Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Medientechnik

Data Compression for Collaborative Visual SLAM

Dominik Van Opdenbosch, M.Sc. (hons)

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Gerhard Kramer
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Eckehard Steinbach
	2.	apl. Prof. Dr.-Ing. Walter Stechele

Die Dissertation wurde am 18.04.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 13.07.2019 angenommen.

Abstract

State-of-the-art visual Simultaneous Localization and Mapping (SLAM) systems achieve a level of accuracy that makes them well suited as alternatives to methods relying on expensive hardware, such as laser scanners. Using low-complexity local binary visual features enables visual SLAM systems to run in real-time on commodity hardware using the input of inexpensive visual sensors. Incorporating features such as relocalization and loop closure detection allows for the deployment in large-scale scenarios ranging from buildings up to city-scale environments and beyond.

However, today's visual SLAM systems are not yet able to run in real-time and for an extended period on a mobile device due to their computational complexity, the resultant battery drain, and the memory consumption when handling large visual maps. This prevents visual SLAM systems to be deployed on robots and mobile devices operating outside a confined lab environment. To provide accurate location and map information based on visual SLAM, alternative architectures apart from local processing on the device have to be investigated.

A promising approach uses cloud computing to outsource the computationally demanding parts of a visual SLAM system. To allow low-latency and real-time capabilities, an edge cloud can be used that moves the computing power close to the location where it is needed. In addition, this allows to process, merge, and store visual information about the surrounding area simultaneously acquired from multiple nearby clients. This approach requires a data-efficient exchange of visual information between client and server, a collaborative mapping scheme, and the possibility to store and exchange visual maps with neighboring computing edges.

To address these issues, first, this thesis investigates an Analyze-Then-Compress architecture. In this scheme, the visual information in the form of local binary features is extracted at the client, compressed exploiting temporal and spatial redundancies, and sent to a server capable of generating a visual SLAM map. Second, a collaborative mapping scheme is added to fuse multiple maps from different agents into a joint representation by detecting overlap based on visual similarity. Third, the issue of exchanging static map information obtained by a visual SLAM system is investigated. The map size is reduced in a lossless fashion by exploiting the inherent structure of the map to encode the contained visual information differentially. Adding a lossy map compression method that discards potentially uninformative points further reduces the required map size. Experimental results verify the feasibility and applicability of the proposed methods.

Kurzfassung

Moderne Systeme zur gleichzeitigen Lokalisierung und Kartierung (engl. Simultaneous Localization and Mapping - SLAM) erreichen eine Genauigkeit, welche ihnen erlaubt, Verfahren basierend auf teurer Lasersensorik abzulösen. Durch die Verwendung von lokalen binären visuellen Bildmerkmalen ist es mittlerweile praktikabel, visuelle SLAM Systeme in Echtzeit auf handelsüblicher Hardware laufen zu lassen. Durch die Verwendung von Komponenten zur Relokalisierung und Schleifenschließung ist es möglich, diese Verfahren in Szenarien, wie beispielsweise in Gebäuden, Städten und darüber hinaus anzuwenden.

Trotzdem sind heutige visuelle SLAM Systeme nicht in der Lage in Echtzeit und für eine längere Zeit auf einem mobilen Gerät zu laufen. Gründe hierfür sind die Komplexität, der resultierende Stromverbrauch und der benötigte Speicherplatz zur Archivierung der Karten. Dies verhindert die Verbreitung von solchen Systemen auf Roboterplattformen und mobilen Endgeräten außerhalb einer Laborumgebung. Um dennoch entsprechende Systeme realisieren zu können, werden Alternativen zur lokalen Verarbeitung auf dem Gerät untersucht.

Ein vielversprechender Ansatz verwendet hierbei Cloud-Computing um anspruchsvolle Berechnungen des Systems auszulagern. Durch die Verwendung einer Edge-Cloud rückt die Verarbeitung näher an den Ort, an dem die Ergebnisse benötigt werden und verringert dadurch die Latenzzeit für Echtzeitanwendungen. Zusätzlich erlaubt diese Architektur das gleichzeitige Sammeln, Verarbeiten und Speichern visueller Informationen von mehreren Endgeräten. Dieser Ansatz benötigt einen effizienten Austausch von visuellen Merkmalen zwischen dem Endgerät und dem Server, einen kollaborativen Ansatz zur Kartierung und die Möglichkeit, Karten zu speichern und mit benachbarten Cloud-Knoten auszutauschen.

Um diese Herausforderungen anzugehen, wird zuerst eine Systemarchitektur untersucht, bei der die visuelle Information in Form von lokalen binären Merkmalen am Endgerät extrahiert wird. Diese Information wird, unter Ausnutzung zeitlicher und räumlicher Zusammenhänge, komprimiert und zu einem Server geschickt, welcher daraus eine visuelle Karte generiert. Zweitens wird das System um einen Ansatz zum Zusammenführen mehrerer Karten in eine gemeinsame Repräsentation erweitert. Als Drittes beschäftigt sich die Arbeit mit dem Austausch von statischen Karten. Die Kartengröße wird dabei verlustlos unter Ausnutzung der Kartenstruktur komprimiert, indem die enthaltene visuelle Information differentiell abgespeichert wird. Durch eine zusätzliche verlustbehaftete Komprimierung, welche potentiell uninformative Kartenpunkte entfernt, kann die Kartengröße weiter reduziert werden. Experimentelle Ergebnisse verifizieren hierbei jeweils die Machbarkeit der vorgeschlagenen Methoden.

Acknowledgements

The work presented in this dissertation was carried out as a member of the academic staff at the Chair of Media Technology (LMT) at the Technical University of Munich (TUM).

First of all, I would like to express my gratitude to my supervisor Prof. Dr.-Ing. Eckehard Steinbach for giving me the opportunity to be part of the "Navvis II" and "VaMEx-VIPE" projects. I want to thank Prof. Dr.-Ing. Steinbach for his continuous support and for providing me numerous opportunities in research and beyond. This includes the supervision of a summer school course and being a guest lecturer at Tongji University in Shanghai for the CDHK.

Furthermore, I would like to thank the second examiner Prof. Dr.-Ing. Walter Stechele to review this thesis and Prof. Dr. sc. tech. Gerhard Kramer for chairing the examination committee.

I would also like to thank Dr.-Ing. Georg Schroth, Robert Huitl, and Sebastian Hilsenbeck. Besides being great colleagues, they initially raised my interest in the topic and supported me during my first steps in academia. On the same note, I would like to thank Dr.-Ing. Dmytro Bobkov and Adrian Garcea who accompanied me through my time at the chair.

A special thanks goes out to all my colleagues and friends for their support, the many fruitful discussions, and activities in and beyond academia. First and foremost, I would like to thank Dr.-Ing. Tamay Aykut and Dr.-Ing. Christoph Bachhuber for their valuable feedback on this thesis. I also want to thank Dr.-Ing. Nicolas Alt, Stefan Lochbrunner, Dr.-Ing. Clemens Schuwerk, Matti Strese, and Jingyi Xu for creating such an enjoyable atmosphere. I also would like to thank Dr.-Ing. Anas Al-Nuaimi, Martin Oelsch, and all other colleagues and students that accompanied me on this journey. I would like to express my sincere thanks to Dr.-Ing. Martin Maier, Martina Schmidt, Marta Giunta, and Simon Krapf for their administrative support.

Last but not least, I would also like to thank my family. In particular, my parents for their support over the years and my grandparents, who are, unfortunately, not able to see the result anymore.

Dominik Van Opdenbosch *Munich, April, 2019*

Contents

Notation

Abbreviations

Abbreviation	Description	Definition
AGAST	Adaptive and Generic Accelerated Segment Test	page 12
ATC	Analyze-then-Compress	page 21
ATE	Absolute Trajectory Error	page 65
BRIEF	Binary Robust Independent Elementary Features	page 13
BRISK	Binary Robust Invariant Scalable Keypoints	page 14
CDVA	Compact Descriptors for Video Analysis	page 24
CDVS	Compact Descriptors for Visual Search	page 23
CNN	Convolutional Neural Networks	page 12
CTA	Compress-then-Analyze	page 21
DSO	Direct Sparse Odometry	page 17
DTAM	Dense Tracking and Mapping	page 16
FAST	Features from Accelerated Segment Test	page 12
FREAK	Fast Retina Keypoints	page 14
HEP	Homography Estimation Precision	page 51
IMU	Interial Measurement Unit	page 18
LBS	Location-based Service	page 2
LIFT	Learned Invariant Feature Transform	page 14
LSD	Large-Scale Direct	page 17
MAV	Micro Aerial Vehicle	page 1
MPEG	Moving Picture Experts Group	page 23
MST	Minimum Spanning Tree	page 92
ORB	Oriented FAST and Rotated BRIEF	page 14
PnP	Perspective-n-Point	page 20
PTAM	Parallel Tracking and Mapping	page 17
RANSAC	Random Sample Consensus	page 10
SFM	Structure From Motion	page 16
SIFT	Scale Invariant Feature Transform	page 11
SLAM	Simultaneous Localization and Mapping	page 1
SSD	Sum of Squared Difference	page 12
SURF	Speeded Up Robust Features	page 11
SVO	Fast Semi-Direct Monocular Visual Odometry	page 17
VLAD	Vector of Locally Aggregated Descriptors	page 15
VO	Visual Odometry	page 1

Scalars, vectors and matrices

Scalars are denoted by letters in italic type. *Vectors* are denoted by lower case letters in boldface type. *Matrices* are denoted by upper case letters in boldface type.

x	scalar
$\hat{x}$	quantized scalar
$\mathbf{x}$	vector
$\mathbf{X}$	matrix

Subscripts

i	feature/observation index
j	descriptor entry index
n	image index
u	map point index

Superscripts

m	coding mode
I	intra coding mode
P	inter coding mode
S	skip mode
M	multi-view coding mode
D	depth coding
T	minimum spanning tree coding mode

Symbols

C	transmission capacity
$\mathbf{d}$	descriptor
d_j	descriptor element j
H	entropy
R	measured bits
N_d	dimensionality of a descriptor
N_D	number of bits for depth value coding
N_f	number of features per frame
N_r	number of reference frames
N_v	visual vocabulary size
$N_{\hat{s}}$	number of quantization bins for the detector response
$N_{\hat{\theta}}$	number of quantization bins for the keypoint orientation
N_k	number of keyframes
N_p	number of map points
N_h	frame height
N_w	frame width
p_0^m	probability of a string entry being zero for coding mode m
σ	keypoint scale-space level
θ	keypoint orientation
x, y	keypoint x and y position
x^s, y^s	keypoint x and y position in the scale-space level

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) is one of the most fundamental problems in robotics. One of the major driving forces for the development of SLAM solutions has been the need for reliable localization and perception approaches in applications, such as autonomous robotic exploration missions and self-driving cars. Although being subject of research for several decades, SLAM has received a considerable boost with the advent of inexpensive sensors. In particular, visual cues have become one of the primary input modalities for various SLAM systems. Due to the mass production of affordable, versatile, and compact visual sensors, initially targeted at the consumer market for smartphones, it is now possible to equip robots and micro aerial vehicles (MAV) with lightweight optical sensors. These robotic systems using visual SLAM are capable of exploring the environment and creating a map while performing localization at the same time. SLAM is also often referred to as the *kidnapped robot problem* [13], as the robot has to deduce its position and surroundings by itself with no additional infrastructure at hand. In outdoor scenarios, the satellite-based Global Positioning System (GPS) can provide feedback on the absolute location but does not provide information about the structure of the environment, which is essential to perform path planning and facilitate collision avoidance. However, many possible application scenarios, such as urban canyons, indoor spaces, or even other planets, are considered as GPS-denied environments and used to require additional costly infrastructure for localization. The list of available solutions ranges from rather low-cost Wi-Fi-based approaches, which allow for room-level accuracy, up to cost-intensive radar-based methods providing centimeter-accuracy. However, many approaches provide no information about obstacles.

Thanks to almost four decades of research on estimating the motion from visual information [14], [15], there exists a broad variety of algorithms ranging from visual odometry (VO), which estimates the motion between frames, up to fully-fledged visual SLAM systems, which build a globally consistent map of the environment [15], [16]. In addition to inexpensive visual sensors, three essential aspects have fueled the usage of visual SLAM in the past: First, recent visual SLAM systems have achieved a high level of accuracy in the sub-centimeter range. Second, state-of-the-art frameworks achieve real-time capabilities on desktop computers. Third, they can map and handle large-scale environments efficiently. With all these requirements fulfilled, current research focuses on collaborative aspects in multi-agent scenarios and distributed SLAM systems.

In *collaborative* scenarios, multiple agents equipped with visual sensors jointly explore an unknown environment. An exploration team can consist of different types of agents such as aerial vehicles, humanoid robots, rovers, or mobile devices carried by a human. Using multiple agents allows to efficiently explore unknown environments by splitting the target area into individual mapping tasks with each one being assigned to a team member. In the case of a heterogeneous agent team, the resulting map can capture the environment from different perspectives, such as ground-based views taken from a mobile device or in the form of aerial views taken by a drone. Besides, multi-agent teams can be used to provide certain fault tolerance in critical robotic exploration missions allowing to take over the task of a failed agent by another team member. Moreover, the different maps generated by the individual team members can be exchanged, which allows for creating a global map and enables other agents to benefit from previously mapped areas as prior knowledge. The map information can be exchanged with other team members or further processed at a centralized instance depending on the architecture of the *distributed* SLAM system.

There is a myriad of application scenarios for multi-agent teams. A large-scale aerial survey allows for crop monitoring, whereas ground-based vehicles could augment the data with information about the soil conditions. Archiving the data acquired from observing areas over a longer period allows documenting its evolution and studying the morphology of objects and nature over time. Rapid exploration of a disaster area enables first responders to obtain an overview of the situation and take appropriate action. Possible scenarios include, but are not limited to, natural disasters, such as earthquakes or floods, but also human-made disasters, such as fires. Providing an initial map of the environment allows the rescue forces to identify regions where survivors are likely to be found quickly.

Besides the application in robotics, collaborative visual exploration can also aid personal navigation. While we spend most of our time indoors, where no GPS signal is available, it is indeed beneficial to have a position estimate available on our smartphone to enable location-based services (LBS), such as navigation instructions. Similar to multi-agent robot teams, a crowd-based approach could be used to gather information from multiple users using smartphone cameras. A globally consistent representation of a building can be obtained by fusing the partial information into a common map. This map can be enriched with further information such as semantic information in the form of labels for different rooms and objects. In return, these maps can be used to enable localization and other LBS for new users. The maps can be kept up-to-date by including further crowd-based visual information. Local maps of the environment can be captured by multiple users, transmitted to a server, and successively combined to obtain an up-to-date digital twin of the building.

Another application scenario that is gaining attention is the domain of virtual reality applications, where inside-out tracking enables head-mounted displays to get rid of external sensors to track the device position. Until now, external infrared sensors are often employed for determining the position. A significant drawback is the limited space that the external sensors can cover. Using built-in wide-angle cameras allows deducing the device position by observing the environment using visual SLAM. This allows overcoming the restriction to single rooms imposed by the use of the external sensors.

However, there are still some problems and room for improvement, especially when considering practical application scenarios. While visual SLAM algorithms reach maturity in terms of accuracy, the computational complexity, and hence the battery life is still an impediment when it comes to mobile and embedded applications. For example, exploring larger areas with a micro aerial vehicle requires returning to the base station and charging the batteries several times, resulting in considerable delay. Also, embedded computers are often restricted regarding their onboard memory, thus prohibiting large maps. Therefore alternatives to running the visual SLAM on the client have to be investigated.

When considering a typical indoor robotic application, the fundamental appearance of a building interior is rather static. While there might be several movable objects, most parts of the infrastructure, such as the floor, the ceiling, and heavy furniture remain located at the same position with the same visual outline during the typical operation time of a robot. This information can be collected after the construction of the site during an initial mapping. This information can be made available as prior knowledge in the form of a visual SLAM map to improve the task performance of other robots. Moreover, this map can be updated with the robots adding new visual information during their operating hours. This updated map can then again be redistributed to other robots operating in the same environment.

These are only two examples that serve as the initial motivation to further investigate the efficient data exchange and the collaborative aspect in the context of visual SLAM applications. In the following, the key contributions of this thesis are listed.

1.1 Major contributions

The focus of this thesis is on the efficient exchange of information in the context of collaborative feature-based visual SLAM systems. The necessary information can mainly be exchanged at three different representation levels: First, the visual information can be transferred between team members in the form of image information, e.g., as a video stream. Second, the information can be transmitted in the form of an image abstraction, e.g., a sequence of local features. Third, local maps created by the visual SLAM system can be shared among the team members. As the first two approaches require a visual SLAM system running in a remote environment to build a map using the visual information, these approaches are referred to as *remote visual SLAM* throughout the text. When running a visual SLAM system in a remote environment, such as cloud-based infrastructure, a natural extension is to use the information collected by multiple agents in a collaborative architecture. The concepts, application scenarios, and the key challenges of visual information exchange, remote visual SLAM, collaborative visual SLAM, and exchanging map information addressed in this thesis are stated in the following.

Visual feature exchange: Providing accurate results for any demanding computer vision task on energy-constrained devices is possible by outsourcing the task to a powerful server. In this thesis, an approach where only the local visual features are extracted, compressed, and transmitted to the server is investigated. To this end, an efficient compression framework for local binary features and global image signatures is introduced. In addition, temporal and spatial correlations are exploited. Approaches for information selection complete the framework.

Remote visual SLAM: In order to allow visual SLAM to run on energy-restricted devices, the computationally demanding parts are outsourced to a more powerful server, possibly located in an edge cloud. To this end, a system architecture allowing for monocular remote visual SLAM is proposed incorporating the aforementioned feature compression framework. This system architecture is then extended to support stereo or depth sensor setups in order to provide metric scale information for the visual SLAM maps.

Collaborative visual SLAM: In this work, a centralized architecture is implemented to collect visual information from multiple agents at a central instance. This scheme allows joining the individual maps from different agents into a more comprehensive global map based on overlap detection using visual similarities. Further applications employing the resulting map, such as the orchestration of an exploration team, can be implemented on top.

Map exchange: This thesis introduces a lossless approach for compressing and storing the visual information of a static visual SLAM map by exploiting visual similarities and geometric constraints inherent in the map structure. Furthermore, a minimization problem is formulated to store only useful map points. This measure of usefulness is based on the costs in bits for storing a point in combination with a visibility metric. The proposed lossy optimization problem is tightly coupled with the compression scheme to store only information that achieves a good compression ratio and is presumably useful for a relocalization task.

This thesis contains the following contributions to address the aforementioned challenges:

1. Joint compression of local binary features and their Bag-of-Words representation.
2. Flexible rate allocation scheme for the proposed compression framework.
3. Monocular remote visual SLAM, including a selection of binary local visual features.
4. Metric scale remote visual SLAM, including depth value and stereo feature coding.
5. Collaborative remote visual SLAM using a centralized system architecture.
6. Rate-aware map compression exploiting visual dependencies in a visual SLAM map.

1.2 Thesis organization

This dissertation is structured as follows: In Chapter 2, the related work, and the relevant background are discussed. An overview of the feature coding framework is presented in Chapter 3. The concept for a joint compression of local binary features and their corresponding Bag-of-Words representation alongside a method for rate allocation is introduced in Chapter 4. In Chapter 5, a monocular remote visual SLAM system exploiting temporal redundancies between features from successive frames and a feature selection are introduced. Feature coding methods for both stereo and depth information to facilitate metric scale visual SLAM are presented in Chapter 6. An extension of the remote visual SLAM framework to a centralized collaborative approach is detailed in Chapter 7. A method for efficient map compression and sparsification is presented in Chapter 8. The dissertation is concluded in Chapter 9. Parts of this thesis have been published in [1]–[5].

Chapter 2

Background and related work

This chapter presents an overview of the relevant background and the related work for this thesis. Section 2.1 provides the prerequisites on data compression. Section 2.2 introduces relevant methods for visual content description. Concepts for distributed, collaborative visual SLAM, and map compression are discussed in Section 2.3. Related work addressing the compression of the visual information and especially local visual features is reviewed in Section 2.4. Cloud-based architectures are discussed in Section 2.5 and employed datasets are introduced in Section 2.6. For an in-depth discussion of the individual topics, the interested reader is referred to the provided references.

2.1 Data compression

Information theory lays the foundation for measuring information contained in symbols produced by a source. In his seminal work on information theory [17], Claude E. Shannon defined the *entropy* H as an appropriate measure for the information. To quantify the entropy, a discrete, memory-less source is modeled by a random variable X with an associated alphabet $A_x = \{\alpha_1, \alpha_2, \ldots, \alpha_{N_x}\}$ containing a total number of N_x possible symbols. An occurrence probability p_q for each symbol $\alpha_q \in A_x$ is given by the associated probability mass function $f_X(\alpha_q) = P(X = \alpha_q) = p_q$. Shannon's requirements regarding the definition of the entropy were:

1. H has to be defined continuously in p_q.
2. If all symbols have the same probability p_q, then H should monotonically increase with the number of symbols produced by the source.
3. If a decision is reformulated into two successive decisions, the original H should be the weighted sum of the individual values of H.

With all these requirements, Shannon formulated the entropy as the weighted sum of the self-information $I(\alpha_q) = -\log_2(p_q)$ over all the symbols produced by a discrete random variable X as:

$$H(X) = -\sum_{q=1}^{N_x} p_q \log_2(p_q). \tag{2.1}$$

When using the logarithm to base 2, the unit is defined as bits. The entropy hereby defines the theoretical lower bound for the number of bits that are needed to transmit a symbol produced by the source. This theoretical limit can be approached using different methods, where the most important ones used in this work are introduced in the following. Considering the notation, the entropy is denoted with H, whereas the experimentally measured rate is denoted by R with equal subscript and superscripts. For the sake of readability, the dependency of $H(X)$ from the random variable X is dropped.

2.1.1 Fixed-length coding

A straightforward way to encode symbols from a source with uniform probability distribution is *fixed-length coding*. The theoretical number of bits necessary to transmit a symbol produced by a random variable with uniform probability $p_q = \frac{1}{N_x}$ is given by

$$H = -\sum_{q=1}^{N_x} p_q \cdot \log_2(p_q) = -N_x \cdot \frac{1}{N_x} \cdot \log_2(\frac{1}{N_x}) = \log_2(N_x). \tag{2.2}$$

However, most sources in practical applications produce non-uniform probability distributions. Using a fixed-length code implies that only an integer number of bits can be transmitted resulting in a loss of $R - H = \lceil \log_2(N_x) \rceil - \log_2(N_x)$ bits per symbol, as N_x is usually not presentable as $N_x = 2^i$ with i being a non-negative integer number. The difference is often referred to as the code redundancy. The input symbol is mapped onto a binary code of fixed length at the encoder, transmitted, and mapped back to its original symbol at the decoder using a common codeword table. The key advantage of fixed-length coding is that it can be implemented with very low complexity resulting in a fast encoding and decoding process.

2.1.2 Variable-length coding

Arbitrary probability distributions of the source symbols can be exploited by *variable-length coding*. The process of transforming symbols produced by a random variable into a sequence of bits close to the theoretical limit given by the entropy is also referred to as *entropy coding*. Intuitively, symbols with higher probability should result in a shorter representation, whereas very unlikely symbols should result in a longer representation. As the short symbols are used more often, the average codeword length (i.e., the average number of bits) is reduced. The number of bits for encoding a symbol should behave proportionally to the self-information of the symbol.

A prominent example is *Huffman coding* [18]. The algorithm uses a tree structure to assign the shortest bit representation to the likeliest symbol. An important property of Huffman and many other codes is that no bit representation of a symbol occurs at the beginning of a representation of another symbol (prefix-free code). A drawback of Huffman coding is that it only performs optimally if all probabilities are powers of two as $p_q = 2^{-i}$ with i being a non-negative integer number. In addition, Huffman codes do not perform very well for symbols with a self-information smaller than 1 bit (i.e., $p_q > 0.5$), as it requires at least one bit

for the most probable symbol. The aforementioned drawbacks can be overcome by grouping several symbols in a block and encode them jointly.

Another approach of entropy coding is *arithmetic coding* [19]. Here, a message is represented as a real-valued number between zero and one. The basic idea is to split the interval between zero and one into sub-intervals proportional to the probabilities of the source symbols. The symbol arriving at the encoder defines, which sub-interval is considered next. Now, this interval is treated as the current interval and is split again into different sub-intervals proportional to the source symbol probabilities. By transmitting a real-valued number falling into this interval, the decoder can uniquely reconstruct the original sequence of symbols. Hence, arithmetic coding is theoretically able to encode a message into a real-valued number with arbitrary precision between zero and one. This approach is usually more efficient than Huffman coding. In practical applications, however, real-valued numbers can only be represented with a limited precision, which requires renormalization techniques to be applied. Also, using floating-point arithmetic adds to the complexity of the algorithm resulting in longer encoding and decoding times compared to fixed-length and Huffman coding.

Regardless of the entropy coding method, the probability tables have to be either fixed beforehand or measured during the encoding process using *probability adaption*. The latter determines the individual probabilities based on the already encoded symbols. The process is started with the same probability tables at the encoder and the decoder. Typically, uniformly distributed probabilities are assumed if no further prior knowledge is available. Then, the probabilities can be simultaneously adapted to the underlying data at both the encoder and the decoder based on the observed symbols. However, this technique comes with two drawbacks. First, a single transmission error renders the remaining data undecodable as the probability tables are not synchronized between the encoder and the decoder. Second, this adds computational overhead as the probabilities have to be updated regularly to benefit from the adapted probability estimates.

To complete the list, a recent promising development in entropy coding is the approach of *asymmetric numeral systems* introduced by Duda et al. [20]. In general, to add a symbol $\alpha_q \in \{0, 1\}$ from a binary source with uniform distribution to a natural number x, the bit representation is shifted and the bit is added in the least significant position resulting in the new number $x' = 2x + \alpha_q$. The information added is $\log_2(x) + \log_2(\frac{1}{0.5}) = \log_2(\frac{x}{0.5})$. The asymmetric numeral systems idea is based on the assumption that the probabilities are not uniformly distributed. In consequence, another rule has to be found to calculate x' based on x and α_q, such that the added information is approximately $\log_2(x) + \log_2(\frac{1}{p_q}) = \log_2(\frac{x}{p_q})$, resulting in $x' \approx \frac{x}{p_q}$. The authors claim to consistently outperform Huffman coding, as well as arithmetic coding in terms of speed and coding efficiency.

Throughout this thesis, fixed-length coding is usually employed for properties, where no additional information about the probability distribution is exploited, and a uniform distribution is assumed, if not noted otherwise. For all symbols with different probability distributions, arithmetic coding without probability adaption is used for encoding. However, both approaches can be easily replaced by any other coding method.

2.2 Visual content description

A fundamental building block of many computer vision algorithms is the ability to compare and associate visually similar parts of images. Although there exist algorithms that employ the raw pixel intensities, most computer vision algorithms use image abstractions, such as visual features. In content-based image retrieval, *global features* are often used to describe the visual outline of an image. However, a global representation cannot capture local patterns or relations among different image parts. Instead, *local features* are used to describe salient parts of the images individually. Generally speaking, having a compact representation of well-localized interest points in an image allows matching points across multiple images depicting the same object in the scene. This forms the basis for many applications in computer vision ranging from geometric verification in the context of content-based image retrieval up to 3D reconstruction techniques and visual SLAM.

2.2.1 Local visual features

Extracting local features in an image can usually be described as a two-stage process. First, salient parts of the image have to be identified. The outcome of a local feature detector is a keypoint that describes the location, spatial extent, and orientation of a point of interest. After local feature detection, the keypoints are fed into the local feature descriptor, which tries to translate the visual information contained in a local patch around the keypoint into a fixed-length representation. In order to provide good matching capabilities, both the detector and descriptor have to fulfill certain requirements [21]:

Brightness:	Invariance to lighting due to changes in the environment or camera settings, such as exposure times.
Scale:	Invariance to changes in scale due to the movement of the camera or the objects in the scene.
Locality:	The keypoint should be localized as accurately as possible, and the descriptor should only consider a small patch around the keypoint.
Rotation:	Invariance to in-plane rotations.
Viewpoint:	Stable detection and description in the presence of substantial viewpoint changes.
Quantity:	A sufficient amount of local features should be extracted in order to allow for the use of random sample consensus (RANSAC) schemes to increase the robustness.
Distinctiveness:	Similar patches should result in similar feature descriptors. Dissimilar patches should be placed further apart in the feature space.
Efficiency:	Feature extraction should be real-time capable.
Storage:	Memory footprint per feature should be as low as possible in order to allow for the storage of large-scale visual information.

In the following, the most important fundamental concepts of local feature detectors and descriptors are introduced. In particular, binary local feature detectors and descriptors, which are used throughout this thesis because of their small memory footprint and real-time capabilities, are highlighted in the following. For more details on the evaluation and comparison of different algorithms, the reader is referred to the related work [21]–[27].

2.2.1.1 Local feature detectors

Traditional *hand-crafted* feature *detectors* can be roughly categorized into three categories: edge, blob, and corner detection. There exists a more recent class of feature detectors, which are *learned features* trained beforehand on a set of images to produce keypoint locations using machine learning approaches.

Edge detectors: The first category is the class of *edge detectors*, which are used to detect mostly discontinuities in surfaces, boundaries, or changing material properties of objects observable within an image. While edge detectors, such as Sobbel-Feldman [28] and Canny [29], have been around for more than five decades and are well suited for applications such as identifying boundaries for image segmentation, the keypoints are not well localized in terms of position along the edge tangential to the gradient direction.

Blob detectors: The representatives of the second category are the *blob detection* algorithms, where the goal is to identify image regions, or more specifically interest points, describing regions with constant brightness properties. One of the most prominent local features is probably the Scale Invariant Feature Transform (SIFT) [30], [31], which is a combination of both a feature detector and descriptor. The detection stage of SIFT achieves the desired property of scale invariance by using a *scale-space* representation, namely the Difference of Gaussians (DoG). The difference of Gaussians can be interpreted as an approximation to applying the Laplacian operation to a Gaussian filtered image [32]. The Laplacian operator is commonly used to create strong responses for blobs of a typical radius associated with the currently considered scale-space level. In SIFT, the locality is achieved by calculating sub-pixel accurate positions using a three-dimensional quadratic function fitted to the DoG scale-space. Subsequently, features detected at edges are rejected as they tend to be poorly localized. Finally, the dominant orientation is estimated from a gradient histogram around the keypoint. The quantity of local features is usually dependent on a detection threshold, which can be adjusted for the desired purpose and depends on the image properties, such as the contrast. Although the SIFT detector provides accurate keypoints, it suffers from the required computation time. This issue was addressed by the feature detection stage of the Speeded-Up Robust Features (SURF) [33]. It employs the Fast-Hessian detector, which uses the determinant of the Hessian matrix to determine the position and scale of a keypoint. The entries of the Hessian matrix are obtained by a convolution of the Gaussian second-order derivatives with the image at a possible keypoint position. In order to speed up the convolution, box filters are used in combination with integral images to approximate the second-order Gaussian derivative calculation.

Corner detectors: The third category is the class of *corner detectors*, where preferably only points located at sharp corners are detected. Many corner detection algorithms are based on the property of two dominant gradient directions present at corner points. Moravec [14] proposed to use small windows to calculate the sum of squared differences (SSD) for different directions (horizontal, vertical, two diagonals) with respect to a window centered at the point of interest. If the current point is located on a planar surface with constant brightness, then the SSD values for all displacements are relatively small. If located at an edge, the SSD along the edge direction is small, and if located at an isolated corner point, the SSD in all directions is relatively high. This basic concept was extended by Harris et al. [34] by evaluating the eigenvalues of a second-moment matrix built from the partial derivatives, thus avoiding the evaluation of the shifted patches. Another example for corner detectors is the Features from Accelerated Segment Test (FAST) by Rosten et al. [35]. They proposed to evaluate pixels located on a Bresenham circle around the potential keypoint position. A corner is detected if N contiguous pixels are either darker or brighter then the central pixel by a previously defined margin. A speed-up is achievable by evaluating not all pixels on the circle but evaluate first a minimal subset of pixels that is necessary to fulfill the corner conditions. Additionally, they proposed to improve this approach by applying machine learning to the problem. They generated a decision tree that quickly identifies corners, given the pixels on the circle. An improvement that is able to adapt to the specific scene structure was proposed as Adaptive and Generic Corner Detection Based on the Accelerated Segment Test (AGAST) [36].

Learned detectors: The representatives of the fourth category are the recently proposed *learned detectors*. A distinction between algorithms that decide based on hand-crafted detectors whether a keypoint is likely to be useful and algorithms that return a new keypoint can be made. Algorithms telling the usefulness based on hand-crafted detectors include the previously mentioned machine learning parts of FAST and AGAST, intuitive properties such as the detector response [37], a random forest trained on the descriptors [38] or Convolutional Neural Networks (CNN) trained on image patches [39]. Recently, different algorithms have been proposed to directly extract keypoints from the images based on learned convolutional filters [40], piece-wise linear regression [41], and a CNN-based decision whether an image patch is centered on a good keypoint [42], [43].

2.2.1.2 Local feature descriptors

Local feature descriptors can roughly be categorized into three categories: Real-valued, binary, and learned feature descriptors. Similar to the feature detectors, the first two classes are typically *hand-crafted descriptors*, whereas the third category includes *learned descriptors*. The differentiation between real-valued and binary descriptors is typically based on the output of the algorithm. A local image patch located at a keypoint position with an appropriate scale, orientation, and size is analyzed according to the employed algorithm. The result is stored as a fixed-length vector representation describing the local neighborhood of the keypoint. The entries of this representation can be real-valued numbers or the outcome of a set of binary decisions.

Real-valued descriptors: Probably, the most famous representative of *real-valued descriptor* algorithms is the descriptor part of SIFT. It achieves brightness invariance by calculating its 128 descriptor entries based on a histogram of the gradients located around a keypoint. Using second-order statistics yields improved invariance towards changing lighting conditions compared to first-order statistics such as the raw intensity values provided by the sensor. According to experimental evaluation [31], SIFT features achieve a matching accuracy of above 50% at a 50-degree viewpoint change. Both the distinctiveness and matching performance have been validated in many experimental evaluations, such as [23], [25]. However, SIFT has two major drawbacks, which are the computational efficiency and storage requirements. The time required to detect and describe SIFT features can range up to several hundred milliseconds per image. This prohibits the usage in real-time applications where tens of images are processed per second. Regarding the storage requirements, a SIFT descriptor without keypoint information has 128 floating-point entries, which require 512 bytes and accumulate when detecting several hundred or thousand features per image. In order to address the two issues, alternative descriptors have been proposed. A famous example is the SURF descriptor part [33], which is based on similar concepts as SIFT but introduces several approximations to speed up the description process. As a result, SURF requires less computation time and stores only 64 floating-point descriptor entries.

Binary descriptors: Another category of local features are the *binary descriptors*, which are typically based on pairwise tests between smoothed pixel intensities. One of the first proposed algorithms is the Binary Robust Independent Elementary Features (BRIEF) [44] descriptor, which produces a N_d dimensional binary vector $\mathbf{d} \in \{0,1\}^{N_d}$ as local visual representation. The individual descriptor entries d_j with $j \in \{z \in \mathbb{N} \,|\, 1 \leq z \leq N_d\}$ are derived from a set of intensity comparisons between pixel intensities $I(x_j^1, y_j^1)$ and $I(x_j^2, y_j^2)$ centered around a keypoint as

$$d_j = \begin{cases} 1 & \text{if } I(x_j^1, y_j^1) < I(x_j^2, y_j^2) \\ 0 & \text{if } I(x_j^1, y_j^1) \geq I(x_j^2, y_j^2), \end{cases} \tag{2.3}$$

where x_j^1, y_j^1 and x_j^2, y_j^2 denote the pixel locations for the test pair j extracted from the image I. Gaussian smoothing is applied prior to the pixel tests to reduce the pixel noise. The set of pixel tests is obtained beforehand from sampling an isotropic Gaussian distribution around the central pixel. The approach of using pairwise pixel tests turned out to be very successful and comes with a great advantage in terms of computation speed due to its simplicity. In addition to the computational efficiency, the binary representation allows for a rapid distance calculation between feature vectors using the Hamming distance. Modern CPUs feature specific instructions that allow evaluating the Hamming distance between two vectors with hardware acceleration. Large-scale retrieval can be facilitated due to the fact that each binary descriptor (typically $N_d^{\text{BRIEF}} = 256$) requires only 32 bytes to be stored and can be efficiently retrieved by using algorithms such as Locality Sensitive Hashing [45].

While the BRIEF features laid the foundation for many efficient binary local features, they come with several drawbacks. The main limitation is the missing invariance towards scale changes and rotation, which has been addressed by its successors, such as Oriented FAST

and Rotated BRIEF (ORB) [46] features. ORB uses the FAST feature detection in combination with a scale-space representation and an orientation estimation based on the intensity centroid. The orientation can be used to rotate the pairwise pixel test pattern accordingly. In addition, the authors proposed a greedy algorithm to determine the $N_d^{\text{ORB}} = 256$ pairwise pixel tests instead of sampling. The pixel test candidates are sorted according to their mean value and their degree of correlation compared to already selected pixel tests. Ideally, the individual pixel tests feature a probability of $p_0 = p_1 = 0.5$ for the binary symbols and are uncorrelated to any other test to capture as much information about the local patch as possible.

Another representative of binary descriptors is the Binary Robust Invariant Scalable Keypoints (BRISK) algorithm [47]. BRISK employs the AGAST detector with a scale-space representation but additionally refines the scale of a keypoint between the octaves. The sampling points for the pixel tests are located on concentric circles centered at the keypoint. Long-distance sampling point pairs are used to estimate the orientation of the keypoint, and short-distance pairs are used to build the descriptors by comparing the intensities resulting in a binary descriptor of size $N_d^{\text{BRISK}} = 512$.

A slightly different approach for determining the sampling pattern is used by the Fast Retina Keypoints (FREAK) [48] algorithm. Inspired by the human visual system, the authors proposed to mimic the retinal ganglion cell distribution and their receptive fields in the pixel test sampling pattern. The result is a binary descriptor of size $N_d^{\text{FREAK}} = 512$.

Learned Descriptors: While the aforementioned approaches are hand-crafted to be invariant against the effects mentioned in the beginning, the question arises whether it is possible to learn a compact description. This *learned descriptor* representation has to ensure that matched local features are co-located in the feature space regardless of different perspectives or image perturbations. With the advent of CNNs, first approaches have been proposed to extract meaningful patch representations such as MatchNet [49], DeepCompare [50], descriptors learned using siamese networks [51], or a Learned Invariant Feature Transform (LIFT) [43]. Recently, deep-learned descriptors are specifically optimized for large-scale localization tasks [52]. An overview of this nascent technology and the evolution from hand-crafted to learned descriptors is given in [53].

2.2.1.3 Discussion

Figure 2.1 shows a comparison of the time and storage requirements of common real-valued and binary feature description algorithms. The keypoints for all descriptors were provided using the BRISK feature detector part and selecting the 1k best features according to the detector response. The experiment was conducted using the MH01 sequence of the EuRoC dataset [54] using OpenCV 3.4.1 [55] feature implementations with their default parameters on an Intel Core i7-7700 CPU running at 3.60 GHz. While this evaluation does not provide details about the matching performance, it demonstrates that SIFT and SURF features are not suitable for large-scale and real-time applications due to the computation time and storage

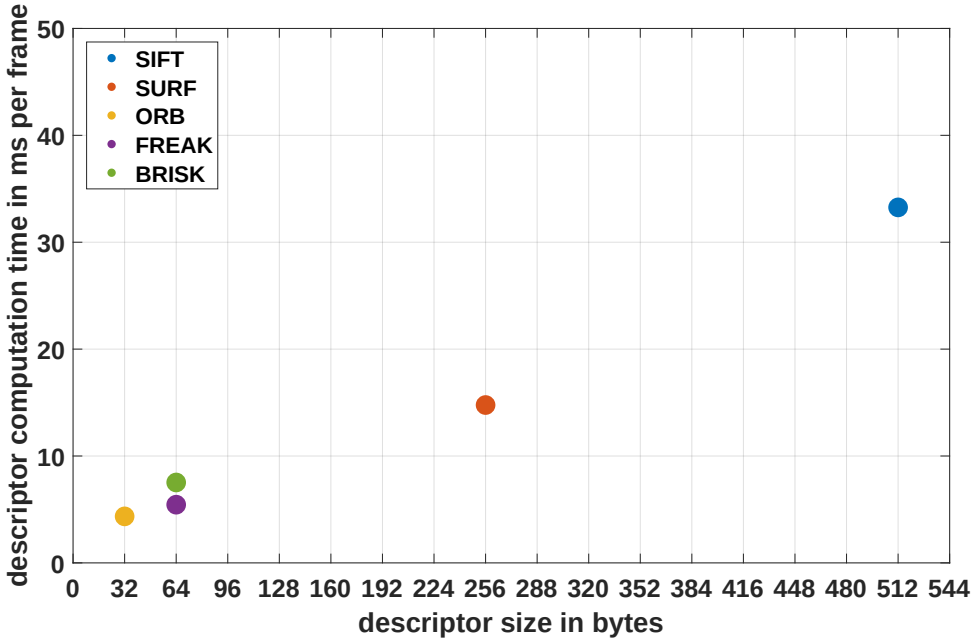


Figure 2.1: Comparison of different feature descriptors with respect to their size and the time required to compute 1k local descriptors. Keypoints were provided by the BRISK feature detector using the top features according to their detector response.

requirements. Although there are some optimized implementations available [56], [57], they further introduce approximations or restrict the algorithm to certain hardware.

2.2.2 Global visual features

Global image signatures are often used in content-based image retrieval to identify possible matches from a vast reference database quickly. Afterward, geometric verification using local features can be carried out to validate the result or to re-order the list of retrieved matches. A common approach for global image signatures is to aggregate the variable number of local features into a fixed-length representation. A prominent example is the Bag-of-Words model [58], where the feature space is first clustered using k-means in an offline procedure. The cluster centers, also denoted as *visual words*, are used as representative feature descriptors forming a *visual vocabulary*. Each feature in an image is quantized to the nearest visual word, and subsequently, the number of occurrences (*term frequency*) weighted with the importance (*inverse document frequency*) is aggregated into a histogram thus forming a global image signature. While local features provide the robustness against viewpoint changes, brightness differences, rotation, and scaling, this pooling approach provides additional robustness against partial occlusion and different fields-of-view. A fast retrieval can be facilitated using the inverted index structure. Here, only images featuring a similar set of visual words have to be traversed using a tree structure. An alternative global image representation named Vector of Locally Aggregated Descriptors (VLAD) was proposed by Jégou et al. [59], [60] and several improvements have been suggested [61], [62], [6], [7]. In contrast

to Bag-of-Words, where visual words are counted, a VLAD descriptor is formed by storing the sum of the residuals, which is the difference vector between a feature descriptor and its closest visual word. The result is a $N_v \cdot N_d$ dimensional vector, where N_v denotes the number of cluster centers (equivalent to the visual words in the context of Bag-of-Words) and N_d the dimensionality of the local feature. Subsequent dimensionality reduction techniques, such as Principal Component Analysis, allow for a compact image representation. This approach was integrated into CNN architectures as an additional layer by interpreting the output of the CNN as a set of descriptors and feeding this representation into a so-called NetVLAD layer [63]. Another VLAD-related technique is the image description using Fisher Vectors [64], where a Gaussian Mixture Model is fitted to the local descriptors. Both VLAD and Fisher vectors contain information about the distribution of the descriptor entries for the features assigned to a cluster center.

2.3 Visual SLAM

2.3.1 Visual SLAM systems

The term visual SLAM refers to the problem of simultaneously estimating the position and a map of the surroundings by using primarily the information obtained by one or multiple visual sensors. *Visual SLAM* keeps track of a globally consistent map of the environment allowing for relocalization or loop-closing to reduce drift [15]. *Visual odometry*, on the other hand, provides only a locally consistent pose estimate relative to the last frame. Another important term in this context is *Structure-from-Motion* (SFM), where a three-dimensional representation of a scene is estimated from multi-view image information. However, SFM techniques usually are not required to run in real-time, the input images must not be an ordered image sequence, and it usually scales better to larger environments such as a city or even a planet-level scale. This is partly due to the relaxed processing time and due to the possibility of using arbitrary images from different crowd-based sources, such as Flickr [65]. In this work, the focus is on real-time visual SLAM systems, which can be categorized according to four key properties [66], [67]:

Front-end:	Direct vs. indirect
Back-end:	Filtering vs. bundle adjustment
Density:	Dense vs. sparse
Sensor:	Monocular, stereo, depth, IMU, ...

Front-end: First, the visual SLAM systems can be differentiated based on the information fed into the system. This describes how visual information is extracted and associated with the visual SLAM front-end. A visual SLAM approach can either be direct or indirect. While indirect methods use an intermediate image abstraction, such as local features or templates extracted from feature patches, the direct methods employ the information contained in the raw pixel intensities by minimizing the photometric error between neighboring frames. Direct methods, such as Dense Tracking and Mapping (DTAM) [68], are also referred to as

	advantages	disadvantages
direct	+ no image abstraction	- susceptible to brightness variations - prone to image distortions
indirect	+ compact visual information	- additional feature extraction - ignores image information

Table 2.1: Advantages and disadvantages of direct and indirect SLAM approaches.

appearance-based approaches [15]. A recent representative of indirect methods is ORB-SLAM2 [69], [70], which is a SLAM framework capable of parallel tracking and mapping using local binary ORB features [46]. There are also hybrid forms using both direct and indirect methods such as the Fast Semi-Direct Monocular Visual Odometry (SVO) [71], [72] approach. The advantages and disadvantages between direct and indirect methods are summarized in Table 2.1.

Back-end: Next, a categorization can be made based on the method used to calculate the current camera pose. The estimation of the pose based on the coherences of sensor inputs detected by the front-end is often referred to as state estimation or, more generally, as the visual SLAM back-end. The predominant approaches are filtering-based methods and graph-based methods [66]. The first class uses algorithms such as Kalman Filters or Extended Kalman Filters to estimate the current position and state. Examples are MonoSLAM [73], [74] or the Multi-State Constraint Kalman Filter [75]. Visual SLAM systems belonging to the optimization-based approaches typically use a subset of frames, so-called *keyframes*, to build the map and perform bundle adjustment using nonlinear least-squares optimization on a graph structure. Instead of using global bundle adjustment, which quickly becomes infeasible with an increasing number of map points and keyframes, windowed optimization is often employed to ensure constant time visual SLAM [76]. One of the important representatives of this class introducing the concept of splitting the tracking and map building into different threads is Parallel Tracking and Mapping (PTAM) [77]. This concept decouples the graph optimization back-end from the tracking front-end, thus allowing continuous tracking without blocking the construction of the map.

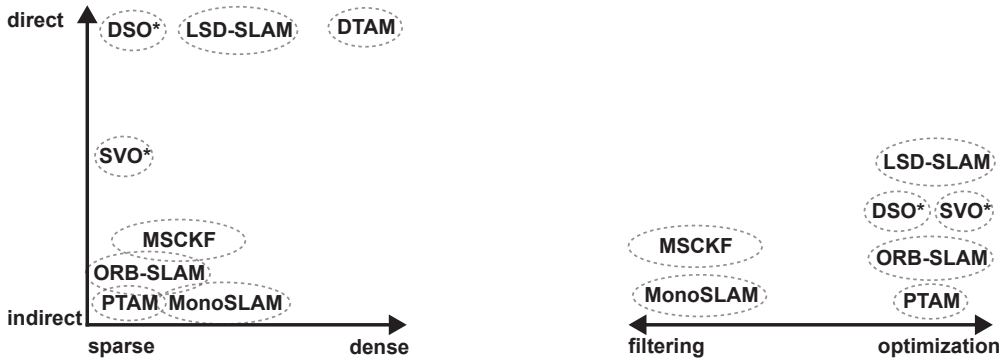
Density: Additionally, a categorization based on the reconstruction density can be made. Dense methods, such as DTAM, try to reconstruct a comprehensive 3D model of the environment based on the visual information from every pixel, whereas sparse methods only operate on a subset of image points to create 3D points. In between, there are some direct approaches such as Large-Scale Direct SLAM (LSD SLAM) [78] trying to reduce the complexity by employing only a subset of pixels, which results in a *semi-dense* representation. Some methods, such as Direct Sparse Odometry (DSO) [67], combine the direct with the sparse approach trying to use the raw image information and keep the complexity reasonable. Image points with high gradients or well localized corner points are usually preferred for determining the sparser subset of image points. The main advantages and disadvantages of methods using different densities are summarized in Table 2.2.

	advantages	disadvantages
sparse	+ computational complexity + compact map	- ignores image information
dense	+ uses all information + dense point-cloud	- complexity - map size

Table 2.2: Advantages and disadvantages of sparse and dense SLAM approaches.

Sensor: The last categorization can be made based on the used sensor suite. The input can range from monocular [67], [69], [78] and stereo [70], [79], [80] over multi-camera setups [81] to panoramic cameras [82], [83]. Some systems support additional input, such as data from an Inertial Measurement Unit (IMU) [84]–[86]. In addition, combinations with sensors such as range scanners, infrared, ultrasonic, global navigation satellite systems, or other modalities are possible.

A categorization of visual SLAM algorithms depending on the visual information and map representation, as well as the underlying back-end methods, is illustrated in Figure 2.2. In order to pick a suitable approach for a collaborative visual SLAM system, not only the accuracy in terms of estimated trajectory, but also the input to the visual SLAM system, as well as the internal map representation is of interest. Most related works concerning collaborative or distributed SLAM systems are based on both indirect and sparse visual SLAM methods. One of the main reasons is the required data rate to exchange visual cues or the map information. Direct dense approaches require the raw pixel intensities to be available, which leads to a considerable amount of data to be transmitted for running the visual SLAM task at a remote site. Indirect sparse methods on the other hand usually require only a set of local features extracted at a limited number of pixel positions to be exchanged. In addition, a sparse map contains by definition fewer data than a dense representation.



(a) Categorization of visual SLAM systems based on visual information and map representation.

(b) Categorization of visual SLAM systems based on the underlying state estimation approach.

Figure 2.2: Categorization of visual SLAM systems based on the information and the estimation approaches. DSO* and SVO* provide only visual odometry.

2.3.2 Collaborative visual SLAM

With real-time capable accurate visual SLAM systems becoming available, some attention has recently been attracted by collaborative visual SLAM. In this section, an overview of *collaborative*, *decentralized* and *distributed* visual SLAM systems is presented. The term collaborative SLAM often refers to collecting information from multiple agents to build a joint map. The terminology decentralized or distributed SLAM is often used to describe systems where the individual components (e.g., the front-end and back-end) of the visual SLAM tasks are placed at different processing entities.

Zou et al. [87] proposed a collaborative system coined CoSLAM using multiple cameras simultaneously. First, the authors differentiate between map points adhered to static scene objects and dynamic map points from transient observations. If enough static map points are visible, they track each camera individually. If temporary objects are visible, they use inter-camera pose estimation to obtain the camera position based on neighboring cameras sharing the same field of view. They assume that the video frames from all cameras are available at a central processing unit. Forster et al. [88] described a monocular SLAM system for collaborative mapping using micro aerial vehicles. They propose to run visual odometry on the MAVs and send selected keyframes to a central ground station. The ground station uses this information to build a map for each MAV. By detecting overlap and subsequent merging of the maps, a common representation is obtained. They specifically account for the scale differences and drift between the monocular visual odometry and the map at the central station. However, they do not provide feedback to the agent. Riazuelo et al. [89] introduced a cloud-based approach named C^2TAM for cooperative tracking and mapping. The authors outsource the costly optimization process to more powerful hardware located in the cloud and keep only a lightweight tracking on the local device. However, they transmit the raw keyframes requiring a considerable amount of transmission capacity. In addition, they send a complete copy of the map after every map optimization back, which is prohibitive in large-scale scenarios. Schmuck et al. [90] introduced a collaborative SLAM system based on ORB-SLAM2 with the support for multiple MAVs. In their approach, each MAV runs a lightweight visual SLAM system by keeping only a limited map in the local memory. All information is collected at a central server, where the computationally expensive tasks, such as global map optimization, are carried out. Schneider et al. [85] provided an extended framework for visual-inertial SLAM that supports multi-session mappings and allows merging maps obtained from multiple sessions into a joint representation.

Fully decentralized systems were proposed by Cieslewski et al. [91]. In their scenario, multiple agents building individual maps are considered. The maps are merged when an overlap is detected among them. The authors proposed a decentralized visual place recognition approach. This approach is based on previous work [92], [93] addressing the issue of data-efficient distributed overlap detection. To facilitate distributed place recognition, a part of the feature space from a global image signature is assigned to each agent. On each new keyframe, an agent sends its global image signature only to the agent responsible for the part of the feature space for this signature. With this approach, no central entity has to collect all image signatures in order to detect overlap, but each agent is responsible for detecting over-

lap within its part of the feature space, which reduces the amount of data to be transmitted. On overlap, inter-robot pose measurements in the form of local features and landmark information are exchanged to estimate the relative pose between the two agents. In order to ensure that data is efficiently exchanged and the required locking of a shared map does not prevent real-time capabilities, several approaches for map synchronization based on version control systems [94], [95] have been proposed.

2.3.3 Feature selection

In order to adapt the amount of data that needs to be processed for the targeted computer vision task, several data reduction methods have been proposed. For the domain of visual search, Francini et al. [37] proposed to use keypoint properties such as the scale-space level, response value, and distance to the image center for calculating a score for the usefulness of a feature. By ranking the local features within an image according to this score and transmitting the features only up to a certain bit budget allows adapting to the available transmission capacity. However, the demands for visual SLAM tasks are different. Features should be distributed over all scales and the whole image to avoid degenerated cases when using, for example, Perspective-n-Point (PnP) algorithms to estimate the camera position. Spreading features has a substantial impact on the performance of visual odometry systems [16]. Distributing features has already been used in early visual odometry [96] systems and is still used in state-of-the-art visual SLAM systems [70]. Also, sparse and direct methods benefit from distributing the used pixels among the image [67]. Hartmann et al. [38] argue that predicting the matchability based on the keypoint properties, such as detector response, is not the best solution. They proposed to train a random forest offline on the descriptors themselves. They achieve a considerable reduction in feature matching time when using the feature classification prior to the matching process. Instead of relying on handcrafted selection criteria, Dymczyk et al. [39] proposed to use a CNN network to classify a local feature based on the image patch around the keypoint into stable and unstable features. According to their evaluation, they are able to reduce the resulting map size of a visual SLAM system by 70% while being able to relocalize about 80% of their keyframes. However, they report the classification time of a single landmark with 2.7 ms using a GPU. The question arises how this method scales with the number of features, as a single frame can easily contain hundreds of local features. While the aforementioned approaches try to predict the usefulness of already extracted features, Cieslewski et al. [97] used a different approach, where they try to detect the minimum set of interest points that is required to run visual odometry or a visual SLAM system. They propose to use a CNN with a Structure-from-Motion algorithm in the loop to detect succinct features and report roughly 20 ms per image for a forward pass through the network when using a shallow network architecture.

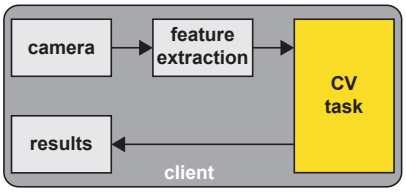
2.3.4 Map compression

A key challenge in collaborative SLAM systems is to exchange map information between multiple agents efficiently. Usually, a reduction in the required data rate is achieved by introducing a certain amount of loss before saving or exchanging the map. When exchanging the 3D information, point-cloud compression approaches such as octrees [98] or more enriched variants, such as OctoMap [99] containing additional information, whether a voxel is occupied, can be used. Some approaches take the trajectory of the camera into account [100]. In their most recent version [101], the authors proposed to split the trajectory into smaller subsets and model the partial trajectories using non-uniform rational B-splines. Afterward, they only consider the map points that are visible from each keyframe in a segment. Lynen et al. [102] proposed a greedy method for map sparsification by iteratively removing map points while ensuring that each keyframe has a minimum number of observations left. Dymczyk et al. [103] compared different criteria such as the number of observations, the covariance of the map point pose, and the descriptor variance of each map point. They also discussed the influence of overlapping parts of the trajectory and extended their work in [104]. Park et al. [105] proposed to formulate the map sparsification as an optimization problem, which is solved by mixed-integer quadratic programming. The problem is related to the maximum cover problem, where a subset of map points should be retained such that every keyframe is covered at least b times. Dymczyk et al. [104] reduced the complexity of the optimization problem by partitioning the map into sub-maps. Cheng et al. [106] proposed to formulate the problem as a weighted k-cover problem with a prediction step for the parameters involved. Merzić et al. [107] derived a quality metric for assessing the quality of a given map. While most of the approaches achieve a data reduction by discarding some of the map information, the lossless compression possibilities using the map structure and the encoding of the visual information is often neglected.

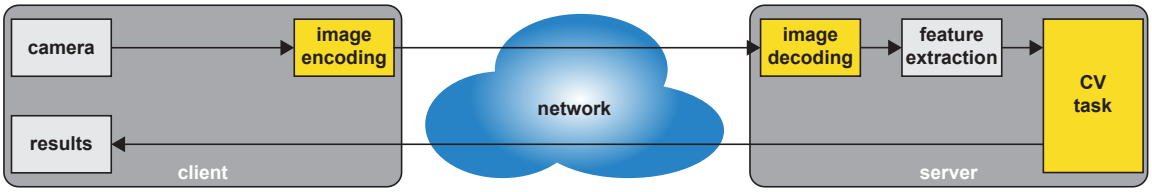
2.4 Visual content coding

In the following, three key system architectures for performing computer vision tasks, as described in [108]–[110], are revisited.

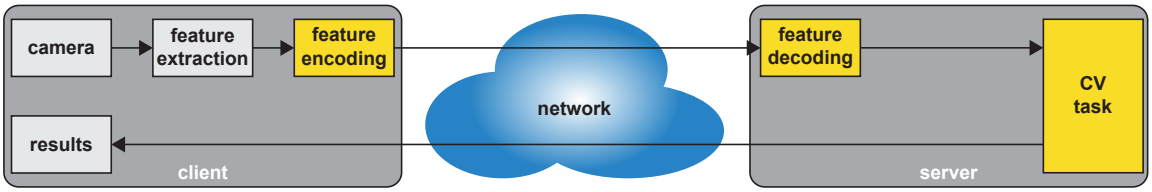
First of all, the image data can be processed at the client device itself, which is referred to as *local processing*. The second architecture employs a client-server setup, where traditional image and video compression techniques are used to transmit the visual content from the client to the server. This server performs the desired computer vision task and sends the result back to the client, which is coined *Compress-then-Analyze* (CTA). The third approach is the *Analyze-then-Compress* (ATC) architecture, where an image abstraction, such as local features, is extracted at the client, compressed, and sent to the server. The server decodes the visual information, performs the computer vision task, and sends the information back to the client. The system architectures are illustrated in Figure 2.3, whereas the advantages and disadvantages are discussed in more detail in the following.



(a) Local processing architecture. All information is obtained and processed locally on the device.



(b) Compress-then-Analyze architecture. A compressed image representation is transmitted to a server, which carries out the computer vision task and signals back the results.



(c) Analyze-then-Compress architecture. An image abstraction is created using features. A compressed feature representation is transmitted to a server, which carries out the computer vision task and signals back the results.

Figure 2.3: Comparison of architectures for mobile computer vision tasks.

2.4.1 Local processing

The local processing architecture (Figure 2.3a) is only feasible if enough computational power is available. A major advantage of running a certain task directly at the client is the independence of a network connection to a server. As the processing is carried out locally, no information has to be exchanged, and no delay is introduced by network transmission. However, there is a considerable performance gap between the energy-constrained hardware on a mobile robot platform and a powerful server. This discrepancy in computational power is observable when trying to run computer vision frameworks that are capable of running in real-time on commodity computers on embedded devices. In order to enable real-time applications on the client, considerable effort has to be made to simplify, adapt, and speed up the process. This affects the image processing, such as local feature extraction [111], as well as the computer vision task, such as visual SLAM [112], [113]. Despite all the effort, there is a considerable gap in performance or sacrifice in terms of accuracy. Another approach is taken by HoloLens [114], where parts of the tracking and mapping algorithm are implemented in a hardware chip called the Holographic Processing Unit that paves the way for always-on computer vision. This hardware-based solution has several drawbacks, such as the considerable amount of initial investment required to develop such a chip, reduced flexibility, and the restriction to devices shipped with matching hardware acceleration. While other visual SLAM systems are solely based on software that can be rolled out at virtually any device that is fast enough, the necessity to deploy hardware hinders the roll-out for the mass market.

2.4.2 Compress-then-Analyze

In the Compress-then-Analyze architecture, the visual information is exchanged by using available image and video compression techniques such as JPEG [115], H.264 [116], or H.265 [117] as shown in Figure 2.3b. The computer vision task is carried out at the server using the information of the previously decoded image. A major advantage is the possibility to reuse mature image and video compression standards. The availability of application-specific integrated circuits allows hardware acceleration of the main encoding and decoding steps, which facilitates both real-time capabilities and energy-efficient compression of visual content. In contrast to specific hardware for visual SLAM, image and video compression acceleration hardware made its way into most mobile and embedded devices due to the ubiquitous demand for mobile video. A further advantage is the availability of visual information at the server-side. This can be used for manual inspection, to archive the data, or to replace the computer vision algorithm. This is desired in case future technologies might be able to improve on the performance, for example, by using upcoming local features. However, this also comes at a cost. A drawback is that today's compression techniques are optimized for the human visual system and are not well suited for machine-based analysis. Some effort was made towards adapting image [118] and video coding [119] approaches for better feature preservation. One of the main problems when using block-based coding methods is the introduction of artificial corners at block boundaries, where features extracted using corner detectors falsely adhere. In order to alleviate this problem, a hybrid method between CTA and ATC here named *Hybrid-Compress-then-Analyze* was introduced by Chao et al. [120]. The authors propose to extract only the keypoints at the client and use this information about the feature locations to compute the descriptors on the decoded images at the server-side. According to their evaluation, this results in improved feature quality at the cost of signaling the keypoint locations as side information. The most important drawback of CTA is the squandering of resources by transmitting the actual visual information. While for a specific task, only parts of the image might be sufficient, a considerable amount of bits is allocated for transmitting visual information that might not be helpful.

2.4.3 Analyze-then-Compress

In order to tackle the limitations of CTA, Analyze-then-Compress was considered [108]. In this concept, local features are extracted at the client-side and sent to the server instead of the pixel information, as illustrated in Figure 2.3c. Starting with real-valued descriptors such as SIFT and SURF, Baroffio et al. [121] proposed to code visual features extracted from video sequences using a scheme similar to hybrid video coding. They extended their approach to local binary features in [122], [123]. For visual feature coding, the Compact Descriptors for Visual Search (CDVS) standard was proposed by the Moving Picture Experts Group (MPEG) [110]. It consists of different normative blocks for local feature detection, selection, description, and compression. It is based on computationally demanding SIFT-like [31] features and is specifically optimized for a visual search task. A feature selection based on the previously introduced work of Francini et al. [37] enables to adapt the data rate to differ-

	advantages	disadvantages
local	+ no network connection required	- local processing power required
CTA	+ hardware acceleration available + images for inspection available	- coding optimized for humans - required data-rate - network connection required
ATC	+ compact visual information	- restriction to selected features - network connection required

Table 2.3: Advantages and disadvantages of different processing architectures.

ent transmission capacities. In addition, the standard includes a normative block for a global image signature based on the Scalable Compressed Fisher Vector [124]. For local feature compression, it includes a transform-based compression scheme followed by scalar quantization and entropy coding. The proposed compression method for the keypoint locations within the MPEG-CDVS standard uses a location histogram [125], [126], which includes a quantization step leading to a loss in accuracy. At the time of writing, the upcoming MPEG Compact Descriptors for Video Analysis (CDVA) standard has reached Committee Draft level [127]. Instead of still images, it focuses on the task of video analysis, including the temporal correlation between visual content extracted from video sequences. An early overview of the expected techniques is given by Duan et al. [128]. The draft proposes a keyframe-based structure and temporal prediction for the local and global descriptors, which are adapted from CDVS. Another novel aspect is the support for deep-learned descriptors. To this end, a framework for CNN based image analysis was proposed with paying special attention to network compression, as the storage required to store the weights defining a CNN can easily exceed several hundred megabytes. In extension to Hybrid-Compress-then-Analyze, a diametrical approach under the name of *Hybrid-Analyze-then-Compress* was proposed by Baroffio et al. [129]. In this scheme, features are extracted from both the original image and the decoded image representation, including the coding artifacts. The authors suggest calculating the differences between the descriptor extracted from the original image and the reconstructed image after compression. They propose to send the compressed image, the keypoint location, and, additionally, the difference vector necessary to compensate for the distortion introduced by the image coding algorithm. A more detailed overview of compact features and different compression methods is given in [26]. The advantages of the different approaches are summarized in Table 2.3.

2.5 Cloud architecture

With regard to the actual deployment of the ATC and CTA-based approaches, several strategies are possible [130], [131]. Approaches using a single server quickly reach their limits and become unfeasible with a growing number of clients querying the service. Today, many applications make use of cloud-based infrastructures, where a scalable number of computing entities is available. The hardware in the form of virtual machines in the cloud can be

adaptively scaled, as more clients are requesting services (on-demand hardware). The visual SLAM solution can also run on a higher level of abstraction as Software as a Service (SaaS, on-demand software) or, more specifically, in this context: *SLAM as a Service* (on-demand SLAM). In CTA-based approaches, the user can be provided with an interface to upload or stream video data and receive the fused map information. This can happen either in real-time or in the form of a reconstructed map. In ATC-based solutions, the suitable image abstraction for the targeted visual SLAM task has to be provided. Different architectures exist in the domain of cloud computing. A centralized cloud computing and an edge cloud computing architecture are discussed in the following.

2.5.1 Centralized cloud computing

Regardless whether ATC or CTA-based approaches are used, employing a *centralized cloud* architecture (Figure 2.4) allows an efficient and energy-saving servicing. An important aspect orthogonal to the previously discussed choice of architecture is the server location. In this example, a centralized cloud means that all the hardware is located in a few data centers. These data centers are designed with a high-speed interconnection between all contained servers. The load of the visual SLAM processing systems can be distributed among the shared computing resources using a suitable load balancing algorithm, allowing to utilize the existing hardware better. In addition, if all the map information from different clients is available in a centralized cloud architecture, the exchange and fusion of multiple maps into a joint representation require less data exchange outside the data center or outside the high-speed cloud infrastructure. However, one of the main drawbacks of centralized cloud-based computing in the context of real-time visual SLAM is the delay introduced by sending the visual information to the cloud and receiving the response. In the case of centralized cloud data-centers, the client and the server that is currently responsible for processing the request could be located on different continents. This adds a considerable amount of transmission time, which is not acceptable in low delay applications. In this case, an edge cloud infrastructure is advantageous.

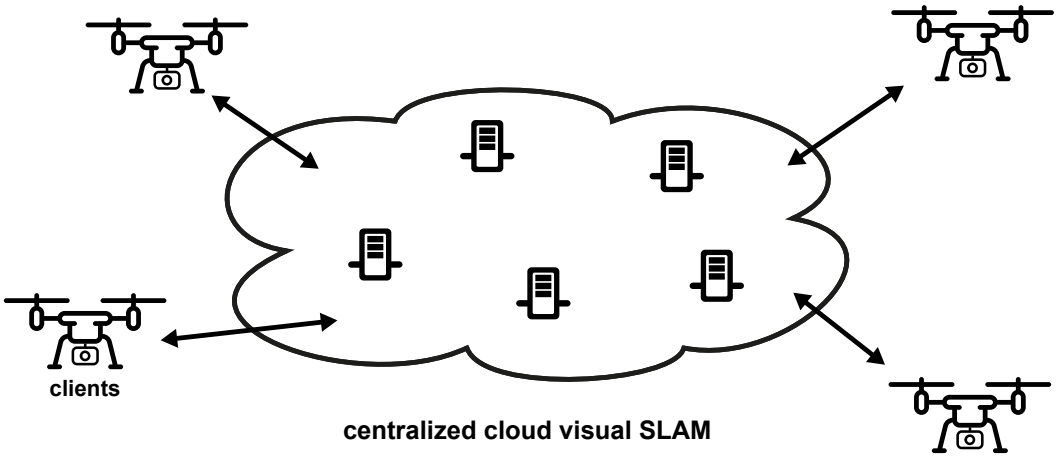


Figure 2.4: Centralized cloud computing architecture using a few centralized data centers.

2.5.2 Edge cloud computing

In traditional cloud computing, the traffic has to be transported from the client to the centralized data centers. With the concept of *edge clouds*, as shown in Figure 2.5, the processing is shifted closer to the client's location. Computing capabilities could be integrated with the hardware of mobile communication networks [131] or located on-premise in factories that use visual localization in their automation strategy. The edge cloud adds an additional layer between a centralized cloud and the client, allowing to perform tasks that require low latency in the proximity of the clients. This could be in the base station, where the client is currently situated. Besides the removed latency, this also reduces the data load from the client on the core network towards the centralized data centers. Another important advantage of localized processing of the data is location awareness. A base station could collect and store the local maps which were captured by users in the current mobile cell. If the local map is updated or substantially extended, the map could be uploaded to the central cloud to aggregate the local maps in a city, a regional, national, or even a planet-covering representation. Also, local maps could be stored at the base station and provided as prior knowledge to newly registered devices. It also adds a certain fault tolerance as the visual SLAM processing is distributed among several separated locations. Besides the advantages, there are also some drawbacks. The base stations have to be equipped with computing hardware, which might not everywhere be profitable. However, in this case, a fall-back solution using the centralized cloud infrastructure could be realized. Another technical difficulty is the issue of roaming if the device is requesting a seamless handover to the neighboring mobile cell. In this case, the SLAM process has to be migrated from a base station to the next without noticeable service interruption.

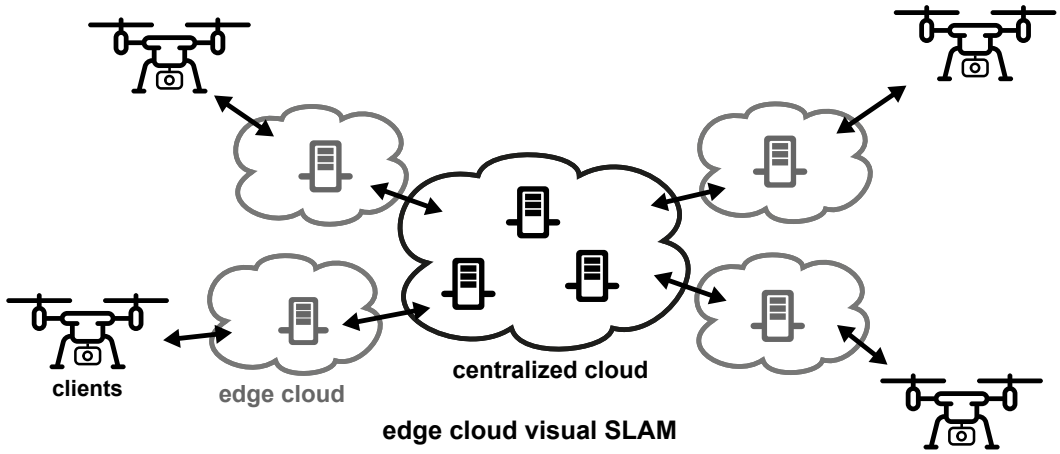


Figure 2.5: Edge cloud computing architecture, where the processing takes place in distributed data centers to reduce the communication delay. They can be connected via a centralized cloud.

2.6 Datasets

This section introduces the datasets used throughout this thesis. An overview of the individual properties of the SLAM evaluation datasets is provided in Table 2.4.

MIRFlickr1M: A dataset predominantly used for training in this work is the *MIRFlickr 1M* [132] dataset. It comprises 1 million Flickr images with manual annotation. While the dataset is designed for retrieval applications, it is used in this work primarily to train the visual vocabularies due to the wide variety of image categories featuring many facets of visual appearances, which ensures generalization of the algorithms.

Tracking: For assessing the resulting compression on a homography estimation task, the public dataset from Gauglitz et al. [24] is used. The dataset provides a variety of videos containing a planar texture each. The dataset was introduced to evaluate the performance of different interest point detectors and feature descriptors for visual tracking. For each category, several sequences exist, showing the scene subject to different motion patterns, such as panning, rotation, zoom, and unconstrained motion. Ground truth is provided as a homography matrix warping the planar texture into a common reference frame. In this work, the unconstrained motion sequence is used, which features a resolution of 640 x 480 pixels and 500 frames captured at a frame rate of 15 fps using a Unibrain Fire-i camera.

KITTI: One prominent SLAM dataset is the *KITTI* visual odometry dataset [133]. It features different sequences in an urban environment captured from a driving car. The images were captured by a stereo grayscale camera system with a baseline of roughly 53.7 cm and a frame rate of 10 Hz. The images have been taken at a resolution of 1392 × 512 pixels and are provided with a resolution of 1241 x 376 pixels after rectification. Ground truth is provided by a GPS/IMU localization unit with real-time kinematic correction of the GPS signal. The dataset additionally features color images and 3D laser scanner data, which are not used in this work.

EuRoC: The *EuRoC* micro aerial vehicle SLAM dataset [54] consists of several sequences collected in an industrial environment and a room scenario using an AscTec Firefly MAV. The image sequences were acquired using a global shutter stereo camera setup based on an MT9V034 sensor providing a WVGA (more specifically 752 x 480 pixels) resolution at a frame rate of 20 Hz for the left and right view. The baseline between the cameras is about 11 cm. The camera calibration parameters for the sequences are provided by the authors.

	environment	properties	sensor	calibration
KITTI	urban	1241 x 376 @ 10 Hz	stereo	rectified, undistorted
EuRoC	industrial, room	752 x 480 @ 20 Hz	stereo	parameters provided
TUM RGB-D	industrial, office	640 x 480 @ 30 Hz	RGB-D	parameters provided

Table 2.4: Tabular overview over the datasets.

The dataset provides ground truth data acquired from a Leica Nova MS50 in the case of the industrial scenario, and a Vicon tracking system in case of the room scenario.

TUM RGB-D: Another well-known dataset is the *TUM RGB-D* dataset [134]. It consists of 39 sequences captured by a Microsoft Kinect camera in both office and industry hall scenarios. The sequences contain color and depth images at a resolution of 640 x 480 frames with a sampling rate of 30 Hz. The intrinsic camera calibration is available. Some sequences are already undistorted. The ground truth data was obtained using a MotionAnalysis motion capturing system.

2.7 Summary

In this chapter, the relevant related work regarding image content description, such as local and global features, has been discussed. An overview of existing visual SLAM techniques has been provided. In addition, challenges and existing methods of collaborative SLAM and map exchange have been introduced. Different architectures for exchanging visual information have been discussed, including their advantages and disadvantages. A summary of cloud-based processing architectures has been presented. Several well-known datasets for evaluating visual SLAM systems have been covered.

The lack of related work addressing the efficient compression of data in collaborative visual SLAM systems serves as motivation to further investigate different aspects, requirements, and discover the untapped potential of compression in this context:

1. An efficient compression algorithm for local features in the context of visual SLAM is necessary.

2. The compression approach should be able to support typical visual sensor data obtained by a stereo camera setup to allow metric scale visual SLAM.

3. A data-efficient collaborative visual SLAM architecture where multiple agents send compressed visual information to a server to build a common map is lacking.

4. In order to provide prior knowledge to other agents, an efficient map compression algorithm is required that not only discards information but also exploits dependencies between visual information in a visual SLAM map to facilitate lossless compression.

Chapter 3

Feature coding framework

3.1 Problem statement

A comprehensive framework capable of coding local binary features, as used in recent visual SLAM applications, should be investigated. The requirements for source coding are threefold. First, it should reduce the number of bits required to transmit a set of features. To this end, the coding should provide independent coding of features, as well as exploiting temporal and spatial redundancies. Second, the framework should be capable of achieving real-time performance, even on embedded devices. Third, the amount of information should be adaptable to the channel conditions. In contrast to the existing solutions in this field [123], this work incorporates the specific properties of the targeted computer vision task into its solution. This chapter provides an overview of the techniques used to approach these goals. The individual parts are then subsequently introduced throughout this thesis, accompanied by their respective experimental evaluation.

3.2 System architecture

The coding framework for binary feature coding developed in this work is based on [123] and inspired by hybrid video coding. Different coding schemes have evolved during the development of hybrid video coding standards (MPEG-X, H.26x). For example, the coding of still images is achieved by techniques summarized as *intra coding*. In order to exploit temporal redundancies in an ordered video sequence, *inter coding* is used. When considering multiple views, extensions to *multi-view coding* have been proposed. Following [123] some of these principles are adapted for coding local binary features extracted from individual frames, video sequences, or stereo-camera setups. A comprehensive block diagram of the proposed framework is provided in Figure 3.1. Camera images obtained by a stereo camera setup serve as input. Subsequently, the features are extracted from both views and stereo matching is performed to estimate the depth information. The depth values can be encoded using a separate *depth coding* mode. Based on a mode decision module, different coding modes can be selected. In the following, the different coding modes, as well as the advantages and disadvantages, are discussed. The results are summarized in Table 3.1. For an in-depth explanation of the individual coding modes, the reader is referred to the corre-

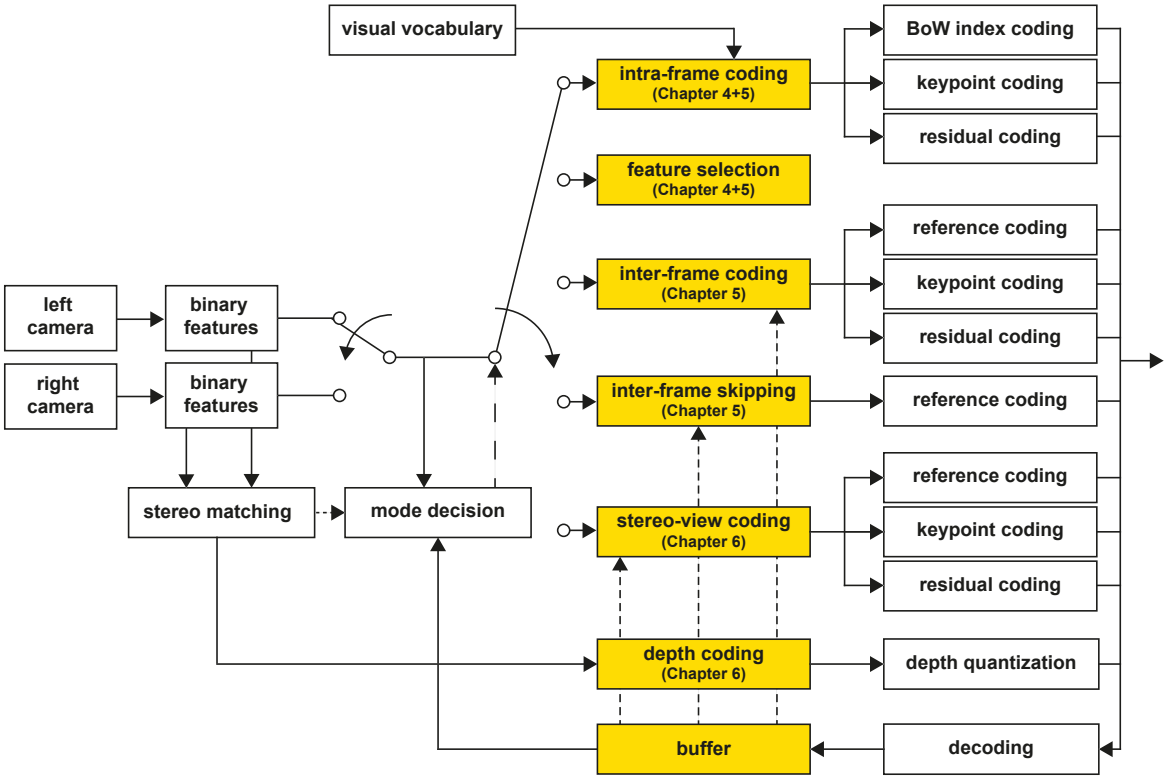


Figure 3.1: Overview of the proposed feature coding framework.

sponding chapters.

Intra-frame coding: This mode provides the possibility to encode features without any dependency on previous frames or neighboring views. Hence, every feature is compressed individually and can be independently decoded. When encoding all features contained within a frame using intra coding, this frame can be decoded without any prior knowledge about the history of the sequence. This allows starting the decoding of a sequence at any intermediate frame, also called *random access*. This independent decoding has the advantage that the effect of previous transmission errors are mitigated and do not influence future frames. As no reference has to be searched, this is expected to be comparably fast in terms of computation time at the cost of increased bitrate as no prior knowledge about the properties of the frames can be exploited. The intra-frame coding is discussed in Chapter 4.

Inter-frame coding: To exploit temporal redundancies, features should be tracked across frames and only the differences should be transmitted. This scheme, denoted in the following as inter-frame coding mode, is inspired by the motion-compensated prediction used in hybrid video coding. It has been introduced to features by Baroffio et al. [123]. Similar to video coding, this concept is responsible for a comparably high gain in terms of bitrate reduction at the drawback of adding computational complexity to find the best reference feature. Another drawback is the added dependency between the frames. If previous frames containing the reference features are affected by transmission errors, the dependent features cannot be decoded without errors. This work goes beyond the related work and extends this mode to multiple reference frames. The inter coding method is detailed in Chapter 5.

	advantages	disadvantages
intra coding	+ random access possible + lowest complexity	- highest bitrate per feature
inter coding	+ lower bitrate compared to intra	- additional complexity - prone to transmission errors
stereo coding	+ lower bitrate compared to intra	- additional complexity - only in stereo-setups available

Table 3.1: Advantages and disadvantages of the main coding methods.

Depth coding: Visual SLAM using solely monocular information is only capable of estimating a map of the environment up to an arbitrary scaling factor. In order to allow metric scale SLAM, additional information collected by an IMU, range scanners, or other sensors is required. However, this work primarily focuses on visual SLAM using additional depth information acquired by either a depth camera or a stereo camera system. The latter can estimate metric scale depth information from matching features across neighboring views in a calibrated stereo camera setup with a known baseline. Hence, the feature coding framework is extended by a specific coding mode to add compressed depth information to the stream of features in Chapter 6.

Stereo coding: More general, features from a stereo camera can be transmitted by exploiting the spatial redundancies between neighboring views. This can be achieved by sending only differences between local features extracted from both views describing the same physical entity. The concept has been proposed for general visual sensor networks [135] and is here adapted for calibrated stereo setups, denoted as stereo-view coding. While reducing the bitrate per feature, this method increases the complexity and is only available in a stereo setup. The stereo coding method is introduced in Chapter 6.

Rate control: The overall goal of reducing the required data rate can be achieved by lossless compression exploiting the signal statistics. However, similar to lossless video coding, the lossless data reduction is bounded by the entropy of the visual data. With lossy approaches, an adaption to arbitrary data rates can be facilitated by removing presumably uninformative parts of the data. In this work, two different approaches for rate control and rate allocation are evaluated. First of all, a selection of features to transmit is proposed. Based on keypoint properties and the targeted task, a score can be calculated for every feature. Ranking the features according to their score and stopping the feature coding when a given time frame or a bit budget is exhausted efficiently reduces the required data rate. This method is included in Chapter 5. The second approach is more flexible and allows to transmit features with a certain accuracy. To this end, features are sorted into different classes and for each class, only a certain part of the original descriptor is reconstructed with the aim of maximizing the overall task performance. This approach is detailed in Chapter 4.

3.3 Notation

In the following, the notation used throughout this work is introduced. For the sake of consistency, a similar notation, as used by [123], is employed to describe the local feature properties.

The first part necessary to define a local feature is a keypoint describing the location where the local feature has been detected. In order to make a feature invariant towards rotation and scaling, an orientation and a patch size is usually estimated during feature detection. Hence, the vector containing the keypoint properties for feature i extracted from the n-th image in a sequence is denoted as $\mathbf{k}_{n,i} = [x, y, \sigma, \theta]$, where $x \in \{z \in \mathbb{R} \,|\, 1 \leq z \leq N_w\}$ and $y \in \{z \in \mathbb{R} \,|\, 1 \leq z \leq N_h\}$ describe the keypoint positions in pixels, $\sigma \in \{z \in \mathbb{N} \,|\, 0 \leq z < N_\sigma\}$ denotes the scale-space level and $\theta \in \{z \in \mathbb{R} \,|\, 0 \leq z < 2\pi\}$ indicates the keypoint orientation. N_w, N_h, and N_σ denote the image width, the height, and the number of scale-space levels. Further keypoint information, such as the detector response, is neglected in this work but can accompany a feature as additional side information.

The second part contains information about the visual outline of a local feature patch in form of a fixed-length binary representation. This image patch descriptor is usually the outcome of a set of pairwise pixel tests contained in a binary vector $\mathbf{d}_{n,i} \in \{0, 1\}^{N_d}$ with length N_d. For example, the ORB descriptor has $N_d^{\text{ORB}} = 256$ entries, whereas FREAK and BRISK have $N_d^{\text{FREAK}} = N_d^{\text{BRISK}} = 512$ dimensions. A specific descriptor element d_j is indexed with the subscript j.

An entry of a binary descriptor can be seen as an outcome of an experiment described by a random variable X with an associated binary alphabet $A_x = \{0, 1\}$. The probability of the outcome being zero is defined by the corresponding probability mass function as $f_X(0) = P(X = 0) = p_0$. The probability of being one is defined as $f_X(1) = P(X = 1) = p_1$ with $p_1 = 1 - p_0$. It is worth noticing that in this notation, the same probability is assumed for all descriptor entries. If individual probabilities for each descriptor entry are used, each entry is modeled with a binary random variable X_j with associated probability mass function $f_{X_j}(d_j) = Pr(X_j = d_j)$. In this case, the probability for entry d_j being zero is denoted as $p_{0,j} = f_{X_j}(0) = P(X_j = 0)$ and conversely $p_{1,j} = 1 - p_{0,j}$ for being one.

More general, the entropy function $H(X)$ for binary random variables is parameterized with the probability of one of both symbols, i.e., in this work the zero symbol as $H(p_0)$. This is also referred to as the binary entropy function [136]. As both symbols are mutually exclusive, this allows inferring the missing probability of the other symbol directly. The entropy for non-binary sources is denoted with H, where the dependency of the random variable X is dropped for the sake of readability. The entropy is denoted with H, whereas the experimentally measured rate is denoted by R with equal subscript and superscripts.

3.4 Summary

In this chapter, a summary of the binary feature coding framework has been provided. The coding modes have been briefly discussed, including their individual advantages and disadvantages. In addition, the notation used to describe the keypoints, the descriptors, the calculated entropy, and the rate are introduced.

In the following chapters, the details of the coding modes and the rate allocation techniques are introduced step by step, starting with the intra coding and a rate allocation method.

Chapter 4

Intra coding and rate allocation

4.1 Problem statement

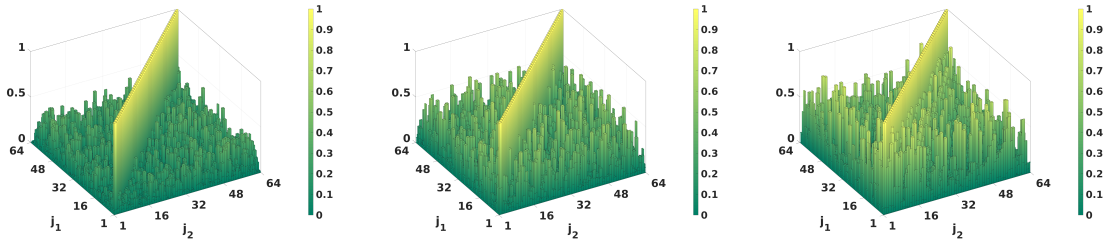
A fundamental problem of data reduction in the context of feature-based visual SLAM is the compression of the visual information. Regardless of whether a visual map or the visual information contained in individual frames should be transmitted, efficient compression of the local visual features, including the keypoint and descriptor information is required. The requirements of the compression introduced in this chapter are threefold. First, the coding should be as efficient in terms of the required data rate as possible. Second, the individual features should not rely on any previously observed feature in a visual SLAM map or video stream. Third, the descriptor coding should provide the possibility of adapting the amount of data. If enough transmission rate is available, the lossless transmission of both the descriptor and the keypoint should be possible to enable accurate geometric reconstruction of the scene.

An analysis of the feature properties of recent binary features reveals possibilities to approach or to achieve the above-stated goals for feature coding and to enable a flexible rate allocation by discarding some of the visual information. Parts of this chapter have been published in [2], [5].

4.2 System architecture

In the work of Baroffio et al. [123], compression of individual features is achieved by exploiting the statistical dependencies among the descriptor entries d_j. To this end, the authors propose to resort the individual descriptor elements according to their mutual correlation. The coding gain stems from modeling the descriptor elements as first-order Markov sources and from coding the current element d_j based on the value of the preceding element d_{j-1}.

While this method works well for specific descriptors, it does not perform well for all feature types. More specifically, the approach was evaluated using BRISK features where the sampling pattern for the pixel tests is defined on concentrically equally spaced circles located around the keypoint position [47]. A drawback of this sampling pattern is that the pixel tests tend to be correlated. In contrast to this approach, a greedy strategy to select pixel tests that are preferably uncorrelated is used in ORB [46]. Similarly, FREAK [48] includes a



(a) ORB features. (b) FREAK features. (c) BRISK features.

Figure 4.1: Absolute values of the Pearson correlation coefficient evaluated between the first 64 descriptor entries for ORB (a), FREAK (b), and BRISK features (c). The BRISK features show the highest correlation among the descriptor entries.

step to learn the best tests based on a training dataset. The result is illustrated in Figure 4.1, where the absolute values of the Pearson correlation coefficients [137] are shown for the first 64 dimensions of ORB, FREAK and BRISK features evaluated on the *EuRoC MH01* sequence using 1k features per frame. A value near zero indicates no correlation between the descriptor entries and values closer to one indicate a higher correlation. The method proposed by Baroffio et al. using the Markov source assumption works for BRISK features as some descriptor entries show a significant correlation mainly due to the missing selection step. On the other hand, this technique is not expected to work well for ORB and FREAK features, as the descriptor entries are selected to minimize the correlation.

To overcome this limitation and to find a coding method that works well for ideal binary feature descriptors with mutual independence of the descriptor elements, the use of visual information in the primary target application, namely ORB-SLAM2, is analyzed. First, local binary ORB features are used to describe salient points in the image, which are then matched for tracking and mapping. Second, a global image signature, namely the Bag-of-Words representation, is employed for rapid feature matching, relocalization, and loop-closing.

The main idea behind rapid feature matching is to quantize all features to their nearest visual words and to consider only features as matching candidates that share the same or similar visual words. The similar visual words matching can be facilitated by allowing to match a visual word to visual words that are connected to the same parent node in the hierarchical vocabulary tree structure (Figure 4.2). The Bag-of-Words representation can also be used to identify visually similar images to perform loop closing.

The visual word representation is already a good approximation of the actual descriptor. Inspired by the use of both these partially redundant representations, the key idea for trans-

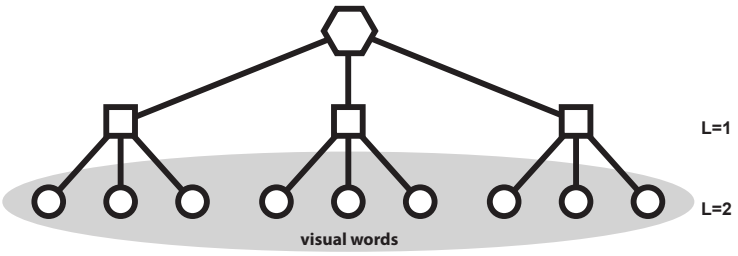


Figure 4.2: Hierachical vocabulary tree. At each level, the descriptor is compared to the child descriptors of the current node in terms of Hamming distance, until a leaf node is reached.

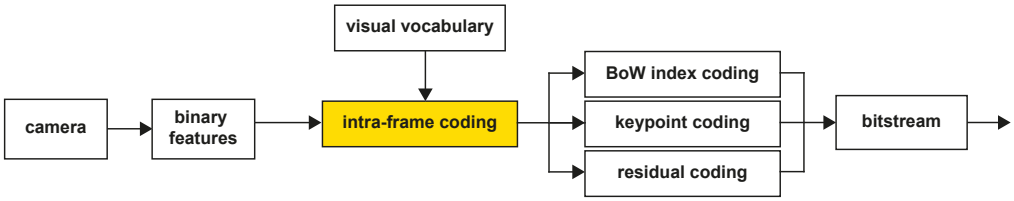


Figure 4.3: Overview of the proposed intra coding mode.

mitting the visual information is to exploit a common visual vocabulary available at both the client and the server. Hence, only an identifier of the closest visual word and the differences between the visual word and the actual descriptor have to be transmitted. As the difference contains mostly zeros, it can be compressed efficiently using entropy coding. The proposed intra coding mode is denoted with I, where m defines the used coding mode for a specific feature as $m \in \{I\}$. The available set of coding modes is extended throughout the work. The client-side processing of the intra coding mode is illustrated in Figure 4.3, which is explained in detail in the following.

4.3 Feature coding

4.3.1 Intra coding

First, a binary hierarchical visual vocabulary tree [138] is trained offline and placed at both the client and the server-side. Based on binary features extracted from images in a training dataset, the feature space is consecutively subdivided using successively applied binary k-median clustering. Each resulting quantization cell is further subdivided, forming a hierarchical representation. In the binary case, the distance function between the cluster cell (i.e., visual word) and the local feature descriptor is evaluated using the Hamming distance. This clustering is used to assign the descriptors to their closest visual word at the leaf nodes of the tree (see Figure 4.2). The visual word index, the keypoint information, and the residual form the information that needs to be compressed. The encoded representation of the individual features contained in a single frame is concatenated into a bitstream and transmitted to the server. At the decoder, the inverse operation is applied to reconstruct the features.

More specifically: First, the descriptor $\mathbf{d}_{n,i}$ describing the outline of visual feature i from image n is assigned to its closest visual word v^* from the vocabulary $\mathbf{C} = \{\mathbf{c}_1 \dots \mathbf{c}_{N_v}\}$ formed by the leaf nodes of the hierarchical representation, such that

$$v^* = \arg\min_{v} h(\mathbf{d}_{n,i}, \mathbf{c}_v), \tag{4.1}$$

where h defines the Hamming distance between descriptor $\mathbf{d}_{n,i}$ and visual word $\mathbf{c}_v$. In the hierarchical clustering scheme, the maximum vocabulary size N_v can be calculated as $N_v = K^L$, where the branching factor is denoted with K and the depth of the tree with L. The advantage of the hierarchically structured tree is that it can be efficiently traversed even on power-restricted devices.

The visual word index is transmitted to the server alongside the residual vector containing the differences between the actual descriptor and the visual word. By determining the

probabilities of an entry being zero in combination with entropy coding, a substantial reduction in terms of bitrate can be achieved. The larger the vocabulary size, the smaller the Hamming distance between the descriptor and the closest visual word, and therefore the higher the expected coding gain. On the downside, the larger the vocabulary, the more bits are required to signal the visual word index. The lower bound for the number of bits for a feature using intra coding can be calculated as

$$H_{n,i}^{I} = H_{n,i}^{I,BoW} + H_{n,i}^{I,res} + H_{n,i}^{I,kpt}, \tag{4.2}$$

where $H_{n,i}^{I,BoW}$ denotes the entropy in bits required for signaling the visual word index, $H_{n,i}^{I,res}$ defines the minimum costs for coding the residual vector necessary to reconstruct the descriptor, and $H_{n,i}^{I,kpt}$ are the bits required to transmit the keypoint information.

4.3.1.1 Bag-of-Words index coding

First, the probability of the visual word occurrences is analyzed in Figure 4.4, showing an excerpt of the probability mass function for a visual vocabulary of size $N_v = 1\text{m}$. The occurrences of the first 10k visual words are shown obtained by quantizing ORB features from the first 1k images of the *EuRoC MH01* and the *V101* sequence. Based on the observation that obtaining sample data from two different sequences results in two different probability mass functions, a common distribution cannot be found. Instead, a uniform distribution of the visual words is assumed. Based on Equation (2.2), the number of bits for signaling the visual word index can be derived using $p_v = \frac{1}{N_v}$ resulting in

$$H_{n,i}^{I,BoW} = \log_2(N_v) \tag{4.3}$$

bits required to signal the visual word index for feature i from image n. For the visual word index coding, arithmetic coding with uniform probabilities is used throughout the work.

4.3.1.2 Residual coding

The residual vector is computed as the binary difference between the descriptor $\mathbf{d}_{n,i}$ and the closest visual word $\mathbf{c}_{v^*}$ using the XOR operation as $\mathbf{r}_{n,i}^{I} = \mathbf{d}_{n,i} \oplus \mathbf{c}_{v^*}$. Starting with the binary entropy function for a string of length N_d, the entropy contained in the residual vector can be determined as

$$\begin{aligned} H^m(p_0^m) &= \sum_{j=1}^{N_d} (-p_0^m \cdot \log_2(p_0^m) - (1-p_0^m) \cdot \log_2(1-p_0^m)) \\ &= -\sum_{j=1}^{N_d} p_0^m \cdot \log_2(p_0^m) - \sum_{j=1}^{N_d} (1-p_0^m) \cdot \log_2(1-p_0^m) \\ &= -N_d \cdot p_0^m \cdot \log_2(p_0^m) - N_d \cdot (1-p_0^m) \cdot \log_2(1-p_0^m), \end{aligned} \tag{4.4}$$

where p_0^m denotes the probability of an entry being zero for the coding mode m. As coding a binary residual vector is also used in other coding modes, the definition of the mode is kept

general. Here, $N_d \cdot p_0^m$ is the expected number of zero elements in the binary string of length N_d and conversely $N_d \cdot (1 - p_0^m)$ is the expected number of non-zero elements.

In the case of intra-frame coding, the number of non-zero entries $h_{n,i}^I$ of the residual vector $\mathbf{r}_{n,i}^I$ corresponds to the Hamming distance between the descriptor and the matching visual word as

$$h_{n,i}^I = h(\mathbf{d}_{n,i}, \mathbf{c}_{v^*}). \tag{4.5}$$

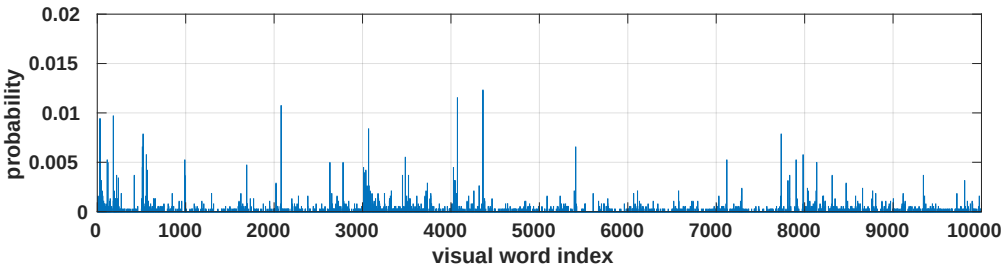
Using Equation (4.4) and Equation (4.5), the lower bound for transmitting the residual information for a specific feature descriptor i from image n can be calculated as

$$H_{n,i}^{I,res} = -(N_d - h_{n,i}^I) \cdot \log_2(p_0^I) - h_{n,i}^I \cdot \log_2(1 - p_0^I), \tag{4.6}$$

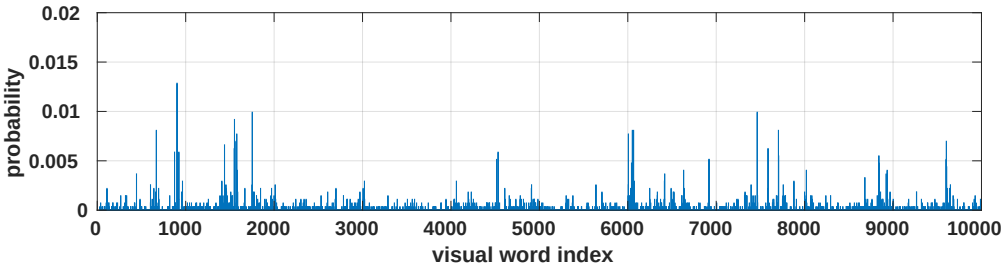
which is approached by using entropy coding techniques, such as arithmetic coding. In Equation (4.6), the assumption is made that each residual vector entry has the same probability of being zero. Violating this assumption would result in more bits required per feature. So, in contrast of using a common probability p_0^I for the entire residual vector, individual probabilities $p_{0,j}^I$ for each residual element $r_{n,i,j} \in \{0, 1\}$ can be obtained resulting in the coding costs for the residual vector denoted as

$$H_{n,i,+}^{I,res} = -\sum_{j=1}^{N_d}(1 - r_{n,i,j}) \cdot \log_2(p_{0,j}^I) - \sum_{j=1}^{N_d} r_{n,i,j} \cdot \log_2(1 - p_{0,j}^I). \tag{4.7}$$

From an encoding perspective, using Equation (4.7) allows an adaption of the probabilities to the individual descriptor entries, but adds the overhead of providing the respective probability tables. Following Equation (4.6) is sufficient, if the probabilities do not vary much among



(a) Probability tables measured for the *EuRoC V101* sequence.



(b) Probability tables measured for the *EuRoC MH01* sequence.

Figure 4.4: Probability of the visual word occurrences for the first 10k visual words in a vocabulary of size $N_v = 1$m using ORB features extracted from 1k images from two different sequences.

the descriptor elements. A detailed comparison of coding the residual vector with Equation (4.6) and Equation (4.7) for different features is included in the experimental evaluation.

4.3.1.3 Keypoint coding

Most computer vision tasks leverage information about the keypoint location. In order to provide the location and the shape information of the local feature, the keypoint properties have to be transmitted as well. This includes the pixel coordinates of the feature, the scale-space level, and the orientation information. In this work, a distinction between generic and ORB keypoints is made. For generic keypoints, the lossy approach by Baroffio et al. [121] is adapted to transmit the feature location. They propose to quantize the x and y coordinates to avoid using a real-valued floating-point representation at the drawback of losing precision. Many local binary features use an interpolation step to determine the keypoint position down to sub-pixel accuracy. BRISK features, for example, fit a 1D parabola to the scale axis to determine a more accurate estimate of the scale. Subsequently, the keypoint position is interpolated, allowing sub-pixel accurate feature locations. On the other hand, features like ORB do not use an interpolation step, which opens the possibility to exploit the scale-space representation to transmit the keypoint location in a lossless fashion.

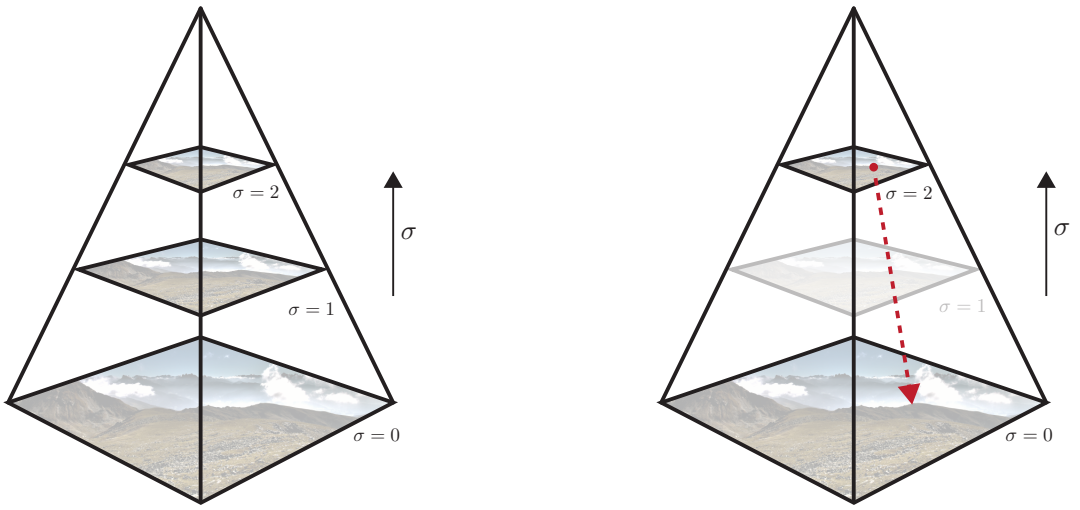
Generic keypoint coding: For generic keypoint position coding, the keypoint locations are quantized to quarter-pixel accuracy. The orientation information is quantized into $N_{\hat{\theta}} = 32$ equally spaced bins of size $\frac{\pi}{16}$. For signaling the scale-space level, the number of bits required is defined by the settings of the local feature.

A floating-point representation with 32 bits for each the x and y coordinates of the keypoint, 32 bits for the orientation, and 8 bit for the pyramid level is assumed for an uncompressed keypoint location. The number of bits required for an uncompressed keypoint is then given as $R_{n,i}^{kpt} = 32 + 32 + 32 + 8 = 104$ bits. The minimum number of bits for the intra coded keypoint depends on the image width N_w, the image height N_h, the number of pyramid levels N_σ, and the number of bins $N_{\hat{\theta}}$ used for the orientation. In summary, it is given by

$$H_{n,i}^{I,kpt} = \log_2(4 \cdot N_w) + \log_2(4 \cdot N_h) + \log_2(N_\sigma) + \log_2(N_{\hat{\theta}}). \tag{4.8}$$

Adding additional properties such as the patch size or the detector response strength is straightforward, but is not needed in many application scenarios and therefore omitted.

ORB keypoint coding: For ORB features, the structure of the scale-space representation is exploited to avoid the lossy quantization step for the keypoint position. In Figure 4.5a, a typical scale-space pyramid representation for three levels is illustrated. At the bottom, the original image is shown. Moving to the top of the pyramid, the image is blurred by applying a low-pass filter and subsampled to create smaller versions of the image. The feature detection is then performed on all scale-space representations. The ORB feature extractor, for instance, uses a FAST corner detector operating on integer pixel positions within every scale. For the final keypoint, the positions within the scale-space are transformed to their corresponding position at full image resolution as indicated in Figure 4.5b. The subsampling



(a) Scale-space representation.

(b) Keypoint in scale-space.

Figure 4.5: Illustration of the scale-space representation using three levels. Features are detected in scaled versions of the original image. The feature locations in each scale-space representation for ORB features are located at integer positions. For the final keypoint location, they are scaled to the original resolution resulting in real-valued keypoint positions (red arrow).

factor between the levels is defined by the algorithm or the application. For example, the default scaling factor between two layers of the image pyramid used in ORB-SLAM2 is 1.2. This results in a total scaling factor $f(\sigma)$ from any level σ to the original image size of:

$$f_s(\sigma) = 1.2^{\sigma}. \tag{4.9}$$

The conversion from the scale-space coordinates x_σ^s, y_σ^s to the full-size image is given by

$$x = x_\sigma^s \cdot f_s(\sigma), \quad y = y_\sigma^s \cdot f_s(\sigma). \tag{4.10}$$

Conversely, the scaling factor from the image coordinates x, y back to scale-space level coordinates x_σ^s, y_σ^s can be written as

$$x_\sigma^s = \frac{x}{f_s(\sigma)}, \quad y_\sigma^s = \frac{y}{f_s(\sigma)}. \tag{4.11}$$

This also allows for adapting the number of bits spent for coding the keypoint position according to the image size in the corresponding scale-space level. The image dimensions $N_w^s(\sigma)$ and $N_h^s(\sigma)$ per level of the scale pyramid are calculated as

$$N_w^s(\sigma) = \frac{N_w}{f_s(\sigma)}, \quad N_h^s(\sigma) = \frac{N_h}{f_s(\sigma)}, \tag{4.12}$$

where N_w and N_h denote the original image height and width, respectively. The cost for coding the keypoint $\mathbf{k}_{n,i}$ is the sum of the costs for coding the keypoint positions x_σ^s and y_σ^s, the scale level σ, and the quantized orientation $\hat{\theta}$ as

$$H_{n,i}^{I,kpt} = \log_2(N_w^s(\sigma_{n,i})) + \log_2(N_h^s(\sigma_{n,i})) + \log_2(N_\sigma) + \log_2(N_{\hat{\theta}}), \tag{4.13}$$

assuming uniform probability. This coding scheme allows for lossless coding of the keypoint position, while at the same time lowering the costs for the transmission. This assertion is true, because $N_w^s(\sigma)$ and $N_h^s(\sigma)$ are always smaller than $4 \cdot N_w$ and $4 \cdot N_h$, which is used for generic keypoints in Equation (4.8). The scale-space level has to be transmitted and decoded first in order to use the correct parameters for the position decoding.

4.4 Rate allocation

The source coding approach introduced previously can assist in reducing the number of transmitted bits in a lossless way. Naturally, the question arises, whether introducing loss could further reduce the number of bits without severely deteriorating the task performance. Following this key idea, two different levels to adapt to the amount of information transmitted to the server have been identified. A reduction can either happen on the feature level by sending only the presumably useful features. The data rate can also be adapted more fine-grained at the descriptor element level by sending only partial descriptors. In the following, a solution that enables a combination of both approaches is proposed. Local features are grouped according to their usefulness and a varying number of bits is spent per group on the descriptors to exchange visual information. In low-bitrate scenarios, a group of features can be completely skipped. The proposed algorithm for flexible rate allocation is tightly coupled to the previously introduced intra coding mode and consists of two steps:

- First, a coarse approximation of the visual descriptor using the shared visual vocabulary is transmitted in the form of the visual word index. To (partially) reconstruct the original descriptor, the binary residual vector containing the missing information is (partially) transmitted. In the proposed rate allocation scheme, the residual elements are reordered to reconstruct the descriptor elements with the highest entropy first.
- Second, the detector response is used as an indicator of the usefulness of a specific feature. According to the response, the features are sorted into N_g different classes. A utility function is used for each class, defining how much gain is expected when adding additional residual elements. As an extension to the intra-frame coding mode, partial residual vectors can be transmitted, thus providing the possibility to approximate the original descriptor at different accuracy levels depending on the available transmission rate. If enough rate is available, the entire residuals can be transmitted, resulting in a lossless reconstruction of the descriptors.

The reordering of the residual and the grouping of the features are described in the following.

4.4.1 Residual reordering

The proposed coding scheme of Baroffio et al. [123] uses the concept of descriptor element reordering to exploit the conditional entropy for coding the elements based on the value of their predecessor. In contrast to this reordering, an entropy prioritization strategy is leveraged here for rate allocation. The key idea is to sort the descriptor elements according to their entropy in descending order. With this particular ranking, the elements containing the most information are prioritized and reconstructed first. The algorithm is similar to the greedy selection strategy of the binary test selection included in ORB and related to the approach of [122].

First, all pair-wise pixel tests of a feature algorithm are part of a set T. For each pixel test, the entropy indicating the information contained in a single descriptor element is estimated

by evaluating feature descriptors obtained from a training dataset in an offline procedure. The information is given by the binary entropy function [136] for the descriptor element j as

$$H(p_{0,j}) = -p_{0,j} \cdot \log_2(p_{0,j}) - (1 - p_{0,j}) \cdot \log_2(1 - p_{0,j}). \tag{4.14}$$

The maximum of the entropy is located at the probability of the element being zero of $p_{0,j} = 0.5$, which corresponds to an entropy of $H(0.5) = -0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5) = 1$ bit. Any different probability results in a lower entropy, thus lower information. Iteratively, the element j with the highest entropy is selected from T, added to the result set U, and removed from T. As side constraint, the element j is only added to U if the correlation between descriptor element j and every other element already in U is lower than a threshold. The correlation among the tests can be calculated using the Pearson correlation coefficient [137]. Otherwise, the next best element in terms of entropy is selected from T. If no element is found fulfilling the correlation criterion, the threshold is increased, and the algorithm is restarted. In the end, an ordered set sorted according to the entropy of the descriptor elements is gathered in U with correlated elements placed further at the end of the list. This order is used to transmit the residual elements that correspond to the descriptor elements with the highest entropy first. However, as ORB and FREAK already employ a similar approach, the impact on these descriptors is expected to be rather limited. In this work, the step is added regardless of the underlying feature descriptor. It is worth noticing that an ideal binary descriptor has only elements with entropy close to $H(p_{0,j}) = 1$ and mutual independence among the descriptor elements.

4.4.2 Feature classification

The visual features are categorized according to their detector response into N_g different classes. As pointed out in [37], the detector response can be used as a basic indicator for the repeatability of features. Local features with high detector response usually correspond to high-contrast corners, which are likely to be rediscovered in related views, whereas low detector responses might be caused by sensor noise and are therefore irreproducible. Each class $g \in \{z \in \mathbb{N} \,|\, 1 \leq z \leq N_g\}$ holds a fixed percentage $p_g \in \{z \in \mathbb{R} \,|\, 0 \leq z \leq 1\}$ with $\sum_{g=1}^{N_g} p_g = 1$ of the total number of features contained in one frame. The advantages of this sorting are twofold: First, this splitting is independent of the contrast properties of the images, thus avoiding fixed response thresholds. Second, fixing the percentage of features per class allows using a look-up table with pre-calculated decisions for the number of sent elements.

4.4.3 Rate optimization

The optimal number of bits to reconstruct the descriptor within each class is determined by a utility function $U_g(n_g)$, where g denotes the feature class index and n_g the number of transmitted residual elements for class g. The function returns a score estimating the task performance depending on the number of the residual elements $n_g \in \{z \in \mathbb{N} \,|\, 0 \leq z \leq N_d\}$. Regarding the corner cases, sending no residual elements ($n_g = 0$) corresponds to reconstructing only the visual word signaled by the visual word index. Adding a residual element

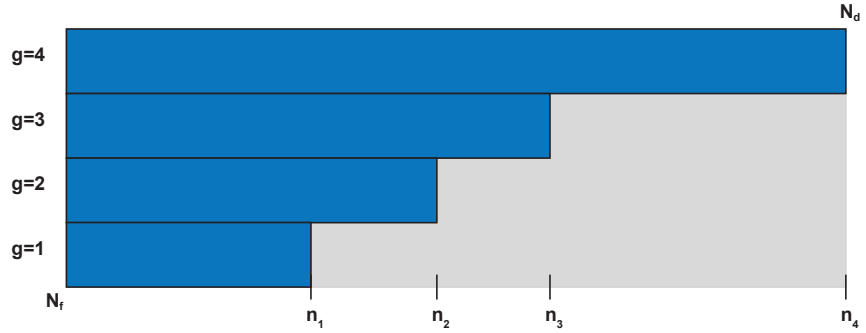


Figure 4.6: Illustration of the transmitted data. All N_f features extracted from a frame are appoximated by their visual word (grey). For class $g = 4$, containing the features with highest detector response, the residual information (blue) is added to reconstruct the full descriptor $n_4 = N_d$. For the remaining classes, less residual elements are transmitted to approximate the original descriptors.

for every descriptor element ($n_g = N_d$) for a descriptor of length N_d translates into a lossless reconstruction of the original descriptor. Hence, the number of bits $R(n_g)$ spent on each residual vector depends on the elements n_g. The rate allocation problem is given by

$$\begin{aligned} U_s = \arg\max_{n_g} \quad & \sum_{g=1}^{N_g} p_g \cdot U_g(n_g) \\ \text{subject to} \quad & N_f \cdot \sum_{g=1}^{N_g} p_g \cdot R(n_g) \leq C. \end{aligned} \tag{4.15}$$

The number of residual elements in each class should be assigned in such a way that the sum over the utility functions weighted with the share of features of each class p_g is maximized. As side constraint, the calculated total rate should obey the upper bound for the number of bits given by the available transmission rate C. The total rate is calculated as the sum over the shares p_g of each class multiplied by the total number of features N_f multiplied with the rate $R(n_g)$ used for this particular class. The concept is illustrated in Figure 4.6, where features are sorted according to their detector response into four classes and approximated with different numbers of residual elements n_g.

The utility function $U_g(n_g)$ for each class is obtained by measuring the results of a feature matching task. Finding corresponding features across several frames is one of the most fundamental challenges and used in many computer vision tasks. Hence, the utility function is defined as the share of successfully matched features in a homography estimation setup, depending on the number of residual elements n_g used for reconstruction. The features are matched across two frames and verified using the available ground truth. More details about the utility function will be provided in the experimental evaluation. In this work, the rate allocation approach has been used in conjunction with the intra coding mode, as detailed previously. However, the method is applicable to other coding methods as well.

4.5 Experimental evaluation

4.5.1 Intra coding

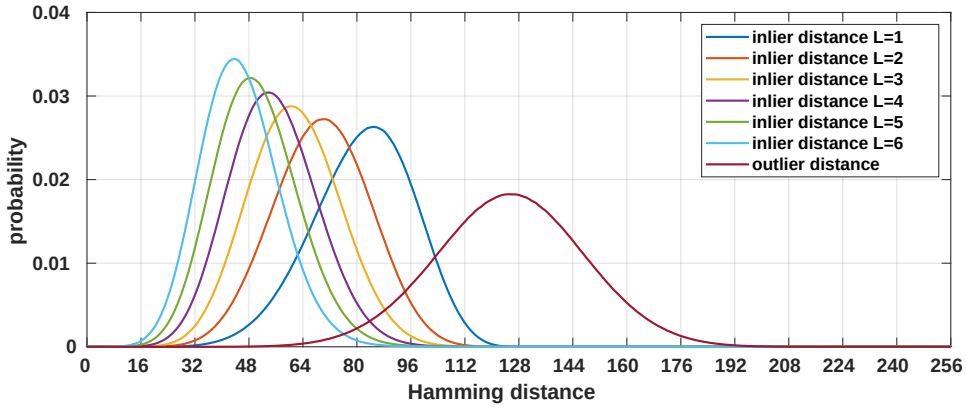
In order to evaluate the proposed intra coding scheme, different features were extracted using OpenCV and the *MIRFlickr1M* dataset as training data. Several vocabularies were trained using a hierarchical implementation of the Bag-of-Words approach for binary descriptors, namely DBoW2 [139], with branching factor $K = 10$ and the depths in the range of $L \in \{z \in \mathbb{N} \,|\, 1 \leq z \leq 6\}$ using the descriptors from the first 100k images of the dataset. A more detailed introduction of the datasets is provided in Section 2.6.

Figure 4.7 shows the Hamming distance between the descriptors and their corresponding visual word representation for different tree depths L and various features for a fixed branching factor of $K = 10$. The outlier distance denotes the Hamming distance when assigning a descriptor to a random non-matching visual word. The results are averaged over all visual words included in a particular vocabulary and are obtained using the 100k last images of the MIRFlickr dataset. The first observation is that the probabilities for Hamming distances resemble a Gaussian distribution. The second observation is that the mean values shift to lower Hamming distances when using a larger vocabulary tree. A lower Hamming distance translates into a higher probability of having zeros in the residual vector and therefore increasing the effectiveness of the residual coding. The drawback is the increased computational complexity by adding additional levels to the tree or by incrementing the branching factor. Besides, additional bits are required to cover the larger range of possible visual word indices. Increasing the vocabulary size also increases the memory footprint required to exchange the visual vocabulary or to keep the vocabulary tree structure in the memory.

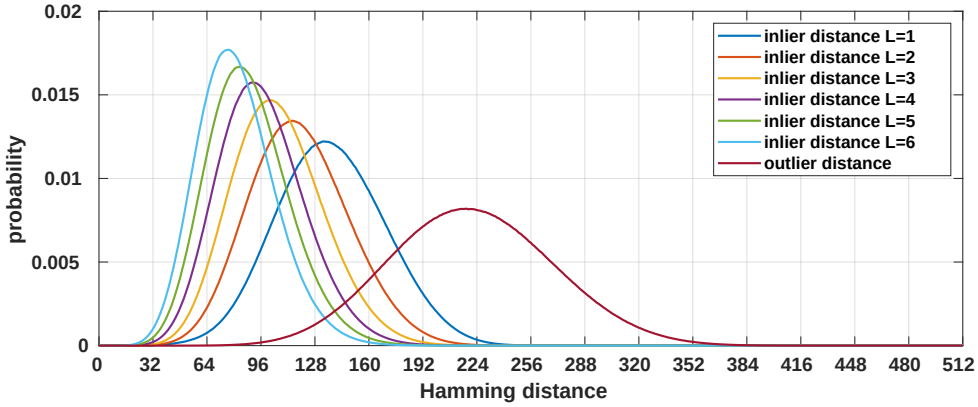
Next, the resulting compression is evaluated on the public dataset from [24], as described in Section 2.6. To this end, the performance of a homography estimation task is assessed. In order to increase the motion between the frames, the sequence was downsampled by a factor of five, as proposed by Baroffio et al. [121]. The results of the intra coding method are summarized for ORB features in Table 4.1, for FREAK features in Table 4.2, and for BRISK features in Table 4.3. The results were obtained by using 500 features extracted on each frame of the *sunset unconstrained motion* sequence. As FREAK defines only the descriptor part, the BRISK detector was used to provide the keypoint information, as proposed by the authors [48]. The keypoint properties for all features were encoded using the generic keypoint coding according to Equation (4.8) in combination with fixed length coding requiring:

$$R_{n,i}^{I,kpt} = \lceil \log_2(4 \cdot 640) \rceil + \lceil \log_2(4 \cdot 480) \rceil + \lceil \log_2(8) \rceil + \lceil \log_2(32) \rceil = 31 \text{ bits} \tag{4.16}$$

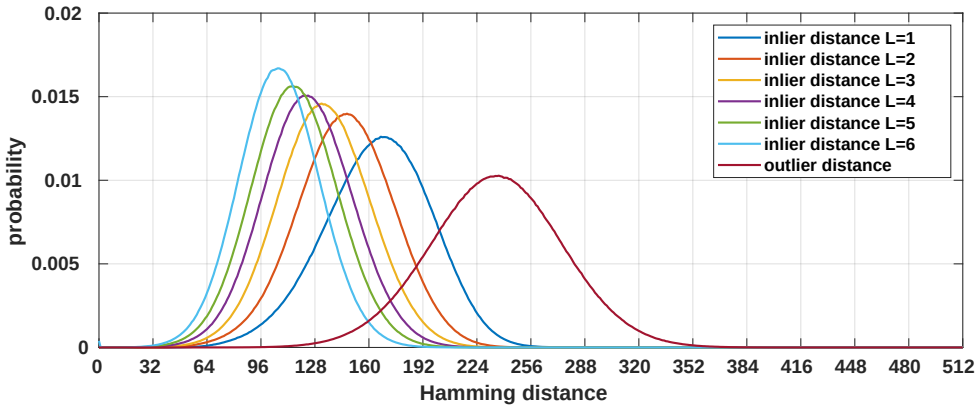
For the sake of completeness, the ORB keypoint coding requires $R_{ORB}^{I,kpt} = 24.6$ bits on this sequence. Further analysis of the lossless ORB keypoint coding is provided in Chapter 5. Additionally, the results of Equation (4.6) are marked as $R^{I,res}$, whereas the results using Equation (4.7) are denoted as $R_{+}^{I,res}$.



(a) ORB features.



(b) FREAK features.



(c) BRISK features.

Figure 4.7: Histogram of Hamming distances between the descriptors and their corresponding visual word for $K = 10$ for varying tree depth L and for ORB (a), FREAK (b), and BRISK features (c). The outlier distance denotes the Hamming distance between the descriptor and a non-matching visual word.

N_v	$R^{I,BoW}$	$R^{I,res}$	$R_+^{I,res}$	$R^{I,kpt}$	R	R_+
10	3.3	229.2	230.1	31	263.5	264.4
100	6.7	212.1	212.4		249.8	250.0
1k	10.0	198.6	197.3		239.5	238.2
10k	13.3	186.7	186.4		231.0	230.7
100k	16.6	176.5	175.7		224.1	223.3
1m	20.0	167.8	167.3		218.8	218.2

Table 4.1: Intra coding results in average number of bits for ORB features with generic keypoint coding for the *sunset unconstrained motion* sequence. ORB keypoint coding requires $R_{ORB}^{I,kpt} = 24.6$ bits. R is the total number of bits, when using the same probability table for all residual vector entries. R_+ uses individual probability tables for each residual element j as in $R_+^{I,res}$.

N_v	$R^{I,BoW}$	$R^{I,res}$	$R_+^{I,res}$	$R^{I,kpt}$	R	R_+
10	3.3	412.9	408.7	31	447.2	443.1
100	6.7	386.7	384.3		424.4	422.0
1k	10.0	364.9	362.6		405.8	403.6
10k	13.3	346.4	344.2		390.7	388.5
100k	16.6	329.3	327.1		376.9	374.7
1m	20.0	315.0	312.6		366.0	363.6

Table 4.2: Intra coding results in average number of bits for FREAK features for the *sunset unconstrained motion* sequence. R is the total number of bits, when using the same probability table for all residual vector entries. R_+ uses individual probability tables for each residual element j as in $R_+^{I,res}$.

N_v	$R^{I,BoW}$	$R^{I,res}$	$R_+^{I,res}$	$R^{I,kpt}$	R	R_+
10	3.3	461.9	448.5	31	496.2	482.8
100	6.7	439.8	423.1		477.5	460.7
1k	10.0	422.3	402.7		463.3	443.7
10k	13.3	408.4	386.1		452.7	430.5
100k	16.6	396.5	371.7		444.1	419.4
1m	20.0	386.1	358.9		437.1	409.8

Table 4.3: Intra coding results in average number of bits for BRISK features for the *sunset unconstrained motion* sequence. R is the total number of bits, when using the same probability table for all residual vector entries. R_+ uses individual probability tables for each residual element j as in $R_+^{I,res}$.

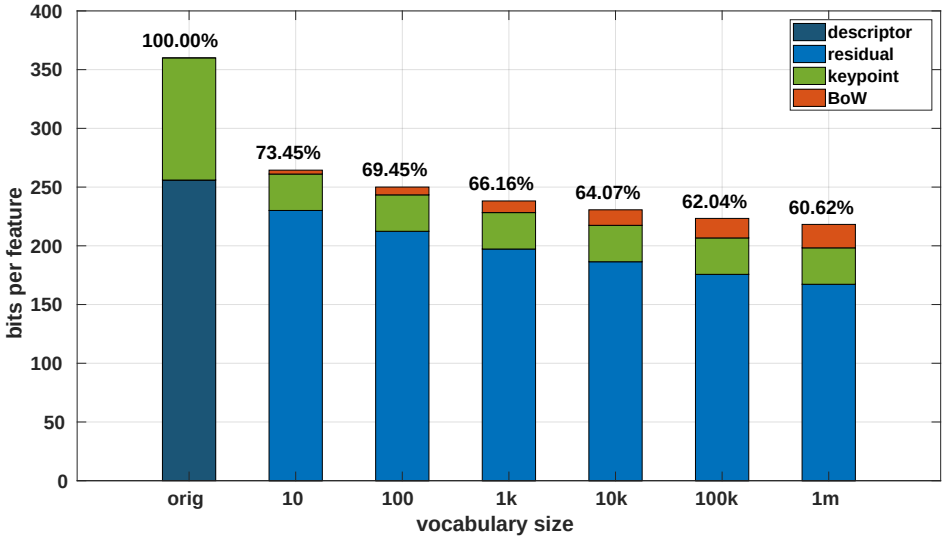


Figure 4.8: Comparison of the average number of bits per ORB feature for different vocabulary sizes extracted from the *sunset unconstrained motion* sequence (adapted from [2] ©2017 IEEE).

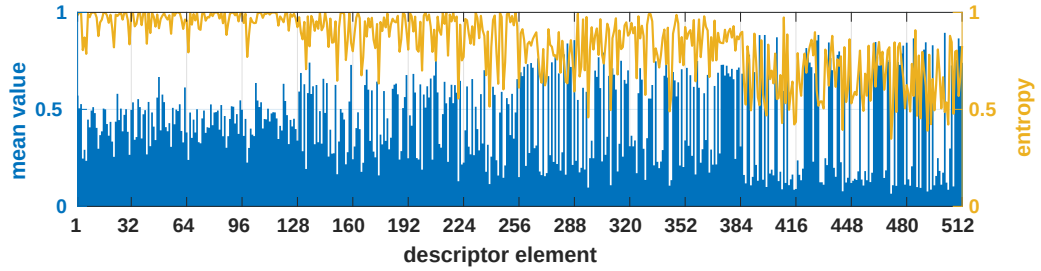
For ORB and FREAK, both formulations yield similar results in the evaluation. However, for the BRISK feature, the results improve when using individual probabilities. For ORB features, the number of bits used to store the descriptor is reduced by 34.7% from 256 bits to 167.3 bits. A reduction by 39% from 512 bits to 312.6 bits per descriptor is achievable for FREAK features. BRISK descriptors can be compressed by 30% down to 358.9 bits.

The size of the keypoint information can be reduced from 104 bits to 31 bits leading to a reduction by 70.1%. For the specific ORB feature keypoint encoding, a reduction by 76.4% from 104 bits to 24.6 bits without losing any information is possible.

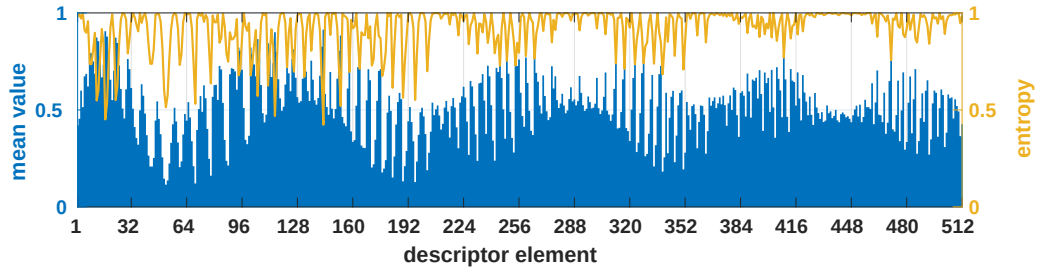
In total, ORB features can be compressed by 39.4% from 360 bits to 218.2 bits per feature, FREAK features by 41% from 616 bits (512 bits + 104 bits) to 363.6 bits, and BRISK features by 33.5% from 616 bits to 409.8 bits. It worth mentioning that the compressed representation includes besides the descriptor and the keypoint information also the visual word index, which can directly be reused in the Bag-of-Words representation.

The results of the compression for ORB features using the element-wise probabilities as in Equation (4.7) for different vocabulary sizes evaluated on the *sunset unconstrained motion* sequence are visualized in more detail in Figure 4.8. As discussed previously, the compression of the descriptor part is getting more efficient with growing vocabulary size. Although more bits are spent on signaling the visual word, the efficiency of the residual coding outweighs the additional cost for signaling values in the broader range of visual word indices.

In order to provide further insights, the expected values of the descriptor elements and the entropy were measured on the *sunset unconstrained motion* sequence and compared in Figure 4.9 for FREAK and BRISK features. The impact of the pixel test selection is visible with the first dimensions showing mean values closer to 0.5 for FREAK features (Figure 4.9a), whereas the elements at the end of the descriptor significantly deviate from this desired

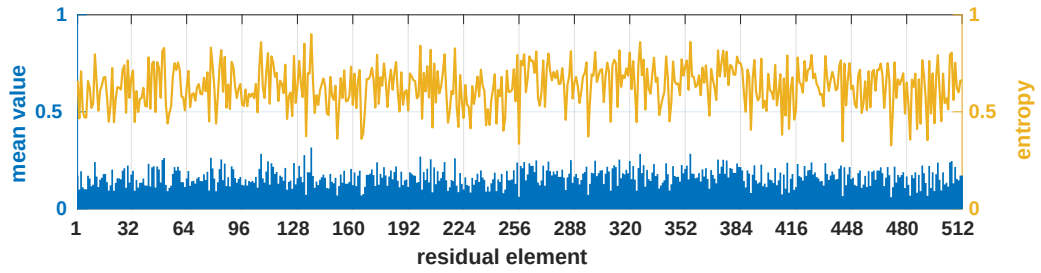


(a) Mean values and entropy for the descriptor entries of FREAK features.

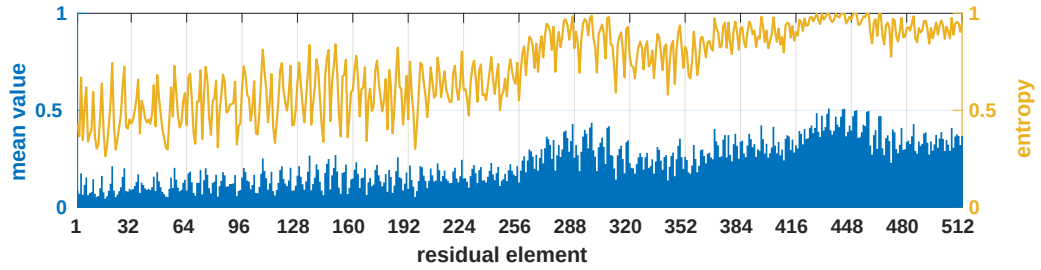


(b) Mean values and entropy for the descriptor entries of BRISK features.

Figure 4.9: Expected values and entropy per descriptor dimension evaluated on the *sunset unconstrained motion* sequence for FREAK features and BRISK features.



(a) Mean values and entropy for the residual entries of FREAK features.



(b) Mean values and entropy for the residual entries of BRISK features.

Figure 4.10: Expected values and entropy per residual dimension evaluated on the *sunset unconstrained motion* sequence for FREAK features and BRISK features.

value resulting in a lower entropy. BRISK features (Figure 4.9b) do not exhibit this ascending order in terms of the expected value. It is worth mentioning that the resorting directly exploits this lack of order in the proposed rate allocation approach.

The mean values and the entropy per entry of the residual vectors using a vocabulary size of 100k are shown in Figure 4.10. While for FREAK (Figure 4.10a), the measured expected values were more or less in a small range around 0.16, the mean values increased for the BRISK features (Figure 4.10b) towards the end of the descriptor. Here, the mean values directly correspond to the probabilities for a residual vector entry being a one, where $p_0^I = 1 - p_1^I$ is used in intra coding. This indicates that the splitting of the hierarchical vocabulary tree was not able to reduce the entropy for every descriptor element equally. In consequence, the approach of defining the probabilities of the residual being zero for each descriptor element individually is advantageous in this case. However, it also adds a computational overhead for changing the probability tables for each element.

A direct comparison of the mean values for the descriptor elements and the residual elements reveals the source of the compression gain for the intra coding mode. While the mean entropy for the first five FREAK descriptor entries is about 0.97 bit per dimension, the mean entropy for the first five elements after subtraction of the visual word is about 0.56 bit. This gap is responsible for the rate reduction.

4.5.2 Homography estimation

In order to prove the advantage of having the descriptors available, the performance of a typical computer vision task is assessed using the *homography estimation precision* (HEP) metric from [121], which is defined as follows: A set of ORB features is extracted for each frame of the sequence within the region of the planar texture as defined by the ground truth. These features are then encoded and decoded using the proposed method. A homography is estimated using feature matches between decoded features of the current and the previous frame using a RANSAC-based scheme. The four corner coordinates of the bounding box of the planar texture from the ground truth data are warped from the previous image into the current image using the estimated homography and compared with the coordinates obtained by using the ground truth directly. If the mean error is larger than 3 pixels, the estimated homography is considered as an outlier. Finally, the homography estimation precision is defined as the ratio between the number of correct estimates and the total number of estimates.

The results for the homography estimation precision are shown in Figure 4.11, where a comparison between feature and Bag-of-Words matching over varying vocabulary sizes N_v is presented. The descriptor-based matching is carried out using the minimum Hamming distance between the descriptors and additional cross-checking. This means a feature pair is only considered a match when the features are considered to be mutually the best matching feature. The Bag-of-Words matching uses only the visual word indices to establish the point correspondences. The decoded keypoints, quantized to quarter pixel resolution, are used for all experiments. For small vocabulary sizes, the number of false matches for the visual word index matching is predominant, leading the RANSAC-based estimation scheme to fail. Starting with vocabulary sizes of about $N_v = 1$k, the matching allows a reliable esti-

mation of the homography matrix. The performance decreases again for larger vocabulary sizes when similar feature descriptors are assigned to different visual words. Although the Bag-of-Words representation provided reasonable performance, it is still outperformed by matching the reconstructed descriptors.

4.5.3 Rate allocation

Next, the performance was evaluated using the proposed rate allocation scheme. To this end, the homography estimation task was reused. Visual vocabularies for the ORB, BRISK, and FREAK features using the implementations from OpenCV 3.4.1 with their default settings have been pre-trained using the DBoW2 [139] library with a branching factor $K = 10$ and depth of $L = 5$ using the *MIRFlickr1M* dataset [132]. Again, the public dataset from Gauglitz et al. [24] was used for testing and obtaining the utility curves. Here, the temporally downsampled *sunset unconstrained motion* sequence was used to obtain the utility functions by matching features across two frames and verifying the results using the provided ground truth. In total, $N_g = 4$ classes were used and the features were distributed ($p_g = 0.25, \forall g$) among all classes according to their detector response, with $g = 4$ containing the features with highest response. The utility curves for ORB, FREAK, and BRISK features with and without entropy reordering are depicted in Figure 4.12, Figure 4.13, and Figure 4.14. As expected, the curves with and without reordering for ORB and FREAK features are similar due to the training step employed in the pixel test selection, whereas for BRISK features, the curve indicates an increased performance for a reduced set of reconstructed descriptor elements. If sufficient transmission capacity is available, the entire residual vector can be transmitted. If the rate limit is reached, the remaining classes and the contained features can be skipped. The saturation of the curves when approaching the full descriptor size is worth noticing.

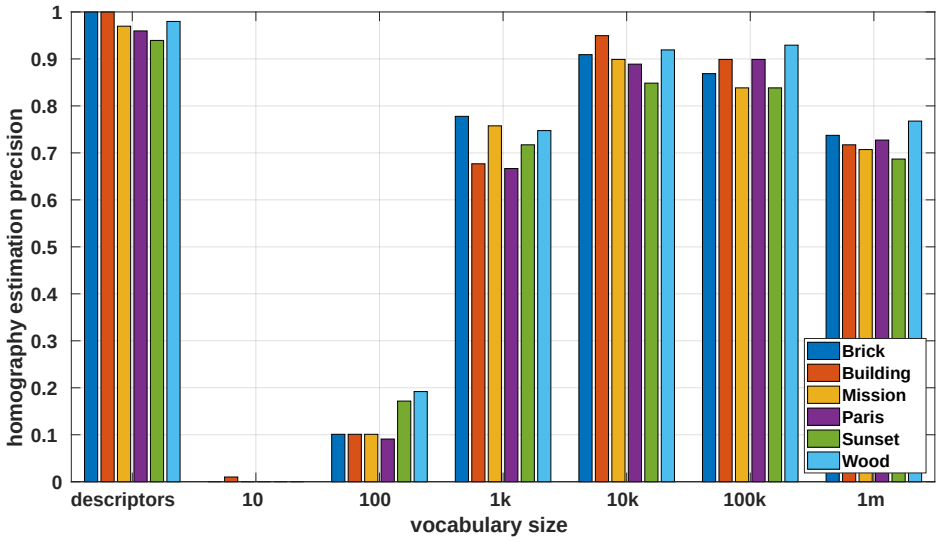
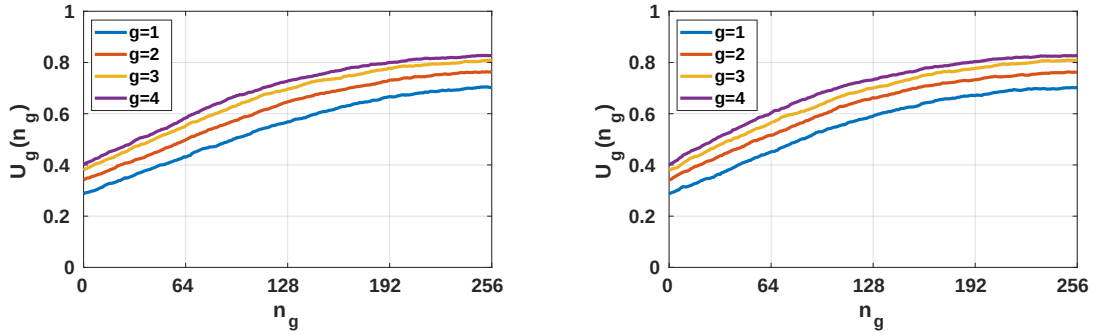
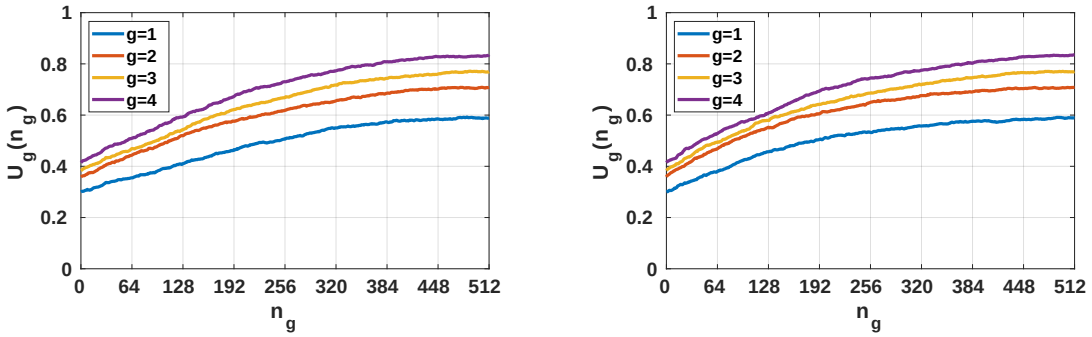


Figure 4.11: Homography estimation precision for different vocabulary sizes for different sequences. On the left, the original descriptors are used for establishing point correspondences. On the right, the visual words are used for matching (adapted from [2] ©2017 IEEE).



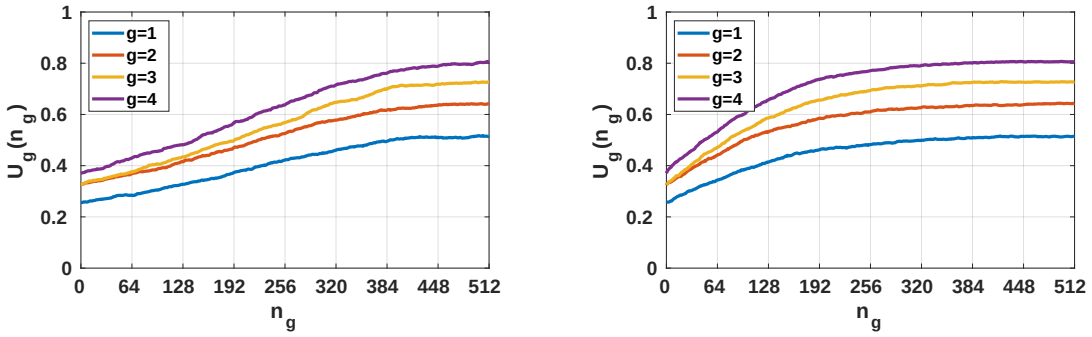
(a) ORB: Without reordering. (b) ORB: With reordering.

Figure 4.12: Utility functions $U_g(n_g)$ for ORB features using four classes at a vocabulary size of $N_v = 100$k over varying number of residual elements. Without and with reordering.



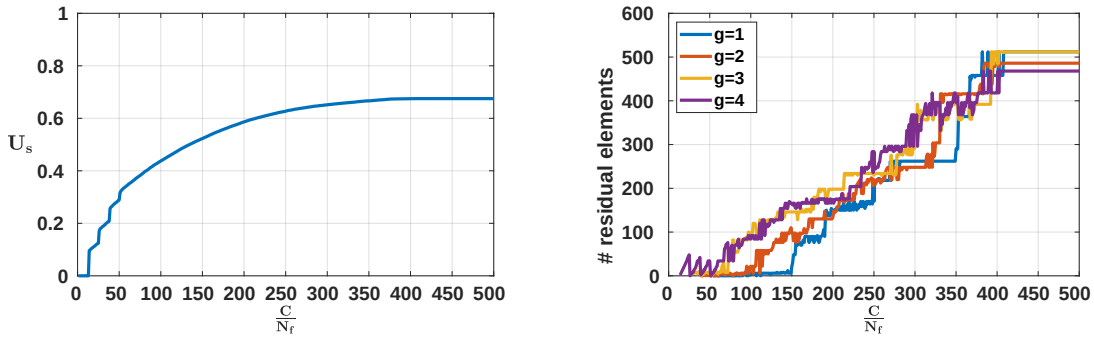
(a) FREAK: Without reordering. (b) FREAK: With reordering.

Figure 4.13: Utility functions $U_g(n_g)$ for FREAK features using four classes at a vocabulary size of $N_v = 100$k over varying number of residual elements. Without and with reordering.



(a) BRISK: Without reordering. (b) BRISK: With reordering.

Figure 4.14: Utility functions $U_g(n_g)$ for BRISK features using four classes at a vocabulary size of $N_v = 100$k over varying number of residual elements. Without and with reordering (adapted from [5] ©2018 IEEE).



(a) Summed utility function U_s.

(b) Residual element selection.

Figure 4.15: Evolution of the summed utility function U_s and the descriptor element selection per class over a varying number of average bits per BRISK feature.

Especially for the reordered BRISK features, a saturation of the utility curves is observable around 400 descriptor entries. This indicates that a similar performance of the descriptor could be achieved using fewer highly descriptive residual entries to approximate the original descriptor. While this reduces the amount of data to be transmitted, further investigation if the whole descriptor can be reduced to incorporate only the descriptor entries with high entropy can be made, but are beyond the scope of this thesis. However, this would reduce the computational complexity when extracting, storing, and matching the descriptors.

In Figure 4.15a, the evolution of the weighted sum of the utility functions U_s is shown as a function of the maximum available transmission rate C normalized by the number of features per image N_f. Figure 4.15b shows the optimal number of residual elements selected to achieve the operating point on U_s. The selection scheme was computed using a brute-force approach in an offline process beforehand. While the function U_s is a monotonically increasing function depending on the available transmission capacity C or the average bits spent per feature $\frac{C}{N_f}$, the number of selected residual elements depends on the characteristics of the utility curves for all classes. In the range of very limited channel capacities, it is beneficial to assign all bits to the highest class and skip the remaining classes. At some point (here at around 35 bits per feature), it is more advantageous to add a second class and reduce the number of bits for the highest class. In the end, the four utility curves from Figure 4.14b reach a saturation, which is also reflected with no further residual vector entries being added in Figure 4.15b. Generally speaking, residual elements from the class with the steepest utility curve at the current operating point are added. Due to noise, the measured utility curves are not monotonically increasing and thus lead to cases where residual elements are removed from a class in favor of additional residual elements from another class.

The proposed method was evaluated using the remaining sequences. Again, the HEP was employed as a quality metric. The coding costs of the individual features for this particular setup are reported in the following. The costs for keypoint coding using the generic keypoint coding and the number of bits for transmitting the Bag-of-Words indices were measured as $R^{I,kpt} = 31$ and $R^{I,BoW} = 16.6$ bits, respectively. It is worth noticing that the values

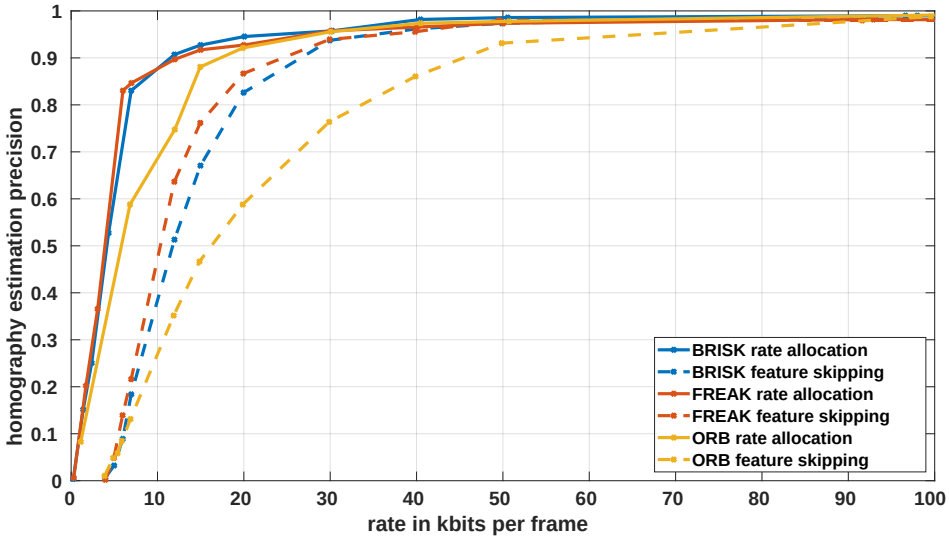


Figure 4.16: Homography estimation precision over varying bitrates obtained for the *brick*, *building*, *mission*, *pairs*, and *wood* sequences. The proposed rate allocation scheme is compared with feature skipping when exhausting the bit budget (adapted from [5] ©2018 IEEE).

for the intra coding in Table 4.1, Table 4.2, and Table 4.3 are reported for the *sunset unconstrained motion* sequence whereas here, the remaining sequences were used. For ORB features, the full residuals required 181.6 bits, FREAK features required 334.9 bits and BRISK features 383.7 bits using individual probabilities for the residual elements as described by Equation (4.7).

The results of the proposed rate allocation scheme are shown in Figure 4.16, where the HEP over varying target bitrates C for the different features is presented. The scheme is compared to a feature skipping mechanism, which is a selection on the feature level. Here, the features are sorted according to their detector response and the features with the highest detector response are prioritized for transmission. If the bit budget is exhausted, the remaining features are skipped. The results indicate a quite substantial gain in performance, especially at low bitrates. At 7 kbits per frame, the HEP for BRISK features is improved from 0.18 to 0.83. It is worth noting that it is crucial allowing to skip features of classes when operating at low bitrates, as every visual feature has fixed coding costs for keypoint and visual word information, which makes it impossible to reach very low bitrates without a feature skipping mechanism. In this setup, skipping all features in a particular class is allowed. The experiments show the results for using four classes and the features being equally distributed among the classes to keep the signaling overhead reasonable. Depending on the target task and the visual feature algorithm, other choices might also be suitable.

4.6 Summary

In this chapter, a novel intra coding mode for local binary features has been introduced. A visual vocabulary is used as shared knowledge to reduce the amount of data to be transmitted. Regarding keypoint coding, a generic keypoint coding has been added based on previous work. In addition, a novel lossless keypoint coding method exploiting the scale-space representation of ORB features has been proposed. Extensive evaluations using a homography estimation task have demonstrated an achievable reduction in required data rate between 33.5% and 41%.

In order to select the most relevant descriptor elements for transmission over a resource-constrained communication channel, a novel rate allocation scheme has been proposed. The local binary descriptors are grouped into classes according to their usefulness. The number of reconstructed descriptor elements is obtained by evaluating utility functions for each class. To transmit the most useful information first, the descriptor elements are rearranged according to their entropy. A substantial improvement has been shown for all tested features. While the approach has been evaluated in conjunction with the intra coding mode, the key idea can also be combined with alternative coding algorithms.

Chapter 5

Monocular remote visual SLAM

5.1 Problem statement

Many computer vision applications are still not feasible in real-time on mobile and embedded devices. One of the many examples is the task of performing visual SLAM. Although some authors argue that their algorithms run on mobile devices, these algorithms are typically constrained to small-scale workspaces [77], do not provide the same accuracy compared to using commodity desktop hardware or need to make sacrifices in terms of image resolution [113]. While the mobile hardware usually lags behind the current development of desktop hardware, alternative system architectures to local processing need to be considered. Parts of this chapter have been published in [3].

5.2 System architecture

In this chapter, a client-server ATC-based concept for visual SLAM is investigated to provide the results of accurate visual SLAM on embedded and mobile devices. To this end, a state-of-the-art visual SLAM system is combined with the feature coding introduced in Chapter 4. The concept is illustrated in Figure 5.1. A client extracts and encodes local binary ORB features. The server receives the compressed visual information over a communication channel, performs decoding, and runs an ORB-SLAM2 instance. A feedback channel provides information about the current tracking status of the visual SLAM system. In this chapter, the feature coding framework from Chapter 4 is extended. In order to exploit the temporal

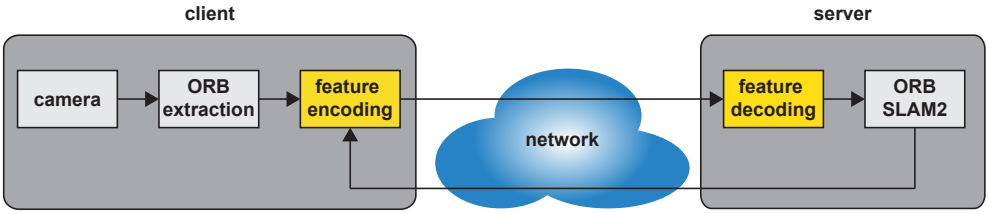


Figure 5.1: Illustration of the proposed monocular remote SLAM system. The client is responsible for acquiring the images, then extracts and encodes ORB features. A server receives the compressed features, decodes the features, and passes them to ORB-SLAM2 (adapted from [3] ©2018 IEEE).

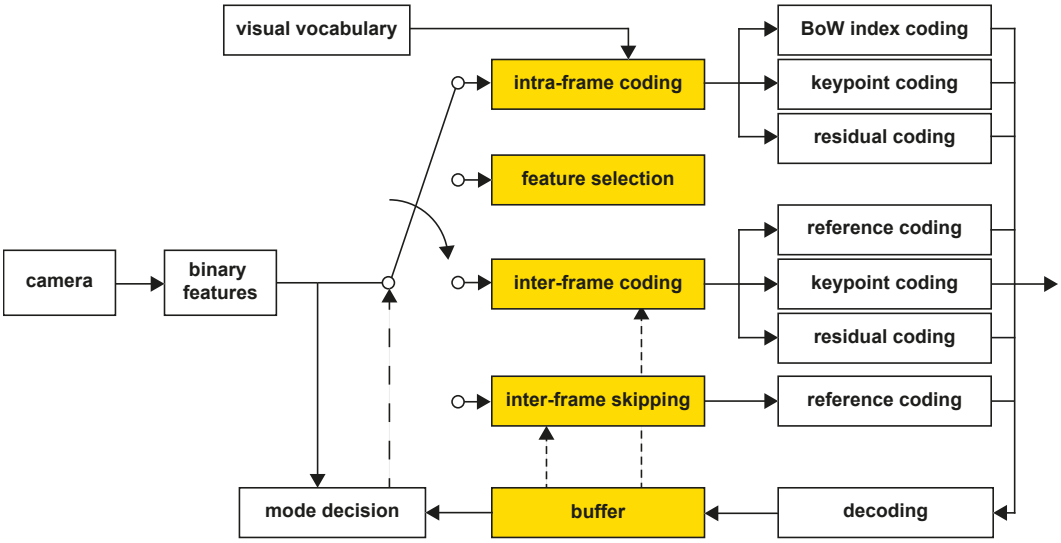


Figure 5.2: Overview of the monocular coding framework. In contrast to the previous chapter, an inter-frame coding, as well as an inter-frame skipping, is added to the existing intra coding mode. The feature prediction requires an additional buffer storing the visual information from the previous frames. A mode decision mechanism determines the best coding mode for each feature (adapted from [3] ©2018 IEEE).

redundancies in a video sequence, an inter-frame predictive coding method is added. An additional skip mode enables the algorithm to copy features from previous frames in static scenarios. A rate adaptation scheme implemented as a feature selection mechanism rapidly decides whether a feature should be transmitted over a resource-constrained communication channel or not. Information such as the tracking state is fed back to the feature encoder to adapt the transmitted data to the state of the visual SLAM system. Different aspects, such as the required data rate, real-time capability, and the trajectory error, are evaluated in an experimental setup. An overview of the coding framework used in this chapter is shown in Figure 5.2.

5.3 Feature coding

The newly proposed feature coding modes, namely the inter coding and the skip mode, implemented as an addition to the coding framework introduced in Chapter 4, are detailed in the following.

5.3.1 Inter coding

With the ability to encode individual local binary features using intra coding, the next step is to exploit temporal redundancies inherent in a video sequence. To this end, a predictive coding mode based on the method by Baroffio et al. [121], [123] is added. This approach is inspired by motion-compensated prediction in hybrid video coding. In the domain of video coding, this technique is one of the main reasons for the remarkable compression ratios achievable today. The key idea is to search in the history of previous frames for features

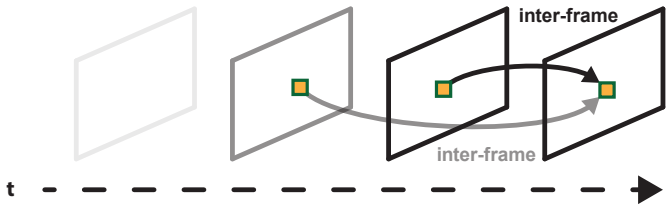


Figure 5.3: Illustration of the inter coding mode. Features from the past N_r frames can serve as a reference for predicting the current feature. In this example, the feature from the past frame is selected as a reference (adapted from [1] ©2018 IEEE).

in the near vicinity that could serve as a reference for the current feature, as illustrated in Figure 5.3. After having found and signaled a suitable candidate feature, only the differences of the keypoint properties such as the position, the orientation, and the scale-space level alongside a residual vector containing the binary differences between the descriptors need to be transmitted. The set of available coding modes is now extended to $\{I, P\}$, where I is the previously introduced intra coding mode, and P denotes the newly added predictive inter-frame coding mode. In contrast to previous work [123], the implemented inter coding mode is not restricted to a single reference frame, which allows for an efficient feature coding in the presence of short-term occlusions using a longer history of reference frames.

The number of bits for coding a local feature i from the n-th image using predictive coding is given by the costs $H_{n,i}^{P,ref}$ for signaling the reference feature, the cost $H_{n,i}^{P,res}(w)$ for the residual vector and the required bits $H_{n,i}^{P,kpt}(w)$ for the differential keypoint information between the current and the reference feature. The reference feature is index by $w \in \{z \in \mathbb{N} \,|\, 1 \leq z \leq N_{ref}^P\}$ with N_{ref}^P denoting the number of reference features. The total cost in minimally required bits is then given by

$$H_{n,i}^P = H_{n,i}^{P,ref} + H_{n,i}^{P,res}(w) + H_{n,i}^{P,kpt}(w). \tag{5.1}$$

To minimize the bitrate, the most suitable feature indexed by w^* according to the estimated number of bits is determined from the previous frames as

$$w^* = \arg\min_w (H_{n,i}^{P,res}(w) + H_{n,i}^{P,kpt}(w)). \tag{5.2}$$

In order to speed up the search, the search range is restricted to a window of size $\Delta x, y = \pm 20$ pixels around the current keypoint location and to neighboring scale-space levels as $\Delta\sigma = \pm 1$ in the past N_r frames. The individual parts of the coding costs are detailed in the following.

5.3.1.1 Reference coding

First, an index indicating the reference feature has to be signaled using

$$H_{n,i}^{P,ref} = \log_2(N_{ref}^P) \tag{5.3}$$

bits assuming uniform probabilities. In the current scenario, the costs $H_{n,i}^{P,ref}$ for coding the reference features is assumed to be static for a fixed number of N_f features per frame and a

fixed number of N_r reference frames calculated as

$$N_{ref}^{P} = N_f \cdot N_r. \tag{5.4}$$

5.3.1.2 Residual coding

The costs for transmitting the residual vector can be derived similarly to the costs for the intra coding mode. The Hamming distance between the current feature descriptor $\mathbf{d}_{n,i}$ and the reference feature descriptor $\mathbf{d}_{w^*}$ is given by

$$h_{n,i}^{P}(w^*) = h(\mathbf{d}_{n,i}, \mathbf{d}_{w^*}). \tag{5.5}$$

The lower bound in bits for a specific feature descriptor i from image n can be calculated similar to Equation (4.6) as

$$H_{n,i}^{P,res}(w^*) = -(N_d - h_{n,i}^{P}(w^*)) \cdot \log_2(p_0^P) - h_{n,i}^{P}(w^*) \cdot \log_2(1 - p_0^P), \tag{5.6}$$

where p_0^P is the specific probability of a zero value in the residual vector for the predictive inter coding mode.

5.3.1.3 Keypoint coding

In contrast to the intra-frame keypoint coding, only the differences with respect to the reference feature have to be transmitted. Again, a differentiation between generic keypoint coding and a novel ORB specific keypoint coding method is possible.

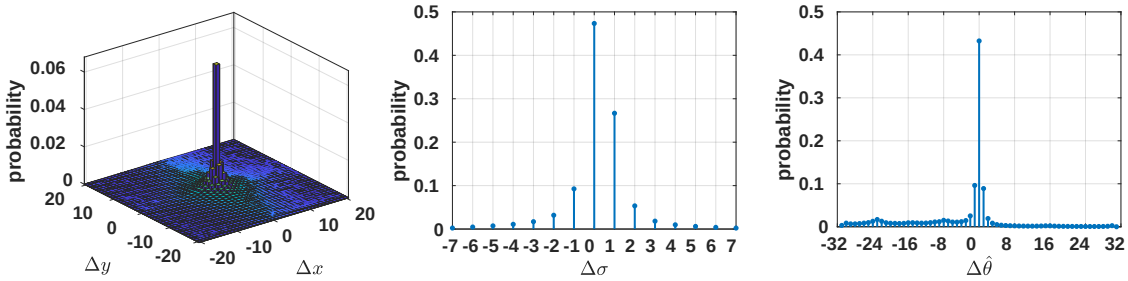
Generic keypoint coding: The number of bits required to transmit the differences is given by the sum of bits needed to transmit the difference in the location $\Delta x_{n,i}(w^*), \Delta y_{n,i}(w^*)$, the scale-space level $\Delta\sigma_{n,i}(w^*)$, and the orientation $\Delta\hat{\theta}_{n,i}(w^*)$. For the sake of readability, the dependence of the values on w^* is dropped in the following notation. The number of bits for the keypoint differences is then given as

$$H_{n,i}^{P,kpt}(w^*) = -\log_2(f_{\Delta p}(\Delta x_{n,i}, \Delta y_{n,i})) - \log_2(f_{\Delta\sigma}(\Delta\sigma_{n,i})) - \log_2(f_{\Delta\hat{\theta}}(\Delta\hat{\theta}_{n,i})). \tag{5.7}$$

Here, the assumption is made that smaller displacements in terms of pixel coordinates occur more often. The same holds for the differences in scale-space level and orientation. Therefore, different probability mass functions f_Δ for each entity are measured and used in combination with arithmetic coding to signal the differences.

ORB keypoint coding: For the ORB specific coding, the probability mass functions $f_{\Delta p}^{s}$ for the difference in keypoint location are obtained individually for each scale-space level. For the keypoint displacement, the difference is scaled back to the scale-space level of the current keypoint to ensure integer accuracy using Equation (4.9) as

$$\Delta x_{n,i}^{s} = round(\frac{x_{n,i} - x_{w^*}}{f_s(\sigma_{n,i})}) \qquad \Delta y_{n,i}^{s} = round(\frac{y_{n,i} - y_{w^*}}{f_s(\sigma_{n,i})}), \tag{5.8}$$



(a) Position difference at $\sigma = 0$. (b) Scale-space level difference. (c) Orientation difference.

Figure 5.4: Probabilities used for differential keypoint coding for the keypoint displacement (a), the scale-space difference (b), and the difference in orientation bins (c). Only very few entries contain significant probabilities thus making entropy coding efficient.

where $\Delta x^s_{n,i}$ and $\Delta y^s_{n,i}$ denote the scaled differences in x and y direction. Depending on the settings for the scaling factor and the maximum allowed scale-space difference, the calculation of the keypoint displacements produces unique solutions for features matched across neighboring scales as well. In order to resolve ambiguity, a restriction to matching feature on the same scale-space level can be enforced.
The number of bits for coding the keypoint difference is then given by

$$H^{P,kpt}_{n,i}(w^*) = -\log_2(f^s_{\Delta p}(\Delta x^s_{n,i}, \Delta y^s_{n,i}, \sigma_{n,i})) - \log_2(f_{\Delta\sigma}(\Delta\sigma_{n,i})) - \log_2(f_{\Delta\hat{\theta}}(\Delta\hat{\theta}_{n,i})), \quad (5.9)$$

where $\Delta\sigma_{n,i}$ denotes the difference in scale-space level and $\Delta\hat{\theta}_{n,i}$ the difference in orientation bins with respect to the best reference feature w^*. The probability mass functions for the ORB keypoint coding are illustrated in Figure 5.4. The distribution of the probabilities indicate that small differences between features are more likely to occur for all properties, thus motivating the use of entropy coding.

5.3.2 Skip mode

In cases where no camera motion is present and the scene is static, inter coding still produces a considerable amount of data. In order to address this issue, a feature skipping mode is implemented. A feature situated at exactly the same location with the same keypoint properties as the current feature has to be present in previous frames. In this case, only the index of the reference feature is transmitted and the reference feature is copied. In order to account for sensor noise and flickering lights, a Hamming distance difference of $h^S_{n,i} < 5$ is tolerated. In consequence, the available set of coding modes is extended by the skip mode S to $\{I, P, S\}$.

5.3.3 Mode decision

The central element of the coding framework is the mode decision algorithm. It calculates the expected number of bits for each coding mode using Equations (4.2, 5.1) and determines the best coding mode m^* for each feature as

$$m^* = \arg\min_m H^m_{n,i}. \quad (5.10)$$

This calculation is done for each feature individually and is implemented in a parallel fashion. Lookup tables are used to speed up the calculation by avoiding the evaluation of the logarithms contained in the rate calculation.

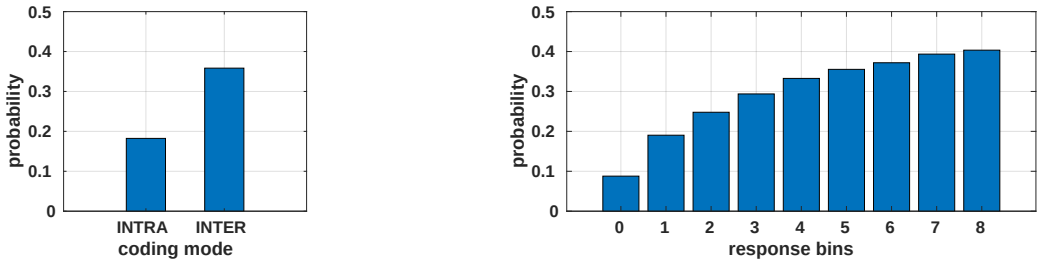
5.3.4 Rate adaption

To ensure constant time encoding and meeting the bitrate constraints predetermined by the transmission capacity C, an approach is introduced that facilitates feature selection. It ranks the features according to a metric that defines how useful a specific local feature is for the targeted task without adding significant overhead. The ORB feature extraction parameters are not changed, but rather a decision based on the feature properties is made whether a visual feature is contributing significantly to the result of the remote visual SLAM system or not. In contrast to the rate allocation introduced in Chapter 4, the feature selection operates on the feature level rather than on the descriptor level. The main reason for using this technique here is that it only adds minimal overhead and provides a simple mechanism to stop the feature encoding when a time budget is exhausted by just skipping the remaining features.

5.3.4.1 Feature selection

Similar to Francini et al. [37], various relevance statistics have been measured for the features used for map point creation in the visual SLAM system. For this purpose, ORB-SLAM2 was used to build a map with features processed by the coding framework. The properties of the features that were successfully triangulated and, thus, serve as observations of map points were analyzed using this static map. While Francini et al. considered most keypoint properties, such as position in the image, detector response, scale-space level, and orientation to determine the usefulness of features for a visual search task, this work focuses on visual SLAM as the target application with different requirements.

After dissecting the individual properties, both the detector response s given by the FAST corner detector [35] employed in ORB and the coding mode decision m from the feature coding algorithm is used for the relevance score. The underlying assumption is that a stronger detector response indicates a high contrast corner, which should be more reproducible in different views of the same scene compared to low contrast corners, which conversely could be a result of sensor noise. The inter coding mode is used for features that are visible in consecutive frames. Intuitively, the more often a corner is seen in adjacent views, the higher the probability of being triangulated to a map point. An analysis of the remaining keypoint properties, such as the orientation $\hat{\theta}$, the scale σ, and the keypoint locations has additionally been performed. While the orientation does not provide additional information, it is observable that higher scale-space levels exhibit a higher triangulation probability. However, the prioritization of certain scale-space levels renders ORB-SLAM2 unable to keep tracking due to the reduced scale invariance. Regarding the keypoint position, Francini et al. use the distance from the image center as ranking criteria. In contrast to a visual search task, where the object of interest is usually located in the center of the image, a visual SLAM system performs best when considering features that are adhered to different objects distributed across



(a) Matching probabilities depending on the coding modes.

(b) Matching probabilities depending on the quantized response values.

Figure 5.5: Triangulation probabilities used for the feature selection based on a map of the *EuRoC MH01* sequence (adapted from [3] ©2018 IEEE).

the scene [96], [140]. This increases the stability of the camera pose estimation by avoiding degenerated cases in the pose estimation process. In contrast to the objectives of visual search, the authors of ORB-SLAM2 specifically try to distribute the feature across both the image and the scale-space to make the tracking more robust and scale invariant [69].

In order to formulate the problem, the notation of Francini et al. [37] is used. The binary variable c is introduced to indicate whether a feature serves as an observation of a map point ($c = 1$) or has not been associated with the map ($c = 0$). For the response value $s \in \{z \in \mathbb{N} \,|\, 0 \leq z < 180\}$, a bin size of 20 is used and the response range is quantized accordingly into bins. Values exceeding this range are assigned to the highest bin. For the quantized response value $\hat{s} \in \{z \in \mathbb{N} \,|\, 0 \leq z < N_{\hat{s}}\}$ with $N_{\hat{s}} = 8$, the probability of correct matches given a response value is calculated using Bayes formulation as follows

$$P(c = 1 | \hat{s} = \hat{s}_{n,i}) = \frac{P(c = 1 \cap \hat{s} = \hat{s}_{n,i})}{P(\hat{s} = \hat{s}_{n,i})}. \tag{5.11}$$

For the coding mode, a distinction between the intra and inter coding modes is made

$$P(c = 1 | m = m_{n,i}) = \frac{P(c = 1 \cap m = m_{n,i})}{P(m = m_{n,i})}. \tag{5.12}$$

The resulting statistics are shown in Figure 5.5. Both the coding mode and the response values are assumed to be independent and thus allowing to calculate the combined relevance score $r_{n,i}$ by a multiplication of the individual scores as

$$r_{n,i} = P(c = 1 | \hat{s} = \hat{s}_{n,i}) \cdot P(c = 1 | m = m_{n,i}). \tag{5.13}$$

Based on these criteria, a score for every feature is calculated and used to sort the features. Only relevant features up to a certain time limit or bit budget are transmitted.

5.3.5 Feedback channel

A reliable initialization is crucial for a visual SLAM system. If the initial map is erroneous, the subsequent tracking has very little chance to estimate the correct frame position. The

authors of ORB-SLAM2 double the number of ORB features for the frames in the uninitialized monocular case. However, they use only the features extracted at the first level of the scale-space pyramid to estimate the initial transformation between two frames. The features detected at other scale-space levels are also stored and can later be used for creating map points or perform loop closing. In order to avoid sending twice the number of features for initialization, the number of extracted features is also doubled, but the number of features sent is kept constant. In this scheme, features extracted at the first scale-space level are prioritized during the initialization phase, and features at higher levels are skipped to keep the number of features constant. The motivation is to avoid a bursty traffic pattern by sending twice the number of features whenever the SLAM system is initializing or has lost track and needs to reinitialize. In order to signal when to return to the usual mode decision, a feedback channel is added where information about the current tracking state is fed back to the sender. More specifically, if the initialization is completed or when tracking is lost, feedback messages are sent. In the latter case, features from the first scale-space level are prioritized.

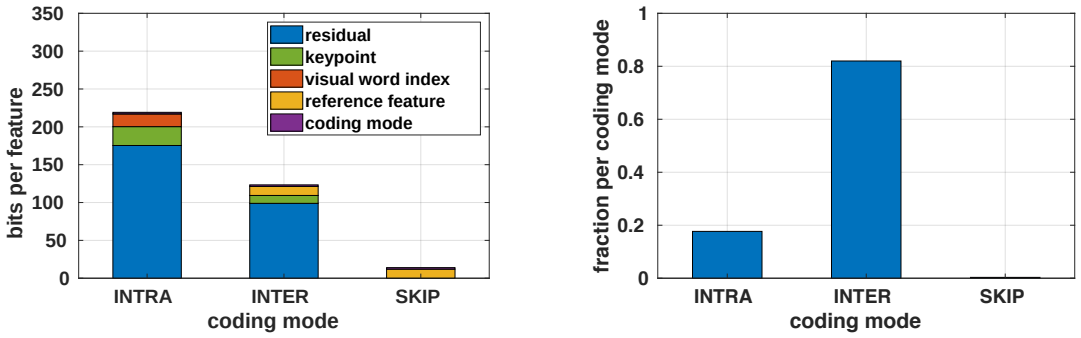
5.4 Experimental evaluation

5.4.1 Feature coding

For the intra coding scheme, the same approach as introduced in Chapter 4 is used. ORB features were extracted from the *MIRFlickr1M* dataset as training data. A hierarchical visual vocabulary using the DBoW2 library has been trained beforehand. Although ORB-SLAM2 already includes a pre-trained visual vocabulary, which has been created with a branching factor $K = 10$ and a tree depth $L = 6$, a smaller vocabulary with the same branching factor, but a depth of $L = 5$ is employed here due to size constraints. The goal is to run the client on embedded devices with a limited maximum memory footprint. Therefore, a smaller vocabulary taking only 14.8 MB replaces the original vocabulary, which consumes 145.3 MB. The probabilities p_0^m have been pre-trained using the industrial sequences from the *EuRoC* dataset. The properties for the feature selection have been obtained from a map of the *EuRoC MH01* sequence. For the evaluation, the *TUM RGB-D* dataset was employed. A more detailed introduction of the datasets is provided in Section 2.6.

For all experiments, the default settings for the *TUM RGB-D* dataset provided by ORB-SLAM2 were employed extracting 1k features per frame. Both encoder and decoder ran simultaneously on a single computer equipped with an Intel i7-3770 CPU @ 3.40 GHz and using the Robotic Operation System (ROS) [141] for the bidirectional communication. First, the feature sizes of the individual coding modes are shown in Figure 5.6. For the intra-coding mode, on average 175.4 bits for the residual coding, 24.9 bits for the keypoint properties and 16.9 bits for the visual word indices are spent. The coding mode is signaled with two additional bits, resulting in 219.2 bits per feature for the intra coding.

For the inter coding, 12 bits are required for signaling the reference features using $N_r = 4$ reference frames with each $N_f = 1$k features. About 98.9 bits for the descriptor and 10.4 bits for the keypoint differences are required, resulting in 123.3 bits per feature including the



(a) Bits per coding mode.

(b) Fraction per coding mode.

Figure 5.6: Bits per feature for all coding modes using $N_r = 4$ reference frames on the *fr3/long_office* sequence. Intra coding mode consumes the most bits per feature. Exploiting temporal redundancies results in a substantial reduction of the required bits. In this sequence, the majority of the features are encoded using the inter coding mode (adapted from [3] ©2018 IEEE).

	$\mathbf{I}$	$\mathbf{I+P_1}$	$\mathbf{I+P_4}$
encoding	19.4 ms	21.0 ms	26.3 ms
decoding	25.4 ms	23.0 ms	23.6 ms
kbits/frame	219.2	161.1	140.6

Table 5.1: Timing results are the median over ten executions using 1k features for varying numbers of reference frames N_r for the inter coding evaluated on the *fr3/long_office* sequence. The consumed time increases when using intra coding and moving to more reference frames, whereas the required number of bits reduces (adapted from [3] ©2018 IEEE).

two bits for the coding mode. The inter-skip mode only needs to send the reference feature index and the coding mode.

The resulting bitrate and coding times for a different number of reference frames are compared in Table 5.1. The uncompressed size is given with 360 kbits/frame using 1k features per frame and 360 bits per feature, as calculated in Chapter 4. Intra coding provides a reduction by 39.1%, inter coding with a single reference frame a reduction by 55.3%, and four reference frames a reduction by 61%. The encoding time increases from 19.4 ms using only intra coding to 26.3 ms using four reference frames.

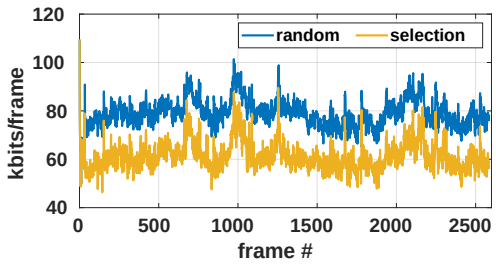
5.4.2 Feature selection

Next, the feature selection is evaluated by limiting the available bit budget to 75 kbits per frame. The SLAM performance is measured in terms of *absolute trajectory error* (ATE) [134] and is shown in Table 5.2. As monocular SLAM does not provide any scale, the resulting trajectory is aligned and scaled with the ground truth. Although, the bitrate is drastically reduced to 75 kbits/frame, which is a reduction by 79.2% compared to the uncompressed size of 360 kbits/frame, the sequences are processed with only a sub-cm loss in accuracy.

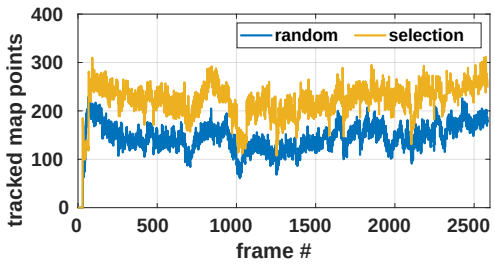
In order to provide additional insights on the effects of the feature selection, Figure 5.7

sequence	75 kbits/frame [cm]	1k features/frame [cm]
fr2/desk	1.18	**1.08**
fr2/desk_person	1.35	**1.18**
fr3/long_office	1.35	**1.23**
fr3/nostr_tex_near_wl	1.57	**1.40**
fr3/sit_halfsphere	1.91	**1.81**
fr3/str_tex_far	**1.11**	1.14
fr3/str_tex_near	1.37	**1.31**

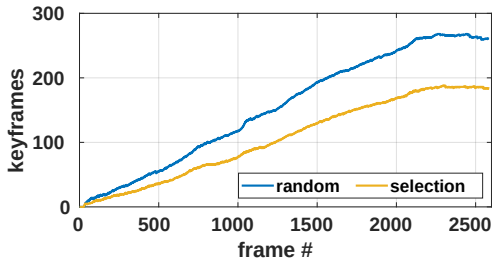
Table 5.2: Comparison of the ATE in cm for sequences from *TUM RGB-D*. Results are reported as median values over 10 executions for each sequence. The trajectories are aligned and scaled to the ground truth using 7DoF alignment by similarity transformation using the provided dataset tools. The results show that accurate results in the range of sub-cm accuracy can be achieved while reducing the bitrate to 75 kbits/frame (adapted from [3] ©2018 IEEE).



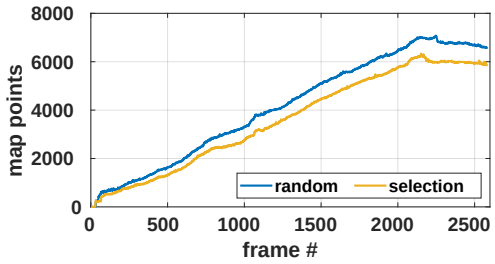
(a) Bitrate profile.



(b) Map points used for tracking.



(c) Evolution of the number of keyframes in the map.



(d) Evolution of the number of map points in the map.

Figure 5.7: Evolution of the map properties using a random selection of 500 features and the proposed feature selection scheme on the *fr3/long_office* sequence. The bitrate (a), the keyframes (c) and the map points (d) are reduced, whereas the number tracked map points (b) is increased (adapted from [3] ©2018 IEEE).

		75 kbits/frame [ms]	1k features/frame [ms]
client	**ORB**	13.5	
	encoding	17.4	26.3
	total	31.4	40.5
server	**decoding**	13.4	23.6
	tracking	11.6	18.8
	total	25.4	43.9

Table 5.3: Timing results in ms for *fr3/long_office* using four reference frames as median over 10 executions. Both the bitrate and the timings are reduced when limiting the bitrate allowing for real-time processing (adapted from [3] ©2018 IEEE).

shows the bitrate profile, the map points used for tracking the current frame, the number of keyframes and the number of map points present in the map measured at each frame of the *fr3/long_office* sequence. The feature selection is compared to random sampling selecting 500 out of the 1k extracted features. It is worth noticing that prioritizing inter coded features improves both the tracking stability and the bitrate (Figure 5.7a) at the same time. Here, the number of tracked map points that are connected with features in the current frame can serve as an indicator for the tracking stability (Figure 5.7b). Intuitively, the more map points are used for estimating the current frame pose, the more reliable is the result. If the map points are highly stable and repeatable, fewer map points are required to provide stable tracking. Hence, the number of both keyframes (Figure 5.7c) and map points (Figure 5.7d) in the map can be significantly reduced by the selection step. Besides, reducing the number of features, map points, and keyframes reduces the memory footprint of the map and lowers the computational burden, as shown in the next section.

5.4.3 Computational complexity

The computational complexity is analyzed in Table 5.3. The median time to track a single frame is reduced to 11.6 ms using only a selected subset of features compared to 18.8 ms using 1k features. To show the performance in mobile applications, additional tests were performed on an embedded platform using an NVIDIA Jetson TX2. Without any specific optimization, 1k features are encoded within 34.2 ms using one and 50.4 ms using four reference frames at 1.4 GHz while using roughly 5.5 Watts. At 2.0 GHz it takes 25.3 ms and 34.6 ms respectively while consuming about 6.7 Watts. In this evaluation, parallelization was only used for the mode decision thus leaving room for further improvement.

5.5 Summary

In this chapter, a monocular remote visual SLAM system architecture has been proposed. Based on the Analyze-then-Compressed approach, the local features are extracted and compressed at the client-side and transmitted to a server for building a visual SLAM map. A feature selection method ensures to stay below the upper bounds regarding real-time processing and transmission capacity. At the server-side, the local binary features are decoded and fed into the state-of-the-art ORB-SLAM2 visual SLAM system. The performance was evaluated on several sequences of the well-known *TUM RGB-D* dataset. In conclusion, the system was capable of reducing the bitrate by 39.1% using intra coding, 55.3% using a single reference frame, and 61% using four reference frames. Moreover, by skipping features, a reduction by 79.2% down to 75 kbits/frame is possible while being able to run ORB-SLAM2 with only a sub-centimeter degradation in accuracy. In addition, the computation time of ORB-SLAM2 and the resulting map size were reduced by using only selected features.

Chapter 6

Stereo remote visual SLAM

6.1 Problem statement

While the remote visual SLAM architecture introduced in the previous chapter is capable of providing monocular visual SLAM, real-world robotic applications usually require a metric scale map representation. Cues to determine the scaling factor can be obtained by using additional sensors, such as range scanners, IMUs, or others. Apart from additional sensors, the size of known objects such as traffic signs [8] or hallways [142] can be used. Another alternative investigated in this work uses multiple visual sensors in a multi-camera setup. In this chapter, the focus lies on the scale information extracted from a calibrated stereo camera setup. Hence, the previously introduced coding methods should be extended by additional modes for the compression of additional depth or stereo information.

6.2 System architecture

The system architecture used in this chapter is an extension of the system described in Chapter 5 and is depicted in Figure 6.1. The client is equipped with a stereo camera system with known intrinsic camera calibration and a known stereo baseline between the two cameras. The ORB feature extraction is running on each stereo image pair and the features are passed to the feature encoding. The compressed features are then sent over a network connection to a server instance. The server is running a feature decoding process and passes the decoded features to a visual SLAM system.

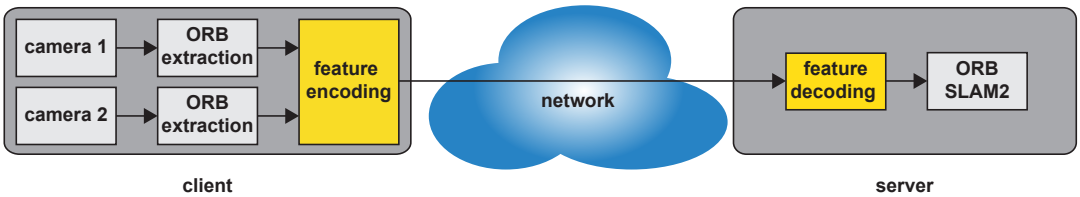


Figure 6.1: Illustration of the ATC-based metric scale visual SLAM architecture. The client extracts visual features from both views and applies feature encoding. The server receives and decodes the feature streams. Subsequently, the features are fed into the visual SLAM system (adapted from [1] ©2019 IEEE).

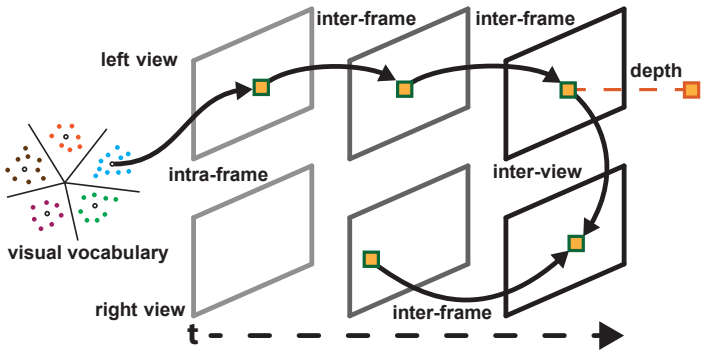


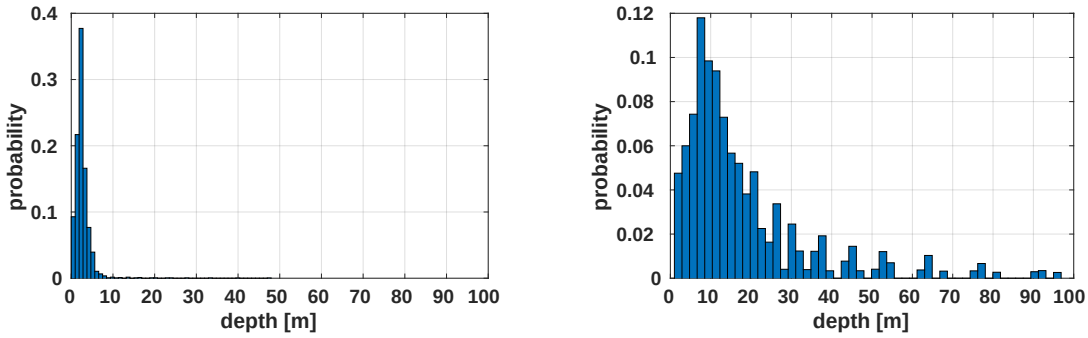
Figure 6.2: Illustration of the different coding modes. The prediction modes exploit either a visual vocabulary (intra-frame), temporal (inter-frame), or spatial correlations (inter-view) between local features. Alternatively, quantized depth values can be transmitted (adapted from [1] ©2019 IEEE).

For purely vision-based SLAM, a stereo, or more general, a multi-camera setup is essential to provide metric scale information. Hence, the monocular feature coding is extended in this chapter to include a *depth value coding*, where the depth values are acquired either directly from a depth sensor in an RGB-D camera or by matching features between stereo-views with a known baseline. The latter is primarily assumed in this work. Instead of sending the floating-point depth representation, a non-uniform scalar quantization scheme, which is tailored to typical depth values, is employed to reduce the required amount of data.

If the local features from both views are required, a *stereo-view coding* scheme exploiting the spatial correlation between the two views is needed. The concept of coding features extracted from arbitrary visual sensor networks [135] is used to encode the visual information efficiently. Similar to the inter coding that exploits temporal redundancies, spatial redundancies are exploited in this scheme by transmitting only the differences between matched stereo features. This chapter adds the missing parts to complete the framework depicted in Figure 3.1 in Chapter 3.

6.3 Feature coding

The set of available coding modes is extended to $\{I, P, S, M\}$, where I denotes the intra-frame coding (Chapter 4), P the predictive inter-frame coding, S the skip mode (both Chapter 5), and M the new multi-view stereo coding mode. The presence of additional depth data is indicated by D. The concepts are illustrated in Figure 6.2. Due to occlusions, estimating the depth is not always possible, and visual features close to the camera system can result in considerably aberrant visual descriptors. Hence, the system is designed such that for features in the right view, the coding modes introduced for monocular feature coding can be selected as well. Regarding the mode decision, the rate for all eligible coding modes (I, P, S for the left view and M additionally for the right view) is calculated per feature, and the mode with the lowest number of bits is selected. The number of bits for the depth and the stereo coding mode is detailed in the following.



(a) Depth histogram for the *EuRoC V101* sequence.

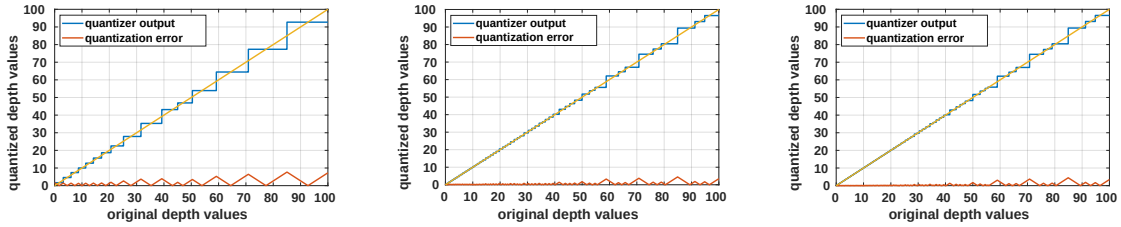
(b) Depth histogram for the *KITTI 07* sequence.

Figure 6.3: Histogram of the depth values obtained from the *EuRoC V101* (a) and the *KITTI 07* (b) sequences. The depth values heavily depend on the dataset scenario (indoor vs. outdoor). Hence, both datasets have been used to obtain the quantization curves.

6.3.1 Depth coding

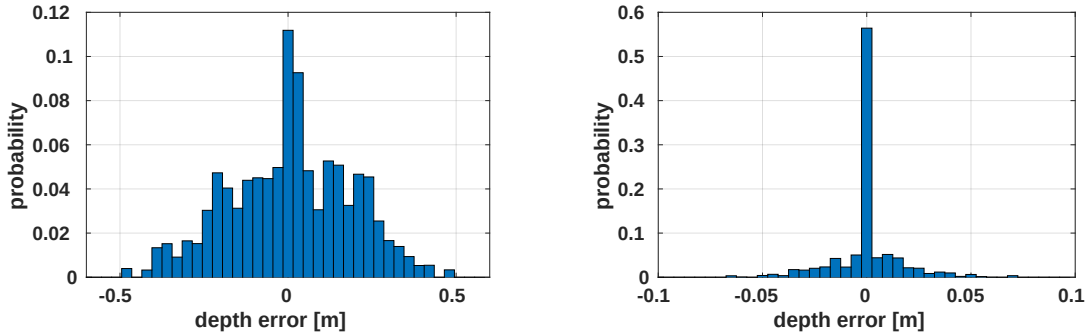
The depth coding is used to augment the visual information with additional depth information for each feature of the monocular feature stream of the left view. To this end, the depth is either acquired directly by reading the depth sensor data from an RGB-D camera or by stereo feature matching in case of a stereo camera setup. For the latter, a pre-calibrated camera system with known intrinsic and extrinsic camera calibration is required. The feature correspondences between neighboring views are established by identifying feature matches located on the same scale-space level along the epipolar line. To account for imperfect camera calibration, a deviation of two pixels from the epipolar line in their respective scale-space representation is allowed. When estimating the depth, the result is usually stored in a single-precision floating-point data structure with 32 bits per depth value. Here, the question arises, whether a quantized version of this representation is sufficient to create accurate metric scale visual SLAM maps. To this end, this work proposes to use a *non-uniform quantization* scheme, which is trained on different sample trajectories with the goal of minimizing the reconstruction error. The data rate can be further reduced by exploiting the observation that, due to occlusion, not every feature can be matched to estimate a corresponding depth value. Hence, for every feature, a single bit is sent, indicating if the feature is accompanied by depth information. The depth reconstruction level is then subsequently signaled using N_D bits.

The quantization characteristics for coding the depth values have been obtained on both the *KITTI* and the *EuRoC Machine Hall* sequences to include different depth ranges in indoor and outdoor scenarios into the pre-trained quantization codebook. Typical depth values obtained from the *EuRoC V101* and *KITTI 07* sequence are shown in Figure 6.3. The difference in the depth values in indoor and outdoor scenarios prove the necessity to train the depth quantization on both scenarios to cover the different environments. For the *EuRoC* sequence, the room size for the *EuRoC V101* dataset is said to be approximately 8 m x 8.4 m x 4 m [54], which is reflected in the depth values staying roughly below 10 m. The majority of features



(a) $N_D = 4$ bits. (b) $N_D = 6$ bits. (c) $N_D = 8$ bits.

Figure 6.4: Depth quantization curves for $N_D = 4$ bits (a), $N_D = 6$ bits (b), and $N_D = 8$ bits (c). The quantization error reduces when using more quantization levels at the cost of increased number of bits for signaling the level.



(a) Quantization error with $N_D = 6$ bits. (b) Quantization error with $N_D = 8$ bits.

Figure 6.5: Histogram of the depth quantization error measured on the *KITTI 00* sequence for $N_D = 6$ bits (a) and $N_D = 8$ bits (b).

for the *KITTI* sequence are placed at about 10 m ranging up to 50 m or more.

The quantization curves including the quantization errors are shown for $N_D = 4$ bit, $N_D = 6$ bit, and $N_D = 8$ bit quantization in Figure 6.4. The curves show a fine quantization for small depth values and a coarser resolution for larger depth values. The quantization scheme trained on *EuRoC V101* and *KITTI 07* is verified with *KITTI 00* as test sequence. The quantization error reduces significantly from using the coarse $N_D = 6$ bit quantizer to using $N_D = 8$ bits, as shown in Figure 6.5. The root mean square error for the $N_D = 6$ bit quantizer is measured with 0.182 m and for $N_D = 8$ bit with 0.015 m.

6.3.2 Inter-view coding

When using a computer vision task or visual SLAM system that makes use of the visual information extracted from both views, it is required to transmit the set of features from both cameras. The data rate for the features from the stereo-view can be reduced by exploiting the inherent *spatial correlation* between both stereo-views using stereo feature correspondences. In the proposed scheme, only the differences between the feature descriptors alongside an index for the reference feature and the missing information to reconstruct the keypoint information needs to be transmitted. The stereo matching, as described previously for the depth

value estimation, is used to establish the necessary feature correspondences between a feature i from the n-th right frame and a reference feature $l \in \{z \in \mathbb{N} \,|\, 1 \leq z \leq N_{ref}^{M}\}$ from the left view, where N_{ref}^{M} denotes the number of reference features in the left view.

The coding costs $H_{n,i}^{M}$ in bits for inter-view coding M contains the individual costs for signaling the reference feature index $H_{n,i}^{M,ref}$, the costs for the differences between the descriptors denoted as $H_{n,i}^{M,res}(l)$, and the cost for the keypoint properties $H_{n,i}^{M,kpt}$ as

$$H_{n,i}^{M} = H_{n,i}^{M,ref} + H_{n,i}^{M,res}(l) + H_{n,i}^{M,kpt}. \tag{6.1}$$

After the set of candidate features from the left view near the epipolar line is identified, the feature l^* that results in the minimum Hamming distance h between the current visual descriptor $\mathbf{d}_{n,i}$ and a reference descriptor $\mathbf{d}_l$ from the left view is identified as

$$l^* = \arg\min_{l} h(\mathbf{d}_{n,i}, \mathbf{d}_l) \tag{6.2}$$

and subsequently used as the reference feature for the coding process.

6.3.2.1 Reference coding

Assuming uniform probability, the lower bound for the number of bits required for signaling the reference feature is given by

$$H_{n,i}^{M,ref} = \log_2(N_{ref}^{M}). \tag{6.3}$$

In this work, N_{ref}^{M} includes all features from the left view. Some bits can be saved by determining the number of possible reference features in the vicinity of the epipolar line and use this as N_{ref}^{M}. This is left for future work.

6.3.2.2 Descriptor coding

For coding the residual information, the difference vector between the current descriptor and the reference descriptor is calculated using the binary XOR operation $\mathbf{r}_{n,i}^{M} = \mathbf{d}_{n,i} \oplus \mathbf{d}_{l^*}$. Similar to Equation (4.5) introduced for intra coding and Equation (5.5) known from inter coding, $h_{n,i}^{M}(l^*)$ denotes the Hamming distance between the current and the reference feature as

$$h_{n,i}^{M}(l^*) = h(\mathbf{d}_{n,i}, \mathbf{d}_{l^*}). \tag{6.4}$$

The minimum number of bits for a specific feature i can be calculated using the binary entropy function similar to Equation (5.6) as follows

$$H_{n,i}^{M,res}(l^*) = -(N_d - h_{n,i}^{M}(l^*)) \cdot \log_2(p_0^{M}) - h_{n,i}^{M}(l^*) \cdot \log_2(1 - p_0^{M}), \tag{6.5}$$

where N_d is the length of the descriptor and p_0^{M} is the probability of any entry of the residual vector being zero for the proposed inter-view coding mode. This lower bound is approached using arithmetic coding. It is worth noticing that in this context the selected feature l^* results not only in the minimum coding costs, but is also the best stereo match in terms of the Hamming distance. This information can be reused in the targeted visual SLAM task.

6.3.2.3 Keypoint coding

To signal the keypoint information, the position and the orientation information have to be transmitted. Feature matches are restricted to the same scale-space level $\sigma_{n,i}$ allowing to reuse the scale-space level of the reference feature. For coding the keypoint locations, a differentiation of generic keypoint coding and ORB keypoint coding is made.

Generic keypoint coding: In the more general case, the keypoint position can be transmitted by quantizing the x and y coordinates to quarter pixel resolution similar to the intra coding mode. The quantized keypoint position is not transmitted differentially. The scale-space level is copied from the reference feature and the orientation is quantized into $N_{\hat{\theta}}$ bins. This results in the keypoint cost of

$$H_{n,i}^{M,kpt} = \log_2(4 \cdot N_w) + \log_2(4 \cdot N_h) + \log_2(N_{\hat{\theta}}). \tag{6.6}$$

Adding differential encoding of both the y coordinate and the orientation $\hat{\theta}$ can further reduce the number of required bits and is left for future work.

ORB keypoint coding: For ORB features, the keypoint position in x direction is scaled into the scale-space level, where the feature has originally been detected. As explained in Section 4.3.1.3, this results in keypoint coordinates located at integer positions. The x coordinate is not transmitted differentially. Conversely, for the y coordinate, the discrepancy with respect to the location of the reference keypoint in the common scale-space is transmitted. A difference of $\pm$ 2 pixels from the epipolar line in the corresponding scale-space representation is allowed, requiring $\log_2(5)$ bits for signaling the deviation. The orientation information is quantized into $N_{\hat{\theta}}$ bins. With the assumption of uniform distributions, the total keypoint costs can be written as

$$H_{n,i}^{M,kpt} = \log_2(N_w^s(\sigma_{n,i})) + \log_2(5) + \log_2(N_{\hat{\theta}}), \tag{6.7}$$

where $N_w^s(\sigma_{n,i})$ denotes the width of the image in the respective scale-space representation, as defined by Equation (4.12). Similar to the generic keypoint coding, adding differential encoding for the orientation $\hat{\theta}$ can further reduce the required number of bits.

6.4 Experimental evaluation

For the existing coding modes, the same probabilities and vocabularies, as in Chapter 5, are used to ensure consistency. The probability p_0^M for stereo coding has been obtained beforehand using the same training dataset as in Chapter 5, namely the *EuRoC Machine Hall* sequences. The scheme is evaluated on the *KITTI* dataset. The *KITTI* dataset is selected as a challenging automotive scenario for the feature coding. The large distances between the frames due to the comparably low frame rate, high camera velocity, and the rather larger baseline between the neighboring stereo-views pose a difficult scenario for the coding modes exploiting temporal and spatial correlations.

quantizer N_D [bit]	KITTI 00 [m] min / max / median
4	5.41 / 5.77 / 5.52
5	1.89 / 2.01 / 1.97
6	1.38 / 1.47 / 1.41
7	1.27 / 1.32 / 1.29
8	1.23 / 1.28 / 1.24
32 (no quant.)	1.21 / 1.26 / 1.23

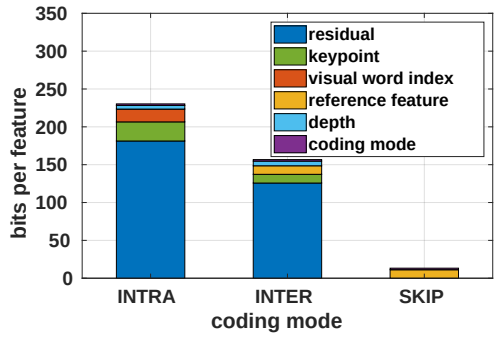
Table 6.1: Performance of depth quantizers measured on the *KITTI 00* sequence in terms of absolute trajectory error obtained from five individual runs. Using a depth quantizer with $N_D = 8$ bits results in similar accuracy as using the original depth information (adapted from [1] ©2019 IEEE).

6.4.1 Depth coding

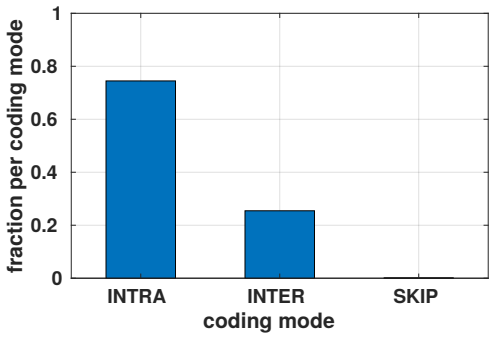
First, the depth coding is evaluated. The mapping results are shown in terms of ATE in Table 6.1. The table shows the values depending on the number of bits used for quantizing the depth values measured using the *KITTI 00* sequence. Throughout the evaluation, the proposed default settings of ORB-SLAM2 for the respective sequences are used. A performance approximately on par with the original depth values is achieved with $N_D = 8$ bits per depth value and an additional bit indicating whether a depth value is present. As ORB-SLAM2 employs the depth information to reproduce the original stereo-view feature coordinates [70], the result of using either the floating-point depth information or the stereo coding are similar.

Second, an experiment showing the results of the complete feature compression framework was conducted. The results are shown in Figure 6.6 for the *KITTI 00* sequence. Figure 6.6a summarizes the results in terms of bits per mode for the left view. About 225.4 bits are required for intra coding without depth information. A similar performance compared to the experimental evaluation from the previous chapters is achieved. Here, the inter coding mode is restricted to use only the past reference frame for prediction. The inter coding mode requires 150.3 bits for the left view. Allowing additional frames to be used as a reference would result in a reduced bitrate, but at the drawback of increased complexity. This effect and the influence of the frame rate is discussed later. In order to signal the coding mode, all features include the overhead of two bits.

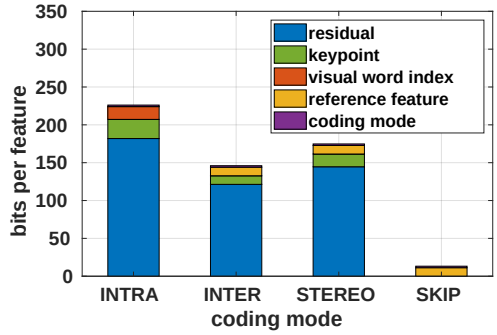
In this evaluation, the depth values for roughly 47.9% of the features contained in the left views using the intra coding and 64.5% of the features using the inter mode could be estimated. This results in adding on average 4.8 bits to the intra coding and 6.2 bits to the inter coding mode in the left view to signal the depth values. Inter coded features are intuitively considered to be more stable, resulting in a reliable tracking over multiple frames and also along the epipolar line in the corresponding stereo-view. The skip mode, which encodes only the reference feature index in the past frame, requires 13.3 bits but is only used by less than 1% of the features in this evaluation. The fraction of features using a particular coding mode is depicted in Figure 6.6b for the left view.



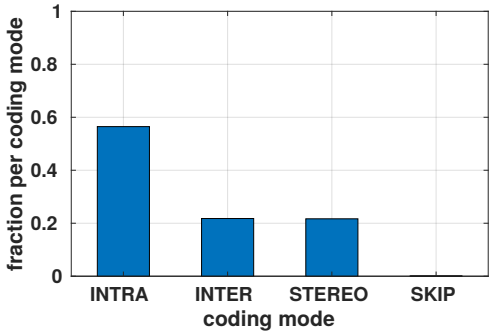
(a) Bits per mode for the left view.



(b) Mode decisions for the left view.



(c) Bits per mode for the right view.



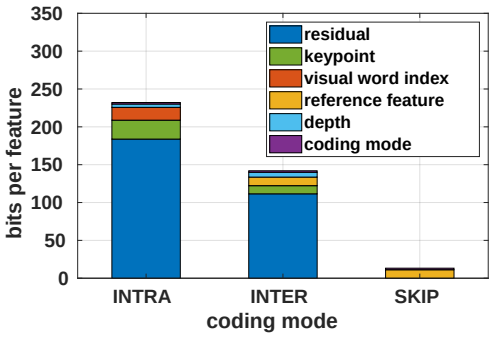
(d) Mode decisions for the right view.

Figure 6.6: Number of bits required for intra coding, inter coding, skip mode, and inter-view stereo prediction as mean values per feature obtained from the *KITTI 00* sequence. The left view (a) additionally includes the bits for the depth value coding and the right view (c) the bits for the stereo coding mode. The fractions of features using a particular coding mode are presented for the left view (b) and the right view (d) (adapted from [1] ©2019 IEEE).

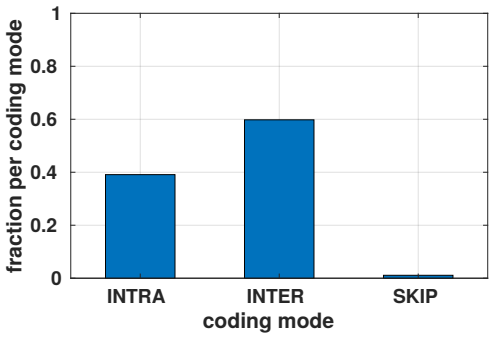
6.4.2 Stereo coding

Figure 6.6 also contains the evaluation of the proposed stereo coding mode. It is worth noticing that although the results are summarized in a single figure, either the depth coding or the stereo-view coding is used as they contain redundant information. The number of bits for the right view are depicted in Figure 6.6c. While the intra coding mode requires roughly the same number of bits for both views, the inter coding mode uses about 145.8 bits for the right view, which is slightly less than for the left view.

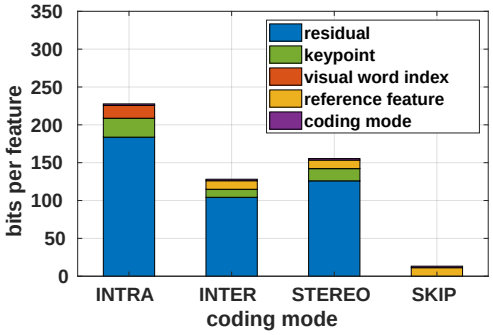
The proposed stereo coding requires about 11.3 bits for indicating the reference feature from the left view, 144.6 bits for signaling the residual vector, and 16.9 bits for transmitting the keypoint differences. Including the two bits signaling costs for the mode, a stereo coded feature takes on average about 174.8 bits. To provide further insights, the fraction of features coded with a specific coding mode is shown in Figure 6.6d for the right view. As a consequence of the large inter-frame distance in the *KITTI* sequences, many features are assigned to the intra coding mode. The amount of inter-coded features for a higher frame rate dataset, such as the *EuRoC* dataset, is considerably higher, which makes the coding in those scenarios



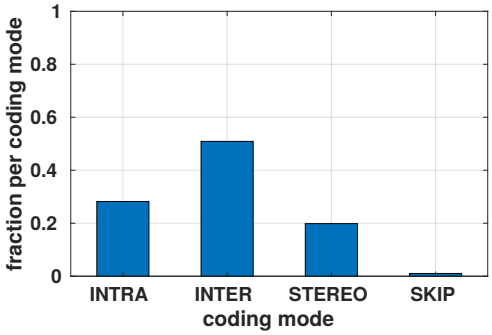
(a) Bits per mode for the left view.



(b) Mode decisions for the left view.



(c) Bits per mode for the right view.



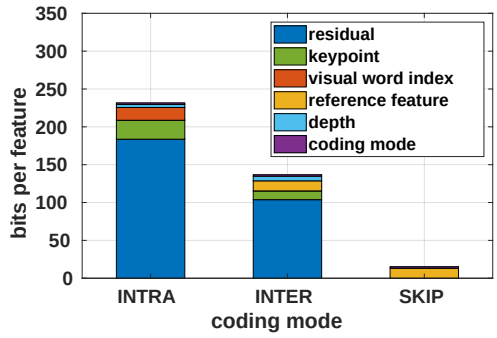
(d) Mode decisions for the right view.

Figure 6.7: Comparison of the number of bits required for intra coding, inter coding, skip mode, and inter-view stereo prediction as mean values per feature for the *EuRoC V101* sequence using a single reference frame ($N_r = 1$) for the left view (a) and the right view (c). The fraction of features using a particular coding mode are shown for the left view (b) and the right view (d). This dataset has higher frame rate, which allows more features to use the inter coding in comparison to *KITTI 00*.

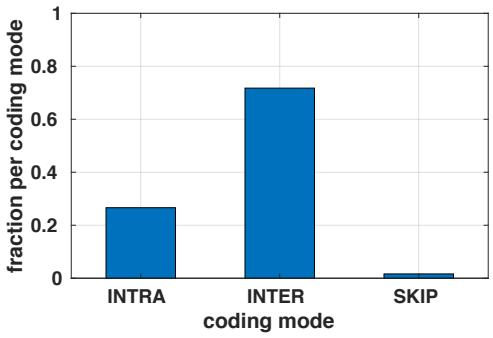
more efficient.

In order to show this effect and demonstrate the impact of using multiple reference frames, the coding results of the *EuRoC V101* sequence are discussed in the following. The results in terms of bits for the individual coding modes are presented in Figure 6.7 for using a single reference frame for the predictive coding. In direct comparison with the results obtained from the *KITTI 00* dataset in Figure 6.6, the share of the inter coded features is considerably higher. Due to the higher framerate, the feature descriptors matched along the temporal axis are much more correlated, thus resulting in lower bitrate for the inter coding mode.

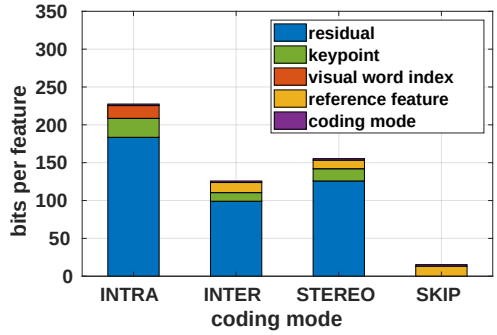
In a second evaluation of the *EuRoC V101* sequence, the number of reference frames is increased to four. The results in Figure 6.8 show that the fraction of inter-coded features increases above 70% for the left view. Due to the smaller baseline of the stereo camera setup in the *EuRoC* sequences, the efficiency of the stereo coding mode is slightly higher than in the wider baseline scenario of the *KITTI* dataset.



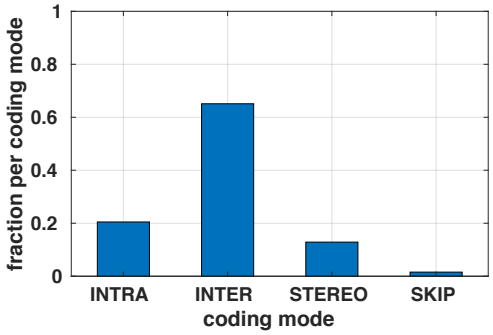
(a) Bits per mode for the left view.



(b) Mode decisions for the left view.



(c) Bits per mode for the right view.



(d) Mode decisions for the right view.

Figure 6.8: Comparison of the number of bits required for intra coding, temporal prediction, skip mode, and inter-view stereo prediction as mean values per feature for the *EuRoC V101* sequence using four reference frames ($N_r = 4$) for the left view (a) and right view (c). The fraction of features using a particular coding mode is shown for the left view (b) and the right view (d). Allowing more reference frames results in more features using the inter coding mode.

6.4.3 Mode configurations

In the following, the impact of different modes on both the coding time and the bitrate is discussed. The coding time was evaluated on two systems, namely an Intel Core i7-7700 with 3.6 GHz and an NVIDIA Jetson TX2. The NVIDIA Jetson TX2 was used in MAX-P ARM mode consuming 7.2 Watts at 2.0 GHz.

The results presented in Table 6.2 were obtained using the monocular and depth coding for two different coding profiles on the *KITTI 00* sequence. Although only the 2k features extracted from the left view are transmitted, the features extracted from both views are required for the depth estimation. In the evaluation presented in the first column, only intra mode and additional depth information (I+D) is used. Here, no temporal or spatial correlations are exploited. This enables the server to start the decoding process at any intermediate frame without any prior knowledge of the frame history. In video coding, this is also known as *random access*, as it allows to directly retrieve the information without waiting for a synchronization point. Moreover, it provides the fastest coding method in the evaluation and can, therefore, be used in *low-delay* scenarios. On the desktop hardware, it takes 11.1 ms

Intel Core i7-7700	I+D	I+P_1+S+D
encoding [ms]	11.1	14.5
decoding [ms]	12.4	12.7
TX2	**I+D**	**I+P_1+S+D**
encoding [ms]	26.5	38.6
decoding [ms]	22.9	25.1
[bits] per feature	229.0	210.6
# features	2k	2k

Table 6.2: Median timings and bits per feature for monocular + depth encoding and decoding measured on the *KITTI 00* sequence with different mode configurations. Using temporal prediction increases the coding time, but reduces the number of bits per feature (adapted from [1] ©2019 IEEE).

for encoding and 12.4 ms for decoding 2k features. On the TX2, encoding requires 26.5 ms and decoding takes 22.9 ms. As this mode does not exploit any correlations, it results in a comparably high bitrate of 229 bits per feature.

Next, both the temporal prediction and feature skipping are allowed (I+P_1+S+D). The temporal prediction uses a single reference frame, which is denoted by the subscript P_1. Searching for reference features increases the run-time, especially for the encoding part. More specifically, encoding and decoding require 14.5 ms and 12.7 ms on a desktop computer. The embedded system requires 38.6 ms and 25.1 ms for encoding and decoding. The result of exploiting the temporal information is a substantial reduction of the mean number of bits per feature down to 210.6 bits per feature.

The results for the stereo coding mode for three different coding profiles are presented in Table 6.3. First, the results are shown when only intra mode (I) is allowed for coding both views requiring 224.3 bits per feature and 17.3 ms for encoding 2x2k features on desktop hardware. Next, the inter-view coding mode (I+M) is added which results in an increase in encoding time to 18.7 ms but at the advantage of spending only 216.5 bits per feature. At last, temporal prediction and the skip mode are added (I+P_1+S+M), which results in an increase in processing time to 25.9 ms, but also provides an additional reduction in bitrate down to 201.3 bits per features.

The timings for the embedded systems are included in Table 6.2 and Table 6.3. So far, no specific adaption or optimization for the embedded platform has been performed. The ORB feature extraction, running on two images in parallel, poses a bottleneck on running directly ORB-SLAM2 on the embedded system. The feature extraction also consumes more time than most of the presented feature coding modes. However, the embedded system is capable of encoding 2k features, including depth estimation, in about 26.5 ms. For many computer vision applications or visual SLAM scenarios, fewer features might be sufficient, thus, lowering the encoding time. In order to further improve the performance of the system, optimized local binary features for embedded devices are available [111] and need to replace the time-consuming ORB feature extraction.

Intel Core i7-7700	**I**	**I+M**	**I+P$_1$+S+M**
ORB features [ms]	20.9		
encoding [ms]	17.3	18.7	25.9
decoding [ms]	20.5	20.7	21.3

TX 2	**I**	**I+M**	**I+P$_1$+S+M**
ORB features [ms]	67.3		
encoding [ms]	52.9	59.5	86.0
decoding [ms]	48.4	48.2	49.5

	I	**I+M**	**I+P$_1$+S+M**
[bits] per feature	224.3	216.5	201.3
# features	2x2k	2x2k	2x2k

Table 6.3: Median timings and mean bits per feature for stereo encoding and decoding measured on the *KITTI 00* sequence with different mode configurations. Using stereo-view coding reduces the number of bits at increased computation time (adapted from [1] ©2019 IEEE).

	I	**I+M**	**I+P$_1$+S**	**I+P$_1$+S+M**	**I+P$_4$+S+M**
ORB features [ms]	14.5				
encoding [ms]	11.5	12.8	15.6	16.4	26.3
decoding [ms]	12.6	13.3	13.8	13.9	13.5
[bits] per feature	224.5	208.9	170.4	165.3	151.5
# features	2x1.2k	2x1.2k	2x1.2k	2x1.2k	2x1.2k

Table 6.4: Median timings and mean bits per feature for feature extraction, stereo encoding and decoding on a Intel Core i7-7700 using 1.2K features per view measured using the *EuRoC V101* sequence with different mode configurations. Adding more reference frames (P_1, P_4) further reduces the number of bits, but increases the coding time by roughly 10 ms.

The timings for the *EuRoC V101* sequence are summarized in Table 6.4. The results were measured using again an Intel Core i7-7700 with 3.6 GHz and 1.2k features per view in stereo mode using the ORB-SLAM2 default settings. The first column denotes the intra only mode. Next, intra and stereo coding are combined adding only dependencies between the two stereo views. The required bitrate reduces from 224.5 bits per feature to 208.9 bits per feature at slightly increased coding times for searching corresponding feature pairs between the views. Next, the intra coding is combined with the predictive coding and the skip mode using a single reference frame. The coding time increases to 15.6 ms due to the feature matching with the previous frame. The bitrate reduces to 170.4 bits per feature by exploiting temporal dependencies. Next, all coding modes are evaluated in the $I + P_1 + S + M$ configuration adding the stereo-view coding, where again the coding time increases to 16.4 ms and the costs decreases to 165.3 bits per feature. Next, four reference frames are tested, which results in an increase in computation time by roughly 10 ms and a gain of 13.8 bits per feature compared to using only a single reference. In comparison to the values reported for the *KITTI 00* sequence, the coding for the *EuRoC V101* greatly benefits more from the stereo and the predictive coding due to the high frame rate and smaller stereo baseline.

In total, a reduction by 57.9% from 360 bits uncompressed down to 151.5 bits using stereo-view coding and four reference frames for the predictive coding is achievable. In contrast to the timings reported in Chapter 5, the enhanced coding framework benefits from an improved implementation featuring, for example, parallel batch processing of features. Also, the current evaluation has been carried out using more recent hardware. This allows encoding 2x1.2k features in the same time as 1k features as reported in the evaluation in Table 5.3 from Chapter 5. Further evaluation of the proposed methods is included in the next chapter.

6.5 Summary

In this chapter, the feature coding framework has been completed with additional coding modes for both depth values and features extracted from a stereo camera setup. The feature coding framework is not restricted to be used in combination with visual SLAM, but can also be used standalone for other applications. The approach has extensively been evaluated in terms of coding efficiency, timing, and absolute trajectory error for the depth value coding on the *KITTI* and the *EuRoC* dataset. A reduction by 57.9% can be achieved on *EuRoC V101* by using multiple reference frames and stereo-view coding without the need for skipping features compared to uncompressed transmission. Real-time processing capabilities can be achieved on embedded devices when replacing the feature extraction by optimized algorithms.

Chapter 7

Collaborative visual SLAM

7.1 Problem statement

The previous chapters addressed the issue of the efficient exchange of visual information. When collecting the compressed visual information from a single client at a server, the question naturally arises, whether the information collected from other nearby clients could contribute to the goal of building a consistent global map. Especially with regard to the edge-cloud infrastructure, collaborative mapping is of interest to quickly gather and update maps. To this end, the system should be extended to include a collaborative visual SLAM server component to facilitate the joint mapping of the environment. Parts of this chapter have been published in [1].

7.2 System architecture

The system architecture used in this chapter is an extension of the system described in Chapter 6 and is depicted in Figure 7.1. Each client comes with a stereo camera system with known intrinsic camera calibration and a known stereo baseline. The ORB extraction is running on each stereo image pair and the features are passed to the feature encoding process. The compressed features are then sent over a network connection to a server instance. The server

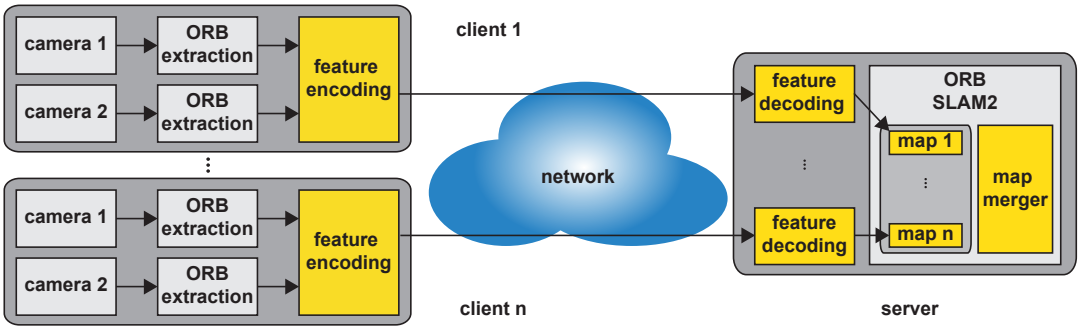


Figure 7.1: Illustration of the ATC-based collaborative mapping architecture. The clients extract visual features and apply feature encoding. The server receives and decodes the feature streams, and creates individual maps for each client. A map merging process is triggered by the insertion of new keyframes. It merges overlapping maps into a unified representation (adapted from [1] ©2019 IEEE).

is running multiple feature decoding processes in parallel that reconstruct the features and pass them on to the collaborative visual SLAM system. Each client starts with an empty map, which is then incrementally extended using the received features. A map merging module is running in the background detecting possible overlaps between all maps contained in the map stack. In case an overlap is detected, the corresponding maps are merged into a unified representation.

The proposed map merging is based on the relocalization and loop-closing technique already present in ORB-SLAM2. It uses fast image retrieval, leveraging the Bag-of-Words representation in combination with an inverted index. Instead of detecting loops within a local map, overlaps across maps are detected. In this case, the maps can be merged and all attached clients can henceforth contribute to a common map representation.

7.3 Map merging

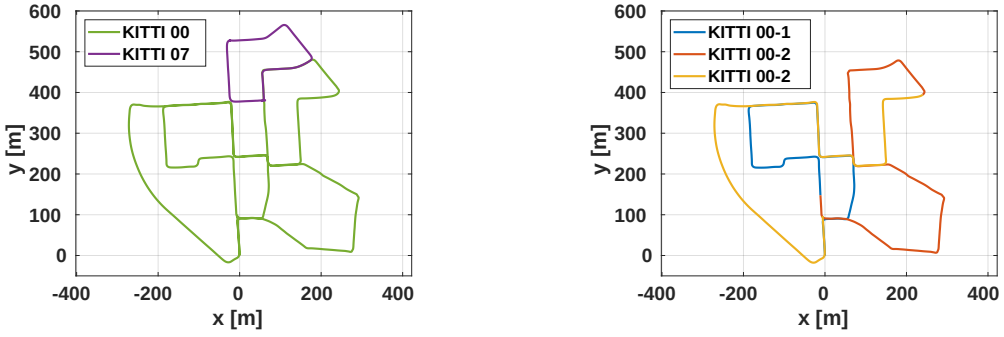
Map merging is realized by adopting the existing approaches for relocalization and loop closing included in ORB-SLAM2 to work not only within, but also across multiple maps. First, loop candidates are detected by assessing the visual similarity between a newly inserted keyframe in the source map and all other keyframes contained in any other possible target map in the map stack. The visual similarity is measured based on the Bag-of-Words representation using *tf-idf* scoring [138]. More specifically, a keyframe is a candidate keyframe if it achieves a comparable similarity score than keyframes inserted prior to the source keyframe on the same trajectory. After identifying a set of match candidates, a similarity transformation is calculated according to the method of Horn [143] using the map points contained in the matched keyframes. This step serves as a geometric verification of the matches to avert false map mergers. Following a successful geometric verification of a candidate, the two maps are merged by migrating all keyframes and map points from the source map to the target map. During this period, the tracking and the mapping threads of all attached clients are paused to facilitate merging without concurrent access to the data. Next, the maps are aligned at the overlap using the estimated similarity transformation. To this end, the poses of both the keyframes and map points are updated. After the alignment, additional duplicate map points are detected and merged. In addition, further connections between map points originally from the source map and keyframes in the target map and vice versa can be established. Next, an optimization of the *Essential Graph* structure is performed. To this end, both Essential Graphs, which are spanning trees connecting all keyframes of a single map, need to be attached. At this point, both maps are coarsely aligned and the tracking and the mapping threads are restarted. At the same time, global bundle adjustment is triggered in the background to optimize and fine align the common map asynchronously.

7.4 Joint mapping

In order to facilitate the map merging, both the tracking and the mapping thread of the source map are paused, detached from the source map, and attached to the target map. In addition, the loop-closing thread of the source map is stopped and not used any more. From now on, the loop-closing process of the target map detects loops within the joint map. A shared locking scheme is implemented to avoid concurrent access to the data and keep consistency during map updates and migration. Hence, only a single mapping thread at a time is allowed to add new map points to a particular keyframe or is allowed to update a keyframe or a map point position. Access is provided in a *first-come-first-serve* manner.

7.5 Experimental evaluation

In the following, the feature coding framework introduced in the previous chapters is combined with the collaborative visual SLAM architecture and evaluated jointly. The same probabilities and training data, as in the previous chapters, have been used. The metric scale collaborative remote visual SLAM is evaluated on the *KITTI* sequences. The *KITTI* dataset poses a challenging scenario not only for the feature coding but also for the collaborative mapping due to the spatial extent of the covered area. The centralized collaborative mapping was first evaluated in an experiment with two clients operating on the *KITTI 00* and *KITTI 07* sequences, as illustrated in Figure 7.2a. The sequences were played back simultaneously, and the result was evaluated in terms of feature coding performance and absolute trajectory error. The experiments were conducted on a virtual machine with 16 virtual CPUs based on an Intel Xeon Platinum 8124M @ 3.00 GHz running in a cloud environment. A baseline for the accuracy achievable with the setup was obtained by initial experiments with map merging deactivated, which is denoted as *standalone* in the following. Afterward, the map merging module was activated. This allows incorporating map information from both clients into a common map. The results for both the baseline and the collaborative mapping are reported in terms of ATE evaluated on each trajectory individually in Table 7.1. To ac-



(a) Ground truth for *KITTI 00* and *KITTI 07*.

(b) Ground truth for *KITTI 00* split into three parts.

Figure 7.2: Overview of the two sequences from the *KITTI* dataset used for the experimental evaluation shown in a common coordinate system (Figures adapted from [1] ©2019 IEEE).

	standalone [m] **min / max / median**	**collaborative [m]** **min / max / median**
KITTI 00	1.21 / **1.26** / 1.23	**1.10** / **1.26** / **1.16**
KITTI 07	0.60 / 0.70 / 0.65	**0.48** / **0.66** / **0.54**

Table 7.1: Collaborative SLAM performance on the *KITTI 00* and *KITTI 07* sequences in terms of absolute trajectory error obtained from five runs. The collaborative approach consistently outperformed the standalone mapping in this scenario (adapted from [1] ©2019 IEEE).

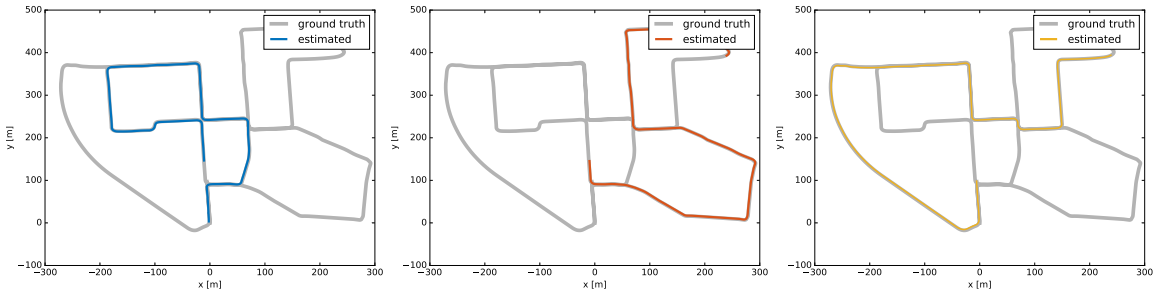
	standalone [m] **min / max / median**	**collaborative [m]** **min / max / median**
KITTI 00-1	**0.99** / **1.09** / **1.08**	1.17 / 1.26 / 1.21
KITTI 00-2	1.44 / 1.52 / 1.49	**1.07** / **1.37** / **1.14**
KITTI 00-3	2.17 / 3.45 / 3.11	**1.31** / **2.18** / **1.44**

Table 7.2: Collaborative SLAM performance on the *KITTI 00* sequence in terms of absolute trajectory error obtained from five runs. The collaborative outperformed the standalone mapping for two out of three clients (adapted from [1] ©2019 IEEE).

count for the non-deterministic behavior of the system, the results were obtained from five individual runs. An improvement by 5.7% is achieved for the *KITTI 00* and 16.9% for the *KITTI 07* sequence. The result of the mapping process is a fully connected map covering both sequences.

In a second experiment, three clients simultaneously explored the *KITTI 00* sequence. The sequence was split into the three disjunct parts shown in Figure 7.2b. For each client, individual maps were created. At some point, a significant overlap between the maps was detected, and the maps were successively merged into a unified representation. The results obtained from five individual runs are provided in Table 7.2. The first client is able to reduce the error in standalone mode, whereas the results significantly improve using the collaborative mapping for the remaining sequences. The error of the third client is reduced from 3.11 m to 1.44 m, which is an improvement by 53.7%. The improvement can be partially explained by the usage of previously mapped areas as provided by the other clients, but the error reduction also stems from additional loops detected within the merged maps, thus allowing to correct for the accumulated drift. An overlay of the ground truth with the individual trajectories is shown in Figure 7.3, whereas the final maps are shown in Figure 7.4.

Considering the exchange of visual features, Figure 7.5a shows the data exchanged between each client and the server. More specifically, the figure compares the cumulative data rate required to signal the features using uncompressed transmission, the proposed stereo-view feature coding, and the depth coding for all clients. The bitrate reduction that is achievable ranges from about 44.1% using stereo-view coding up to 70.8% using the monocular and depth value transmission. Again, 360 bits per uncompressed feature are assumed as

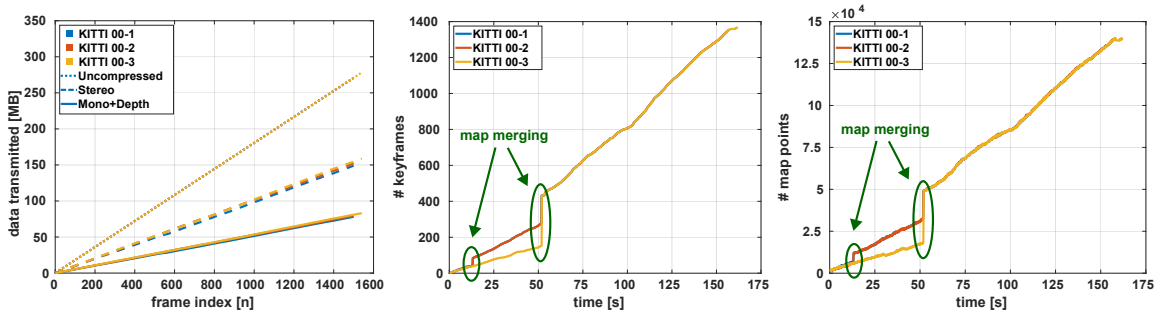


(a) *KITTI 00-1*. (b) *KITTI 00-2*. (c) *KITTI 00-3*.

Figure 7.3: Comparison of the ground truth with the SLAM results for *KITTI 00* using three clients for collaborative mapping. The client collaboratively reconstruct their partial trajectory.

(a) Map from *KITTI 00* and *KITTI 07*. (b) Map from *KITTI 00* split into three parts.

Figure 7.4: Globally consistent map after merging individual maps. Obtained from clients operating on the *KITTI 00* and the *KITTI 07* sequences (a), and from three clients operating on the *KITTI 00* sequence (b). The color indicates, which client created the map points ((b) adapted from [1] ©2019 IEEE).



(a) Cumulative data rate. (b) Number of keyframes. (c) Number of map points.

Figure 7.5: Cumulative data rate measured at the encoder per client (a), where stereo uses the I+P_1+S+M modes and mono+depth the I+P_1+S+D modes. Next to it, the evolution of keyframes (b) and map points (c) using three clients on the *KITTI 00* sequence is shown. The occurrence of map merging is highlighted in green (adapted from [1] ©2019 IEEE).

detailed in Chapter 4. In addition, the evolution of the maps during the sequences in terms of the number of keyframes is shown in Figure 7.5b. Besides, the evolution in terms of map points is shown in Figure 7.5c. The occurrences of map merging are highlighted to indicate when the existing keyframes and map points were merged into a single map representation. It is worth noticing that the time required to build a complete map of the *KITTI 00* sequence is reduced to a third of the original length by parallelizing the work using the three clients simultaneously. Another consequence of the proposed client-server architecture is that the ORB extraction is carried out at the client, thus reducing the median time required for tracking individual frames. For example, the median tracking time at the server was measured with 28.8 ms for the client operating on *KITTI 00-3*.

In the proposed centralized system architecture, the number of clients is limited by the computational power of the central server. For larger teams, extending the system to support multiple servers that are capable of exchanging information among them is required.

7.6 Summary

In this chapter, both the feature coding and the collaborative visual SLAM have been combined into a unified system architecture, where the computationally intensive visual SLAM system is running on a powerful server and only the visual features are extracted and compressed at the client-side. The approach has been extensively evaluated in terms of coding efficiency, timing, and absolute trajectory error on the *KITTI* dataset. The results show a substantial reduction in the required data-rate up to 70.8% and an improvement in ATE by 53.7% using the collaborative mapping. Hence, this system closes a gap by addressing the urgent need for data-efficiency in collaborative visual SLAM setups.

Chapter 8

Map compression for visual SLAM

The previous chapters introduced schemes that reduce the amount of data required to exchange visual cues among clients and a server running the actual visual SLAM system. However, exchanging already processed information in the form of map data has not been considered so far. In order to facilitate efficient exchange of map information, suitable approaches for compressing a visual map have to be found. Hence, in this chapter, the map properties of ORB-SLAM2 [70], as a recent representative of state-of-the-art visual SLAM systems, have been analyzed. Based on the obtained insights, a lossless compression scheme using feature coding and exploiting the geometric relationships in such a visual SLAM map is conceived. In addition, a map sparsification step is added to reduce the required data rate using lossy compression. Parts of this chapter have been published in [4].

8.1 Problem statement

In order to support the robotic task at hand, it is beneficial to exchange static map information either directly between the cooperating clients in a multi-client setup or making the

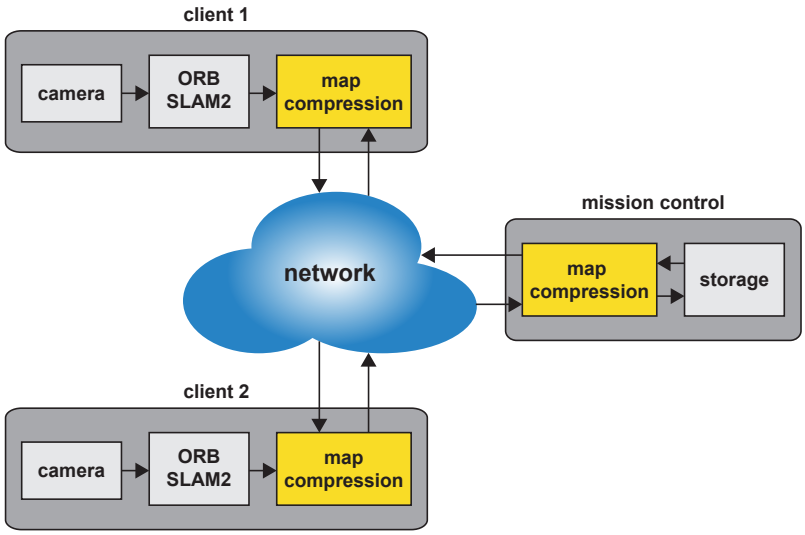


Figure 8.1: Illustration of an application scenario for compressed map exchange. Multiple clients exchange map information among the team members or with mission control.

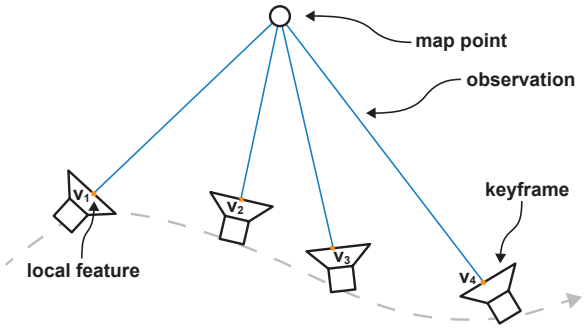


Figure 8.2: A sample trajectory is shown in dashed grey with keyframe poses illustrated in black. A map point is shown as a black circle with several observations v_i (blue) denoting the connection to features in the keyframes. The key idea is to exploit that the observations v_i should have similar visual outlines, which facilitates differential encoding (adapted from [4] ©2018 IEEE).

information available in form of prior knowledge obtained from previous mappings. In many application scenarios, where collaborative exploration is applicable, e.g., creating a map after a natural disaster, the communication capabilities are usually strongly limited. A possible system architecture is depicted in Figure 8.1, where multiple clients share map information over a network with either other clients or mission control. The key tenet is to reduce the map information without losing its relocalization capabilities, allowing to share the map information even with limited communication resources.

8.2 System architecture

To tackle the problem, the typical structure of a visual SLAM map is analyzed. A typical excerpt of a map representation is shown in Figure 8.2. The map usually consists of keyframes and map points inserted along the traversed trajectory. The keyframes contain information such as the timestamps, the estimated position, but also the visual outline in the form of local features. In the process of generating a visual SLAM map, these features are matched with features situated in neighboring keyframes and are then used for subsequent triangulation. The result is a map point with observations attached, which denote the local features from the keyframes that are linked to a map point. In the graph-based optimization back-end, the keyframes and map points are used as vertices, whereas the observations are used to add edges as constraints between these vertices. Although the ORB-SLAM2 framework is used in this work, the same underlying structure and bundle adjustment techniques are found in many recent visual SLAM systems.

In order to identify the primary source of generated data, the typical parts of a visual SLAM map from ORB-SLAM2 have been analyzed and evaluated in Table 8.1 using the *EuRoC MH01* sequence. Necessary information about the characteristics of the camera and features are stored in the map constants. Additional information, such as the number of keyframes and the number of map points, allows storing these entities in a consecutive fashion. Each keyframe includes information such as a timestamp and a parent keyframe. The first allows comparing the keyframe position to the ground truth by matching the timestamps. The latter allows rebuilding the *Essential Graph* connecting all the keyframes in ORB-

	full map	essential map
map constants	< 0.1%	< 0.1%
keyframe header	< 0.1%	0.2%
keyframe poses	<0.1%	0.2%
feature descriptors	**56.6**%	**50.2**%
feature keypoints	**23.0**%	**20.4**%
feature stereo information	14.1%	-
map point poses	0.6%	2.9%
map point observations	5.6%	26.1%
total	33.5 MB	7.2 MB

Table 8.1: Memory footprint of the individual map properties measured on the *EuRoC MH01* sequence. The full map encompasses all features and stereo information, whereas the essential version contains only the features attached to map points and no stereo information (adapted from [4] ©2018 IEEE).

SLAM2. In addition, the pose $\mathbf{T}_n \in \mathrm{SE}(3)$ of a keyframe n is stored as quaternion $\mathbf{q}_n \in \mathbb{H}$ for the orientation and translation $\mathbf{t}_n \in \mathbb{R}^3$ for the position with respect to the world coordinate frame. The visual information is saved in the form of ORB features extracted from the keyframes. The typical number of ORB features ranges from 1k to 2k depending on the configuration for the specific dataset or the targeted application scenario. In the case of a stereo camera setup, the keyframe additionally stores the keypoints in the corresponding stereo-view. Using an RGB-D or a stereo camera setup, the estimated depth is available through readings of the depth sensor or is acquired by stereo feature matching. The map points are defined by their position in the form of a translation vector $\mathbf{t}_p \in \mathbb{R}^3$ and by the observations in the form of local features contained in the keyframes. So each map point stores a list with the keyframes and the corresponding feature indices to establish the connections.

The map itself could be represented only by the map points and the visual outline in the form of a single visual descriptor obtained from averaging the visual descriptions from all attached observations. However, the goal is a representation that allows incorporating the information from previous mappings into the graph optimization problem. So the map should include all vertices in the form of map points and keyframes, as well as all the links between them in the form of observations. This allows extending the map but still including the optimization constraints from the previous mappings. In order to store only the essential information necessary to achieve this goal, the ORB features which were not linked to any map point, i.e., did not serve as observation during map creation, are removed. Additionally, the stereo and depth information is discarded. This information providing the scale is not required after the map is completed, and the scale is fixed. This reduced representation is therefore coined *essential map* in the following.

A comparison of both the full and the essential map representation is provided in Table 8.1, where in both cases the feature keypoints and descriptors are responsible for a major fraction of the storage required for the map representation. This serves as motivation to apply some of the previously introduced concepts of visual feature coding to reduce the

required amount of storage. While this step is implemented using only *lossless compression* techniques ensuring that the original essential map representation can be recovered, a second step allowing *lossy compression* is added. To this end, an optimization problem is formulated keeping only map points that exhibit low coding costs as calculated in the lossless step. To summarize, the algorithm introduced in this chapter contains two parts:

- First, the approach introduced in Chapter 4 is used for joint coding of the visual descriptors with their corresponding Bag-of-Words representation. The visual similarities among the observations of map points are exploited by constructing a *minimum spanning tree* (MST) connecting all features using the Hamming distance as the weight for the edges. The resulting tree structure is exploited to determine the coding order for differential feature coding. This allows for *lossless compression* of the visual information by exploiting geometric dependencies.
- Second, a *lossy compression* method is proposed. To this end, a map sparsification step is added to reduce the number of map points to be stored. This step is tightly coupled with the lossless compression scheme and favors map points, which exhibit good compression properties using *Integer Linear Programming* (ILP).

8.3 Lossless map compression

The lossless map compression is inspired by the concept of coding visual descriptors extracted from video sequences, as discussed in the previous chapters. In contrast to the inter and stereo coding modes, making use of temporal and spatial correlations, the structure of a visual SLAM map can be exploited for differential feature coding as well. The key idea is to jointly encode all features attached to a single map point, as these observations describe the same physical structure in the world. This is ensured by passing several checks based on visual similarity and geometric consistency within the visual SLAM algorithm. Thus, the observations should exhibit similar descriptors, which can be used in the proposed scheme to encode the local feature descriptors differentially. The concept is shown in Figure 8.3. The observations of a map point are denoted with v_i, where i denotes the observation index. The observations of a single map point are connected using a fully connected graph structure. The weight $w_{i,j}$ of the edge connecting observation with index i and index j is defined as the Hamming distance between the descriptors. In order to find the optimal coding order of the observations, a minimum spanning tree is constructed based on the edge weights. In order to start the coding process, the observation exhibiting the least coding costs for the intra coding mode from Chapter 4 is determined. Afterward, the minimum spanning tree is traversed using the already encoded features as a reference for encoding the remaining features. The intra observation coding and MST coding are introduced in more detail in the following.

8.3.1 Intra observation coding

In order to start the encoding of the information attached to a map point, the first observation needs to be stored similarly to the approach employed for the intra-frame coding. Here,

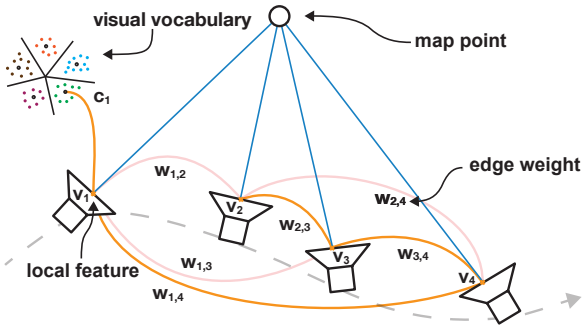


Figure 8.3: A sample trajectory is shown in dashed grey with keyframe poses illustrated in black. The fully connected graph is shown with red lines connecting the observations v_i (shown in blue) of a map point. Based on this graph, a minimum spanning tree is constructed (shown in orange) connecting the observations along the edges with the minimum Hamming distances. The observation with the least coding costs for intra coding using a shared visual vocabulary is selected as a starting point. Here, the coding starts at observation v_1, continues with MST coding in the following order: v_4, v_3, v_2 (adapted from [4], ©2018 IEEE).

the keypoint and the descriptor have to be stored alongside an additional identifier to which keyframe and feature this observation belongs. The total number of bits required for intra observation coding, denoted with I_o in the following, is given by

$$H_{u,i}^{I_o} = H_{u,i}^{I,BoW} + H_{u,i}^{I,res} + H_{u,i}^{I,kpt} + H_{u,i}^{I_o,obs}, \tag{8.1}$$

where $H_{u,i}^{I,BoW}$ denotes the cost for coding the visual word index, $H_{u,i}^{I,res}$ denotes the costs for transmitting the residual vector between the descriptor and the visual word. $H_{u,i}^{I,kpt}$ characterizes the costs for transmitting the keypoint information. Both the keyframe information and the feature indices are included in $H_{u,i}^{I_o,obs}$.

Throughout this chapter, the index u denotes the map point index and i the observation index. The visual word index coding, residual coding, and the keypoint coding are equivalent to the intra coding. This is reflected by the usage of the superscript I. The theoretical limits are approached using arithmetic coding in the experimental evaluation. For the sake of completeness, all coding costs will be briefly recapitulated. For further details on the intra coding, the reader is referred to Chapter 4.

8.3.1.1 Bag-of-Words index coding

The number of bits required for signaling the visual word index assuming uniform probabilities for the N_v indices is given by Equation (2.2) as

$$H_{u,i}^{I,BoW} = \log_2(N_v). \tag{8.2}$$

8.3.1.2 Residual coding

The required number of bits for sending the differences between the descriptor and its nearest visual word for a specific observation can be written as

$$H_{u,i}^{I,res} = -(N_d - h_{u,i}^{I}) \cdot \log_2(p_0^I) - h_{u,i}^{I} \cdot \log_2(1 - p_0^I), \tag{8.3}$$

with $h_{u,i}^{I}$ being the number of non-zero elements and p_0^I the probability of a zero entry.

8.3.1.3 Keypoint coding

Here, the ORB specific keypoint coding is exploited by scaling the feature location back to the scale-space level where the feature has been originally detected ensuring integer pixel locations. The cost for coding the keypoint information $\mathbf{k}_{u,i}$ is the sum of the costs for coding the keypoint location, the scale-space level σ, and the quantized orientation $\hat{\theta}$ similar to Equation (4.13) as

$$H_{u,i}^{I,kpt} = \log_2(N_w^s(\sigma_{u,i})) + \log_2(N_h^s(\sigma_{u,i})) + \log_2(N_\sigma) + \log_2(N_{\hat{\theta}}), \tag{8.4}$$

with $N_w^s(\sigma_{u,i})$ and $N_h^s(\sigma_{u,i})$ being the image sizes at the respective scale-space level $\sigma_{u,i}$ of the observation. The total number of scale-space levels is denoted by N_σ and the number of bins used to quantize the orientation information is denoted by $N_{\hat{\theta}}$. The information about the keyframe and feature indices requires a minimum amount of

$$H_{u,i}^{I_o,obs} = \log_2(N_k) + \log_2(N_f) \tag{8.5}$$

bits, where N_k denotes the number of keyframes in the map and N_f denotes the number of features per keyframe. While the number of keyframes is fixed only after the mapping process is finished, the number of features per keyframe is usually fixed for all frames defined by a dataset specific configuration.

8.3.2 Minimum spanning tree coding

The goal is to exploit visual similarities among the visual features attached to the same map point. The order of observations that minimizes the Hamming distance between successively encoded features needs to be determined. To this end, the observations of a single map point u are modeled as vertices v_i with connecting edges $e_{i,j}$ in an undirected graph structure $G_n = (V_u, E_u)$. The weights $w_{i,j}$ assigned to each edge between vertex v_i and v_j are defined as the Hamming distance between the respective binary feature descriptors $\mathbf{d}_i$ and $\mathbf{d}_j$. Kruskal's algorithm [144] is employed to retrieve the minimum spanning tree that connects all the vertices. The coding of the observations of a map point starts at the observation i^* that exhibits the minimum coding costs by evaluating the costs for all attached observations $i \in \{z \in \mathbb{N} \,|\, 1 \leq z \leq |V_u|\}$ as

$$i^* = \arg\min_i H_{u,i}^{I_o}, \tag{8.6}$$

where $|V_u|$ is the number of vertices, i.e., features observing a map point u. Then, the minimum spanning tree is traversed, and the observations along the connected edges are differentially encoded using the proposed MST coding, as illustrated in Figure 8.3 and detailed in the following. Similar to the inter-frame prediction from Chapter 5 or the stereo-view coding introduced in Chapter 6, the algorithm strives to exploit the similarities between multiple features by sending only differences between connected features exhibiting minimal differences, i.e., maximum visual similarities. The concept is not solely restricted to exploiting either temporal coherence or immediate spatial relations among the keyframes. On the contrary, it can utilize observations from keyframes on the trajectory that are both temporally

and spatially further apart, as long as they are attached to the same map point by the visual SLAM system.

The cost for coding an observation using MST coding, which is denoted as $H_{u,i}^{T}$ in the following, can be written as

$$H_{u,i}^{T}(g(i)) = H_{u,i}^{I,kpt} + H_{u,i}^{I_o,obs} + H_{u,i}^{T,vert} + H_{u,i}^{T,res}(g(i)). \tag{8.7}$$

The keypoint coding uses the same method as the intra coding mode (Equation (8.4)). In case features extracted from video sequences are coded, there are usually only small camera movements between frames, which allows for coding the keypoint properties differentially as detailed in Chapter 5 and Chapter 6. In the current application scenario, however, observations of a map point are not necessarily restricted to a consecutive sequence of frames and, therefore, the positions must neither be located in a similar part of the image nor have the same feature orientation or scale-space level, so the proposed approach refrains from coding the keypoint properties differentially.

The observation information is transmitted using the same coding method as for the intra observation coding introduced in Equation (8.5). The observation information is necessary to assign the feature to the correct keyframe at a consistent feature index position. In addition, the index of the reference vertex, i.e., observation, has to be signaled alongside the descriptor residuals. The reference vertex is defined by the function $g(i)$ that returns the connected feature exhibiting the minimum Hamming distance to the current observation i given by the MST.

8.3.2.1 Vertex coding

The costs for signaling the reference vertex is denoted as $H_{u,i}^{T,vert}$, where the number of bits is defined by the number of attached observations $|V_u|$ given by

$$H_{u,i}^{T,vert} = \log_2(|V_u|). \tag{8.8}$$

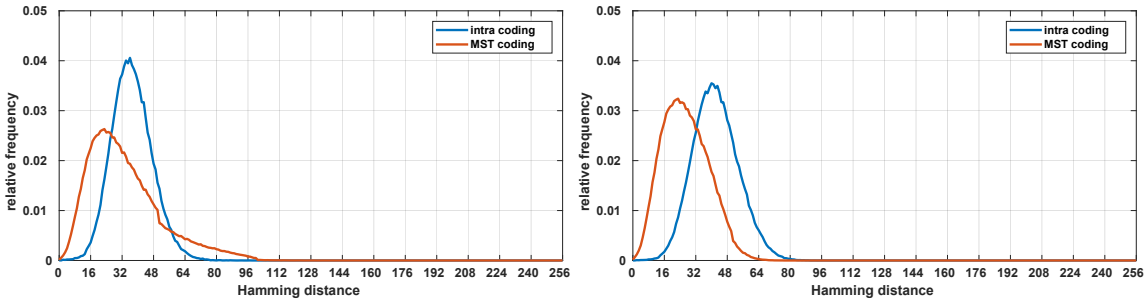
Further coding gain can be anticipated by exploiting the fact that the number of possible reference vertices starts at zero and is incremented each time an additional observation has been coded and becomes available as a reference. However, this work refrains from implementation, as the result is expected to provide only a minor contribution to the overall coding performance.

8.3.2.2 Residual coding

The cost for reconstructing the descriptor is denoted as $H_{u,i}^{T,res}(g(i))$ and defines the cost for sending the differences between the reference descriptor $\mathbf{d}_{g(i)}$ and the current descriptor $\mathbf{d}_i$ as

$$H_{u,i}^{T,res}(g(i)) = -(N_d - h_{u,i}^{T}(g(i))) \cdot \log_2(p_0^T) - h_{u,i}^{T}(g(i)) \cdot \log_2(1 - p_0^T), \tag{8.9}$$

where $h_{u,i}^{T}(g(i))$ denotes the Hamming distance between both descriptors. Here, similar to the intra coding mode, p_0^T denotes the probability of a residual element being zero for the minimum spanning tree coding.



(a) MST coding.

(b) Intra+MST coding.

Figure 8.4: Histogram of the Hamming distances between observations and visual words (intra coding) or reference descriptor (MST coding) respectively. Some of the observations encoded using the MST coding produce high costs due to large Hamming distances (a). Therefore, intra coding should be allowed as a fallback solution for each observation (b) (adapted from [4], ©2018 IEEE).

8.3.3 Map point coding

The cost for coding a map point with all observations is given by the sum of coding the first observation using the intra coding and the remaining observations using the minimum spanning tree approach as

$$H_u^{'} = H_{u,i^*}^{I} + \sum_{i=1, i \neq i^*}^{|V_u|} H_{u,i}^{T}(g(i)). \tag{8.10}$$

In Figure 8.4, the Hamming distances between connected features and the corresponding visual words obtained from the *EuRoC Machine Hall* sequences are shown. In Figure 8.4a, the histogram for using intra coding for the first observation and MST coding for the remaining features is shown. The comparison provides two main insights:

- First, two salient points can be identified in the curve for MST coding. These correspond to fixed thresholds used in ORB-SLAM2 to determine feature correspondences. The lower threshold is $th_l = 50$, whereas the higher threshold has a value of $th_h = 100$. The lower threshold results in a small brink in the probability distribution, whereas no feature exceeds the upper threshold.
- Second, some edges show a comparably high Hamming distance between connected observations in the MST, e.g. at Hamming distances larger than 60. This entails inefficiencies in the coding process as these features require a comparably large amount of bits using MST coding.

In this case, it is beneficial to allow also intra coding for these observations. The cost for coding a map point including all observation is then given as

$$H_u = \sum_{i \in V_u^{I_o}} H_{u,i}^{I_o} + \sum_{j \in V_u^{T}} H_{u,j}^{T}(g(j)) + |V_u| - 1, \tag{8.11}$$

where $V_u^{I_o}$ denotes the set of indices for intra coded observations, including the starting ob-

servation i^* and V_u^T the MST coded vertex indices. $|V_u|$ denotes the number of observations attached to map point u. A supplemental bit per observation, except the first one, is added to indicate whether intra or MST coding is used. The decision is made based on the minimum of the estimated coding costs for $H_{u,i}^{I_o}$ and $H_{u,i}^{T}(g(i))$. This results in the histogram shown in Figure 8.4b. The curve for MST coding now resembles a Gaussian curve with the features resulting in Hamming distances between the lower and the higher threshold almost always being assigned to intra coding. Compared to the results shown for the intra coding in Figure 4.7 from Chapter 4, the curves are shifted to the left. Apart from the different properties of the used datasets, the curves are shifted as each observation has two coding modes available, where always the best in terms of minimum required bits is selected, resulting in an overall minimization of the Hamming distances to their respective references.

8.4 Lossy map sparsification

While feature coding techniques provide the opportunity for lossless map compression, the compression gain is rather limited. Similar to video coding, where lossless approaches provide only comparably small compression ratios, the majority of today's compression gains stem from lossy compression techniques by omitting information that is not perceivable or not required to achieve a certain goal. While for video coding, the goal is to provide the maximum user experience in terms of subjective tests or objective measurements given a target bitrate or vice versa, the goal of lossy map compression is to maintain the relocalization performance to assist further clients entering the map to determine their current whereabouts. In other words: Given a set of map points, the target is to identify the subset of points that is essential for successful relocalization. The identified requirements for the map points are threefold:

- Most important, the remaining set of map points should cover the whole mapped area. This ensures that the map covers the same spatial extent before and after map compression.
- Second, frequent observation of the map points is beneficial. The assumption is that these points are, due to their visibility and stability, also more likely to be discovered during a relocalization attempt. However, this introduces a bias towards locations that have been visited more frequently by the sequence used for the map creation.
- Third, the map points should provide a good compression performance using the proposed coding approach. This allows keeping more map points within a fixed bit budget. As a side effect, map points that exhibit good compression ratios implicitly also match the second requirement, as explained later on.

Following the same key tenet as [104], [105], the problem of map sparsification is formulated as a minimization problem. In contrast to the previous works, where each map point received a weight proportional to a visibility score, each map point is weighted according to

the individual coding costs. Following previous notation, the problem is defined as

$$\begin{aligned} \text{minimize} \quad & \mathbf{q}^T\mathbf{x} + \lambda_1 \mathbf{1}^T\boldsymbol{\xi} + \lambda_2\zeta \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \boldsymbol{\xi} \geq b\mathbf{1} \\ \text{and} \quad & \mathbf{c}^T\mathbf{x} + \zeta = H_p \\ & \mathbf{x} \in \{0,1\}^{N_p}, \\ & \boldsymbol{\xi} \in \{\mathbb{Z}_0^+\}^{N_k}, \\ & \zeta \in \mathbb{Z}_0^+, \end{aligned} \tag{8.12}$$

where $\mathbf{x}$ denotes a vector of size N_p containing the binary decision whether to keep or to remove a map point. Hence, N_p is defined as the total number of map points contained in the map. The vector $\mathbf{q}$ contains an integer weight for each map point. In previous work [105], this vector accounted for the visibility of each map point, where a higher weight is assigned to the map points that have very few observations. However, in this work, the lossless compression should be tightly coupled with the optimization step.

To this end, the weight vector $\mathbf{q}$ is not directly derived from the number of observations, but uses an estimate of the mean coding costs per observation for a particular map point. More specifically, each entry $\mathbf{q}_u$ contains the number of bits required to store a particular map point u (Equation (8.11)) normalized by the number of attached observations. The entries of $\mathbf{q}$ are then obtained by evaluating

$$\mathbf{q}_u = \frac{H_u}{|V_u|} \tag{8.13}$$

for each map point. To benefit from Integer Linear Programming, the values are rounded to the next integer value. The motivation behind this formulation is threefold:

- First, map points that exhibit observations that are visually similar and, therefore, easy to compress due to their small Hamming distances are favored. A small Hamming distance indicates that these local features are true matches, whereas high Hamming distances can be an indicator of unstable features or false correspondences.

- Second, with the coding cost for observations using the minimum spanning tree being lower than the intra observation coding costs, this approach favors map points with many observations attached. Hence, this problem formulation implicitly incorporates the number of observations per map point as optimization criteria. The effect is illustrated in Figure 8.5, where the rate normalized by the number of observations is shown. The more observations are attached to a map point, the fewer bits are required for coding each individual observation due to the higher chance of finding a reference feature with lower Hamming distance in the minimum spanning tree.

- Third, retaining a sparser set of map points, which feature a good compression ratio, shifts the probabilities p_0^I and p_0^T to higher values and allows even more efficient coding. However, this effect is not exploited in this work.

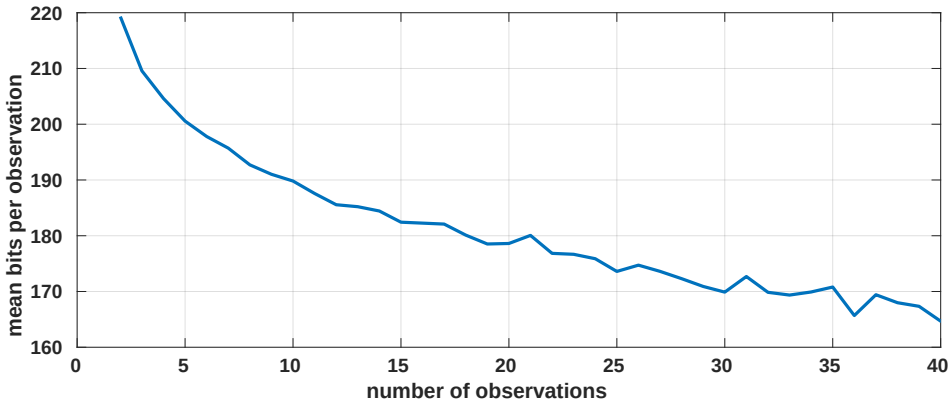


Figure 8.5: Average number of bits required to code a map point normalized by the number of observations. The data has been obtained from all *EuRoC Machine Hall* sequences. The average number of bits per observation reduces for map points with many observation attached as the MST tree coding becomes more efficient due to the increased number of reference features (adapted from [4], ©2018 IEEE).

As pointed out by Dymczyk et al. [104], a formulation as a quadratic optimization problem to penalize co-visible and co-located map points is omitted here because of the computational complexity that can quickly render the problem unsolvable in a reasonable time. However, two side constraints complement the optimization problem.

First, following related works [104], [105], the map points should be retained such that every keyframe contains at least b observations after the map point reduction. To achieve this goal, a $N_k \times N_p$ visibility matrix $\mathbf{A}$ is defined where N_k denotes the number of keyframes. Each entry of $\mathbf{A}$ is a binary indicator of whether a keyframe can observe a map point u.

Second, a constraint is added that restricts the size of the map by using $\mathbf{c}$ containing the unnormalized number of bits per map point. The total number of bits for the map points can be adapted with H_p to reflect the number of bits available. This constraint allows to restrict the size of the target map and adapt to superimposed constraints such as fixed transmission rates. Fulfilling the two constraints simultaneously is usually not possible. To relax the optimization problem, the slack variables $\boldsymbol{\xi}$ and ζ are added, allowing deviations from the side constraints.

8.5 Experimental evaluation

8.5.1 Map compression

Following previous chapters, a vocabulary of size $N_v = 100$k has been used for the intra observation coding. For the evaluation, ORB-SLAM2 [70] was used in stereo mode in combination with the *EuRoC* MAV dataset. The experiments were conducted on an Intel i7-3770 CPU @ 3.40 GHz. The probabilities for intra observation and MST coding have been obtained beforehand. The probabilities are set to $p_0^I = 0.85$ and $p_0^T = 0.86$ when using the intra observation coding only for the first observation. This mode is denoted as MST in the fol-

	essential [MB]	intra [MB]	MST [MB]	intra+MST [MB]
MH01	7.2	3.5	3.1	3.0
MH02	9.0	4.4	3.7	3.6
MH03	11.1	5.4	4.6	4.5
MH04	8.2	4.0	3.3	3.2
MH05	10.0	4.9	4.0	3.9

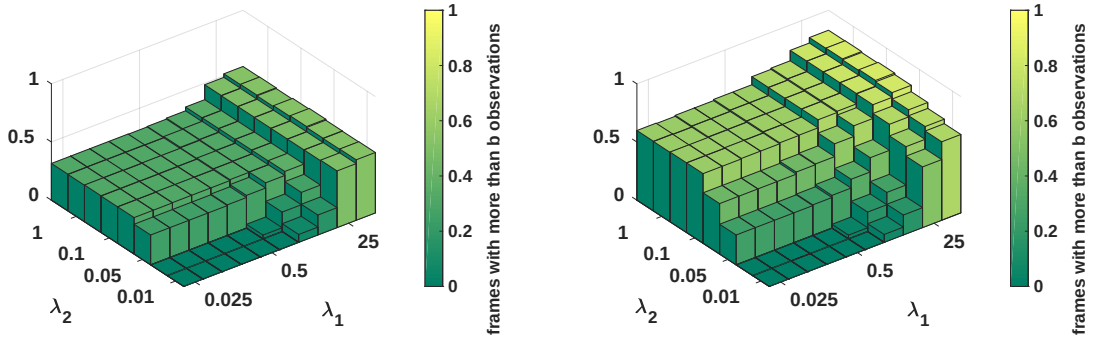
Table 8.2: Comparison of the map sizes without compression in the essential version, with all visual information coded using intra coding, with all features using the coding order defined by MST and the possibility to switch for each observation. The values were obtained on different sequences of the *EuRoC* dataset (adapted from [4], ©2018 IEEE).

lowing and follows the formulation of Equation (8.10). When allowing both intra and MST coding, the probabilities are set to $p_0^I = 0.83$ and $p_0^T = 0.90$. This coding mode using Equation (8.11) is denoted as intra+MST in the following. The main reason for the probability p_0^I for intra+MST coding being lower is that apart from the best observation i^* in terms of minimum Hamming distance also features with a high Hamming distance in the minimum spanning tree are coded with this mode thus increasing the mean Hamming distance. On the other side, p_0^T increases for intra+MST as linked observations with high Hamming distances are now intra coded. This particular variation of the probabilities can be observed indirectly in Figure 8.4, where the histograms of the Hamming distances are plotted.

The essential map is compared to compressed versions using different coding modes in Table 8.2. The column named intra coding shows the results when relying only on intra coding without the usage of the minimum spanning tree. Next, the results for MST coding are presented, where only the first observation is intra coded and all other observations use MST coding as given in Equation (8.10). Finally, intra+MST denotes the possibility to flexibly switch between intra and MST coding for each observation, as defined in Equation (8.11). Already more than 50% of the required storage for the map is saved by applying intra coding. The combination of both MST and intra coding result in storage requirements of only 39% of the uncompressed size for a map of the *EuRoC MH04* sequence. Apart from the quantized keypoint orientation, all coding methods are lossless and contain the same visual information as the essential map. The number of bits spent to signal the visual word index is measured with 16.9 bits. The mean size of coding the keypoint information is $R^{I,kpt} = 25.1$ bits. This value is dependent on the properties of the sequence, such as the resolution and the distribution of the scale-space levels. For signaling the observation information, $R^{I_o,obs} = 19.3$ bits are required to store the keyframe and feature identifier.

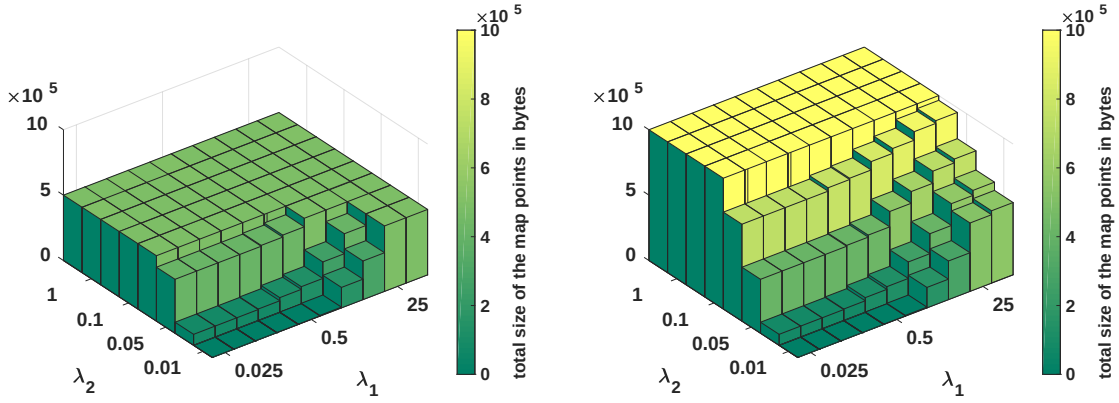
8.5.2 Map sparsification

While λ_1 in Equation (8.12) regulates the impact of the coverage constraint, which tries to retain at least $b = 50$ observations per frame, λ_2 has an impact on how close the targeted map size is approached. For lossless compression, experiments were conducted to determine suitable choices of λ_1 and λ_2. For these experiments, the desired number of bits for storing the



(a) Coverage constraint 0.5 MB. (b) Coverage constraint 1 MB.

Figure 8.6: Influence of λ_1, λ_2 on the coverage side constraint as the fraction of keyframes fulfilling the constraint for a target size of map points of 0.5 MB (a) and 1 MB (b). Higher values of λ_1 result in approaching the coverage constraints (adapted from [4], ©2018 IEEE).



(a) Cost constraint 0.5 MB. (b) Cost constraint 1 MB.

Figure 8.7: Influence of λ_1, λ_2 on the total cost side constraint for a target size of map points of 0.5 MB (a) and 1 MB (b). Higher values of λ_2 result in achieving the cost constraints (adapted from [4], ©2018 IEEE).

map points has been set to 1 MB and 0.5 MB. The influence on the coverage constraint trying to keep more than $b = 50$ observations per keyframe is shown in Figure 8.6. The influence on the cost constraint approaching the desired map size is illustrated in Figure 8.7. For values of $\lambda_1 \geq 25$, a reasonable percentage of frames keep more than 50 observations. For values of $\lambda_2 \geq 0.1$, the target map sizes are achieved. Considering the results of all experiments, the values are set to fixed values as $\lambda_1 = 25.0$ and $\lambda_2 = 1.0$ for the remaining evaluation. It is worth noticing that other values might be more suitable for other maps and applications.

The tight coupling of the optimization step and the proposed compression approach is investigated in Table 8.3. Prioritizing map points that feature minimum coding costs per observations favors map points with many observations attached due to the lower coding costs. Hence, the mean number of observations per map point $|V|$ increases for higher compression ratios showing the implicit consideration of the number of observations in the proposed

	$R^{T,vert}$ [bits]	$R^{I,res}$ [bits]	$R^{T,res}$ [bits]	$\|V\|$	N_p
full	2.9	167.2	127.2	6.4	17,426
2.0 MB	3.4	166.3	124.6	10.7	7,464
1.0 MB	4.1	165.6	121.6	17.4	2,370
0.5 MB	4.3	162.0	117.1	19.5	1,063

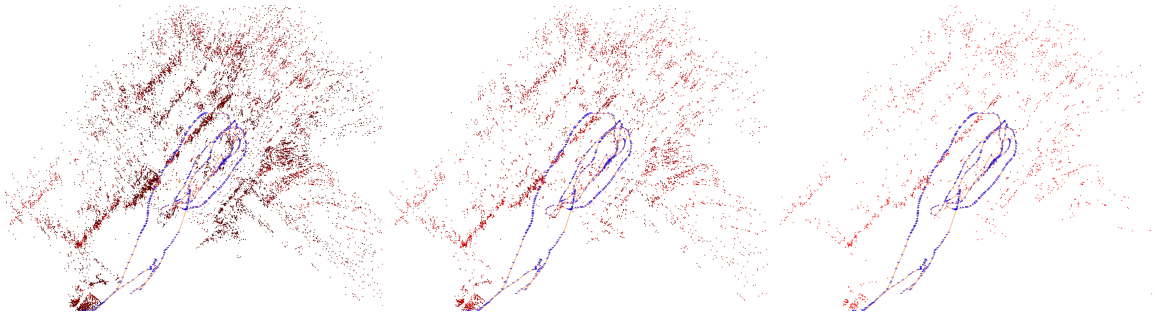
Table 8.3: Mean number of bits for different map compression methods and properties that are affected when combined with map sparsification. Results are mean values obtained from the *EuRoC MH01* sequence. $|V|$ denotes mean number of observations per map point and N_p the number of retained map points. The map with fewer map points contains only map points with high number of observations showing the implicit consideration when using the cost-aware optimization (adapted from [4], ©2018 IEEE).

sparsification scheme. In addition, the map points exhibiting smaller coding costs for both coding methods are preferred. The result is visible for both $R^{I,res}$ and $R^{T,res}$, where the cost reduces for the residual when obtaining sparser maps. Regarding the time consumption, the optimization problem with a target map size of 1 MB takes 5.77 s to converge to the final solution. Some examples of the resulting maps are shown in Figure 8.8.

The full map of the *KITTI 00* sequence requires about 160.9 MB, the essential map 38.5 MB, and the intra+MST compressed map uses 16.9 MB consisting of 1,392 keyframes and 139,246 map points. Figure 8.9 shows the original map and two reduced versions to 10 MB and 5 MB preserving 53,863 and 17,606 map points, respectively.

8.5.3 Relocalization performance

In order to gain insights about the influence of the map sparsification on the relocalization performance, two experiments were conducted. First, the relocalization capabilities of the reduced maps are evaluated by considering only individual frames for the task using the Bag-of-Words based relocalization module from ORB-SLAM2. It identifies similar keyframes



(a) Original map (b) 2.0 MB map (c) 1.0 MB map

Figure 8.8: View on different compressed versions of the map obtained from the *EuRoC MH01* sequence. The keyframes are shown in blue. The map points are color coded depending on the number of observations ranging from few observations (black) to many observations (red). Although not explicitly modeled in the optimization, the number of map points with few observations decreases when moving to higher compression levels (adapted from [4], ©2018 IEEE).

(a) Original map (b) 10.0 MB map (c) 5.0 MB map

Figure 8.9: Top-down view on different compressed versions of the map obtained from the *KITTI 00* sequence. The original map occupies 160.9 MB uncompressed and 16.9 MB using intra+MST coding. The map points are color coded depending on the number of observations ranging from few observations (black) to many observations (red).

and performs a 6DoF pose estimation by solving the Perspective-n-Point (PnP) problem. To this end, the relocalization module uses a RANSAC-based scheme with map points observed by the candidate keyframe. Second, the impact on a system that can estimate the trajectory by using not only the map information but also additional visual odometry (PnP+VO) is evaluated. The latter is able to provide a relative motion estimate for frames where no map points could be associated with the prior map. For example, when the part of the current camera trajectory is not covered by the prior map. In this case, a relative motion estimate using the visual odometry can support the relocalization process. For both experiments, a map of the *EuRoC MH01* sequence has been created beforehand. ORB-SLAM2 is evaluated on the map of *EuRoC MH01* using the *EuRoC MH02* sequence as a query. First, the full map of *EuRoC MH01* is used to obtain ground truth poses of the *EuRoC MH02* sequence in the map of *EuRoC MH01* using the full capabilities of ORB-SLAM2 employing the PnP+VO approach. Then, the map sparsification is applied to the *EuRoC MH01* map and subsequently, the performance is assessed by performing relocalization. In all experiments, the mapping capabilities of ORB-SLAM2 are deactivated.

The results for both PnP and PnP+VO are shown in Table 8.4. The relocalization quality is measured in terms of successfully localized frames that are located in the vicinity of 10 cm to their ground truth position. The proposed approach, including the map point coding costs in the optimization process (denoted here as MILP), is compared with the results obtained by employing an objective function similar to the one used by [105]. The difference is that this approach uses the number of observations as the weight for the entries of $\mathbf{q}$ as $\mathbf{q}_u = V_{max} - |V_u|$, where V_{max} denotes the maximum number of observations attached to any map point in the map and where $|V_u|$ denotes the number of observations attached to the specific map point u. This formulation is referred to as ILP in the following. The ground truth experiment is able to successfully localize 3037 frames out of 3040 frames contained in the MH02 sequence using both the full map information and additional visual odometry. The same setup using only PnP for relocalization on the full map is able to determine valid locations for 2161 frames.

method	map size	PnP	PnP+VO	N_p
full	3.00 MB	2161	3037	17426
ILP	1.00 MB	1777	3036	2290
	0.75 MB	1724	2825	1518
	0.60 MB	1729	2866	1144
	0.50 MB	1520	2859	909
	0.40 MB	1226	2681	657
	0.30 MB	1085	2615	416
	0.20 MB	1011	1959	219
MILP	1.00 MB	**1803**	2994	2370
	0.75 MB	**1864**	**2989**	1641
	0.60 MB	**1824**	**2991**	1302
	0.50 MB	**1651**	**2834**	1063
	0.40 MB	**1510**	**2794**	791
	0.30 MB	**1267**	**2650**	507
	0.20 MB	**1026**	**2154**	278

Table 8.4: Number of successfully localized frames by querying different sized map representations of the *EuRoC MH01* sequence with frames of the *EuRoC MH02* sequence. Localized frames have to be within 10 cm compared to ground truth poses obtained by PnP+VO on the full map. N_p denotes the number of retained map points. The proposed MILP approach is able to keep more map points for the same target map size compared to ILP. MILP almost always outperformed the ILP approach (adapted from [4], ©2018 IEEE).

The first observation from Table 8.4 is that the MILP approach consistently outperforms the ILP approach in terms of the number of localized frames. Only at 1.0 MB, the results are slightly worse compared to the ILP approach. The second observation is that due to the optimization for reducing the costs per map point, more map points could be stored compared to the ILP approach. This enables more frames to be successfully localized for both PnP and PnP+VO. For larger map sizes above 0.6 MB, a saturation is reached for both PnP and PnP+VO. At a target map size of 0.6 MB, resulting in a total compression of 91.7% compared to the essential map of size 7.2 MB, the proposed method is able to localize 5.5% more frames using PnP and 4.4% more frames using PnP+VO compared to the ILP approach.

8.6 Summary

In this chapter, a novel approach for efficiently compressing the visual information contained in a map from a state-of-the-art visual SLAM system has been proposed. To this end, the map properties have been analyzed and the visual information has been identified as one of the major sources of storage requirement. To compress the visual outline, the inherent dependencies within a visual map are exploited to encode the contained visual information differentially. This lossless compression approach has been evaluated on different sequences and the results show that the size of maps can be reduced to 39% of their original size without losing relevant visual information.

Moreover, the properties of the compression are exploited in a subsequent lossy map sparsification approach, which can be solved through integer linear programming techniques. With the lossy compression, a data reduction by 91.7% is achievable, while similar tracking capabilities compared to the full map are preserved. The approach can be extended by quadratic linear programming [104], [105] to include additional constraints. Also, related work targeting the compression of keyframe poses [101], as well as other visual SLAM systems relying on local feature descriptors, can be combined with this algorithm.

Chapter 9

Conclusion and outlook

9.1 Conclusion

The recent improvements of visual SLAM systems and the variety of application scenarios promise a large-scale adoption of visual SLAM in our everyday life. Providing accurate locations down to centimeter accuracy paves the way for a myriad of applications ranging from navigation tasks over robotic exploration to fully autonomous driving. However, when using embedded and mobile devices, sacrifices have to be made considering the accuracy of a visual SLAM system when enforcing local processing. This thesis focuses on techniques to provide accurate visual SLAM results on energy-constrained devices by outsourcing the visual SLAM to a more powerful processing node possibly integrated into an edge-cloud architecture.

To this end, a complete binary feature coding framework is proposed to transmit the visual cues to the server efficiently. A novel intra coding mode is introduced in Chapter 4 that allows the encoding and decoding of individual features using a visual vocabulary as shared knowledge between client and server. Experimental results show that a bitrate reduction between 33.5% and 41% can be achieved. Based on this coding method, a novel rate allocation method is proposed to adapt the amount of information transmitted to the available transmission capacity on the descriptor level. This allows computer vision tasks to be carried out at low bitrates such as 7 kbits/frame for a homography estimation task with 83% success rate.

In Chapter 5, an inter-frame coding approach inspired by the block-based motion compensation used in hybrid video coding is added. Going beyond the state-of-the-art, the implemented coding methods are combined with a rate adaptation on feature level tailored to the specific needs of a visual SLAM system. Consequently, these building blocks are combined to a client-server monocular visual SLAM architecture and are evaluated in terms of coding gain and visual SLAM results. Experimental results for the *TUM RGB-D* dataset show that the proposed system is capable of reducing the bitrate by 39.1% using intra coding, by 55.3% using a single reference frame, and by 61% using four reference frames compared to uncompressed transmission. In addition, a reduction by 79.2% down to 75 kbits/frame is possible using feature selection while being able to run ORB-SLAM2 with only a sub-centimeter degradation in accuracy.

Metric scale remote visual SLAM is achieved in Chapter 6 by extending the existing coding architecture to support a stereo camera setup. The coding of the necessary information is facilitated by either sending depth information or the additional visual information extracted from the stereo-view. A reduction in required bits by 57.9% compared to uncompressed transmission is possible on the *EuRoC* dataset using multiple reference frames and stereo-view coding. The coding approaches have been tested on embedded devices and are able to achieve real-time performance.

In order to include information provided by clients operating in the nearby environment, the visual SLAM architecture is extended to a collaborative approach in Chapter 7. Individual maps are created for each client which are merged on the detected overlap. The system performance is evaluated in terms of coding gain and visual SLAM results using the proposed centralized collaborative scheme on the *KITTI* dataset. Using the depth value coding from Chapter 6 allows a reduction in terms of bits of up to 70.8% compared to uncompressed transmission. The improvement stemming from the collaborative mapping of an environment is up to 53.7% in terms of ATE compared to individual mapping.

Map exchange is facilitated by using the inherent structure of a visual SLAM map in Chapter 8. Interpreting the visual features in keyframes as observations of map points allows the differential coding of all attached observations of a map point. Incorporating the resulting coding costs into an optimization problem for map sparsification tightly couples the proposed compression method with the map point reduction. The results are evaluated in terms of successful relocalization attempts. The results for the lossless compression approach show that the size of maps can be reduced to 39% of the original size without losing relevant visual information. With the lossy compression, a data reduction by 91.7% is achievable, while similar tracking capabilities compared to the original map are preserved. In contrast to related work between 4.4% and 5.5% more frames can be successfully localized.

In summary, this dissertation closes a gap by focusing on the efficient exchange of information in the context of visual SLAM and addresses the urgent need for data-efficiency in future collaborative mapping scenarios. Although this work provides some steps towards efficient collaborative visual SLAM, some research is left for future work.

9.2 Outlook

The approaches presented for visual feature coding are evaluated mainly using local binary ORB features. However, recent advances in deep-learned features are boosting more and more the performance of computer vision tasks and, hence, also visual SLAM. These features are not yet considered here and need further investigation.

Although ORB-SLAM2 condenses the recent advantages in feature-based visual SLAM into a single framework, alternatives and successors to ORB-SLAM2 might have different requirements in terms of input and information selection. A straightforward example is direct visual SLAM, which has not been considered. While approaches that are both dense and direct are not well suited for remote visual SLAM, the nascent technology of sparse direct approaches, such as DSO, is. In order to compress the information, a pixel selection can be

implemented on the client-side. Transmitting only the best pixel intensity patches according to a gradient-based selection function and reducing the number of required bits by exploiting statistical properties can be a topic of further research.

If a server-based SLAM processing is employed, the feedback is delayed by the time required for communication and processing, which still might exceed practical limitations for certain applications. A typical example would be a drone-based exploration, where immediate feedback at a high frequency is required for the flight controller. To circumvent this problem, novel technologies such as 5G, in combination with an edge cloud located in the near vicinity of the client, could be investigated. Alternatively, a lightweight local visual odometry could be implemented at the client to provide short-term positioning. These small local maps can be augmented with the more precise and larger-scale mapping carried out at the server.

When discussing a roll-out of edge-cloud based architectures with a visual SLAM instance running at a base station of a mobile network, more challenges are faced. Technologies for seamless visual SLAM handover between neighboring cells need to be investigated. Aggregating the individual proximity-based SLAM maps collected at the edge servers to a globally consistent map at the central cloud poses further challenges also with respect to scaling the architecture.

The proposed map exchange approach reaches its limits by considering only complete static maps. Extending the approach to partial coding of maps, where only newly created map points and observations have to be transmitted, would be beneficial. Also, approaches for updating map points and version control play a vitally important role in future applications.

Bibliography

Publications by the author

Journal publications

[1] D. Van Opdenbosch and E. Steinbach, "Collaborative Visual SLAM using Compressed Feature Exchange," *IEEE Robotics and Automation Letters (RAL)*, vol. 4, no. 1, pp. 57–64, 2019.

Conference publications

[2] D. Van Opdenbosch, M. Oelsch, A. Garcea, and E. Steinbach, "A Joint Compression Scheme for Local Binary Feature Descriptors and their Corresponding Bag-of-Words Representation," in *IEEE Conference on Visual Communications and Image Processing (VCIP)*, 2017, pp. 1–4.

[3] D. Van Opdenbosch, M. Oelsch, A. Garcea, T. Aykut, and E. Steinbach, "Selection and Compression of Local Binary Features for Remote Visual SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7270–7277.

[4] D. Van Opdenbosch, T. Aykut, M. Oelsch, N. Alt, and E. Steinbach, "Efficient Map Compression for Collaborative Visual SLAM," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, USA, 2018, pp. 992–1000.

[5] D. Van Opdenbosch and E. Steinbach, "Flexible Rate Allocation for Binary Feature Compression," in *IEEE International Conference on Image Processing (ICIP)*, Athens, Greece, 2018, pp. 3259–3263.

[6] D. Van Opdenbosch, G. Schroth, R. Huitl, S. Hilsenbeck, A. Garcea, and E. Steinbach, "Camera-based Indoor Positioning using Scalable Streaming of Compressed Binary Image Signatures," in *IEEE International Conference on Image Processing (ICIP)*, Paris, France, 2014, pp. 2804–2808.

[7] D. Van Opdenbosch and E. Steinbach, "AVLAD: Optimizing the VLAD Image Signature for Specific Feature Descriptors," in *IEEE International Symposium on Multimedia (ISM)*, 2016, pp. 545–550.

[8] A. Hanel, A. Mitschke, R. Boerner, D. Van Opdenbosch, D. Brodie, and U. Stilla, "Metric Scale Calculation For Visual Mapping Algorithms," in *ISPRS Technical Commission II Symposium 2018*, Riva del Garda, Italy, 2018, pp. 433–440.

[9] M. Oelsch, D. Van Opdenbosch, and E. Steinbach, "Survey of Visual Feature Extraction Algorithms in a Mars-like Environment," in *IEEE International Symposium on Multimedia (ISM)*, Taichung, Taiwan, 2017, pp. 322–325.

[10] T. Aykut, C. Zou, J. Xu, D. Van Opdenbosch, and E. Steinbach, "A Delay Compensation Approach for Pan-Tilt-Unit-based Stereoscopic 360° Telepresence Systems Using Head Motion Prediction," in *IEEE International Conference of Robotics and Automation (ICRA)*, Brisbane, Australia, 2018, pp. 1–9.

[11] A. Garcea, J. Zhu, D. Van Opdenbosch, and E. Steinbach, "Robust Map Alignment for Cooperative Visual SLAM," in *IEEE International Conference on Image Processing (ICIP)*, Athens, Greece, 2018, pp. 4083–4087.

[12] A. Dettmann, D. Kuehn, D. Van Opdenbosch, T. Stark, H. Peters, S. Natarajan, S. Kasperski, A. Boeckmann, A. Garcea, E. Steinbach, and F. Kirchner, "Exploration in Inaccessible Terrain Using Visual and Proprioceptive Data," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Madrid, Spain, 2018.

General publications

[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[14] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover.," Stanford University, Tech. Rep., 1980.

[15] D. Scaramuzza and F. Fraundorfer, "Visual Odometry : Part I: The First 30 Years and Fundamentals," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[16] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II: Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[17] C. Shannon, "The mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[18] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, 1952.

[19] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987. [Online]. Available: http://doi.acm.org/10.1145/214762.214771.

[20] J. Duda, K. Tahboub, N. J. Gadgil, and E. J. Delp, "The use of asymmetric numeral systems as an accurate replacement for Huffman coding," in *Picture Coding Symposium (PCS 2015)*, 2015, pp. 65–69.

[21] T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[22] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615 –1630, 2005.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1, pp. 43–72, 2005.

[24] S. Gauglitz, T. Hoellerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *International Journal of Computer Vision*, vol. 94, no. 3, pp. 335–360, 2011.

[25] J. Heinly, E. Dunn, and J. M. Frahm, "Comparative evaluation of binary features," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 759–773.

[26] L. Baroffio, A. Redondi, M. Tagliasacchi, and S. Tubaro, "A survey on compact features for visual content analysis," *APSIPA Transactions on Signal and Information Processing*, vol. 5, pp. 1–22, 2016.

[27] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6959 –6968.

[28] I. Sobel, "History and Definition of the so-called Sobel Operator," in *Pattern Classification and Scene Analysis*, 1973, pp. 271–272.

[29] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679 –698, 1986.

[30] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision (ICCV)*, 1999, pp. 1150 –1157.

[31] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[32] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of Applied Statistics*, vol. 21, pp. 224–270, 1994.

[33] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[34] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Procedings of the Alvey Vision Conference 1988*, 1988, pp. 23.1–23.6.

[35] E. Rosten and T. Drummond, "Machine Learning for High-speed Corner Detection," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 430–443.

[36] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 183–196.

[37] G. Francini, S. Lepsoy, and M. Balestri, "Selection of local features for visual search," *Signal Processing: Image Communication*, vol. 28, no. 4, pp. 311–322, 2013.

[38] W. Hartmann, M. Havlena, and K. Schindler, "Predicting matchability," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 9 –16.

[39] M. Dymczyk, E. Stumm, J. Nieto, R. Siegwart, and I. Gilitschenski, "Will it last? Learning stable features for long-term visual localization," in *IEEE International Conference on 3D Vision (3DV)*, 2016, pp. 572 –581.

[40] A. Richardson and E. Olson, "Learning convolutional filters for interest point detection," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 631 –637.

[41] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "TILDE: A Temporally Invariant Learned DEtector," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5279 –5288.

[42] H. Altwaijry, A. Veit, S. J. Belongie, and C. Tech, "Learning to Detect and Match Keypoints with Deep Architectures.," *British Machine Vision Conference (BMVC)*, 2016.

[43] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 467–483.

[44] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 778–792.

[45] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," in *International Conference on Very Large Data Bases (VLDB '99)*, 1999, pp. 518–529.

[46] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[47] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2548–2555.

[48] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 510–517.

[49] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "MatchNet: Unifying feature and metric learning for patch-based matching," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3279–3286.

[50] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4353 –4361.

[51] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 118–126.

[52] A. Loquercio, M. Dymczyk, B. Zeisl, S. Lynen, I. Gilitschenski, and R. Siegwart, "Efficient descriptor learning for large scale localization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3170–3177.

[53] L. Zheng, Y. Yang, and Q. Tian, "SIFT Meets CNN: A Decade Survey of Instance Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1224 –1244, 2018.

[54] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1–7, 2016.

[55] G. Bradski, "The OpenCV Library," *Dr Dobbs Journal of Software Tools*, 2000.

[56] S. Heymann, K. Müller, A. Smolic, B. Froehlich, and T. Wiegand, "SIFT implementation and optimization for general-purpose GPU," *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2007.

[57] F. Schweiger, G. Schroth, R. Huitl, Y. Latif, and E. Steinbach, "Speeded-up SURF: Design of an efficient multiscale feature detector," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 3475–3478.

[58] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," *IEEE International Conference on Computer Vision (ICCV)*, pp. 1470–1477, 2003.

[59] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 3304–3311.

[60] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Pérez, and C. Schmid, "Aggregating Local Image Descriptors into Compact Codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[61] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez, "Revisiting the VLAD image representation," in *ACM International Conference on Multimedia*, 2013, pp. 653–656.

[62] R. Arandjelovic and A. Zisserman, "All About VLAD," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1578–1585.

[63] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1437–1451, 2018.

[64] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-scale image retrieval with compressed Fisher vectors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 3384–3391.

[65] J. Heinly, J. L. Schönberger, E. Dunn, and J. M. Frahm, "Reconstructing the world in six days," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3287–3295.

[66] H. Strasdat, J. M. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?" In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2657–2664.

[67] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[68] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[69] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[70] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[71] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[72] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[73] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *IEEE International Conference on Computer Vision (ICCV)*, 2003, p. 1403.

[74] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[75] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 3565–3572.

[76] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2352–2359.

[77] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234.

[78] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Direct Monocular SLAM," in *European Conference on Computer Vision (ECCV)*, vol. 8690, 2014, pp. 834–849.

[79] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[80] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3923 –3931.

[81] S. Urban and S. Hinz, "MultiCol-SLAM - A Modular Real-Time Multi-Camera SLAM System," *arXiv preprint arXiv:1610.07336*, 2016.

[82] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct SLAM for omnidirectional cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 141 –148.

[83] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers, "Omnidirectional DSO: Direct Sparse Odometry with Fisheye Cameras," *IEEE Robotics and Automation Letters (RAL)*, vol. 3, no. 4, pp. 3693 –3700, 2018.

[84] R. Mur-Artal and J. D. Tardós, "Visual-Inertial Monocular SLAM With Map Reuse," *IEEE Robotics and Automation Letters (RAL)*, vol. 2, no. 2, pp. 796–803, 2017.

[85] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization," *IEEE Robotics and Automation Letters (RAL)*, vol. 3, no. 3, pp. 1418–1425, 2018.

[86] L. von Stumberg, V. Usenko, and D. Cremers, "Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2510–2517.

[87] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 354–366, 2013.

[88] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular SLAM with multiple Micro Aerial Vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 3963–3970.

[89] L. Riazuelo, J. Civera, and J. M. Montiel, "C2TAM: A Cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[90] P. Schmuck and M. Chli, "Multi-UAV collaborative monocular SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3863–3870.

[91] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM," in *IEEE International Conference of Robotics and Automation (ICRA)*, 2018, pp. 2466–2473.

[92] T. Cieslewski and D. Scaramuzza, "Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index," *IEEE Robotics and Automation Letters (RAL)*, vol. 2, no. 2, pp. 1–8, 2017.

[93] ——, "Efficient Decentralized Visual Place Recognition From Full-Image Descriptors," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2017, pp. 78–82.

[94] T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat, and R. Siegwart, "Map API - Scalable decentralized map building for robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6241–6247.

[95] M. Gadd and P. Newman, "Checkout my map: Version control for fleetwide visual localisation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5729–5736.

[96] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 1–8.

[97] T. Cieslewski and D. Scaramuzza, "SIPS: Unsupervised Succinct Interest Points," *arXiv:1805.01358v1*, 2018. [Online]. Available: http://arxiv.org/abs/1805.01358.

[98] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 778–785.

[99] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[100] L. Contreras and W. Mayol-Cuevas, "Trajectory-driven point cloud compression techniques for visual SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 133–140.

[101] ——, "O-POCO: Online point cloud compression mapping for visual odometry and SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4509–4514.

[102] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, "Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization," in *Robotics: Science and Systems XI*, 2015.

[103] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps - Summarizing experience for lifelong localization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2767–2773.

[104] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2536–2542.

[105] H. S. Park, Y Wang, E Nurvitadhi, J. C. Hoe, Y Sheikh, and M Chen, "3D Point Cloud Reduction Using Mixed-Integer Quadratic Programming," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013, pp. 229–236.

[106] W. Cheng, W. Lin, X. Zhang, M. Goesele, and M. T. Sun, "A data-driven point cloud simplification framework for city-scale image-based localization," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 262–275, 2017.

[107] H. Merzić, E. Stumm, M. Dymczyk, R. Siegwart, and I. Gilitschenski, "Map Quality Evaluation for Visual Localization," in *IEEE International Conference of Robotics and Automation (ICRA)*, 2017, pp. 3200–3206.

[108] A. Redondi, L. Baroffio, M. Cesana, and M. Tagliasacchi, "Compress-then-analyze vs. analyze-then-compress: Two paradigms for image analysis in visual sensor networks," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2013, pp. 278 –282.

[109] A. Redondi, L. Baroffio, L. Bianchi, M. Cesana, and M. Tagliasacchi, "Compress-then-Analyze vs Analyze-then-Compress: what is best in Visual Sensor Networks?" *IEEE Transactions on Mobile Computing*, vol. 1233, no. c, pp. 1–1, 2016.

[110] L.-Y. Duan, V. Chandrasekhar, J. Chen, J. Lin, Z. Wang, T. Huang, B. Girod, and W. Gao, "Overview of the MPEG-CDVS Standard," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 179–194, 2016.

[111] L. Baroffio, A. Canclini, M. Cesana, A. Redondi, and M. Tagliasacchi, "Briskola: BRISK optimized for low-power ARM architectures," in *IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 5691–5695.

[112] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009, pp. 83–86.

[113] T. Schöps, J. Engel, and D. Cremers, "Semi-dense visual odometry for AR on a smartphone," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014, pp. 145–150.

[114] Microsoft, *HoloLens HPU*, 2018. [Online]. Available: https://www.microsoft.com/en-us/research/blog/second-version-hololens-hpu-will-incorporate-ai-coprocessor-implementing-dnns/.

[115] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, 1992.

[116] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264 / AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 560 –576, 2003.

[117] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649 –1668, 2012.

[118] J. Chao, H. Chen, and E. Steinbach, "On the design of a novel JPEG quantization table for improved feature detection performance," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 1675 –1679.

[119] J. Chao, R. Huitl, E. Steinbach, and D. Schroeder, "A Novel Rate Control Framework for SIFT/SURF Feature Preservation in H.264/AVC Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 958–972, 2015.

[120] J. Chao and E. Steinbach, "Keypoint Encoding for Improved Feature Extraction From Compressed Video at Low Bitrates," *IEEE Transactions on Multimedia*, vol. 18, no. 1, pp. 25–39, 2016.

[121] L. Baroffio, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro, "Coding visual features extracted from video sequences," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2262–2276, 2014.

[122] A. Redondi, L. Baroffio, J. Ascenso, M. Cesana, and M. Tagliasacchi, "Rate-accuracy optimization of binary descriptors," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 2910 –2914.

[123] L. Baroffio, A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro, "Coding Local and Global Binary Visual Features Extracted from Video Sequences," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3546–3560, 2015.

[124] J. Lin, L. Y. Duan, Y. Huang, S. Luo, T. Huang, and W. Gao, "Rate-adaptive compact fisher codes for mobile visual search," *IEEE Signal Processing Letters*, vol. 21, no. 2, pp. 195 –198, 2014.

[125] S. S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, J. P. Singh, and B. Girod, "Location coding for mobile image retrieval," in *Mobile Multimedia Communications Conference (ICST)*, 2009, pp. 1–7.

[126] S. S. Tsai, D. Chen, and G. Takacs, "Improved Coding for Image Feature Location Information," *Proc. SPIE*, vol. 8499, pp. 1–10, 2012.

[127] C. Timmerer, *MPEG 121 Meeting Report*, 2018. [Online]. Available: https://mpeg.chiariglione.org/meetings/121.

[128] L.-Y. Duan, V. Chandrasekhar, S. Wang, Y. Lou, J. Lin, Y. Bai, T. Huang, A. C. Kot, and W. Gao, "Compact Descriptors for Video Analysis: the Emerging MPEG Standard," *arXiv:1704.08141*, 2017.

[129] L. Baroffio, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro, "Hybrid coding of visual content and local image features," in *IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 2530–2534.

[130] R. Hastings, *Making the Most of the Cloud: How to Choose and Implement the Best Services*. Scarecrow Press Inc., 2013.

[131] Nokia, *The edge cloud: an agile foundation to support advanced new services*, 2014. [Online]. Available: https://onestore.nokia.com/asset/202184 (visited on 11/28/2018).

[132] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *ACM International Conference on Multimedia Information Retrieval*, vol. 4, 2008, pp. 39–43.

[133] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[134] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[135] L. Bondi, L. Baroffio, M. Cesana, A. Redondi, and M. Tagliasacchi, "Multi-view coding of local features in visual sensor networks," in *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2015, pp. 1–6.

[136] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002.

[137] K. Pearson, "Note on Regression and Inheritance in the Case of Two Parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240–242, 1895.

[138] D Nistér and H Stewénius, "Scalable Recognition with a Vocabulary Tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 2161–2168.

[139] D. Galvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[140] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, vol. 94, no. 2, pp. 198–214, 2011.

[141] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[142] J. Straub, S. Hilsenbeck, G. Schroth, R. Huitl, A. Möller, and E. Steinbach, "Fast relocalization for visual odometry using binary features," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 2548–2552.

[143] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, p. 629, 1987.

[144] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, p. 48, 1956.

List of Figures

List of Tables