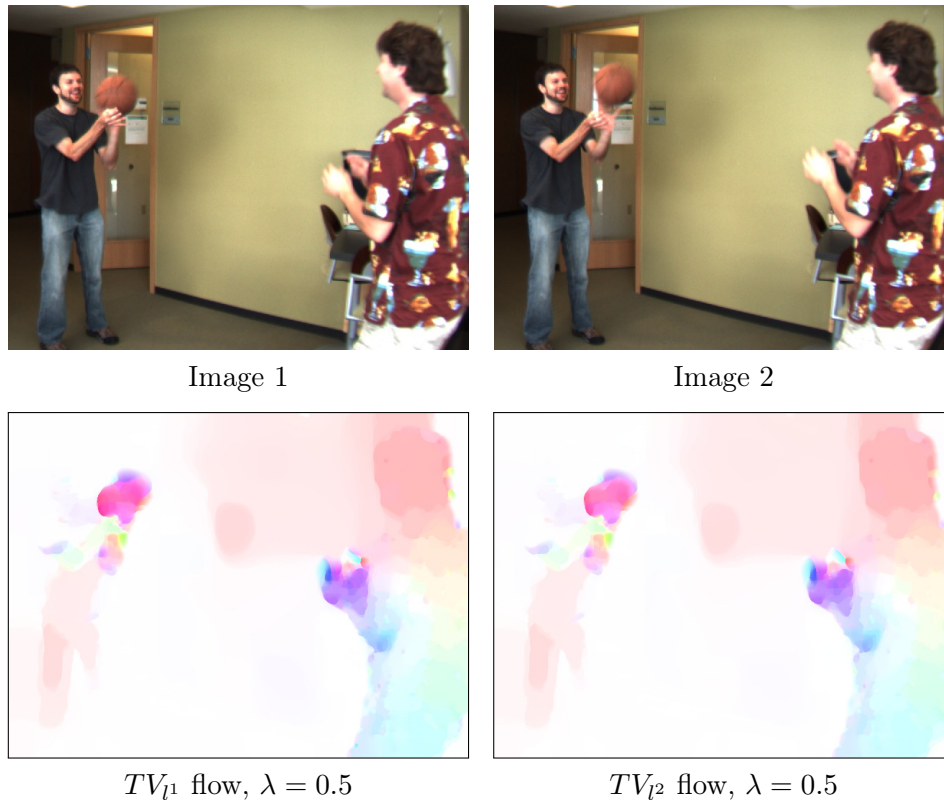
The task of optical flow estimation, or image matching, is to find point correspondences between two images. Given two color images  $I_1 : \Omega_1 \rightarrow \mathbb{R}^3$ ,  $I_2 : \Omega_2 \rightarrow \mathbb{R}^3$  which show one and the same scene, but taken from different viewpoints or at different times, one seeks a function  $u : \Omega_1 \rightarrow \mathbb{R}^2$  such that  $I_1(x) = I_2(x + u(x))$ . In practice, this relation will not be satisfiable exactly, and therefore one seeks  $u$  as a minimizer of the energy functional

$$E(u) = \int_{\Omega_1} c(x, u(x)) \, dx + TV(u) \quad (9.102)$$

where the data term is given e.g. by color differences

$$c(x, t) = |I_1(x) - I_2(x + t)|. \quad (9.103)$$

One can also sum up the image differences in a small window around  $x$ , e.g. with 1 pixel radius. A regularization term, such as  $TV(u)$  here, is needed to ensure a spatial coherence of the flow, as the local estimates by minimizing  $c(x, \cdot)$  point-wise in each point  $x$  may differ considerably. The *separable*  $TV_{l^1}$  regularizer is

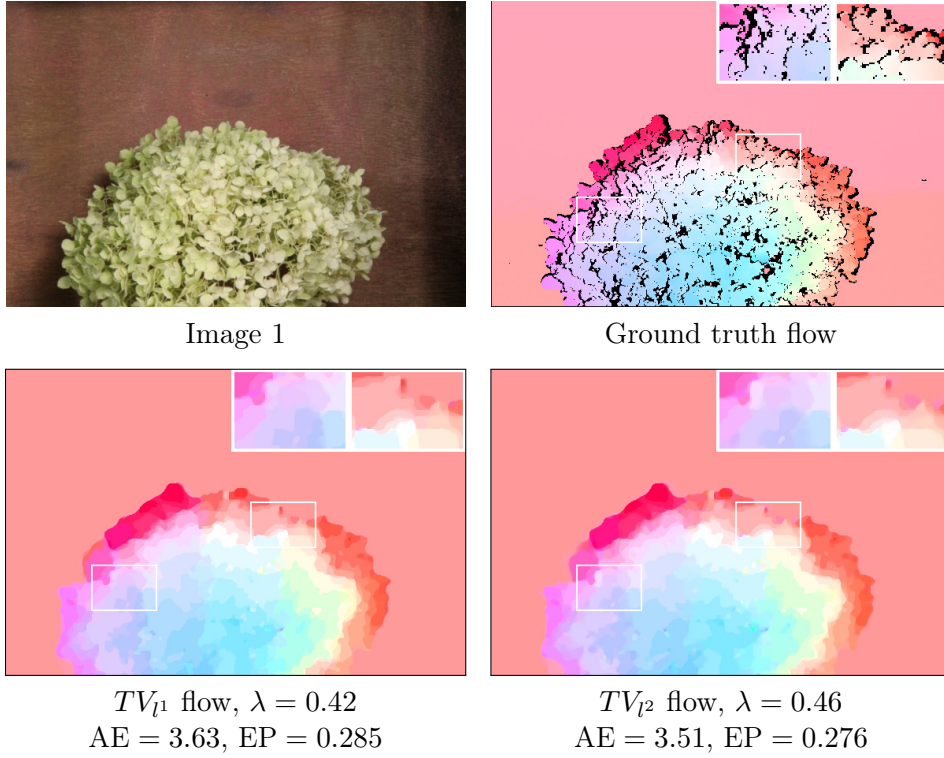


**Figure 9.1: Optic flow.** “Basketball” image pair from the Middlebury dataset. If there is no need for much smoothing, as here, the results with  $TV_{l1}$  and  $TV_{l2}$  are visually similar. The difference is more pronounced for larger weights, see Figure 9.3. Images  $640 \times 480$  using  $50 \times 50$  labels.

often employed in optical flow computations. With our framework, it becomes possible to also use the *coupled*  $TV_{l2}$ , which is rotationally invariant and thus a more favorable choice. Note that some literature refers by the term “optical flow” to a *linearized* version of the model (9.102), which is then already convex. However, the linearization is only a technical means to cope with the nonconvexity and does not describe the image matching problem correctly, e.g. for large scale flow.

**Comparison of Coupled  $TV_{l2}$  with Uncoupled  $TV_{l1}$ .** Figure 9.1 shows an optical flow computation on a real world image taken from the Middlebury optical flow dataset [10]<sup>2</sup>. There is no need for much smoothing in this case, therefore the results for  $TV_{l1}$  and  $TV_{l2}$  are visually almost the same. The  $640 \times 480$  image with a  $50 \times 50$  label space requires 68.16 seconds for  $TV_{l1}$  and 71.04 seconds for  $TV_{l2}$ . The most time is spent for the convexification scheme (9.90) of the non-separable data term (9.103), while the updates due to  $TV_{l1}/TV_{l2}$  are negligible. GPU memory required for  $TV_{l1}$  and  $TV_{l2}$  is 6918 MB, respectively 7275 MB.

<sup>2</sup><http://vision.middlebury.edu/flow>



**Figure 9.2: Comparing to ground truth for optic flow.** Shown are the results for the “Hydrangea” image pair from the Middlebury dataset. The flow using  $TV_{l_2}$  has smaller error measures (average angular error  $AE$ , average endpoint error  $EP$ ) than with  $TV_{l_1}$ . Image size  $584 \times 388$  using  $22 \times 22$  labels. The run time for  $TV_{l_1}$  and  $TV_{l_2}$  is 13.95 resp. 15.19 seconds, requiring 1182 resp. 1299 MB of memory.

A comparison with the ground truth optical flow is shown in Figure 9.2. In terms of common error measures, the  $TV_{l_2}$  regularization yields a more accurate flow than  $TV_{l_1}$  for the same data term. In this example, for the  $584 \times 388$  images we used a  $22 \times 22$  label space.

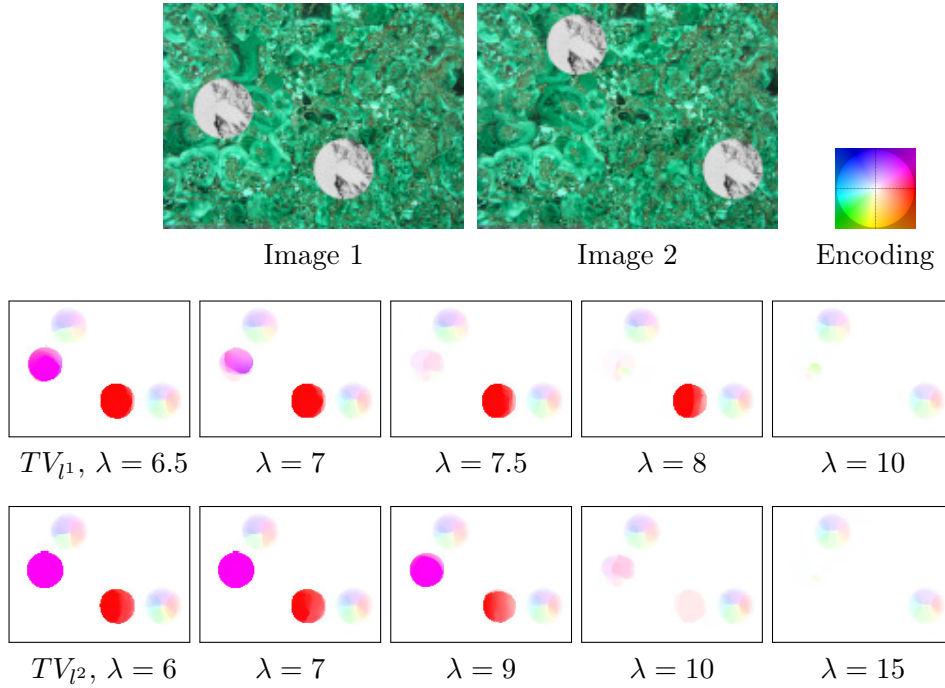
The difference between  $TV_{l_1}$  and  $TV_{l_2}$  becomes more apparent for larger weights  $\lambda$ . Since  $TV_{l_1}$  is not rotationally invariant, one and the same optical flow may be penalized differently if the underlying images — and along with them also the flow — are rotated by a certain angle. This is stated in the following proposition:

**Proposition 9.8** ( $TV_{l_1}$  is not rotationally invariant). *Set  $\Omega = \{x \in \mathbb{R}^2 \mid |x| < 1\}$ . For fixed angles  $\theta \in \mathbb{R}$  define the vector field  $v_\theta : \Omega \rightarrow \mathbb{R}^2$  by*

$$v_\theta(x) = \begin{cases} e_\theta & \text{if } |x| \leq r, \\ 0 & \text{else.} \end{cases} \quad (9.104)$$

for all  $x \in \Omega$  with  $e_\theta := (\cos \theta, \sin \theta)$ . Then

$$TV_{l_1}(v_\theta) = 2\pi r (|\cos \theta| + |\sin \theta|). \quad (9.105)$$



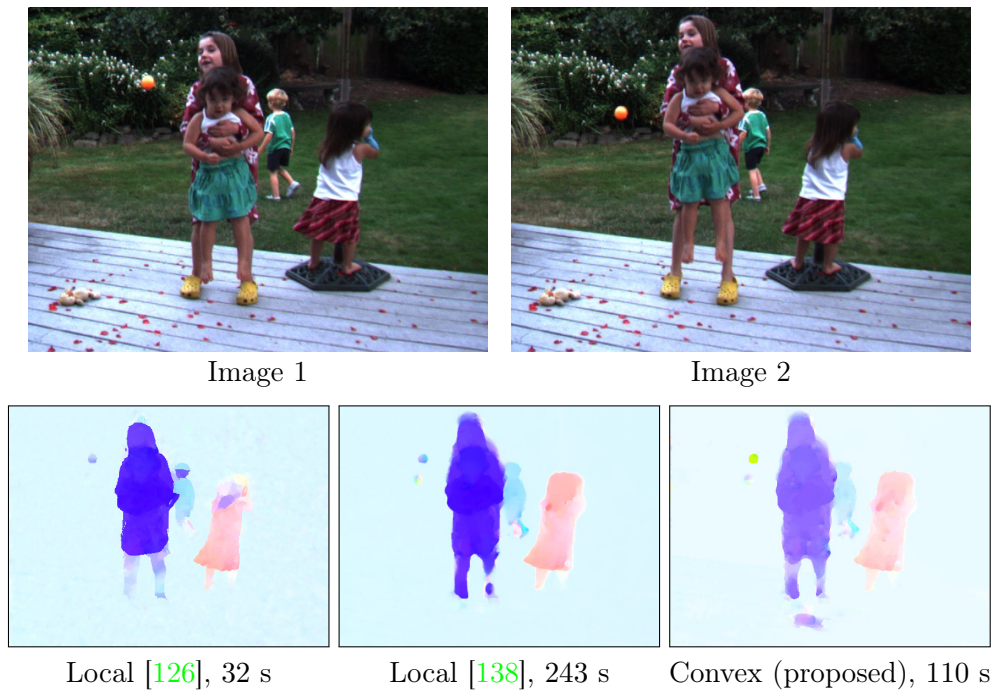
**Figure 9.3: Rotational invariance.** The synthetic image pair consists of two identical circles, traveling along the  $x$ -axis and diagonally. Although the two motions are equal up to direction,  $TV_{l^1}$  prefers grid aligned motions. The diagonal one disappears first when the regularizer weight  $\lambda$  increases. In contrast, the rotationally invariant  $TV_{l^2}$  handles the two motions equally. The right and top artifacts arise because of occlusions, which are not modeled by the simple data term (9.103).  $160 \times 120$  images using  $80 \times 80$  labels.

*Proof.* This follows directly from the representation

$$\begin{aligned} TV_{l^1}(v_\theta) &= TV((v_\theta)_1) + TV((v_\theta)_2) = TV(\chi_{B_r(0)} |(e_\theta)_1| + TV(\chi_{B_r(0)} |(e_\theta)_2| \\ &= 2\pi r (|\cos \theta| + |\sin \theta|), \end{aligned} \tag{9.106}$$

together with the property of  $TV$  that  $TV(\chi_A) = \mathcal{H}^{m-1}(\partial_* A)$  for rectifiable sets  $A \subset \Omega$ .  $\square$

In contrast, the  $l^2$ -coupled  $TV_{l^2}$  regularizer is rotationally invariant. For the above vector field we would get  $TV_{l^2}(v_\theta) = 2\pi r$ , which is the same for all  $\theta$ . This difference is demonstrated in Figure 9.3. The two circles are moving in essentially the same way, but in different directions. The background motion outside the circles will be nearly zero, as the texture ensures that the zero vector field will have the smallest data term. Since  $TV_{l^1}$  penalizes diagonal motions more (by factor  $\sqrt{2}$  due to (9.105)) than the ones parallel to the axes, choosing the weight  $\lambda$  in a certain range may result in unequal treatment of the two basically identical motions. This is not the case for  $TV_{l^2}$ . For every value of  $\lambda$  the two motions are recognized at the same time. The run time for the  $160 \times 120$  image with a  $80 \times 80$  label space is 22.05 seconds for  $TV_{l^1}$  and 22.45 seconds for  $TV_{l^2}$ , memory requirements are 1047 MB, respectively 1080 MB.






**Figure 9.4: Comparison with local approaches.** Shown is the optical flow between frames 10 and 12 “Backyard” sequence from the Middlebury dataset. Due to linearization and coarse to fine schemes, local approaches usually fail to correctly recognize large motion of small scale objects, such as the orange ball (moving downwards, green color in the flow encoding). In contrast, convex approaches, such as the proposed one, can use the original nonconvex data term and are able to recognize such motion. Image size  $544 \times 408$  using  $75 \times 75$  labels. Run time for local methods is on the CPU (no GPU versions are available), for convexification on the GPU.

**Comparison with Locally Convergent Approaches.** In case of a non-separable data term, such as in the optical flow example, the proposed approach is computationally expensive in terms of memory and run time. In view of this, a natural question is whether this complexity is really necessary and whether one can use *locally* convergent methods instead. An advantage of these methods is that they work in the original image domain and thus require much less memory than the convexification approach.

Their main disadvantage is of course their local optimality, which means that only certain flows can be reliably detected. Recent local approaches for optical flow estimation, such as [126] and [138], employ some kind of convex approximation to cope with the nonconvexity of the original problem. For instance, the data term is usually linearized around the current solution. Since this is only valid for small motions, a heuristic coarse-to-fine scheme needs to be employed in order to also recognize large scale motions: Starting with down-scaled images the optical flow is transferred to and iteratively refined on the next higher resolution.

As a result, these methods generally cannot detect large scale motion of

Image	$TV_{l1}$	$TV_{l2}$
 $368 \times 270$	2.04	2.85
 $481 \times 321$	3.51	5.16
 $326 \times 244$	2.01	2.79

**Table 9.3:** Run times [s] for Figure 9.5. Label space  $32 \times 32 \times 32$ .

small scale objects. This is demonstrated in Figure 9.4: The ball on the left side of the images is not recognized for any choice of the parameters. In contrast, our convex approach handles the original, non-linearized data term and avoids this limitation. We used the publicly available source code for [126]<sup>3</sup>, resp. binaries for [138]<sup>4</sup> for the experiments. The CPU run times for [126, 138] (no GPU versions are available) are comparable with the GPU run times for the convexification.

On the other hand, for small scale motion, state-of-the-art local methods tend to outperform the convexification method in terms of flow quality, see the comparison in Table 8.6 for the case of  $TV_{l1}$ . However, this comes at the cost of increased complexity: They usually employ highly engineered data terms, and features such as handling of occlusions and illumination changes, combined in a multi-stage optimization framework. In contrast, we use a very simple data term since the focus of this chapter lies on the regularizer part. With more elaborate data terms the convexification method is likely to achieve competitive results, which is left for future work.

### 9.6.2 Color Denoising

A natural application of our framework is to apply the coupled regularizers to denoise color images  $f : \Omega \rightarrow \mathbb{R}^3$ . We seek an image  $u : \Omega \rightarrow \mathbb{R}^3$  minimizing

$$E(u) = \int_{\Omega} c(x, u(x)) dx + TV(u) \quad (9.107)$$



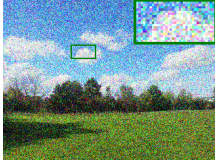
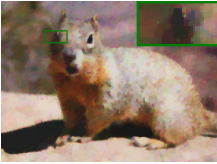
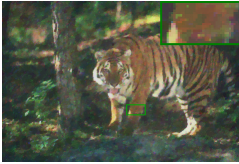

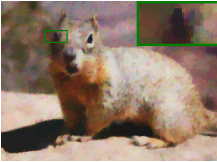
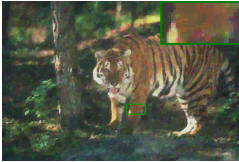







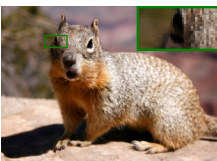
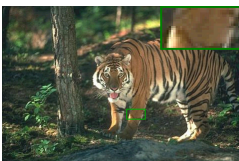

where  $TV$  is either  $TV_{l2}$  or  $TV_{l1}$ . Note that our framework allows possibly *non-convex* data terms  $c$ . Specifically, we choose the truncated quadratic differences, or the truncated linear ones:

$$c_2(x, t) = \sum_{i=1}^3 \min \{T, |t_i - f_i(x)|^2\}, \quad c_1(x, t) = \sum_{i=1}^3 \min \{T, |t_i - f_i(x)|\} \quad (9.108)$$

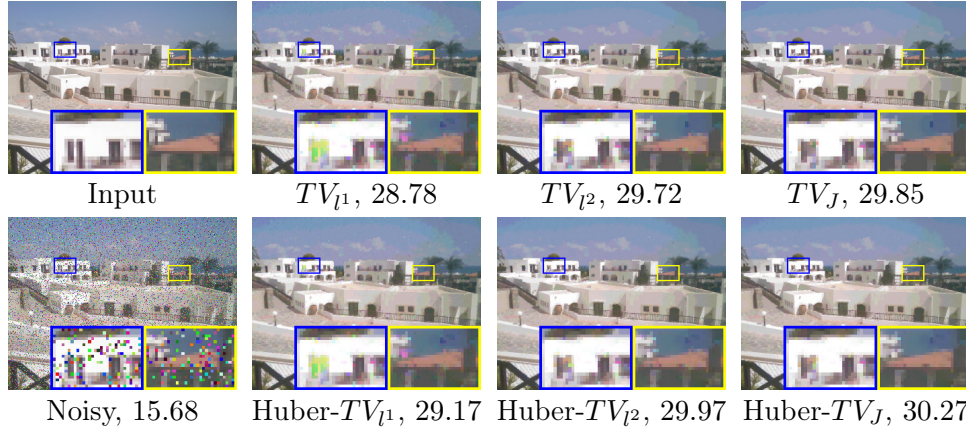
with a fixed threshold  $0 < T \leq 1$ . Since  $c$  is separable in each case, the model allows a fast implementation with only linearly many constraints, as described in Section 9.4.3. Note that the data term is convex for  $T = 1$ , and nonconvex for  $T < 1$ . More precisely, e.g. for  $c_2$  it becomes nonconvex once  $T$  has an effect on  $c_2$ , i.e. for  $T < \max_{x \in \Omega, 1 \leq i \leq k} \max(f_i(x), 1 - f_i(x))^2$ .

<sup>3</sup><http://ps.is.tuebingen.mpg.de/person/black>

<sup>4</sup><http://www.cse.cuhk.edu.hk/~leojia/projects/flow>

Noisy image			
	14.76	14.83	14.65
Separable $TV_{l_1}$ , convex data term, $T = 1$			
	24.47	23.49	25.88
Separable $TV_{l_1}$ , nonconvex data term, $T = 0.3$			
	24.49	23.53	25.90
Coupled $TV_{l_2}$ , convex data term, $T = 1$			
	25.01	24.10	26.35
Coupled $TV_{l_2}$ , nonconvex data term, $T = 0.3$			
	25.03	24.12	26.39
Original image			

**Figure 9.5: Denoising.** Input images were degraded by additive channel-wise Gaussian noise with standard deviation  $\sigma = 0.2$ . For each image and regularizer, the optimal weight  $\lambda$  was chosen manually to maximize the PSNR value. The *coupled*  $TV_{l_2}$  regularizer leads to systematically higher PSNR values and thus denoising quality. Using a *nonconvex* data term,  $c_2$  in (9.108), reconstructions of a higher quality can be achieved. Label space  $32 \times 32 \times 32$ .



**Figure 9.6: Regularizer comparison.** Input image has been degraded by impulse noise (20% of pixels set to random values). For each regularizer, the denoising model with a nonconvex data term ( $c_1$  in (9.108),  $T = 0.2$ ) was solved with manually chosen optimal parameters  $\lambda$  and  $\varepsilon$ . The *coupled* regularizers  $TV_{l_2}$  and  $TV_J$  outperform the separable  $TV_{l_1}$  in denoising quality in terms of the PSNR values. The Huber versions consistently improve the result. Label space  $16 \times 16 \times 16$ .

	$TV_{l_1}$	Huber- $TV_{l_1}$	$TV_{l_2}$	Huber- $TV_{l_2}$	$TV_J$	Huber- $TV_J$
Run time	1.04	1.07	1.42	1.58	126	329
Memory	131	131	181	181	8314	9710

**Table 9.4: Run times [s] and required GPU memory [MB] for Figure 9.6.** Image resolution  $341 \times 256$ , label space  $16 \times 16 \times 16$ .

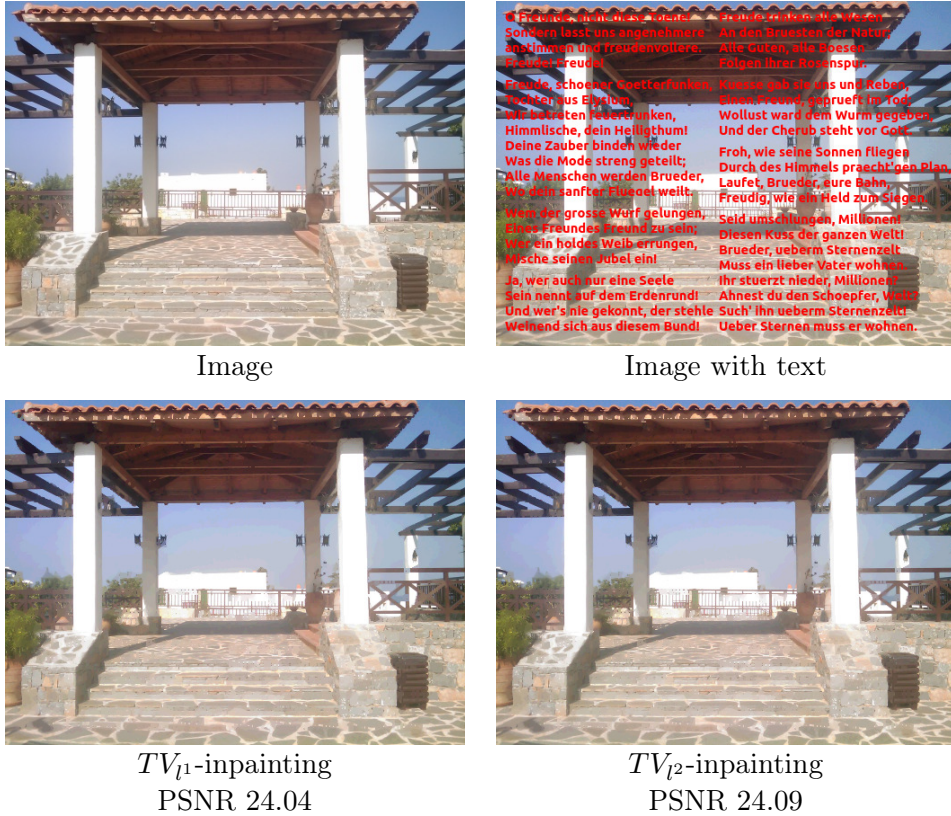
To compare the denoising capabilities of  $TV_{l_2}$  and  $TV_{l_1}$ , we perform an experiment where we add a certain amount of noise to several given clean images, and then try to recover these images by means of  $TV$  denoising. We compute the maximal achievable Peak-Signal-to-Noise-Ratios (PSNR), defined by

$$PSNR(u, f) = 10 \log_{10} \frac{3}{\sum_{x \in \Omega} |u(x) - f(x)|^2} \quad (9.109)$$

where we take the Euclidean distance of the RGB-values  $u(x), f(x) \in \mathbb{R}^3$ . Higher PSNR values indicate a better quality of the reconstruction  $u$ .

As seen in Figure 9.5 in case of Gaussian noise and a truncated quadratic data term  $c_2$  in (9.108),  $TV_{l_2}$  systematically leads to a better denoising quality, which is indicated by the higher PSNR values. We have chosen the weights  $\lambda$  to maximize these values in each case. The best denoising results are achieved using the *coupling*  $TV_{l_2}$  regularizer in combination with the *nonconvex* truncated data term with  $T = 0.3$ . Note that minimizing energies with nonconvex data terms *and* coupled regularizers only becomes possible with our proposed convex relaxation framework. The approach in the previous Chapter 8 allows nonconvex data terms but the regularizer must be separable, and [54] allows





**Figure 9.7: Inpainting.** A text has been placed on top of the input image (*top row*). The corresponding regions are marked as to be inpainted. The inpainted image values are computed by minimizing their  $TV_{l_1}$  or  $TV_{l_2}$  energy (*bottom row*). The coupled  $TV_{l_2}$  achieves a slightly higher PSNR value than  $TV_{l_1}$ , comparing with the known solution. Label space  $32 \times 32 \times 32$ .

coupling regularizers but the data term must be convex. The run times are independent of the data term and are listed in Table 9.3.

Figure 9.6 further compares the denoising capabilities of the different regularizers for a fixed data term. In the input image 20% of all pixel were set to random RGB values. For this case we use the robust truncated linear data term  $c_1$  in (9.108) with  $T = 0.2$ . The *coupling* regularizers  $TV_{l_2}$  and  $TV_J$  only become possible with our approach and lead to better denoising results than with the separable one  $TV_{l_1}$ . While  $TV_J$  yields the best results, it is also the most costly regularizer memory-wise. Switching to the Huber-regularized versions of the total variations has a positive effect on the denoising quality in each case. This is because it allows a smoother variation of the solution, which is then able to more closely resemble the natural image. The best result is achieved with Huber- $TV_J$  which however requires the general costly implementation scheme (9.97). The run times for this experiment, as well as the GPU memory requirements are given in Table 9.4.

### 9.6.3 Color Inpainting

Another useful application is inpainting. When a color image  $f : \Omega \rightarrow \mathbb{R}^3$  has some missing parts in an area  $A \subset \Omega$ , the task is to inpaint the missing colors in  $A$  using the available information from  $\Omega \setminus A$ . This can be formulated as an energy minimization problem

$$E(u) = \int_{\Omega} c(x, u(x)) \, dx + R(u). \quad (9.110)$$

The data term ensures that  $u = f$  in  $\Omega \setminus A$  and lets  $u$  vary freely in  $A$ . The resulting inpainted values in  $A$  will then be interpolated from the surrounding values in a way such that the regularizer value  $R(u)$  is minimal. We set

$$c(x, t) = \sum_{i=1}^k \begin{cases} \min \{T, (t_i - f_i(x))^2\} & \text{if } x \in \Omega \setminus A, \\ 0 & \text{if } x \in A. \end{cases} \quad (9.111)$$

We use  $R = TV_{l_1}$  and  $R = TV_{l_2}$  with a very small weight  $\lambda = 0.001$ , so that the given values in  $\Omega \setminus A$  remain unchanged.

Figure 9.7 shows the removal of a text on top of the image ( $T = 0.3$ ). Comparing the obtained inpainted values with the known true ones, we can compute the corresponding PSNR values. The coupled  $TV_{l_2}$  achieves a slightly higher value than the separable  $TV_{l_1}$ . For the  $512 \times 384$  image using  $32 \times 32 \times 32$  labels, the run time for  $TV_{l_1}$  is 16.75 seconds, while that for  $TV_{l_2}$  is 24.33 seconds.

## 9.7 Conclusion

We introduced convex relaxations for nonconvex variational models on vector-valued functions which are computationally tractable and in a certain sense as tight as possible. In contrast to existing relaxations of vectorial multilabel problems, we can handle the combination of nonconvex data terms with coupled regularizers such as  $TV_{l_2}$  and Huber- $TV_{l_2}$ . The key idea is to consider a collection of hyperplanes with a relaxation that takes into account the entire functional rather than separately treating data term and regularizers. We provided a theoretical analysis, detailed the implementations for different functionals, and presented run time and memory requirements. In particular, for the isotropic  $TV_{l_2}$  and Huber- $TV_{l_2}$  regularizers we proposed efficient equivalent constraint reformulations. This allows one to account for channel coupling with negligible overhead in memory and run time compared to the uncoupled versions of these regularizers. In numerous experiments on denoising, optical flow and inpainting, we showed that coupled  $l^2$ -regularizers give systematic improvements regarding rotational invariance and quantitative performance.

## 9.8 Appendix: Proof of the Main Lemma

We give here a proof of Lemma 9.3 in case  $u$  is a function with bounded variation.

*Proof of Lemma 9.3.* When  $u \in \text{BV}(\Omega; \Gamma^k)$ , its derivative  $Du$  has an absolutely continuous part with respect to the Lebesgue measure (traditionally denoted by  $\nabla u(x) dx$ ) and a singular part  $D^s u$  (consisting in a ‘‘Cantor’’ part and a ‘‘Jump’’ part, see [5]). Then, the integral in (9.25) is classically defined [42] as

$$\int_{\Omega} f(x, dDu) = \int_{\Omega} f(x, \nabla u(x)) dx + \int_{\Omega} f_{\infty}(x, \nu_u(x)) d|D^s u|(x)$$

where  $\nu_u(x) = \frac{D^s u}{|D^s u|}$  is the (unit) Radon-Besicovitch derivative of the measure  $D^s u$  with respect to its total variation, and  $f_{\infty}(x, p)$  is the convex, one-homogeneous recession function of  $f$  w.r.t. the last variable, see (2.4). See [42, 15] for details. Observe that since we have assumed  $f(x, 0) = \min_p f(x, p) = 0$ , we have  $f(x, sp)/s \leq f(x, tp)/t$  if  $s \leq t$ , so that

$$f_{\infty}(x, p) = \sup_{t>0} \frac{1}{t} f(x, tp)$$

(and  $f_{\infty}$  is l.s.c. as a supremum of lower-semicontinuous functions).

We will admit the following fact: the function  $u \mapsto \int_{\Omega} f(x, dDu)$  is lower-semicontinuous in  $L^1(\Omega, \mathbb{R}^k)$ , that is, if  $u_n \rightarrow u$  in  $L^1(\Omega, \mathbb{R}^k)$ ,

$$\int_{\Omega} f(x, dDu) \leq \liminf_{n \rightarrow \infty} \int_{\Omega} f(x, dDu_n),$$

see for instance [40, 16, 15]. Here the (semi)-continuity of  $f$  is important, whereas in the  $W^{1,1}$  case,  $f$  could be merely measurable in the first variable.

A first remark is that we may assume that there exists  $L > 0$  with

$$f(x, p) \leq L|p|. \tag{9.112}$$

Indeed, it is always possible to replace  $f(x, p)$  with the inf-convolution  $f_L(x, p) := \min_q f(x, q) + L|p - q|$  which satisfies (9.112) (since  $f(x, 0) = 0$ ), and such that the union for  $L > 0$  of the corresponding sets  $\mathcal{C}_0$  is the set  $\mathcal{C}_0$  of  $f$ . The conclusion easily follows.

Next, we also assume that  $f$  is uniformly continuous in  $x$  (for  $p$  bounded), but it is a bit more complicated to explain why. The idea is to define, for  $\lambda > 0$ ,

$$f_{\lambda}(x, p) = \min_{y \in \Omega} \lambda|y - x||p| + f(y, p)$$

which satisfies

$$|f_{\lambda}(x, p) - f_{\lambda}(y, p)| \leq \lambda|x - y||p|$$

and  $\sup_{\lambda>0} f_{\lambda}(x, p) = f(x, p)$ . However this new  $f$  is not convex in  $p$ , and one must show that its convex envelope  $f_{\lambda}^{**}(x, p)$  (with respect to the second variable) also enjoys these properties. Given any  $x, y \in \Omega$ ,  $p \in \mathbb{R}^{m \times k}$  and  $\varepsilon > 0$ , since  $\dim \mathbb{R}^{m \times k} = mk$  there exists  $(\theta_i, p_i)_{i=1}^{mk} \subset (\mathbb{R} \times \mathbb{R}^{m \times k})^{mk}$  with  $\sum_i \theta_i p_i = p$ ,  $\sum_i \theta_i = 1$ ,  $\theta_i \geq 0$  and such that  $f_{\lambda}^{**}(y, p) \geq \sum_i \theta_i f_{\lambda}(y, p_i) - \varepsilon$ .

Hence, using (9.24) and (9.112) (which are trivially still satisfied by  $f_\lambda$ )

$$\begin{aligned} f_\lambda^{**}(x, p) - f_\lambda^{**}(y, p) + \varepsilon &\leq \lambda|x - y| \sum_i \theta_i |p_i| \\ &\leq \lambda|x - y| \sum_i \theta_i \left(1 + \frac{f_\lambda(y, p_i)}{C}\right) \\ &\leq \lambda|x - y| \left(1 + \frac{f_\lambda^{**}(y, p) + \varepsilon}{C}\right) \\ &\leq \lambda|x - y| \left(1 + \frac{L}{C}|p| + \frac{\varepsilon}{C}\right) \end{aligned}$$

so that, letting  $\varepsilon \rightarrow 0$ , we see that  $f_\lambda^{**}$  satisfies

$$|f_\lambda^{**}(x, p) - f_\lambda^{**}(y, p)| \leq C'|x - y|(1 + |p|) \quad (9.113)$$

for some constant  $C'$ .

Then, we must check that the convex, l.s.c. functions  $\sup_\lambda f_\lambda^{**}(x, p)$  and  $f(x, p)$  are the same. If not (thanks to the separation theorem), there exists  $a \in \mathbb{R}^{m \times k}$ ,  $b' < b \in \mathbb{R}$  and a point  $(x, p)$  such that  $f_\lambda^{**}(x, p) \leq a : p + b'$  for all  $\lambda$ , while

$$a : q + b \leq f(x, q) \quad (9.114)$$

for all  $q \in \mathbb{R}^{m \times k}$ , where ‘ $:$ ’ denotes the scalar product of matrices. This means that one can find  $(\theta_i^\lambda, p_i^\lambda)_{i=0}^{mk} \subset (\mathbb{R} \times \mathbb{R}^{m \times k})^{mk}$  with  $\lim_{\lambda \rightarrow \infty} \sum_i \theta_i^\lambda p_i^\lambda = p$  and

$$\sum_i \theta_i^\lambda f_\lambda(x, p_i^\lambda) \leq a : p + b'' \quad (9.115)$$

where  $b'' = (b + b')/2$ . Up to a subsequence, one has  $\theta_i^\lambda \rightarrow \bar{\theta}_i$ , and either  $p_i^\lambda \rightarrow \bar{p}_i$  (if  $i \in I \subset \{0, \dots, mk\}$ ), or  $t_i^\lambda = |p_i^\lambda| \rightarrow \infty$  and  $\xi_i^\lambda = p_i^\lambda/t_i^\lambda \rightarrow \bar{\xi}_i$ , a unit vector (when  $i \notin I$ ). In the latter case, one introduces  $y_i^\lambda$  such that

$$f_\lambda(x, p_i^\lambda) = \lambda|x - y_i^\lambda| |p_i^\lambda| + f(y_i^\lambda, p_i^\lambda)$$

and one observes that given any  $t > 0$ , if  $\lambda$  is large enough so that  $t_i^\lambda > t$ ,

$$f_\lambda(x, p_i^\lambda) \geq f\left(y_i^\lambda, t \frac{t_i^\lambda}{t} \xi_i^\lambda\right) \geq \frac{t_i^\lambda}{t} f(y_i^\lambda, t \xi_i^\lambda).$$

Hence, denoting  $\varrho_i^\lambda = \theta_i^\lambda t_i^\lambda$  for  $i \notin I$ , by (9.115) one has

$$\sum_{i \in I} \theta_i^\lambda f_\lambda(x, p_i^\lambda) + \sum_{i \notin I} \varrho_i^\lambda \frac{1}{t} f(y_i^\lambda, t \xi_i^\lambda) \leq a : p + b''$$

and since  $\varrho_i^\lambda$  must therefore be bounded from above, we may also assume that it converges to some  $\bar{\varrho}_i$ , for each  $i \notin I$  (in particular, we must have  $\bar{\theta}_i = 0$  in these cases). Since  $f$  is lower-semicontinuous, in the limit we obtain

$$\sum_{i \in I} \bar{\theta}_i f(x, \bar{p}_i) + \sum_{i \notin I} \bar{\varrho}_i \frac{1}{t} f(x, t \bar{\xi}_i) \leq a : p + b''.$$

for any  $t > 0$ , and it follows from (9.114)

$$a : \left( \sum_{i \in I} \bar{\theta}_i \bar{p}_i + \sum_{i \notin I} \bar{\varrho}_i \bar{\xi}_i \right) + b \left( \sum_{i \in I} \bar{\theta}_i + \sum_{i \notin I} \frac{\bar{\varrho}_i}{t} \right) \leq a : p + b''.$$

Sending  $t \rightarrow \infty$  and observing that  $\sum_{i \in I} \bar{\theta}_i \bar{p}_i + \sum_{i \notin I} \bar{\varrho}_i \bar{\xi}_i = p$ , we obtain

$$a : p + b \sum_{i \in I} \bar{\theta}_i = a : p + b \leq a : p + b''$$

where we have used  $\theta_i = 0$  if  $i \notin I$ . This is a contradiction, since  $b'' < b$ .

It follows that  $\sup_\lambda f_\lambda^{**}(x, p) = f(x, p)$ , and we easily deduce that also  $\sup_\lambda (f_\lambda^{**})_\infty(x, p) = f_\infty(x, p)$ . Therefore, for any  $u$  with bounded variation,  $\sup_\lambda \int_\Omega f_\lambda^{**}(x, dDu) = \int_\Omega f(x, dDu)$ . Moreover the set  $\mathcal{C}_0(\lambda)$  of  $f_\lambda^{**}$  is clearly a subset of  $\mathcal{C}_0$ , so that if Lemma 9.3 holds for  $f_\lambda^{**}$ , it will also hold for  $f$ .

We observe eventually that it is not restrictive to assume that  $f$  is smooth in the  $p$  variable (a mollification of  $(f(x, p) - \varepsilon)^+$  will provide a smooth, convex function below  $f$ , enjoying the same properties as  $f$ , and arbitrarily close).

To sum up, we are reduced to the case where  $f$  is convex,  $L$ -Lipschitz (satisfying (9.112)) and smooth in  $p$ , and moreover with the spatial regularity (9.113).

Given  $u \in \text{BV}(\Omega; \Gamma^k)$ , we fix  $\delta > 0$  and choose a subset  $\Omega' \subset\subset \Omega$  such that  $\int_\Omega f(x, dDu) < \int_{\Omega'} f(x, dDu) + \delta$ . We let  $\varrho$  be a symmetric mollifier (convolution kernel) and  $u^\varepsilon = \varrho_\varepsilon * u$ , which is well defined in  $\Omega'$  if  $\varepsilon > 0$  is small enough. Moreover, by lower-semicontinuity,  $\int_{\Omega'} f(x, dDu) \leq \liminf_{\varepsilon \rightarrow 0} \int_{\Omega'} f(x, \nabla u^\varepsilon(x)) dx$  so that if  $\varepsilon$  is small enough,

$$\int_{\Omega'} f(x, \nabla u^\varepsilon(x)) dx > \int_{\Omega'} f(x, dDu) - \delta. \quad (9.116)$$

Since this new function  $u^\varepsilon$  is smooth, we can let for all  $x$  (we drop the dependence in  $(t_i)_{i=1}^k$ , i.e.  $\varphi$  is defined as the same value for every  $t$ )

$$(\varphi_i^x(x))_{i=1}^k = \nabla_p f(x, \nabla u^\varepsilon(x))$$

(which is continuous, since by using the convexity one can check that  $\nabla_p f$  is continuous) and for each  $i$ ,

$$\varphi_i^t(x) = \frac{1}{k} f^*(x, \nabla_p f(x, \nabla u^\varepsilon(x))).$$

Legendre-Fenchel's identity [108]

$$f(x, \nabla u^\varepsilon(x)) + f^*(x, \nabla_p f(x, \nabla u^\varepsilon(x))) = (\nabla u^\varepsilon(x)) : (\nabla_p f(x, \nabla u^\varepsilon(x))) \quad (9.117)$$

(denoting by  $':$  the scalar product of matrices) shows that also  $\varphi_i^t$  is continuous. By definition, using (9.117) again and (9.116),

$$\begin{aligned} \sum_{i=1}^k \int_{\Omega'} \varphi_i \cdot dD1_{u_i^\varepsilon} &= \sum_{i=1}^k \int_{\Omega'} \varphi_i^x(x) \cdot \nabla u_i^\varepsilon(x) - \varphi_i^t(x) dx \\ &= \int_{\Omega'} f(x, \nabla u^\varepsilon(x)) dx > \int_{\Omega'} f(x, dDu) - \delta. \end{aligned}$$

It is enough to observe, now, that (extending  $\varphi$  by the value 0 in  $\Omega \setminus \Omega'$ )

$$\begin{aligned} \int_{\Omega'} \varphi_i^x(x) \cdot \nabla u_i^\varepsilon(x) \, dx &= \int_{\Omega'} \varphi_i^x(x) \cdot \left( \int_{\Omega} \varrho_\varepsilon(x-y) \, dD u_i(y) \right) \, dx \\ &= \int_{\Omega} \left( \int_{\Omega'} \varphi_i^x(x) \varrho_\varepsilon(x-y) \, dx \right) \cdot dD u_i(y) \\ &= \int_{\Omega} (\varrho_\varepsilon * \varphi_i^x)(y) \cdot dD u_i(y) \end{aligned}$$

for each  $i$ , while, in the same way,  $\int_{\Omega'} \varphi_i^t(x) \, dx = \int_{\Omega} \varrho_\varepsilon * \varphi_i^t(y) \, dy$  for each  $i$ . We deduce that

$$\sum_{i=1}^k \int_{\Omega} (\varrho_\varepsilon * \varphi_i) \cdot dD 1_{u_i} \geq \int_{\Omega} f(x, dD u) - \delta$$

and Lemma 9.3 follows if we show that  $\varrho_\varepsilon * \varphi$  is in (or close to)  $\mathcal{C}_0$ . For any  $y \in \Omega$ ,  $p \in \mathbb{R}^{m \times k}$ , one has (including in  $\Omega \setminus \Omega'$  where it is 0) using (9.117) and then (9.113)

$$\begin{aligned} \sum_{i=1}^k \varphi_i^t(y) &\geq \sum_{i=1}^k \varphi_i^x(y) \cdot p_i - f(y, p) \\ &\geq \sum_{i=1}^k \varphi_i^x(y) \cdot p_i - f(x, p) - C'|x-y|(1+|p|) \end{aligned}$$

so that, thanks also to (9.24),

$$\begin{aligned} \sum_{i=1}^k (\varrho_\varepsilon * \varphi_i^t)(x) &\geq \sum_{i=1}^k (\varrho_\varepsilon * \varphi_i^x)(x) \cdot p_i - f(x, p) - C'\varepsilon(1+|p|) \\ &\geq \sum_{i=1}^k (\varrho_\varepsilon * \varphi_i^x)(x) \cdot p_i - \left(1 + \varepsilon \frac{C'}{C}\right) f(x, p) - 2\varepsilon C'. \end{aligned}$$

Hence, the field  $\tilde{\varphi}$ , defined by  $\tilde{\varphi}_i^x = (\varrho_\varepsilon * \varphi_i^x)/(1 + \varepsilon C'/C)$  and  $\tilde{\varphi}_i^t = (\varrho_\varepsilon * \varphi_i^t + 2\varepsilon C'/k)/(1 + \varepsilon C'/C)$  for each  $i$ , is an element of  $\mathcal{C}_0$ , which is such that

$$\begin{aligned} \sup_{\psi \in \mathcal{C}_0} \sum_{i=1}^k \int_{\Omega} \psi_i \cdot dD 1_{u_i} &\geq \sum_{i=1}^k \int_{\Omega} \tilde{\varphi}_i \cdot dD 1_{u_i} \\ &\geq \left( \int_{\Omega} f(x, dD u) - \delta - 2\varepsilon C'|\Omega'| \right) \left(1 + \varepsilon \frac{C'}{C}\right)^{-1}. \end{aligned}$$

We can then send  $\varepsilon$ , and then  $\delta$  to zero to get the thesis.  $\square$

## 9.9 Appendix: Projections

### 9.9.1 Projection onto Cones $y \geq \alpha|x|$

Let  $\alpha \geq 0$ . For  $x_0 \in \mathbb{R}^m$  and  $y_0 \in \mathbb{R}$  consider the projection

$$\operatorname{argmin}_{\substack{x \in \mathbb{R}^m, y \in \mathbb{R}, \\ y \geq \alpha|x|}} \frac{(x-x_0)^2}{2} + \frac{(y-y_0)^2}{2}. \quad (9.118)$$

If already  $y_0 \geq \alpha |x_0|$ , the solution is  $(x, y) = (x_0, y_0)$ . Otherwise set

$$v := \max \left( 0, \frac{|x_0| + \alpha y_0}{1 + \alpha^2} \right). \quad (9.119)$$

The solution is then given by

$$x = \begin{cases} v \frac{x_0}{|x_0|} & \text{if } x_0 \neq 0, \\ 0 & \text{else} \end{cases}, \quad y = \alpha |x|. \quad (9.120)$$

*Proof.* First, for  $y_0 \geq \alpha |x_0|$  the projection is obviously  $(x, y) = (x_0, y_0)$ . Otherwise, we set  $x = t\omega$ ,  $t \geq 0$ ,  $\omega \in \mathbb{R}^m$  with  $|\omega| = 1$ . For fixed  $t$  the expression  $(x - x_0)^2 = (t\omega - x_0)^2$  is minimized for  $\omega = \frac{x_0}{|x_0|}$  if  $x_0 \neq 0$  and with arbitrary  $\omega$  else. Since  $(t\omega - x_0)^2 = (t - |x_0|)^2$ , the solution  $(t, y)$  is given by the projection  $(t, y) = \pi_{y \geq \alpha t}(|x_0|, y_0)$ . This projection can be easily computed by projecting onto the line  $y = \alpha t$ , respectively on the point  $(0, 0)$ , depending on whether  $(|x_0|, y_0)$  is above or below the corresponding orthogonal line through the origin.  $\square$

### 9.9.2 Projection onto Parabolas $y \geq \alpha |x|^2$

Let  $\alpha > 0$ . For  $x_0 \in \mathbb{R}^m$  and  $y_0 \in \mathbb{R}$  consider the projection onto a parabola:

$$\underset{\substack{x \in \mathbb{R}^m, y \in \mathbb{R}, \\ y \geq \alpha |x|^2}}{\operatorname{argmin}} \frac{(x - x_0)^2}{2} + \frac{(y - y_0)^2}{2}. \quad (9.121)$$

If already  $y_0 \geq \alpha |x_0|^2$ , the solution is  $(x, y) = (x_0, y_0)$ . Otherwise, with  $a := 2\alpha |x_0|$ ,  $b := \frac{2}{3}(1 - 2\alpha y_0)$  and  $d := a^2 + b^3$  set

$$v := \begin{cases} c - \frac{b}{c} & \text{with } c = \sqrt[3]{a + \sqrt{d}} & \text{if } d \geq 0, \\ 2\sqrt{-b} \cos \left( \frac{1}{3} \arccos \frac{a}{\sqrt{-b^3}} \right) & \text{if } d < 0. \end{cases} \quad (9.122)$$

If  $c = 0$  in the first case, set  $v := 0$ . The solution is then given by

$$x = \begin{cases} \frac{v}{2\alpha} \frac{x_0}{|x_0|} & \text{if } x_0 \neq 0, \\ 0 & \text{else} \end{cases}, \quad y = \alpha |x|^2. \quad (9.123)$$

**Remark.** In the case  $d < 0$  it always holds  $\frac{a}{\sqrt{-b^3}} \in [0, 1]$ . To ensure this also numerically, one should compute  $d$  by  $d = (a - \sqrt{-b^3})(a + \sqrt{-b^3})$  for  $b < 0$ .

*Proof.* First, for  $y_0 \geq \alpha |x_0|^2$  the projection is obviously  $(x, y) = (x_0, y_0)$ . Otherwise, we dualize the parabola constraint using  $\delta_{z \geq 0} = \sup_{\lambda \geq 0} -\lambda z$  (for  $z \in \mathbb{R}$ ):

$$\min_{x \in \mathbb{R}^m, y \in \mathbb{R}} \max_{\lambda \geq 0} \frac{(x - x_0)^2}{2} + \frac{(y - y_0)^2}{2} - \lambda(y - \alpha |x|^2). \quad (9.124)$$

Since this expression is convex in  $x, y$  and concave in  $\lambda$ , we can interchange the ordering of min and max. The inner minimization problem in  $x$  and  $y$  is

then easily solved, giving the following necessary representation, with a certain  $\lambda \geq 0$ :

$$x = \frac{x_0}{1 + 2\alpha\lambda}, \quad y = y_0 + \lambda. \quad (9.125)$$

For instance,  $x$  has the same direction as  $x_0$ , so only the norm of  $x$  is unknown. The solution must also necessarily satisfy  $y = \alpha|x|^2$ . Plugging this into the second equation of (9.125), as well as the expression for  $\lambda$  obtained from the first equation by taking the norms, we obtain the cubic equation

$$2\alpha^2|x|^3 + (1 - 2\alpha y_0)|x| - |x_0| = 0. \quad (9.126)$$

Set  $a := 2\alpha|x_0|$ ,  $b := \frac{2}{3}(1 - 2\alpha y_0)$  and  $t := 2\alpha|x|$ . Then (9.126) becomes

$$t^3 + 3bt - 2a = 0. \quad (9.127)$$

Since the derivative  $3t^2 + 3b$  of the left hand side is monotonically increasing for  $t \geq 0$ , the  $t$  we are looking for is the unique nonnegative solution of (9.127) for  $x_0 \neq 0$  (so that  $a > 0$ ). This cubic equation can be solved using the method of elementary hyperbolic/trigonometric function identities [83], yielding the claimed solution. The second case in (9.122) corresponds to “ $x_2$ ” in equation (23) of [83].

For  $x_0 = 0$ , because of the assumed inequality  $y_0 < \alpha|x_0|^2 = 0$  we have the first case in (9.122), which leads to the correct solution  $(x, y) = (0, 0)$ .  $\square$

### 9.9.3 Projection for Huber- $TV_J$

Let  $\alpha > 0$  and  $\lambda > 0$ . For  $x_0 \in \mathbb{R}^{m \times k}$  and  $y_0 \in \mathbb{R}$  consider the projection:

$$\underset{\substack{x \in \mathbb{R}^{m \times k}, y \in \mathbb{R}, \\ y \geq \alpha\|x\|_*^2, \|x\|_* \leq \lambda}}{\operatorname{argmin}} \frac{(x - x_0)^2}{2} + \frac{(y - y_0)^2}{2} \quad (9.128)$$

with the nuclear norm  $\|\cdot\|_*$  in (9.68). Here we will only consider the case of  $2D$  color images, i.e.  $m = 2$  and  $k = 3$ . Let  $x_0 = U_0 \Sigma_0 V_0^{-1}$  be the singular value decomposition of  $x_0$ , with some  $U_0 \in \text{SO}(2)$  and  $V_0 \in \text{SO}(3)$ , and the matrix  $\Sigma_0 \in \mathbb{R}^{2 \times 3}$ ,  $\Sigma_0 = \text{diag}(\sigma_1^0, \sigma_2^0)$ , containing the singular values  $\sigma_1^0 \geq \sigma_2^0 \geq 0$ . Using the definition (9.68), the projection (9.128) can be reformulated in terms of the singular values  $\sigma_1, \sigma_2$  of  $x$ :

$$\underset{\substack{x \in \mathbb{R}^{m \times k}, y \in \mathbb{R}, \\ y \geq \alpha(\sigma_1 + \sigma_2)^2, \sigma_1 + \sigma_2 \leq \lambda}}{\operatorname{argmin}} \sum_{i=1}^2 \frac{(\sigma_i - \sigma_i^0)^2}{2} + \frac{(y - y_0)^2}{2}. \quad (9.129)$$

Having a  $(\sigma_1, \sigma_2, y)$  which solves (9.129), the solution  $(x, y)$  of (9.128) can be obtained by  $x = U_0 \text{diag}(\sigma_1, \sigma_2) V_0^{-1}$ . Finally, the solution  $(\sigma_1, \sigma_2, y)$  of (9.129) is  $(\sigma_1^0, \sigma_2^0, y^0)$  if this point already satisfies the constraints, and otherwise with



$\delta := \sigma_1^0 - \sigma_2^0$  given by

$$\left\{ \begin{array}{ll} (\lambda, 0, \max(\alpha\lambda^2, y_0)) & \text{if } \delta \geq \lambda \text{ and } y_0 \geq \alpha\lambda^2 - \frac{\sigma_1^0 - \lambda}{2\alpha\lambda}, \\ (\frac{\lambda+\delta}{2}, \frac{\lambda-\delta}{2}, \max(\alpha\lambda^2, y_0)) & \text{if } \delta \leq \lambda \text{ and } y_0 \geq \alpha\lambda^2 - \frac{\sigma_1^0 + \sigma_2^0 - \lambda}{4\alpha\lambda}, \\ \left( \frac{\hat{x}}{\sqrt{2}} + \frac{\delta}{2}, \frac{\hat{x}}{\sqrt{2}} - \frac{\delta}{2}, \hat{y} \right) \\ \quad \text{with } (\hat{x}, \hat{y}) := \pi_{y \geq 2\alpha x^2} \left( \frac{\sigma_1^0 + \sigma_2^0}{\sqrt{2}}, y_0 \right) & \text{if } \delta \leq \lambda \text{ and } y_0 \geq \alpha\delta^2 - \frac{\sigma_2^0}{2\alpha\delta}, \\ (\hat{x}, 0, \hat{y}) \\ \quad \text{with } (\hat{x}, \hat{y}) := \pi_{y \geq \alpha x^2}(\sigma_1^0, y_0) & \text{otherwise.} \end{array} \right. \quad (9.130)$$

*Proof.* First, let us establish the equivalence of (9.128) and (9.129). For candidate solutions  $(x, y)$  of (9.128) let  $x = U\Sigma V^{-1}$  with  $U \in \text{SO}(2)$ ,  $V \in \text{SO}(3)$ ,  $\Sigma \in \mathbb{R}^{2 \times 3}$ ,  $\Sigma = \text{diag}(\sigma_1, \sigma_2)$  be the singular value decomposition of  $x$ . By Mirsky's inequality [85] we have  $(x - x_0)^2 \geq \sum_{i=1}^2 (\sigma_i - \sigma_i^0)^2$  with equality if  $U = U_0$ ,  $V = V_0$ . Thus, an optimal  $x$  for (9.128) will have the form  $U_0 \Sigma V_0^{-1}$  where the singular values  $\sigma_1, \sigma_2$  satisfy the constraints of (9.129).

The derivation of (9.130) is rather lengthy, but straightforward. The third case corresponds to the projection onto the paraboloid segment  $y = \alpha(\sigma_1 + \sigma_2)^2$ ,  $\sigma_1, \sigma_2 \geq 0$ ,  $\sigma_1 + \sigma_2 \leq \lambda$ , and is active when there is an outer surface normal passing through the point  $(\sigma_1^0, \sigma_2^0, y^0)$ . The fourth case is the projection onto the parabola line  $y = \alpha(\sigma_1 + \sigma_2)^2 = \alpha\sigma_1^2$ ,  $0 \leq \sigma_1 \leq \lambda$ ,  $\sigma_2 = 0$ , active when there is an orthogonal line on it passing through the given point. The second case projects onto the plane segment  $y \geq \alpha(\sigma_1 + \sigma_2)^2 = \alpha\lambda^2$ ,  $\sigma_1, \sigma_2 \geq 0$ ,  $\sigma_1 + \sigma_2 = \lambda$ , and the first one onto the line  $y \geq \alpha(\sigma_1 + \sigma_2)^2 = \alpha\lambda^2$ ,  $\sigma_1 = \lambda$ ,  $\sigma_2 = 0$ .  $\square$



## Chapter 10

# The Vectorial Mumford-Shah Functional

While the last two Chapters 8 and 9 addressed general classes of vectorial functionals, here we will consider the concrete and very special vectorial functional of Mumford-Shah, which has a long history of proposed approaches and theoretical analysis due to the wide range of its potential applications. This chapter is based on joint work with Antonin Chambolle and Daniel Cremers [119].

### 10.1 Introduction

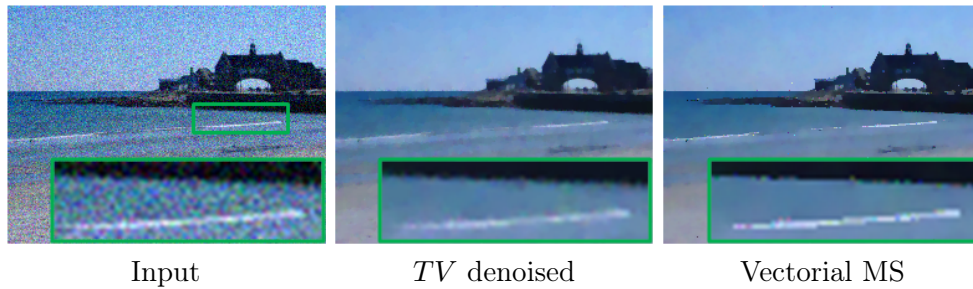
#### 10.1.1 The Mumford-Shah Problem

Regularization is of central importance in image analysis and beyond as it provides a prior for a number of otherwise ill-posed inverse problems. The Mumford-Shah functional [92] is a prototypical form of all regularizers which aim at combining a smoothing of homogeneous regions with the enhancement of edges: Given  $\Omega \subset \mathbb{R}^m$ ,  $m \geq 1$ , a bounded open set, the vectorial Mumford-Shah problem with  $k \geq 1$  channels is given by

$$\min_{u,K} \left\{ \int_{\Omega} |u - f|^2 \, dx + \alpha \int_{\Omega \setminus K} |\nabla u|^2 \, dx + \lambda \mathcal{H}^{m-1}(K) \right\} \quad (10.1)$$

where  $f : \Omega \rightarrow \mathbb{R}^k$  is the vectorial input image,  $\alpha, \lambda > 0$  are weights, and  $u = (u_1, \dots, u_k) : \Omega \rightarrow \mathbb{R}^k$  is the unknown which is assumed to be smooth except on a possible jump set  $K$ . This set is the same for all components  $u_i$  and introduces a coupling of the channels. Minimization of the energy (10.1) leads to *piecewise smooth* approximations  $u$  of  $f$ . Intuitively, the first term ensures that  $u$  is similar to  $f$ . The last term ensures that there are not “too many” jumps by penalizing the overall length of the jump set. Finally, the second term makes sure that  $u$  is smooth between the jumps by penalizing the gradient. The norm of the gradient  $|\nabla u|$  which appears in (10.1) is the Frobenius (Euclidean) norm  $|\nabla u|^2 = \sum_i |\nabla u_i|^2$ , and the norm in the term  $|u - f|$  is also Euclidean.

The Mumford-Shah problem has been intensively studied in the applied math community [89]. In practice its applicability is substantially limited because of its nonconvexity. As a consequence, researchers typically revert to the



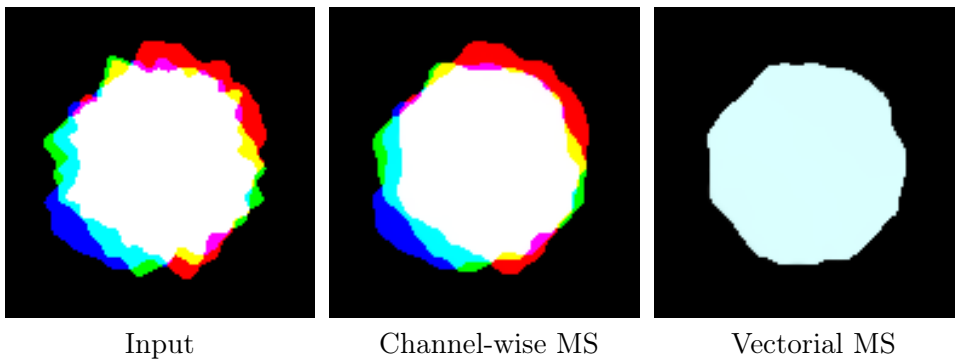
**Figure 10.1: Vectorial Mumford-Shah in comparison to total variation.** We propose a convex relaxation and an efficient implementation for the vectorial Mumford-Shah functional. It allows to compute piecewise smooth approximations of color images independently of initialization. Compared to the Total variation (*center*), the Mumford-Shah model (*right*) preserves image contrast.

(also nonconvex) phase-field approximation of Ambrosio and Tortorelli [2, 3]. Extensions of this approximation to color images have been proposed in [22]. Alternatively it is often replaced with the convex total variation  $TV$ . However, the tendency of  $TV$  to lower the contrast at edges and oversmooth flat regions (the “staircasing” phenomenon) makes it a poor substitute to more elaborate functionals — see Figure 10.1. In [65] a different approach for color channels coupling is proposed by means of Riemannian Geometry.

### 10.1.2 Related Work

In the recent past, several authors have overcome the issue of nonconvexity by suggesting convex relaxations for respective functionals [1, 51, 28]. Specific examples include the convex relaxation for the two-region piecewise-constant Mumford-Shah model [33], for multilabel problems with convex regularizer [102, 101], for the multi-region piecewise constant Mumford-Shah [77, 29, 140], and — possibly most closely related to the approach in this chapter — a convex relaxation for the *scalar* piecewise smooth Mumford-Shah model [100]. These developments, based on multilabel and functional lifting relaxations, are described in Chapters 3 and 7, respectively. Some of these approaches were clearly inspired from the Markov random field (MRF) community where researchers have introduced graph cut algorithms for minimizing discrete version of such energies [56, 61, 19, 60]. In the MRF community, the Mumford-Shah regularizer corresponds to a truncated quadratic penalizer.

While the above works are predominantly focused on functionals over *scalar-valued* functions (grayscale images), the present chapter is focused on the case of multi-component signals, i.e. color or multi-spectral images. The major challenge in formulating convex relaxations for vectorial models is that the straightforward channel-by-channel application of respective scalar approaches [100] invariably produces suboptimal results because the individual color channels cannot be treated independently: Indeed, the jump set  $K$  in (10.1) combines all color channels, as it denotes all points in the image plane where the signal



**Figure 10.2: Vectorial versus channel-wise Mumford-Shah.** The vectorial MS model penalizes jumps only once if two or more components jump at the same place. As a consequence, the correct representation (*right*) favors component edges to coincide leading to coherent segmentations. This is not the case for the channel-wise MS model (*center*) even for large weights  $\lambda$ .

is discontinuous, no matter in which color channel.

A convex relaxation for the *vectorial* case has been given in [88] using the method of calibrations. However, it utilizes the full three-dimensional label space for color images and is therefore by no means tractable, neither in memory, nor in computation time. Moreover, it appears infeasible to actually devise a numerical algorithm for solving the relaxation of [88] because it depends only on a part of the solution.

### 10.1.3 Contributions

In this chapter, we propose a convex representation for the vectorial Mumford-Shah functional (10.1) which captures the coupling among the different color channels correctly. Among other conceivable generalizations of the scalar case [100], the proposed relaxation is special, as it combines several important advantages:

- It is the first tractable convex relaxation for the vectorial Mumford-Shah functional. We propose an efficient implementation of the method for which both memory and run time scale linearly with the number of channels. In particular, channel coupling is achieved with the same run time as for the channel-wise version.
- The proposed method indeed favors solutions where discontinuities in the individual color channels coincide. As a consequence, in contrast to the channel-wise approach it does not introduce artificial color edges in the solution.
- In comparison to the channel-by-channel solution, the commonly employed Ambrosio-Tortorelli approximation [2, 3] and  $TV$ , the method leads to improved and more natural results for discontinuity-preserving denoising of color images and various other applications.

## 10.2 The Convex Representation

Recall that for vectorial functions  $u \in \text{SBV}(\Omega, \mathbb{R}^k)$  the distributional gradient can be decomposed as (2.23) into a continuous part and a jump part. This means there exists a well-defined  $(m-1)$ -dimensional jump set  $S_u$  on which  $u$  jumps in a direction  $\nu_u \in \mathbb{S}^{m-1}$  from  $u^- \in \mathbb{R}^k$  to  $u^+ \in \mathbb{R}^k$ .

For such functions in  $\text{SBV}(\Omega, \mathbb{R}^k)$ , the “weak” Mumford-Shah functional is well-defined and given by

$$MS^{\alpha, \lambda}(u) = \int_{\Omega} |u - f|^2 \, dx + \alpha \int_{\Omega \setminus S_u} |\nabla u|^2 \, dx + \lambda \mathcal{H}^{m-1}(S_u). \quad (10.2)$$

It has been shown [4] that the minimization of  $MS^{\alpha, \lambda}(u)$  in  $\text{SBV}(\Omega, \mathbb{R}^k)$  is a well-posed problem.

### 10.2.1 Relaxation for the Scalar Case

Let us recall here the convexification approach for the scalar case  $k = 1$ . Given a scalar function  $u : \Omega \rightarrow \mathbb{R}$ , the idea is to consider its corresponding graph function  $1_u : \Omega \times \mathbb{R} \rightarrow \{0, 1\}$ . It is defined as

$$1_u(x, t) = \begin{cases} 1 & \text{if } u(x) > t, \\ 0 & \text{else.} \end{cases} \quad (10.3)$$

The general functional lifting approach in Chapter 7 suggests to introduce the functional

$$\mathcal{E}(v) := \sup_{\varphi \in \mathcal{K}} \int_{\Omega \times \mathbb{R}} \varphi(x, t) \cdot dDv(x, t), \quad (10.4)$$

defined for  $v \in \text{BV}_{\text{loc}}(\Omega \times \mathbb{R})$ , where the constraint set is given by

$$\mathcal{C} := \left\{ \varphi \mid \begin{aligned} &(\varphi^x, \varphi^t) \in C_c^\infty(\Omega \times \mathbb{R}; \mathbb{R}^m \times \mathbb{R}) \\ &\varphi^t(x, t) \geq \frac{1}{4\alpha} |\varphi^x(x, t)|^2 - (t - f(x))^2, \\ &\left| \int_t^{t'} \varphi^x(x, s) \, ds \right| \leq \lambda \quad \forall x \in \Omega, t < t' \end{aligned} \right\}. \quad (10.5)$$

It is shown in [1] that then for any  $u \in \text{SBV}(\Omega, \mathbb{R})$ ,

$$\mathcal{E}(1_u) = MS^{\alpha, \lambda}(u). \quad (10.6)$$

The key advantage of this representation is that  $\mathcal{E}(v)$  is convex in  $v$ . This can be used to numerically minimize  $MS^{\alpha, \lambda}$  [100] with interesting practical applications.

### 10.2.2 Contribution: Relaxation for the Vectorial Case

In the vectorial case  $k \geq 1$ , we consider  $k$  graph functions

$$1_u(x, t) = (1_{u_1}, \dots, 1_{u_k}) \quad (10.7)$$

corresponding to the  $k$  channels  $u_i$ . We propose the following convex relaxation (in  $v = 1_u$ ) of the Mumford-Shah functional (10.2):

$$\mathcal{E}(v) := \sup_{\varphi \in \mathcal{K}} \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i(x, t) \cdot dDv_i(x, t), \quad (10.8)$$

with the convex set

$$\begin{aligned} \mathcal{C} := \left\{ \varphi \mid (\varphi_i^x, \varphi_i^t) \in C_c^\infty(\Omega \times \mathbb{R}; \mathbb{R}^m \times \mathbb{R}), \right. \\ \left. \varphi_i^t(x, t_i) \geq \frac{1}{4\alpha} |\varphi_i^x(x, t_i)|^2 - (t_i - f_i(x))^2, \right. \\ \left. \sum_{j=1}^k \left| \int_{t_j}^{t'_j} \varphi_j^x(x, s) ds \right| \leq \lambda \quad \forall x \in \Omega, 1 \leq i \leq k, t_j < t'_j \right\}. \quad (10.9) \end{aligned}$$

The dual variables  $\varphi^x, \varphi^t$  now also have  $k$  components, while the first constraint in (10.9) is the same as in (10.5). The central part of the generalization is the second constraint of (10.9). Intuitively, the upper bound  $\lambda$  on the dual variables corresponds to the local penalization if each  $u_j$  jumps from  $t_j$  to  $t'_j$ . Let us explain how the set  $\mathcal{C}$  in (10.9) is derived. First, if  $u_i \in \text{SBV}(\Omega, \mathbb{R})$ , one can check (following the representation in [1, Lemma 2.8]) that

$$\begin{aligned} \int_{\Omega \times \mathbb{R}} \varphi_i \cdot dD1_{u_i} &= \int_{\Omega} \varphi_i^x(x, u_i(x)) \cdot \nabla u_i(x) - \varphi_i^t(x, u_i(x)) dx \\ &+ \int_{S_{u_i}} \left( \int_{u_i^-(x)}^{u_i^+(x)} \varphi_i^x(x, s) ds \right) \cdot \nu_{u_i}(x) d\mathcal{H}^{m-1}(x). \end{aligned} \quad (10.10)$$

If  $\varphi_i$  satisfies the first constraint in (10.9), standard convex duality shows that

$$\begin{aligned} \varphi_i^x(x, u_i(x)) \cdot \nabla u_i(x) - \varphi_i^t(x, u_i(x)) \\ \leq (u_i(x) - f_i(x))^2 + \alpha |\nabla u_i(x)|^2 \end{aligned} \quad (10.11)$$

a.e. in  $\Omega$ , with equality at  $x$  if and only if  $\varphi_i^t = |\varphi_i^x|^2 / (4\alpha) - (t - f_i)^2$  and  $t = u_i(x)$ . The right hand side of (10.11) summed up over all  $i$  are the first two integrands in (10.2). On the other hand, if  $\varphi$  satisfies the second constraint in (10.9), then one has a similar inequality for the jump part. Indeed, recalling that  $S_u = \bigcup_{i=1}^k S_{u_i}$ , and  $\nu_{u_i} = \nu_u$  a.e. in  $S_u \cap S_{u_i}$ , one can check that

$$\begin{aligned} \sum_{i=1}^k \int_{S_{u_i}} \left( \int_{u_i^-(x)}^{u_i^+(x)} \varphi_i^x(x, s) ds \right) \cdot \nu_u(x) d\mathcal{H}^{m-1}(x) \\ \leq \int_{S_u} \sum_{i=1}^k \left| \int_{u_i^-(x)}^{u_i^+(x)} \varphi_i^x(x, s) ds \right| d\mathcal{H}^{m-1}(x) \\ \leq \lambda \mathcal{H}^{m-1}(S_u). \end{aligned} \quad (10.12)$$

This derivation shows that at least

$$\mathcal{E}(1_u) \leq MS^{\alpha, \lambda}(u). \quad (10.13)$$

Beyond this inequality we will in fact show in Theorem 10.2 below that the proposed relaxation (10.8) indeed coincides with the original Mumford-Shah model (10.2). The proof will utilize a crucial efficient reformulation of the constraint set (10.9) which we introduce next.

### 10.2.3 Efficient Constraint Set Reformulation

In practice, the range of each channel  $u_i$  must be discretized into  $n_i \geq 1$  levels. In its original formulation (10.9), the constraint set  $K$  requires  $\mathcal{O}(n_1^2 \cdots n_k^2)$  constraints, which is not feasible in practice. However, it turns out that these can be equivalently reformulated using only  $\mathcal{O}(n_1^2 + \dots + n_k^2)$  constraints. Thus, the proposed relaxation (10.8) has the key advantage that its minimization is just *as tractable* as the simple channel-by-channel model. In Section 10.3 we will also propose an approximation which has even *linear* instead of quadratic complexity.

The idea of the constraint set decoupling is to introduce auxiliary variables  $\mu_i : \Omega \rightarrow \mathbb{R}$  for each  $1 \leq i \leq k$ .

**Proposition 10.1.** *The constraint set  $\mathcal{C}$  in (10.9) is equivalent to the following constraint set:*

$$\mathcal{C}' := \left\{ (\varphi, \mu) \mid \begin{aligned} &(\varphi_i^x, \varphi_i^t) \in C_c^\infty(\Omega \times \mathbb{R}; \mathbb{R}^m \times \mathbb{R}), \\ &\varphi_i^t(x, t_i) \geq \frac{1}{4\alpha} |\varphi_i^x(x, t_i)|^2 - (t_i - f_i(x))^2, \\ &\left| \int_{t_i}^{t'_i} \varphi_i^x(x, s) \, ds \right| \leq \mu_i(x), \\ &\sum_{j=1}^k \mu_j(x) \leq \lambda \quad \forall i, x \in \Omega, t_i < t'_i \end{aligned} \right\}, \quad (10.14)$$

in the sense that, for all  $\varphi$ , it holds

$$\varphi \in \mathcal{C} \iff \exists \mu : (\varphi, \mu) \in \mathcal{C}'. \quad (10.15)$$

*Proof.* Let  $(\varphi, \mu) \in \mathcal{C}'$ . Then obviously also  $\varphi \in \mathcal{C}$ . On the other hand, if  $\varphi \in \mathcal{C}$ , define the functions  $\mu_i : \Omega \rightarrow \mathbb{R}$  by

$$\mu_i(x) := \sup_{t_i < t'_i} \left| \int_{t_i}^{t'_i} \varphi_i^x(x, s) \, ds \right|. \quad (10.16)$$

Then we have  $\sum_{j=1}^k \mu_j(x) \leq \lambda$  by the second inequality of (10.9), and together clearly  $(\varphi, \mu) \in \mathcal{C}'$ .  $\square$

The following theorem shows that the relaxation coincides with the actual functional for binary graph functions:

**Theorem 10.2.** *Let  $u \in \text{SBV}(\Omega, \mathbb{R}^k)$ . Then*

$$\mathcal{E}(1_u) = MS^{\alpha, \lambda}(u). \quad (10.17)$$



*Proof.* The inequality (10.13) has already been shown above, so it remains to show

$$\mathcal{E}(1_u) \geq MS^{\alpha,\lambda}(u). \quad (10.18)$$

Consider functions  $\mu$  from the set

$$\mathcal{M} := \left\{ \mu \in C(\Omega; \mathbb{R}^k) \mid \sum_{i=1}^k \mu_i(x) \leq \lambda \quad \forall x \in \Omega, \quad \min_{x \in \Omega} \mu_i(x) > 0 \quad \forall i \right\}. \quad (10.19)$$

For each  $i$ , consider the non-uniform variant of the scalar Mumford-Shah functional:

$$MS_i^{\alpha,\mu_i}(u_i) := \int_{\Omega} |u_i - f_i|^2 dx + \alpha \int_{\Omega \setminus S_{u_i}} |\nabla u_i|^2 dx + \int_{S_{u_i}} \mu_i(x) d\mathcal{H}^{m-1}(x). \quad (10.20)$$

Then, it can be quite easily shown that the vectorial MS functional (10.2) is given by

$$MS^{\alpha,\lambda}(u) = \sup_{\mu \in \mathcal{M}} \sum_{i=1}^k MS_i^{\alpha,\mu_i}(u_i). \quad (10.21)$$

Moreover, for fixed  $\mu_i$ , an adaption of the proofs in [1, 100] will show that a representation similar to (10.6) holds, specifically

$$MS_i^{\alpha,\mu_i}(u_i) = \sup_{\varphi_i \in \mathcal{C}_i^{\mu_i}} \int_{\Omega \times \mathbb{R}} \varphi_i(x, t) \cdot dD1_{u_i}(x, t) \quad (10.22)$$

for  $u_i \in \text{SBV}(\Omega, \mathbb{R})$ , where  $\mathcal{C}_i^{\mu_i}$  is defined as in the scalar case (10.5) but with  $\lambda$  replaced by  $\mu_i(x)$ :

$$\begin{aligned} \mathcal{C}_i^{\mu_i} := \left\{ \varphi \mid (\varphi^x, \varphi^t) \in C_c^\infty(\Omega \times \mathbb{R}; \mathbb{R}^m \times \mathbb{R}) \right. \\ \left. \varphi^t(x, t) \geq \frac{1}{4\alpha} |\varphi^x(x, t)|^2 - (t - f(x))^2, \right. \\ \left. \left| \int_t^{t'} \varphi^x(x, s) ds \right| \leq \mu_i(x) \quad \forall x \in \Omega, t < t' \right\}. \end{aligned} \quad (10.23)$$

Plugging (10.22) into (10.21):

$$MS^{\alpha,\lambda}(u) = \sup_{\mu \in \mathcal{M}} \sup_{\substack{\varphi \\ \varphi_i \in \mathcal{C}_i^{\mu_i} \forall i}} \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i(x, t) \cdot dD1_{u_i}(x, t) \quad (10.24)$$

$$\leq \sup_{(\varphi, \mu) \in \mathcal{C}'} \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i(x, t) \cdot dD1_{u_i}(x, t) \quad (10.25)$$

$$= \sup_{\varphi \in \mathcal{C}} \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i(x, t) \cdot dD1_{u_i}(x, t). \quad (10.26)$$

Above, the inequality holds because  $(\varphi, \mu)$  is more general in (10.14) (less assumptions on  $\mu$ ), and the last equality is due to the constraint set equivalence (10.1). The right hand side of (10.26) is just the definition (10.8) of  $\mathcal{E}(1_u)$ , so that (10.18) follows.  $\square$

In particular, this theorem assures that in case of a binary minimizer  $v = 1_u$  of  $\mathcal{E}(v)$  the solution  $u$  must indeed be a global optimum of the vectorial Mumford-Shah model (10.2).

### 10.2.4 Generalizations and Variants

One can generalize the Mumford-Shah model (10.2), replacing the data term  $|u - f|^2$  and the regularization term  $\alpha |\nabla u|^2$  with separable terms of the form  $\sum_{i=1}^k c_i(x, u_i(x))$  and  $\sum_{i=1}^k g_i(\nabla u_i)$  with convex  $f_i$ , respectively:

$$\int_{\Omega} \sum_{i=1}^k c_i(x, u_i(x)) \, dx + \int_{\Omega \setminus S_u} \sum_{i=1}^k g_i(\nabla u_i) \, dx + \lambda \mathcal{H}^{m-1}(S_u). \quad (10.27)$$

In our relaxation this amounts to replacing the first constraint in (10.14) with

$$\varphi_i^t(x, t) \geq g_i^*(\varphi_i^x(x, t)) - c_i(x, t) \quad \forall i, x, t. \quad (10.28)$$

Here,  $g_i^*(q) := \sup_p p \cdot q - g_i(p)$  is the Legendre-Fenchel conjugate of  $g_i$ . Then, provided  $c_i$  is continuous, and  $g_i$  is convex and superlinear, i.e.  $\lim_{t \rightarrow \infty} g_i(tp)/t = \infty$  for  $p \neq 0$ , Theorem 10.2 will hold.

For example, the data term  $c_i(x, u_i) = |u_i - f_i|$  is suited to remove salt-and-pepper noise, while  $|u - f|^2$  is applicable for Gaussian noise. More generally, one is free to choose possibly *nonconvex* data terms. With  $f_i(\nabla u_i) = \alpha_i |\nabla u_i|^2$  one can choose different gradient weightings  $\alpha_i > 0$ .

Furthermore, one can use the weighted variant

$$\int_K w(x) \, d\mathcal{H}^{m-1}(x) \quad (10.29)$$

of the edge set length  $\mathcal{H}^{m-1}(K)$  in (10.1) with a weight  $w : \Omega \rightarrow \mathbb{R}_{>0}$ . For this,  $\lambda$  must be replaced by  $\lambda w(x)$  in (10.14).

## 10.3 Implementation

### 10.3.1 Domain Relaxation

In order to minimize  $\mathcal{E}(1_u)$  and thus  $MS^{\alpha, \lambda}(u)$ , we use the convexity of  $\mathcal{E}$  and instead directly minimize over the collection of graph functions  $v = 1_u$ . The overall problem becomes

$$\inf_{v \in \mathcal{D}} \sup_{(\varphi, \mu) \in \mathcal{C}'} \sum_{i=1}^k \int_{\Omega \times \mathbb{R}} \varphi_i(x, t) \cdot dDv_i(x, t). \quad (10.30)$$

The constraint set for the solutions

$$\mathcal{D} := \left\{ v \in \text{BV}_{loc}(\Omega \times \mathbb{R}; [0, 1])^k \mid \begin{aligned} &v_i(x, t) = 1 \quad \forall t \leq a_i, \quad v_i(x, t) = 0 \quad \forall t > b_i, \\ &v_i(x, \cdot) \text{ non-increasing} \end{aligned} \right\} \quad (10.31)$$

is the convex hull of the valid graph functions, i.e. we relax the binary constraint  $v_i(x, t) \in \{0, 1\}$  allowing values in-between. The scalars  $a_i < b_i$  define the range of the channels

$$u_i : \Omega \rightarrow \Gamma_i := (a_i, b_i). \quad (10.32)$$

Theorem 10.2 assures that if we have a binary solution  $v = 1_u$  of (10.30), then  $u$  must indeed be a global optimum of the vectorial MS model (10.2). Otherwise, in general we need to project back onto the set of binary graph functions, and then recover  $u$  from the relation  $v = 1_u$ . We use the method (9.88) to obtain the final solution  $u$  of the original problem (10.1).

### 10.3.2 Discretization

The convex relaxation of this chapter is based on the same representation as in the previous chapter Chapter 9, namely using an individual graph function for each of the channels. The discretization is therefore very similar to Section 9.5.1.

We discretize the image domain  $\Omega$  into a rectangular pixel grid. For each channel  $1 \leq i \leq k$ , the range (10.32) is discretized into  $n_i \geq 1$  levels

$$a_i, a_i + \Delta t_i, \dots, a_i + (n_i - 1)\Delta t_i = b_i \quad \text{with spacing } \Delta t_i = \frac{b_i - a_i}{n_i - 1}. \quad (10.33)$$

The variables  $v, \varphi, \mu$  are discretized as follows:

$$\begin{aligned} v_i(x, \frac{j}{n_i-1}) &= v_i^j(x) \in \mathbb{R}, \\ \varphi_i^x(x, \frac{j}{n_i-1}) &= \frac{1}{\Delta t_i} \varphi_i^{x,j}(x) \in \mathbb{R}^m, \\ \varphi_i^t(x, \frac{j}{n_i-1}) &= \varphi_i^{t,j}(x) \in \mathbb{R}, \\ \mu_i(x) &= \mu_i(x) \quad (\text{same notation}) \end{aligned} \quad (10.34)$$

for all pixels  $x \in \Omega$ , channels  $1 \leq i \leq k$ , and channel values  $0 \leq j < n_i$ . The differential operators are discretized exactly as in Section 9.5.1.

The energy becomes

$$\min_{v \in \mathcal{D}_d} \max_{(\varphi, \mu) \in \mathcal{C}'_d} \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j < n_i} \left\langle \varphi_i^{x,j}(x), \nabla_x^+ v_i^j(x) \right\rangle + \varphi_i^{t,j} \partial_t^+ v_i^j(x) \quad (10.35)$$

with the discrete primal set without the monotonicity constraint

$$\mathcal{D}_d = \left\{ v = (v_i)_{1 \leq i \leq k} \mid v_i : \Omega \rightarrow [0, 1]^{n_i}, v_i^0(x) = 1 \quad \forall x \in \Omega, 1 \leq i \leq k \right\}, \quad (10.36)$$

and the dual set

$$\begin{aligned} \mathcal{C}'_d = \left\{ (\varphi_i^x, \varphi_i^t)_{1 \leq i \leq k}, \mu \mid (\varphi_i^x, \varphi_i^t) : \Omega \rightarrow \mathbb{R}^m \times \mathbb{R} \quad \forall i, \mu : \Omega \rightarrow \mathbb{R} \right. \\ \left. \varphi_i^{t,j}(x) \geq \frac{1}{4\alpha \Delta t_i^2} |\varphi_i^{x,j}(x)|^2 - c_i^j(x), \quad \forall x \in \Omega, 1 \leq i \leq k, 0 \leq j < n_i \right. \\ \left. \left| \sum_{j_1 < j \leq j_2} \varphi_i^{x,j}(x) \right| \leq \mu_i(x) \quad \forall x \in \Omega, 0 \leq j_1 < j_2 < n_i \right. \\ \left. \sum_{j=1}^k \mu_j(x) \leq \lambda \quad \forall x \in \Omega \right\} \end{aligned} \quad (10.37)$$

with the data term values  $c_i^j(x) := (t_j - f_i(x))^2 \in \mathbb{R}$  for all  $x, i, j$ . These can be precomputed for efficient look-up.

We solve the saddle-point problem (10.35) by the preconditioned primal-dual Algorithm 3. The projection onto the constraint set for  $v$  in (10.36) is straightforward. In the following we will describe a decoupling strategy to also implement the constraints of the dual set (10.37).

### 10.3.3 Projection for the Duals

Essentially, the only difficult part for (10.37) are the nonlocal sum-constraints which couple  $\varphi^x$  and  $\mu$ . Observe that introducing auxiliary variables  $p_i^j(x) \in \mathbb{R}^m$  by the requirement

$$\partial_t^- p_i^j(x) = \varphi_i^{x,j}(x) \quad \forall x, i, j, \quad (10.38)$$

the sum constraints in (10.14) simplify to

$$|p_i^{j_2}(x) - p_i^{j_1}(x)| \leq \mu_i(x) \quad \forall x, i, j_1, j_2. \quad (10.39)$$

While (10.38) can be enforced by Lagrange multipliers using (2.74), for (10.39) we can use the dual relation (2.76). The dualization procedure is analogous to Section 7.4.2, so that we will skip the details. The final decoupled energy is

$$\min_{v \in \mathcal{D}_d, (a,b) \in A, \xi} \max_{(\varphi, \mu) \in \widehat{\mathcal{C}}_d, p} \quad (10.40)$$

$$\begin{aligned} & \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j < n_i} \left\langle \varphi_i^{x,j}(x), \nabla_x^+ v_i^j(x) \right\rangle + \varphi_i^{t,j} \partial_t^+ v_i^j(x) \\ & + \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j < n_i} \left\langle \xi_i^j(x), \partial_t^- p_i^j(x) - \varphi_i^{x,j}(x) \right\rangle \\ & + \sum_{x \in \Omega} \sum_{i=1}^k \sum_{0 \leq j_1 < j_2 < n} \left\langle -a_i^{j_1 j_2}(x), p_i^{j_2}(x) - p_i^{j_1}(x) \right\rangle + b_i^{j_1 j_2}(x) \mu(x). \end{aligned} \quad (10.41)$$

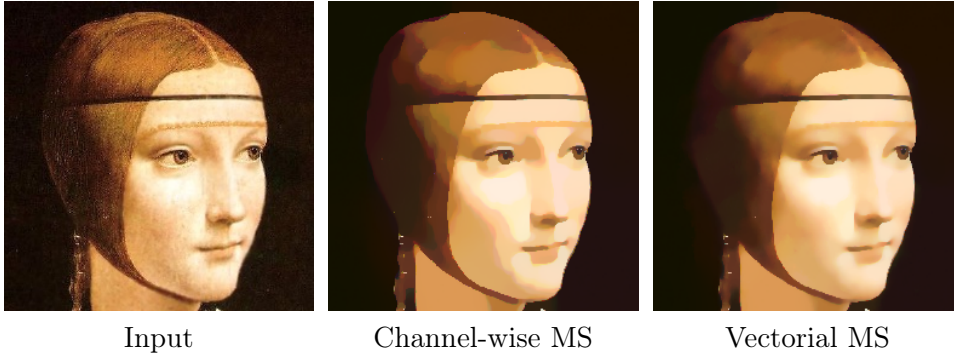
The dual set for  $\varphi$  and  $\mu$  is now reduced to only

$$\begin{aligned} \widehat{\mathcal{C}}_d = \left\{ (\varphi_i^x, \varphi_i^t)_{1 \leq i \leq k}, \mu \mid (\varphi_i^x, \varphi_i^t) : \Omega \rightarrow \mathbb{R}^m \times \mathbb{R} \quad \forall i, \mu : \Omega \rightarrow \mathbb{R}, \right. \\ \left. \varphi_i^{t,j}(x) \geq \frac{1}{4\alpha\Delta_i^2} |\varphi_i^{x,j}(x)|^2 - \varrho_i^j(x), \quad \forall x \in \Omega, 1 \leq i \leq k, 0 \leq j < n_i, \right. \\ \left. \sum_{j=1}^k \mu_j(x) \leq \lambda \quad \forall x \in \Omega \right\}, \end{aligned}$$

i.e. without the nonlocal sum-constraints. The constraints on  $\varphi$  and  $\mu$  are independent from each other. While the projection for  $\mu$  is very simple as there is only a linear constraint, the projection for  $\varphi$  is in essence a projection onto a parabola for which we can use the explicit formula of Section 9.9.2.

The new auxiliary dual variables  $p_i^j(x) \in \mathbb{R}^m$  have no constraints, just as the corresponding new primal variables  $\xi_i^j(x) \in \mathbb{R}^m$ .

Finally, the new primal variables  $a_i^{j_1 j_2}(x) \in \mathbb{R}^m$  and  $b_i^{j_1 j_2}(x) \in \mathbb{R}$  arise from (2.76) and the corresponding set  $A$  is defined by independent constraints  $|a_i^{j_1 j_2}(x)| \leq b_i^{j_1 j_2}(x)$ , which are also easy to project onto, see Section 9.9.1.



**Figure 10.3: Piecewise smooth approximation without color artifacts.** A comparison on the Leonardo da Vinci’s “Dama con l’ermellino” shows that the channel-wise solution introduces independent discontinuities in the color channels, thereby producing colors not present in the original image (blue on the cheek and green in the hair for example). In contrast, the proposed convex relaxation of the vectorial Mumford-Shah model provides more natural color transitions.

### 10.3.4 Linear Complexity Approximation

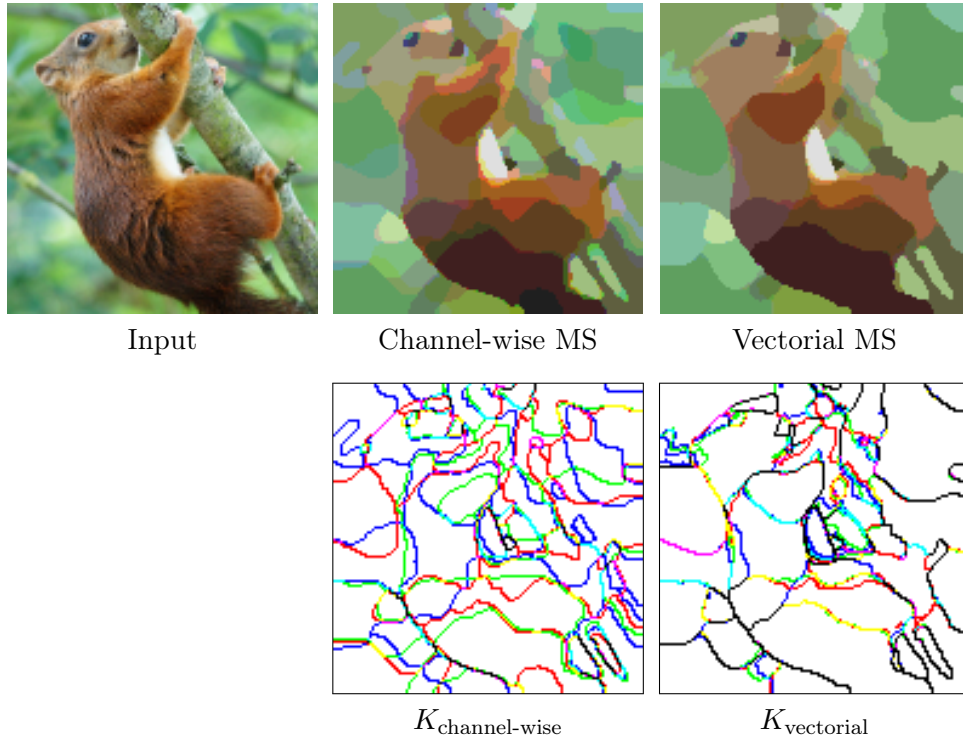
The quadratically many sum-constraints in (10.37) are responsible for correctly representing the length of the jump interface of the vectorial function  $u$ . As shown above they can be equivalently written as the difference constraints (10.39) when introducing new auxiliary variables  $p$ . This can be seen as a generalization to the vectorial case of the tight convex relaxation (3.34) for the multilabel Potts model, which also measures the interface length. More precisely, through (10.39) with a fixed  $\mu_i$  our relaxation measures the jump set length for the  $i$ -th channel  $u_i$ , and the last constraint in (10.37) on  $\mu$  ensures the correct handling of the situation when several channels jump at the same image location.

As discussed in Section 3.4.4, the constraints (3.34) yield a very tight relaxation but are computationally expensive, so that a viable alternative in practice is to use the simpler but more efficient relaxation (3.25).

This idea of simplifying the constraints for a more efficient optimization can be transferred to our vectorial case as well. Instead of (10.39), in analogy to (3.25) we can use constraints

$$|p_i^j(x)| \leq \frac{1}{2} \mu_i(x) \quad \forall x \in \Omega, \quad 1 \leq i \leq k, \quad 0 \leq j < n_i. \quad (10.42)$$

They can be implemented using the same dualization method (2.76) as was used above for (10.39). Using this linear approximation yields a less tight relaxation of the Mumford-Shah functional. However, the time and memory complexities are reduced substantially from quadratic to just *linear*. We observed similar results using both relaxations. Because of this we suggest to preferably use the linear approximation (10.42) in practice for quick results, and the more elaborate original version of the constraints when high accuracy is needed.

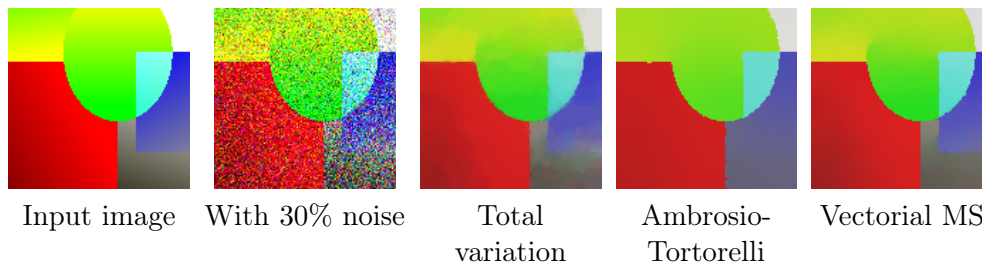


**Figure 10.4: Coupling of edge sets.** In contrast to the channel-by-channel Mumford-Shah (*center*), the vectorial Mumford-Shah model (*right*) favors discontinuities in the channels to coincide. This is confirmed by the respective discontinuity sets  $K$  for each color component (*lower row*), where black lines represent edges in all three channels.

## 10.4 Experimental Results

In the following, we will provide experimental comparisons of the proposed vectorial Mumford-Shah relaxation with several alternative algorithms on a variety of inverse problems. The channels  $u_i$  are obtained from the computed graph functions  $v_i$  in (10.3) by taking the 0.5-isolevel. When comparing channel-wise and vectorial MS, we set  $\lambda_{\text{channel-wise}} = \lambda_{\text{vectorial}}/3$ , where  $k = 3$  is the number of color channels, so that both functionals are two different convex relaxations of the same energy (10.2), since channel-wise MS counts common boundaries up to three times.

On NVIDIA GTX 480, a typical run time for  $128 \times 128$  color images with 32 levels for each color channel is about 20 seconds. For comparison, the channel-wise version runs in 19, the proposed relaxation with the simplified constraint set (10.42) in 3.5, Ambrosio-Tortorelli approximation in 1, and  $TV$  in 0.02 seconds. Thus, the proposed method accounts for the channel coupling while remaining as efficient as the simple channel-by-channel version.



**Figure 10.5: Denoising test case.** Total variation leads to a loss of contrast and staircasing (piecewise constant regions) in the lower right corner. The Ambrosio-Tortorelli method optimizes only locally, missing the blue region. In contrast, the proposed convex relaxation of the vectorial Mumford-Shah model provides the best reconstruction.

#### 10.4.1 Piecewise Smooth Approximations

Figure 10.2 shows a synthetic  $128 \times 128$  image with three different blobs for each color channel. In the piecewise smooth approximation, the vectorial MS model clearly favors solutions where the edge sets coincide. This is not the case for the channel-wise variant, which processes one color at a time and is thus “color blind” w.r.t. color as a whole. This is further confirmed in Figure 10.4 on a real world image by directly visualizing the edge sets of the different colors (coloring them accordingly). We used a  $8 \times 8 \times 8$  color discretization for both experiments.

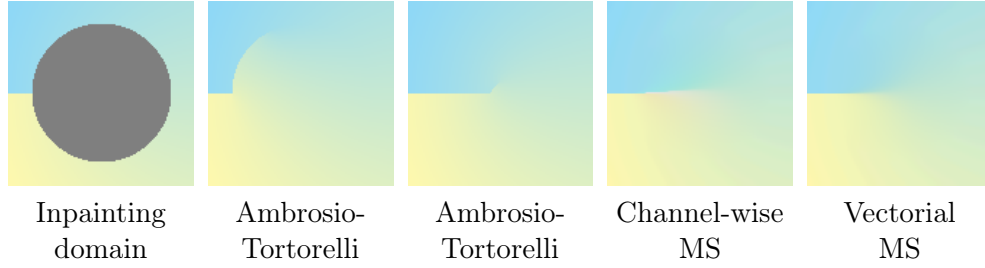
In Figure 10.3 we compute piecewise smooth approximations for the  $256 \times 256$  image “Dama con l’ermellino” by Leonardo da Vinci using vectorial MS and channel-wise MS. The parameters are  $\alpha = 100$  and  $\lambda = 0.1$ , with a  $32 \times 32 \times 32$  discretization. The large parameter  $\alpha$  makes the approximation nearly piecewise constant, showing significant color artifacts of the channel-wise MS model, as opposed to the vectorial one.

#### 10.4.2 Denoising

To compare various regularizers, we devised a synthetic denoising test case in Figure 10.5, degrading the  $128 \times 128$  test image by adding severe 30% noise. The result using simple total variation regularization produces loss of contrast and staircasing effects are visible in the grey lower right part. The result of the Ambrosio-Tortorelli method [3] is also shown. It is a nonconvex approximation of the Mumford-Shah energy and thus merely allows to compute a local minimum, missing the blue region. Finally, the proposed vectorial MS relaxation provides more natural results than the other approaches ( $32 \times 32 \times 32$ ). Figure 10.1 further demonstrates the loss of contrast of  $TV$  on a real world image.

#### 10.4.3 The Cracktip Problem

Among the fascinating aspects of the Mumford-Shah functional is that it allows to model open boundaries. In the scalar case, for inpainting a circular domain



**Figure 10.6: Inpainting test case: The cracktip problem.** *From left to right:* Grey inpainting domain and colors prescribed on the boundary. Ambrosio-Tortorelli method can get stuck in bad local minima. Ambrosio-Tortorelli method for a good initial guess. Channel-wise MS introduces faulty red colors in the center. Vectorial MS produces a good approximation to the cracktip solution.

with a Mumford-Shah regularizer and specific values prescribed on the boundary, a well-known analytical solution is the so called cracktip function [14], given in polar and Cartesian coordinates by

$$f_p(r, \varphi) = \sqrt{r} \sin \frac{\varphi}{2}, \quad \text{resp.} \quad f_c(x, y) = \frac{y}{\sqrt{2(x + \sqrt{x^2 + y^2})}}. \quad (10.43)$$

In the vectorial case, transforming  $f$  linearly for each color channel, one can easily see that “color cracktips” are also MS minimizers. Thus it is interesting to see, how well these solutions are recovered with different algorithms.

After discretization of the image domain into  $w \times h$  pixels, we set  $a = 1/f_c(-\frac{w-1}{2}, \frac{h-1}{2}) > 0$ . We choose colors  $c_1, c_2 \in \mathbb{R}^3$  and consider the function

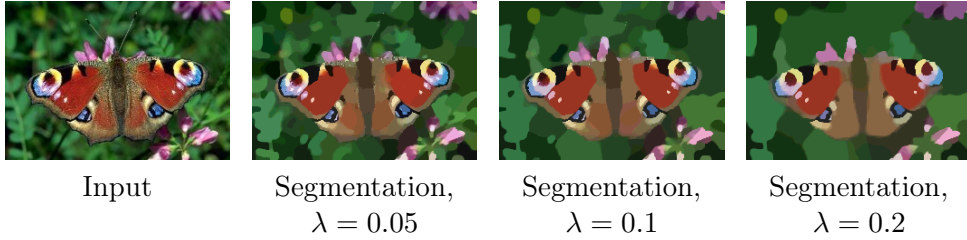
$$\hat{f}(x, y) = \frac{c_1 + c_2}{2} + \frac{c_2 - c_1}{2} a f_c(x - \frac{w-1}{2}, y - \frac{h-1}{2}). \quad (10.44)$$

Then  $\hat{f}$  is a “color cracktip”, interpolating smoothly between  $\hat{f}(0, 0) = c_1$  in the upper left and  $\hat{f}(0, h-1) = c_2$  in the lower left corner going clockwise around the origin, see Figure 10.6. The parameters  $\alpha$  and  $\lambda$  in (10.1) must be chosen so that  $\alpha \int_{\Omega \setminus K} |\nabla \hat{f}|^2 dx = \lambda \mathcal{H}^{m-1}(K)$  where  $\Omega$  the a disc of some radius  $R$  around  $(\frac{w-1}{2}, \frac{h-1}{2})$ , and  $K$  is the portion of  $\Omega$  along the semiline going to the left of this point. This leads to

$$\lambda = \alpha \cdot \frac{\pi a^2}{8} |c_2 - c_1|^2. \quad (10.45)$$

Figure 10.6 shows the corresponding  $128 \times 128$  inpainting experiment using a  $64 \times 64 \times 64$  color discretization, with  $c_1 = \frac{(254, 248, 174)}{255}$  (light yellow) and  $c_2 = \frac{(142, 216, 247)}{255}$  (light blue). We set  $\lambda = 1$  and compute  $\alpha$  using the formula above. The dataterm is set to  $c = 0$  inside the circle and to  $M(u - f)^2$  outside with a big  $M > 0$  (using the data term generalization of Section 10.2.4). As expected, the nonconvex Ambrosio-Tortorelli approximation greatly relies on the initial solution to produce acceptable results: With a bad initialization it generates strong artifacts (second image). Yet, even with a good initial guess,





**Figure 10.7: Joint segmentation and color selection.** In contrast to previous relaxations where the set of allowed color models must be specified beforehand, the proposed approach automatically selects the appropriate color models during the segmentation process based on the scale parameter  $\lambda$ . Color discretization is  $16 \times 16 \times 16$ .

the same artifact emerges at a smaller scale (third image). The channel-wise MS model treats the color channels independently, and thus introduces faulty red colors in the center. In contrast, the proposed convex relaxation of the vectorial MS model provides a good approximation of the color cracktip.

#### 10.4.4 Unsupervised Image Partitioning

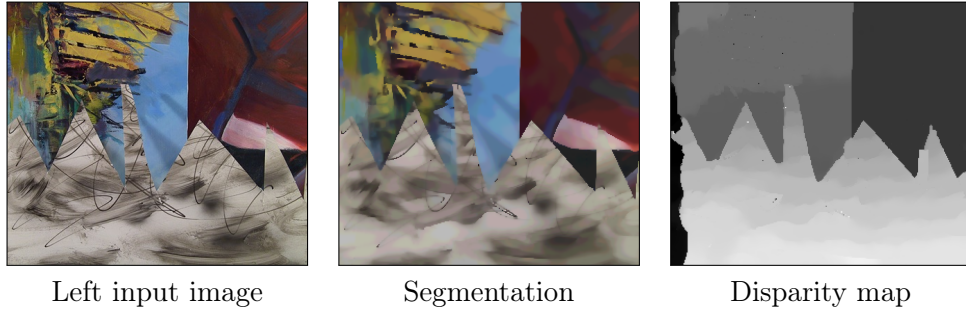
In the limiting case  $\alpha \rightarrow \infty$  in (10.2), the smoothness constraint is enforced more and more leading to the well-known *piecewise constant* approximation, also known as the *minimal partition problem*. The first constraint in (10.14) reduces to just  $\varphi_i^t(x, t) \geq -(t - f_i(x))^2$ . Only one parameter  $\lambda$  remains in the model (10.2), which controls the overall length of the interfaces between the regions where  $u$  is constant.

The standard approach to compute such a multi-region partitioning is by alternatingly determining a small number of color models (with fixed regions) and optimizing for the regions (with fixed color models). Of course, for such iterative approaches results are invariably suboptimal and will depend on the choice of the initial color models.

In contrast, the proposed convex relaxation allows to *jointly* optimize over the regions of constancy and the corresponding colors in these regions in a single convex optimization problem. In particular, the algorithm determines the appropriate number of color models depending on the input image and the scale parameter  $\lambda$ . Figure 10.7 shows the partitionings computed for various values of  $\lambda$ .

#### 10.4.5 Joint Disparity and Segmentation

Our final experiment is a more advanced application of the vectorial Mumford-Shah model (10.2) to stereo image analysis. Given a stereo image pair, the task is to jointly compute a disparity map and a color segmentation, the central idea being that discontinuities in disparity and color tend to coincide. While this problem has been addressed in the nonconvex Ambrosio-Tortorelli framework [103], we can apply the proposed convex relaxation of the vectorial Mumford-Shah functional to compute solutions independently of initialization.



**Figure 10.8: Joint segmentation and stereo depth reconstruction.** Applying the proposed relaxation of the vectorial Mumford-Shah model to the joint estimation of color segmentation and disparity, we can impose that discontinuities of the color values preferably coincide with depth discontinuities. The label space is four dimensional with three color channels and one depth channel.

The proposed problem simply corresponds to the case of  $k = 4$  channels corresponding to the three color and one depth channel:  $u = (u^{\text{color}}, u^{\text{depth}})$ . For the data term and gradient penalization we use the generalized variants of Section 10.2.4. We use the data term from [103]:

$$\begin{aligned} c_i^{\text{color}}(x, t) &= (1 - \gamma) (t - I_{\text{left}}^i(x))^2, \\ c^{\text{depth}}(x, t) &= \gamma \sum_{j=1}^D \frac{(t - d_j(x))^2}{1 + (t - d_j(x))^2} \end{aligned} \quad (10.46)$$

with  $\gamma = 0.05$  and  $D = 4$  depth hypotheses  $d_j : \Omega \rightarrow \mathbb{R}$  calculated as in [103, Section 5.1]. Furthermore, we use custom coefficients for the gradient regularizations:  $g_i^{\text{color}}(\nabla u_i) = 2 |\nabla u_i|^2$  and  $g^{\text{depth}}(\nabla u) = 100 |\nabla u|^2$  in (10.27).

Figure 10.8 shows the jointly computed segmentation and disparity for the  $289 \times 253$  “sawtooth” test image of the Middlebury stereo dataset. We set  $\lambda = 0.01$ , using a  $16 \times 16 \times 16 \times 20$  discretization. The clear correlation between the color and disparity edges, especially at the right (non-occluding) boundaries of the sawtooths confirms the advantage of the proposed model.

## 10.5 Conclusion

We proposed a convex relaxation for the vectorial Mumford-Shah problem. In contrast to a naive sequential processing of each color channel, this approach allows to correctly handle the coupling of all color channels with a single discontinuity set. As a consequence, it assures that color discontinuities tend to coincide and thus avoids color artifacts. Furthermore, the relaxation is computationally tractable. We proposed an efficient algorithmic implementation which allows to capture the color coupling at essentially no additional run time. Experimental comparisons with the channel-wise application of the scalar Mumford-Shah model, total variation and the nonconvex Ambrosio-Tortorelli approach confirm the superiority of the proposed Mumford-Shah relaxation for contrast-preserving and edge-enhancing regularization in vectorial inverse problems.

# Chapter 11

## Conclusion

### Summary

In this thesis we presented a number of techniques for the convexification of image analysis problems. We considered two important classes of energies, multilabel and vectorial problems, and developed novel convex relaxations for several examples in each case. Though each energy required its own approach, one of the key ingredients for finding efficient relaxations was the choice of a suitable representation of the solutions.

For multilabel problems, the idea was to start with the indicator function representation of the label regions. We considered three different types of higher-level knowledge for the labeling configurations and showed how each can be incorporated in a convex way in terms of the indicator functions:

- The length regularization prior was extended to nonmetric distances, thus allowing for a more accurate multi-object segmentation.
- The convexification of geometric ordering constraints makes it possible to obtain special layouts for the label regions.
- The proposed proportion priors make image sequence segmentation more robust to shape and scale changes.

For vectorial functionals, we proposed to express the energy in terms of the graph functions, introducing one graph function for each of the channels. We developed especially tailored convexifications for three special cases of vectorial functionals:

- For the case of separable regularization, on the one hand this representation allowed us to reduce the overall complexity by orders of magnitude in comparison to previous methods. On the other hand, the data term becomes nonconvex in this representation. To cope with this, we introduced a novel convex relaxation and showed it to be the best possible one.
- For coupling regularizers we identified novel cases which still allow to keep the complexity as low as in the separable case.
- Finally, for the vectorial Mumford-Shah energy we developed a convex relaxation which is as efficient as when processing the channels separately.

For the initially obtained convex formulations, in each case we indicated ways to arrive at efficient parallel implementations on GPUs. For this we showed how the models can be gradually reformulated by utilizing suitable decoupling strategies, so that the final models become more amenable to parallelization.

## Advantages of Convexification Methods

Convex relaxation methods strive to replace the original nonconvex energy by a convex one, and to recover solutions of the original problem by solving the relaxed one. In this thesis we showed that this presents a powerful tool to assess a wide range of functionals. The experimental results demonstrate that it can be successfully applied to obtain high-quality solutions.

Although the techniques presented in the two main parts of the thesis are different from case to case, they share a number of advantages which are typical to convexification methods. Let us emphasize the following:

- **Optimality Guarantees.** Generally, it is possible to obtain optimality bounds which allow to determine how far one is from the global optimum. Having computed a solution, natural energy bounds give an estimate of the unknown optimal energy. This can be used to see how much higher the energy of the obtained solution is in comparison to the optimal energy value. The bound is observed to be very small in practice, with values as low as 5%, which means that near-optimal solutions are achieved. Special classes of energies can even be solved globally optimally through convex relaxation. Notable examples are the cases of two-label segmentation, and the total variation regularization of scalar functions with nonconvex data terms.
- **Initialization Independence.** Since the final optimization problem is convex, there is no dependency on an initialization. For instance, one can conveniently set all variables to zero in the beginning, as was done in our experiments. Thus, no expertise on choosing a suitable initial approximation is required from the user. The required input is reduced to only the model parameters (and possibly scribbles or similar input for the learning of the data term), so that even fully automatic algorithms become possible once the parameters are fixed or learned.
- **Parallelization.** Variational methods and their convex relaxations have the major advantage of parallelization by GPUs. The obtained models can be typically solved by general and flexible primal-dual algorithms which are well suited for massive parallelization. They can be conveniently implemented on GPUs using the CUDA framework of NVIDIA. Run times of a few seconds can be achieved for moderate problem sizes and several GPUs can be used simultaneously for more parallelization. Note that variational approaches are based on the viewpoint of continuous image domains, as assumed throughout this thesis. In contrast, discrete approaches such as  $\alpha$ -expansion are inherently serial and hardly parallelizable.

## Limitations of Convexification Methods

Despite their advantages, convex relaxation methods nonetheless can be more or less suitable for each concrete application depending on the requirements imposed on the solution or the available resources. Regarding the relaxations from in the main two parts of the thesis, there are also a number of limitations:

- **Optimality Bounds are per Instance.** The available energy bounds mentioned above are only a-posteriori. This means that one first has to compute the solution in order to get a bound on its optimality. Though the bound is typically very tight in practice, there is no guarantee that this always will be the case, and it may be different for each problem instance. Special energies do allow true a-priori bounds, i.e. bounds computable in advance, which are then a property of the convex relaxation itself and not of the solution. However, these bounds are usually quite large and thus of little practical interest. Thus, the use of general convexification methods is limited for critical applications which require firm guarantees on the quality of the solution, as e.g. in certain medical imaging applications.
- **Memory Requirements.** Certain convexification methods pose high demands on memory requirements for the implementation, as the final models exhibit a large number of variables. This is especially the case for the approaches based on functional lifting as in the second part of the thesis, since the range of the functions must be discretized. Multi-label problems may also require a large number of variables depending on the relaxation. However, this becomes less and less an issue with the recent developments in high-performance computing. One can now resort to more computational resources, making it possible to consider models which were previously intractable. Most notably, this comes from the significant advances in the direction of GPUs in the recent years. The increase of on-board memory allows larger problems and massive parallelization permits to keep the run time low.

## Outlook and Future Research

In this thesis we have proposed and studied convex relaxations for special cases of multilabel and vectorial energies. There is of course still room for interesting further developments, so that we would like to indicate some possible directions of future research:

- **Relaxations for Further Functionals.** It would be advantageous to develop convex relaxations for further interesting and important functionals frequently occurring in practice, thus enabling to find good approximations of the solutions independently of the initialization. For instance:
  - Deblurring and deconvolution applications with a nonconvex data term or regularizer. This requires convolutional operators in the

data term. However, functional lifting and the approaches for vectorial functionals in Part II assume a pointwise data term. Current approaches typically rely on decoupling strategies [144] leading to alternating minimization with possibly suboptimal results.

- Higher-order derivatives. For segmentation, additional curvature penalization is an interesting alternative to length-only penalization, which allows to preserve thin and long structures. Current relaxations are restricted to certain special cases [21] or are more general but with a high computational demand [20]. Another interesting example of a higher-order functional is regularization of nonconvex data terms with total generalized variation (TGV), where one also resorts to alternating minimization [106].
- **Real-Time Applications.** One question is whether convex optimization methods can be applied in scenarios with real-time requirements. For example, one may be required to compute wide-range optical flow as quickly as 30 frames per second. Currently, convexification methods are not ideal for such applications. Although they can provide high-quality solutions, current relaxations still need a couple of seconds per frame. An alternative to convex relaxation is a direct optimization of the nonconvex model at hand. Recently, for the primal-dual algorithm a number of extensions to nonconvex energies were introduced [123, 86]. They do not require range discretization so that the memory and run time requirements are significantly lower. For the Mumford-Shah functional case [123], one can achieve result even in real-time and of similar quality as with the convex relaxation of Chapter 10.

- **Functions with Values in a Manifold.** Considering the minimization of an energy  $E(u)$ , we can look at the optimization approaches from the point of view of what they assume about the domain and range of the solution functions  $u$ . In the initial approach of Ishikawa [60] for convex regularization of scalar functions, both the domain and the range are assumed to be discrete:  $u : \Omega_{\text{discrete}} \rightarrow \Gamma_{\text{discrete}}$ .

This motivated the use of the functional lifting approach for imaging applications in [102, 101], which allows for continuous image domains  $\Omega$  and a continuous but scalar range (which is discretized only in the end for the actual implementation):  $u : \Omega \rightarrow \mathbb{R}$ .

In the second part of this thesis we considered the regularization of functions with a vectorial domain:  $u : \Omega \rightarrow \mathbb{R}^k$  with  $k \geq 1$ .

A natural extension would be to consider energies defined more generally on functions mapping into a manifold  $\mathcal{M}$ :  $u : \Omega \rightarrow \mathcal{M}$ . For example,  $\mathcal{M}$  could be the space of angles or directions in the plane, the space of normals in  $\mathbb{R}^3$ , or the space of rotations in  $\mathbb{R}^3$ . Several advances in this direction for special cases of manifolds  $\mathcal{M}$  and energies  $E$  include [39, 80] and [12, 135].

# Bibliography

- [1] Alberti, G., Bouchitté, G., Maso, G.D.: The calibration method for the Mumford-Shah functional and free-discontinuity problems. *Calculus of Variations and Partial Differential Equations* 16(3), 299–333 (2003) [8](#), [74](#), [110](#), [111](#), [112](#), [158](#), [200](#), [202](#), [203](#), [205](#)
- [2] Ambrosio, L., Tortorelli, V.M.: Variational problems in SBV and image segmentation. *Acta Applicandae Mathematicae* 17(1), 1–40 (1989) [200](#), [201](#)
- [3] Ambrosio, L., Tortorelli, V.M.: Approximation of functionals depending on jumps by elliptic functionals via  $\Gamma$ -convergence. *Communications on Pure and Applied Mathematics* 43, 999–1036 (1990) [200](#), [201](#), [211](#)
- [4] Ambrosio, L.: A compactness theorem for a new class of functions of bounded variation. *Bollettino della Unione Matematica Italiana. Serie VII. B* 3(4), 857–881 (1989) [202](#)
- [5] Ambrosio, L., Fusco, N., Pallara, D.: Functions of bounded variation and free discontinuity problems. *Oxford Mathematical Monographs*, The Clarendon Press Oxford University Press, New York (2000) [19](#), [20](#), [21](#), [70](#), [74](#), [113](#), [156](#), [191](#)
- [6] Attouch, H., Buttazzo, G., Michaille, G.: Variational Analysis in Sobolev and BV Spaces. *MPS-SIAM Series on Optimization*, Society for Industrial and Applied Mathematics (SIAM) (2006) [19](#), [71](#), [154](#)
- [7] Bae, E., Yuan, J., Tai, X.C.: Global minimization for continuous multi-phase partitioning problems using a dual approach. *International Journal of Computer Vision* 92(1), 112–129 (2011)
- [8] Bai, X., Sapiro, G.: A geodesic framework for fast interactive image and video segmentation and matting. In: *IEEE International Conference on Computer Vision (ICCV)* (2007) [54](#)
- [9] Baker, S., Kanade, T.: Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(9), 1167–1183 (2002) [109](#)
- [10] Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. *International Journal of Computer Vision* 92(1), 1–31 (2011) [182](#)

- [11] Batra, D., Kowdle, A., Parikh, D., Luo, J., Chen, T.: iCoseg: Interactive co-segmentation with intelligent scribble guidance. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2010) **89, 90, 98, 99**
- [12] Bergmann, R., Laus, F., Steidl, G., Weinmann, A.: Second order differences of cyclic data and applications in variational denoising. *SIAM Journal on Imaging Sciences* 7(4), 2916–2953 (2014) **218**
- [13] Blake, A., Zisserman, A.: *Visual Reconstruction*. MIT Press, Cambridge, MA (1987) **2, 3**
- [14] Bonnet, A., David, G.: *Cracktip is a global Mumford-Shah minimizer*. Société mathématique de France, Paris (2001), astérisque, ISSN 0303-1179, vol. 274 **212**
- [15] Bouchitté, G., Valadier, M.: Integral representation of convex functionals on a space of measures. *Journal of Functional Analysis* 80(2), 398–420 (1988) **113, 191**
- [16] Bouchitté, G., Valadier, M.: Multifonctions s.c.i. et régularisée s.c.i. essentielle. *Annales de l’Institut Henri Poincaré. Analyse Non Linéaire* 6(suppl.), 123–149 (1989), analyse non linéaire (Perpignan, 1987) **191**
- [17] Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: *IEEE International Conference on Computer Vision (ICCV)* (2003) **2**
- [18] Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(9), 1124–1137 (2004) **141**
- [19] Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(11), 1222–1239 (2001) **39, 40, 47, 53, 82, 141, 142, 200**
- [20] Bredies, K., Bock, T., Wirth, B.: A convex, lower semi-continuous approximation of Euler’s elastica energy. Preprint (2013), [http://gpu4vision.icg.tugraz.at/index.php?content=Cat\\_0](http://gpu4vision.icg.tugraz.at/index.php?content=Cat_0) **218**
- [21] Bredies, K., Pock, T., Wirth, B.: Convex relaxation of a class of vertex penalizing functionals. *Journal of Mathematical Imaging and Vision* 47, 278–302 (2012) **218**
- [22] Brook, A., Kimmel, R., Sochen, N.A.: Variational restoration and edge detection for color images. *Journal of Mathematical Imaging and Vision* 18(3), 247–268 (2003) **200**
- [23] Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: *European Conference on Computer Vision (ECCV)* (2004) **2**



- [24] Brox, T., Cremers, D.: On local region models and a statistical interpretation of the piecewise smooth Mumford-Shah functional. *International Journal of Computer Vision* 84, 184–193 (2009) 144
- [25] Capel, D., Zisserman, A.: Super-resolution from multiple views using learnt image models. In: *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2001) 109
- [26] Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. In: *IEEE International Conference on Computer Vision (ICCV)* (1995) 2
- [27] Chambolle, A.: Finite-differences discretizations of the Mumford-Shah functional. *Journal of Mathematical Modelling and Numerical Analysis* 112(2), 261–288 (1999) 125
- [28] Chambolle, A.: Convex representation for lower semicontinuous envelopes of functionals in  $L^1$ . *Journal of Convex Analysis* 8(1), 149–170 (2001) 200
- [29] Chambolle, A., Cremers, D., Pock, T.: A convex approach for computing minimal partitions. Technical Report TR-2008-05, Department of Computer Science, University of Bonn, Bonn, Germany (November 2008) 2, 37, 40, 45, 50, 53, 91, 129, 132, 149, 200
- [30] Chambolle, A., Cremers, D., Pock, T.: A convex approach to minimal partitions. *SIAM Journal on Imaging Sciences* 5(4), 1113–1158 (2012) 40, 45, 110, 118
- [31] Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision* 40, 120–145 (2011) 24, 25, 26, 27, 28
- [32] Chambolle, A., Pock, T.: On the ergodic convergence rates of a first-order primal-dual algorithm. Preprint (2014), [http://gpu4vision.icg.tugraz.at/index.php?content=Cat\\_0](http://gpu4vision.icg.tugraz.at/index.php?content=Cat_0) 27
- [33] Chan, T., Esedoğlu, S., Nikolova, M.: Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics* 66(5), 1632–1648 (2006) 2, 4, 7, 37, 40, 136, 200
- [34] Chan, T., Vese, L.: Active contours without edges. *IEEE Transactions on Image Processing* 10(2), 266–277 (2001) 2
- [35] Chan, T., Yip, A., Park, F.: Simultaneous total variation image inpainting and blind deconvolution. *International Journal of Imaging Systems and Technology* 15(1), 92–102 (2005) 109
- [36] Chekuri, C., Khanna, S., Naor, J., Zosin, L.: A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics* 18, 608–625 (2005) 54, 59, 60, 61, 63

- [37] Collins, M., Xu, J., Grady, L., Singh, V.: Random walks based multi-image segmentation: Quasiconvexity results and GPU-based solutions. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) [90](#), [99](#), [102](#)
- [38] Cremers, D., Kolev, K.: Multiview stereo and silhouette consistency via convex functionals over convex domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(6), 1161–1174 (2011) [2](#), [38](#)
- [39] Cremers, D., Strelakovski, E.: Total cyclic variation and generalizations. *Journal of Mathematical Imaging and Vision* 47(3), 258–277 (2012) [218](#)
- [40] Dal Maso, G.: Integral representation on  $BV(\Omega)$  of  $\Gamma$ -limits of variational integrals. *Manuscripta Mathematica* 30(4), 387–416 (1979) [191](#)
- [41] DeLong, A., Osokin, A., Isack, H., Boykov, Y.: Fast approximate energy minimization with label costs. *International Journal of Computer Vision* 96(1), 1–27 (2012) [141](#), [142](#)
- [42] Demengel, F., Temam, R.: Convex functions of a measure and applications. *Indiana University Mathematics Journal* 33(5), 673–709 (1984) [191](#)
- [43] Dobson, D., Santosa, F.: Recovery of blocky images from noisy and blurred data. *SIAM Journal on Applied Mathematics* 56(4), 1181–1198 (1996) [109](#)
- [44] Evans, L.C., Gariepy, R.F.: Measure theory and fine properties of functions. *Studies in Advanced Mathematics*, CRC Press, Boca Raton, FL (1992) [161](#)
- [45] Faugeras, O., Keriven, R.: Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *IEEE Transactions on Image Processing* 7(3), 336–344 (1998) [2](#)
- [46] Federer, H.: Geometric measure theory. Springer-Verlag (1969) [110](#)
- [47] Federer, H.: Real flat chains, cochains and variational problems. *Indiana University Mathematics Journal* 24, 351–407 (1974) [110](#)
- [48] Felzenszwalb, P.F., Veksler, O.: Tiered scene labeling with dynamic programming. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2010) [9](#), [67](#), [78](#), [80](#), [82](#), [83](#), [84](#), [85](#)
- [49] Geiger, D., Girosi, F.: Parallel and deterministic algorithms from MRFs: Surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(5), 401–412 (1991) [3](#)
- [50] Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6(6), 721–741 (1984) [3](#)

- [51] Giaquinta, M., Modica, G., Souček, J.: Cartesian Currents in the Calculus of Variations I: Cartesian Currents, vol. 37. Springer-Verlag, Berlin (1998) [110](#), [200](#)
- [52] Glocker, B., Paragios, N., Komodakis, N., Tziritas, G., Navab, N.: Optical flow estimation with uncertainties through dynamic MRFs. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2008) [121](#)
- [53] Goldluecke, B., Cremers, D.: Convex relaxation for multilabel problems with product label spaces. In: European Conference on Computer Vision (ECCV) (2010) [38](#), [122](#), [126](#), [127](#), [128](#), [129](#), [130](#), [132](#), [140](#), [146](#), [147](#), [148](#)
- [54] Goldluecke, B., Strelakovsky, E., Cremers, D.: The natural total variation which arises from geometric measure theory. SIAM Journal on Imaging Sciences 5(2), 537–563 (2012) [156](#), [172](#), [178](#), [188](#)
- [55] Goldluecke, B., Strelakovsky, E., Cremers, D.: Tight convex relaxations for vector-valued labeling. SIAM Journal on Imaging Sciences 6(3), 1626–1664 (2013) [8](#), [119](#), [122](#)
- [56] Greig, D.M., Porteous, B.T., Seheult, A.H.: Exact maximum a posteriori estimation for binary images. Journal of the Royal Statistical Society B 51(2), 271–279 (1989) [39](#), [200](#)
- [57] Hochbaum, D., Singh, V.: An efficient algorithm for co-segmentation. In: IEEE International Conference on Computer Vision (ICCV) (2009) [90](#)
- [58] Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. International Journal of Computer Vision 75, 151–172 (2007) [78](#), [82](#), [83](#)
- [59] Horn, B., Schunck, B.: Determining optical flow. Artificial Intelligence 17, 185–203 (1981) [2](#), [3](#)
- [60] Ishikawa, H.: Exact optimization for Markov random fields with convex priors. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(10), 1333–1336 (2003) [8](#), [39](#), [40](#), [109](#), [110](#), [200](#), [218](#)
- [61] Ishikawa, H., Geiger, D.: Segmentation by grouping junctions. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (1998) [110](#), [200](#)
- [62] Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2010) [89](#), [91](#), [99](#), [102](#)
- [63] Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision 1(4), 321–331 (1988) [2](#)
- [64] Kichenassamy, S., Kumar, A., Olver, P.J., Tannenbaum, A., Yezzi, A.J.: Gradient flows and geometric active contour models. In: IEEE International Conference on Computer Vision (ICCV) (1995) [2](#)

- [65] Kimmel, R., Malladi, R., Sochen, N.A.: Image processing via the Beltrami operator. In: Asian Conference on Computer Vision (ACCV) (1998) 200
- [66] Kleinberg, J., Tardos, E.: Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In: IEEE Annual Symposium on Foundations of Computer Science (1999) 47
- [67] Klodt, M., Cremers, D.: A convex framework for image segmentation with moment constraints. In: IEEE International Conference on Computer Vision (ICCV) (2011) 91
- [68] Klodt, M., Schoenemann, T., Kolev, K., Schikora, M., Cremers, D.: An experimental comparison of discrete and continuous shape optimization methods. In: European Conference on Computer Vision (ECCV) (2008) 2, 3, 143, 144
- [69] Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(10) (2006) 141
- [70] Kolmogorov, V., Rother, C.: Minimizing non-submodular functions with graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(7), 1274–1279 (2007) 2, 39
- [71] Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(2), 147–159 (2004) 39, 141
- [72] Komodakis, N., Tziritas, G.: Approximate labeling via graph-cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(8), 1436–1453 (2007) 39
- [73] Kulkarni, S., Mitter, S., Richardson, T., Tsitsiklis, J.: Local versus non-local computation of length of digitized curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(7), 711–718 (1994) 109
- [74] Kusch, G., Cremers, D.: Fast and accurate large-scale stereo reconstruction using variational methods. In: ICCV Workshop on Big Data in 3D Computer Vision (2013) 32
- [75] Ladicky, L., Russell, C., Kohli, P., Torr, P.H.: Graph cut based inference with co-occurrence statistics. In: European Conference on Computer Vision (ECCV) (2008) 54, 62, 63, 64
- [76] Lellmann, J., Becker, F., Schnörr, C.: Convex optimization for multi-class image labeling with a novel family of total variation based regularizers. In: IEEE International Conference on Computer Vision (ICCV) (2009) 2, 37, 40, 44, 54, 73, 129, 130, 131, 132, 141, 146, 148

- [77] Lellmann, J., Kappes, J., Yuan, J., Becker, F., Schnörr, C.: Convex multi-class image labeling by simplex-constrained total variation. Tech. rep., IPA, HCI, Heidelberg University (October 2008) [91](#), [200](#)
- [78] Lellmann, J., Lenzen, F., Schnörr, C.: Optimality bounds for a variational relaxation of the image partitioning problem. *Journal of Mathematical Imaging and Vision* 47(3), 239–257 (2013) [47](#)
- [79] Lellmann, J., Schnörr, C.: Continuous multiclass labeling approaches and algorithms. *SIAM Journal on Imaging Sciences* 4(4), 1049–1096 (2011) [40](#), [44](#), [53](#), [58](#)
- [80] Lellmann, J., Strekalovskiy, E., Koetter, S., Cremers, D.: Total variation regularization for functions with values in a manifold. In: *IEEE International Conference on Computer Vision (ICCV)* (2013) [218](#)
- [81] Lellmann, J., Breitenreicher, D., Schnörr, C.: Fast and exact primal-dual iterations for variational problems in computer vision. In: *European Conference on Computer Vision (ECCV)* (2010) [50](#)
- [82] Liu, X., Veksler, O., Samarabandu, J.: Order-preserving moves for graph-cut-based optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 1182–1196 (2010) [9](#), [67](#), [78](#), [80](#)
- [83] McKelvey, J.P.: Simple transcendental expressions for the roots of cubic equations. *American Journal of Physics* 52(3), 269–270 (1984) [103](#), [196](#)
- [84] Michelot, C.: A finite algorithm for finding the projection of a point onto the canonical simplex of  $R^n$ . *Journal of Optimization Theory and Applications* 50, 195–200 (1986) [49](#), [137](#)
- [85] Mirsky, L.: Symmetric gauge functions and unitarily invariant norms. *Quarterly Journal of Mathematics, Oxford Journals* 2(11), 50–59 (1960) [197](#)
- [86] Moellenhoff, T., Strekalovskiy, E., Moeller, M., Cremers, D.: The primal-dual hybrid gradient method for semiconvex splittings. *SIAM Journal on Imaging Sciences* 8(2), 827–857 (2015) [218](#)
- [87] Möllenhoff, T., Strekalovskiy, E., Möller, M., Cremers, D.: Low rank priors for color image regularization. In: *International Conference on Energy Minimization Methods for Computer Vision and Pattern Recognition (EMMCVPR)* (2015)
- [88] Mora, M.G.: The calibration method for free-discontinuity problems on vector-valued maps. *Journal of Convex Analysis* 9(1), 1–29 (2002) [201](#)
- [89] Morel, J.M., Solimini, S.: *Variational Methods in Image Segmentation*. Birkhäuser, Boston (1995) [199](#)
- [90] Mukherjee, L., Singh, V., Dyer, C.: Half-integrality based algorithms for cosegmentation of images. In: *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2009) [89](#), [90](#)

- [91] Mukherjee, L., Singh, V., Xu, J., Collins, M.: Analyzing the subspace structure of related images: Concurrent segmentation of image sets. In: European Conference on Computer Vision (ECCV) (2012) [89](#), [91](#), [98](#), [99](#), [102](#)
- [92] Mumford, D., Shah, J.: Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics* 42, 577–685 (1989) [2](#), [199](#)
- [93] Nieuwenhuis, C., Cremers, D.: Spatially varying color distributions for interactive multi-label segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(5), 1234–1247 (2013) [98](#)
- [94] Nieuwenhuis, C., Strekalovskiy, E., Cremers, D.: Proportion priors for image sequence segmentation. In: IEEE International Conference on Computer Vision (ICCV) (2013) [8](#), [89](#)
- [95] Nieuwenhuis, C., Toeppe, E., Cremers, D.: A survey and comparison of discrete and continuous multilabel segmentation approaches. *International Journal of Computer Vision* 104(3), 223–240 (2013) [40](#)
- [96] Osher, S.J., Sethian, J.A.: Fronts propagation with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics* 79, 12–49 (1988) [40](#)
- [97] Parzen, E.: On the estimation of a probability density function and the mode. *Annals of Mathematical Statistics* 33, 1065–1076 (1962) [98](#)
- [98] Pock, T., Chambolle, A.: Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In: IEEE International Conference on Computer Vision (ICCV) (2011) [24](#), [27](#)
- [99] Pock, T., Chambolle, A., Bischof, H., Cremers, D.: A convex relaxation approach for computing minimal partitions. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2009) [45](#), [125](#)
- [100] Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the piecewise smooth Mumford-Shah functional. In: IEEE International Conference on Computer Vision (ICCV) (2009) [2](#), [11](#), [24](#), [38](#), [60](#), [61](#), [110](#), [111](#), [118](#), [125](#), [129](#), [132](#), [200](#), [201](#), [202](#), [205](#)
- [101] Pock, T., Cremers, D., Bischof, H., Chambolle, A.: Global solutions of variational models with convex regularization. *SIAM Journal on Imaging Sciences* 3(4), 1122–1145 (2010) [2](#), [8](#), [38](#), [40](#), [110](#), [113](#), [115](#), [117](#), [125](#), [129](#), [131](#), [132](#), [133](#), [157](#), [159](#), [165](#), [200](#), [218](#)
- [102] Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. In: European Conference on Computer Vision (ECCV) (2008) [40](#), [110](#), [125](#), [200](#), [218](#)

- [103] Pock, T., Zach, C., Bischof, H.: Mumford-Shah meets stereo: Integration of weak depth hypotheses. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2007) [213](#), [214](#)
- [104] Potts, R.B.: Some generalized order-disorder transformations. *Proceedings of the Cambridge Philosophical Society* 48, 106–109 (1952) [38](#)
- [105] Ramalingam, S., Kohli, P., Alahari, K., Torr, P.: Exact inference in multi-label CRFs with higher order cliques. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2008) [121](#)
- [106] Ranftl, R., Pock, T., Bischof, H.: Minimizing TGV-based Variational Models with Non-Convex Data terms. In: International Conference on Scale Space and Variational Methods in Computer Vision (SSVM) (2013) [218](#)
- [107] Riklin Raviv, T., Sochen, N., Kiryati, N.: Shape-based mutual segmentation. *International Journal of Computer Vision* 79, 231–245 (2008) [90](#)
- [108] Rockafellar, R.T.: *Convex analysis*. Princeton Landmarks in Mathematics, Princeton University Press, Princeton, NJ (1997), reprint of the 1970 original, Princeton Paperbacks [15](#), [25](#), [151](#), [193](#)
- [109] Rother, C., Minka, T., Blake, A., Kolmogorov, V.: Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2006) [89](#), [90](#)
- [110] Rubio, J.C., Serrat, J., Lopez, A.M., Paragios, N.: Unsupervised cosegmentation through region matching. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) [89](#), [91](#), [99](#)
- [111] Rudin, L.I., Osher, S.J., Fatemi, E.: Nonlinear total variation based noise removal algorithms. *Journal of Physics D* 60(1–4), 259–268 (1992) [158](#)
- [112] Schlesinger, D., Flach, B.: Transforming an arbitrary min-sum problem into a binary one. Tech. rep., Dresden University of Technology (2006) [39](#)
- [113] Shekhovtsov, A., Garcia-Arteaga, J., Werner, T.: A discrete search method for multi-modal non-rigid image registration. In: CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment (2008) [121](#)
- [114] Shekhovtsov, A., Kovtun, I., Hlavac, V.: Efficient MRF deformation model for non-rigid image matching. *Computer Vision and Image Understanding* 112, 91–99 (2008) [121](#)
- [115] Souiaï, M., Nieuwenhuis, C., Strelakovski, E., Cremers, D.: Convex optimization for scene understanding. In: ICCV Workshop on Graphical Models for Scene Understanding (2013)

- [116] Souiai, M., Strekalovskiy, E., Nieuwenhuis, C., Cremers, D.: A co-occurrence prior for continuous multi-label optimization. In: International Conference on Energy Minimization Methods for Computer Vision and Pattern Recognition (EMMCVPR) (2013)
- [117] Sroubek, F., Cristobal, G., Flusser, J.: A unified approach to superresolution and multichannel blind deconvolution. *IEEE Transactions on Image Processing* 16(9), 2322–2332 (2007) 109
- [118] Strang, G.: Maximal flow through a domain. *Mathematical Programming* 26(2), 123–143 (1983) 39
- [119] Strekalovskiy, E., Chambolle, A., Cremers, D.: A convex representation for the vectorial Mumford-Shah functional. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 8, 199
- [120] Strekalovskiy, E., Chambolle, A., Cremers, D.: Convex relaxation of vectorial problems with coupled regularization. *SIAM Journal on Imaging Sciences* 7(1), 294–336 (2014) 8, 155
- [121] Strekalovskiy, E., Cremers, D.: Generalized ordering constraints for multilabel optimization. In: IEEE International Conference on Computer Vision (ICCV) (2011) 8, 67, 72, 76
- [122] Strekalovskiy, E., Cremers, D.: Total variation for cyclic structures: Convex relaxation and efficient minimization. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2011) 132, 133
- [123] Strekalovskiy, E., Cremers, D.: Real-time minimization of the piecewise smooth Mumford-Shah functional. In: European Conference on Computer Vision (ECCV) (2014) 218
- [124] Strekalovskiy, E., Goldluecke, B., Cremers, D.: Tight convex relaxations for vector-valued labeling problems. In: IEEE International Conference on Computer Vision (ICCV) (2011) 8, 119, 122, 149
- [125] Strekalovskiy, E., Nieuwenhuis, C., Cremers, D.: Nonmetric priors for continuous multilabel optimization. In: European Conference on Computer Vision (ECCV) (2012) 8, 53, 58
- [126] Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2010) 185, 186
- [127] Szeliski, R.: Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision* 5(3), 271–301 (1990) 1
- [128] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields. In: European Conference on Computer Vision (ECCV) (2006) 141



- [129] Tappen, M., Freeman, W.: Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In: IEEE International Conference on Computer Vision (ICCV) (2003) 141
- [130] Unger, M., Pock, T., Werlberger, M., Bischof, H.: A convex approach for variational super-resolution. In: Annual Symposium of the German Association for Pattern Recognition (DAGM) (2010) 109
- [131] Veksler, O.: Star shape prior for graph-cut image segmentation. In: European Conference on Computer Vision (ECCV) (2008) 9, 67, 84
- [132] Vicente, S., Rother, C., Kolmogorov, V.: Object cosegmentation. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2011) 89, 90, 98, 99
- [133] Villani, C.: Topics in optimal transportation, Graduate Studies in Mathematics, vol. 58. American Mathematical Society, Providence, RI (2003) 160
- [134] Wainwright, M., Jaakkola, T., Willsky, A.: Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory* 51(11), 3697–3717 (2005) 2, 39, 141
- [135] Weinmann, A., Demaret, L., Storath, M.: Total variation regularization for manifold-valued data. *SIAM Journal on Imaging Sciences* 7(4), 2226–2257 (2014) 218
- [136] Welk, M., Theis, D., Brox, T., Weickert, J.: PDE-based deconvolution with forward-backward diffusivities and diffusion tensors. In: International Conference on Scale Space and PDE Methods in Computer Vision, Lecture Notes in Computer Science, vol. 3459, pp. 585–597. Springer-Verlag (2005) 109
- [137] Werlberger, M., Unger, M., Pock, T., Bischof, H.: Efficient minimization of the non-local potts model. In: International Conference on Scale Space and Variational Methods in Computer Vision (SSVM) (2011) 109
- [138] Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(9), 1744–1757 (2012) 185, 186
- [139] Yezzi, A., Zöllei, L., Kapur, T.: A variational framework to integrate segmentation and registration through active contours. *Medical Image Analysis* 7, 171–185 (2003) 90
- [140] Zach, C., Gallup, D., Frahm, J., Niethammer, M.: Fast global labeling for real-time stereo using multiple plane sweeps. In: Workshop on Vision, Modeling and Visualization (2009) 2, 37, 40, 43, 91, 200
- [141] Zach, C., Hane, C., Pollefeys, M.: What is optimized in convex relaxations for multilabel problems: Connecting discrete and continuously inspired

- map inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(1), 157–170 (2014) 40, 51, 58
- [142] Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime  $TV - L^1$  optical flow. In: Annual Symposium of the German Association for Pattern Recognition (DAGM) (2007) 131, 147
- [143] Zach, C., Hane, C., Pollefeys, M.: What is optimized in tight convex relaxations for multi-label problems? In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 40, 51, 58, 143, 144
- [144] Zheng, S., Xu, L., Jia, J.: Forward motion deblurring. In: IEEE International Conference on Computer Vision (ICCV) (2013) 218