TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik
Computer Aided Medical Procedures & Augmented Reality / I16

Tracking by Detection of 3D Human Shapes in Multi-camera Environments

Chun-Hao Huang

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Thomas Huckle
Prüfer der Dissertation:		
	1.	Priv.-Doz. Dr. Slobodan Ilic
	2.	Prof. Dr. Daniel Cremers
	3.	Prof. Dr. Adrian Hilton

Die Dissertation wurde am 20.06.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 16.11.2016 angenommen.

Abstract

Understanding and analyzing human movements has long been studied and yet remains an active topic in computer vision field as it is a fundamental task for the development of more advanced artificial intelligence systems. It is particularly challenging when the observations are only visual-based, namely, RGB images from photometric sensors or range data from depth sensors; neither markers nor inertial sensors are available. This thesis aims at addressing this problem, which is also known as marker-less human motion capture (mocap.) or in a more general sense, *4D modeling*.

One common approach to this problem is the so-called *top-down* approach. That is, given a pre-defined 3D humanoid surface, one recovers its temporal evolutions in a frame-by-frame manner according to observations from various modalities. Such model-based strategies generally consist of two steps: 1) estimating the associations between primitives of input data and those of the reference template; 2) estimating the motion parameters based on the correspondences obtained above. The former handles data explanations, typically involving statistics or machine learning techniques, whereas the latter deals with parameterizations of motions such as articulations or non-rigid deformations, which are well studied in robotic and computer graphic fields. As such, top-down human shape tracking is an applied problem that requires knowledge from various background. In particular, this thesis contributes to the two aspects above respectively.

On one hand, most existing methods adopt the well-known Iterative-Closest-Point (ICP) framework that alternates between the aforementioned two steps iteratively. It provides decent results when converges, but requires close initializations which is not always available. To alleviate this problem, we employ a machine learning technique, random forests, to learn the associations between input data and reference surfaces offline. During the online tracking phase, forests predict the correspondences discriminatively, discover human shapes in a clutter scene and initialize the tracking quickly. This strategy is applied to both surface data (visual hulls from 3D reconstruction) and volumetric data (centroidal Voronoi tessellation). The frame-wise correspondences prevent errors from being accumulated and hence avoid tracking failures. With this frame-independent nature, our tracking pipeline shares the similar spirit to the tracking-by-detection paradigm in image domain, termed *3D tracking-by-detection of human shapes*.

On the other hand, in the context of marker-less human mocap, two most widely-used motion parameterizations are articulated skeletons that follows a kinematic chain and surfaces that deforms completely non-rigidly. While the former is a compact representation that resembles human anatomical structures, the latter is more general and models realistic deformations like loose apparel. We introduce a versatile joint skeleton-and-surface optimization concept that holds the merits of the two. As the nature of surface-based method, our approach is more accurate in shape deformations than traditional skeleton-based methods. Yet, it also offers correct skeletal poses, which are more semantic meaningful and often desired in applications. On top of that, a keyframe-based tracking pipeline is introduced to online identify distinctive skeletal poses as key-frames, which in turn improves the handling of missing data and large deformations.

Zusammenfassung

Das Erkennen und Analysieren menschlicher Bewegungen ist eine aktiv untersuchte und ungelöste Aufgabe des Fachgebiets Computer Vision und wichtig für die Entwicklung fortgeschrittener Systeme der Künstlichen Intelligenz. Es ist besonders anspruchsvoll, wenn es lediglich basierend auf Intensitäts- oder Tiefenbildern durchgeführt wird und nicht durch Marker oder Beschleunigungssensoren unterstützt wird. Diese Dissertation befasst sich mit dieser Problemstellung, die manchmal auch Bewegungserfassung (motion capture) in Mehrkamerasystemen oder auch 4D modeling genannt wird.

Ein Ansatz für dieses Problem ist das sogenannte top-down-Verfahren, welches die Verformung eines 3D-Modells des Menschen über die aufgenommenen Zeitschritte bestimmt. Dies geschieht in zwei Schritten: Zunächst werden die Korrespondenzen zwischen dem Modell und den aufgenommenen Daten ermittelt. Darauf aufbauend werden anschließend die Bewegungsparameter des Modells geschätzt. Während der erste Schritt die Daten etwa mittels statistischer Ansätze oder maschinellem Lernen erklärt, beschäftigt sich der zweite Schritt mit der Parametrisierung der Bewegung etwa mittels nicht starren Verformungen oder Gelenkartikulierungen. Da diese Verfahren im Einzelnen bereits intensiv untersucht wurden, kann der top-down-Ansatz auch als angewandtes Problem gesehen werden welches Kenntnisse aus verschiedenen Gebieten benötigt. Diese Arbeit leistet einen Beitrag zu beiden der oben beschriebenen Schritte.

Der erste Beitrag verwendet Random Forests, ein Klassifikationsverfahren aus unkorrelierten Entscheidungsbäumen. Zur Vermeidung von Fehlerakkumulation werden diskriminative Assoziationen vorgestellt, die zu langzeitstabilen Systemen führen. Die 3D-Modelle werden sowohl mittels einer oberflächenbasierten, als auch einer volumenbasierten Parametrisierung dargestellt. In beiden Fällen wird das Problem der Assoziationsbestimmung als Regressionsaufgabe definiert. Mit den diskriminativen Assoziationen funktioniert das System ähnlich wie tracking-by-detection in 2D, weshalb wir es als 3D tracking-by-detection der menschlichen Gestalt bezeichnen.

Im zweiten Beitrag wird ein informativerer Suchraum vorgeschlagen, der weiche Verformungen zulässt und zugleich die menschliche Körperhaltung beibehält. Die Körperhaltung wird dabei über die Lage der Gelenkpositionen parametrisiert. Diese ist in vielen Anwendungen nützlich, allerdings weniger allgemein als eine Verformungen des Modells. Wir führen die Idee ein, dass man beide Parametrisierungen gleichzeitig abschätzen kann.

Acknowledgments

It has been my pleasure to be part of the Computer Aided Medical Procedures (CAMP) chair for the past four years. The first person I would like to cordially thank is my Doktorvater PD. Dr. Slobodan Ilic. Accepting my application, he brought a young student whom he had never met before from the other side of the world to TUM. Meanwhile, Germany was, at that time, not a very common choice for Taiwanese students to pursue a doctoral degree in computer science. His open-mindedness and my audacity together resulted in this Ph.D. journey. Slobodan has long been a great support for my research. I am very grateful to him for his supervision and guidance.

The interactions with many CAMPers also widely broadened my horizon. I would like to thank the director of the chair, Prof. Dr. Nassir Navab, for his motivational words and inspirational discussions, always encouraging us to stretch out our capacity into different problems and enjoying the 'beauty' of freedom in academy. I would like to thank particularly the whole vision team for the brainstorming of ideas and paper corrections, and thank Dr. Federico Tombari for the uncountable hours of discussions on many technical details. Notably, as CAMP is one of the chairs that have most international students, there are many CAMPers whom I had no scientific collaborations with but I feel very happy to have met them. I would also like to pay whole-hearted thanks to the Hilla family, especially Martina, for helping a non-German speaker in so many administrative tasks.

Moreover, I would like to thank my external partners for the great collaborations. I had a great time working with Morpheo group from INRIA Grenoble in France. I enjoyed each time visiting them and always got to learn new things to help my own research. There were for sure many work which could not be done by myself alone. It was the synergy of the whole team that made them possible. A special thank goes to Prof. Dr. Edmond Boyer, who always provides me with valuable opinions, both on research direction and paper writing.

Not limited to research environment, a heartfelt thank goes to Taiwanese Ministry of Education which founded the first two year of my study and to Hsing-Fen for being the guarantee of this scholarship. The pursue for Ph.D. abroad may dates back to six years ago. Thanks to all my friends in Taiwan for standing by my side all the way along and friends in Germany for enriching my life here. Thanks to Chia-Wei, Tsu-Han and my cousin Tzu-Wei for attending my defense and support me.

Last but not least, I am sincerely thankful to my parents 鄭金鳳 and 黃文生. Despite the adversity my family might have encountered, they believe in me and never let me worry about anything at home, so that I get to concentrate fully on my studies from the very beginning till today. I owe this doctoral degree to many people but they certainly deserve the biggest credit.

-俊豪

世界很大，reach what you cannot

Contents

List of Figures

List of Tables

Part I

Introduction and Background

1 Introduction

During the last few decades, the advent of commodity digital cameras has made a revolutionary change on how people record, store and transmit snapshots of their daily lives. Digital cameras have become ubiquitous because the cost of imaging sensors and computing capabilities has been substantially reduced. As a result, such prevalence also opens the possibilities to new technologies. Beyond the acquisition, storage and rendition of appearance enabled by photography, there is a growing need for tools that automatically measure and interpret the world underlying the pictures. *Computer vision*, as a part of artificial intelligence, strives to enable machines human beings' visual cognitive perception. A well-studied subject within this field deals with human subjects. For instance, human detection and pose estimation from images are fundamental high-level problems that the community has addressed several times during the past [2, 102, 113, 151].

A great part of vision research is dedicated to measuring three-dimensional shapes from 2D visual data, usually referred to as a *reconstruction* problem. In particular, considerable efforts have been devoted to contrive algorithms that automatically build models of objects that were observed from multiple views. Each view offers only partial appearance information of the reconstructed object. The challenge is to merge all these sources of information. The survey of [120] and more recent results, *e.g.* [60] show that precise photometric models can be recovered from multiple images and that Multiple View 3D Reconstruction has grown into a mature topic within the computer vision community.

After perceiving static 3D shapes, the next step that follows naturally is to interpret dynamic 3D objects. The problem has been approached from many directions [80, 104, 144, 149]. When the aforementioned multi-view 3D reconstruction algorithms are applied to multiple videos of moving and deforming objects, most of the available methods treat each frame independently, ignoring the dynamic nature of the observed event and thus the temporal redundancy in the data. Understanding dynamics, especially human motion, is an active topic in the community, with a wide spectrum of applications. In this thesis, we consider human motion as the *temporal evolutions of shapes* and aim at capturing robustly such evolutions with various meaningful motion parameterizations.

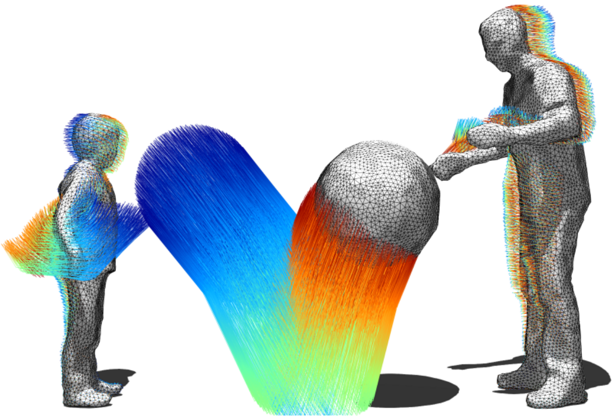


Figure 1.1: 4D modeling refers to the ability to produce animated shape sequences using videos of real dynamic scenes [34, 35].

1.1 Applications of Human Motion Tracking

The core interest of this thesis is the inference for human motion in 3D space. It is sometimes referred to as *4D modeling* [49, 74, 103][1] since one estimates not only static 3D points but also their temporal trajectories as shown in Fig. 1.1. The automatic computation of four-dimensional descriptions of scenes has a wide range of applications. In this section, we list a number of representative ones which we feel already made impacts in the industry.

- **Content production.** Manipulating digital media content is perhaps the most important application of spatial-temporal modeling. Consider for example the task of adding a virtual logo to the shirt of an actor. Editing the 2D video frame-by-frame to draw the logo would be extremely laborious, as one needs to account for occlusions, foldings of the shirt, self-cast shadows resulted from various illumination conditions. Drawing on a series of 3D meshes frame-by-frame and then rendering 2D views would overcome part of these issues but still remains tedious and impractical. If the *4D model* of the same performance is available, the task becomes rather straightforward. As 4D models contain the trajectories of each point of the shirt across time, the logo can be added to the first frame just once and its deformation can be automatically computed for the rest of frames. In other words, this information allows to *automatically propagate edits through time*. Moreover, the editions are not limited only to the appearance of objects. The geometry itself can be modified in a temporally consistent fashion, *e.g.* the actor can be made taller or skinnier [78]. The captured motion can also be transfered to a new character [12, 106]. Finally, inserting 4D models into dynamic simulations also becomes possible. This allows for example to capture dynamic objects and simulate their influence in particle systems, or to give an actor virtual long hair or clothing that will interact more realistically with his captured body.

[1] Yet note that 4D modeling applies not only to humans but generally to every object that deforms.

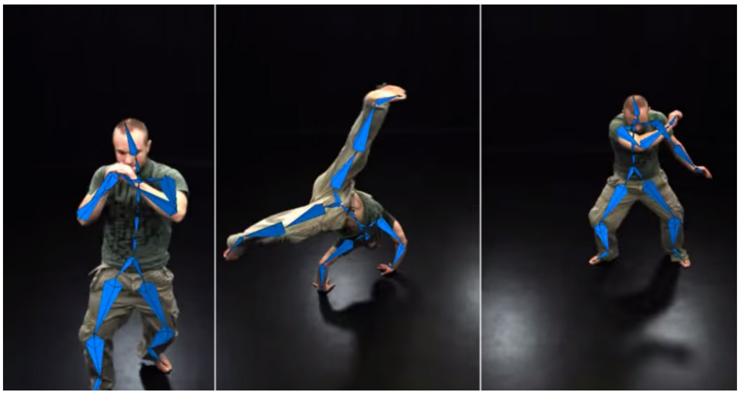


Figure 1.2: **Marker-less motion capture**. (©Captury)

- **Marker-less motion capture (mocap.)** Prior to content manipulation, another important application of human motion tracking is capturing actors' movement, because the content to be edited first has to be acquired anyway. More precisely, motion capture actually aims to solve the equivalent problem as 4D modeling does, rather than being its application. Currently the most widespread way of acquisition in the film industry deploys optical markers, where users are required to wear tight suits [160]. Sparsely attached on the body, markers are tracked quite accurately by infrared cameras during acting. The background software reasons with their locations to generate the positions of human joints. Since attaching and detaching markers are obviously tedious but inevitable, capturing motion without markers also attracts considerable research efforts in the past decade. As shown in Fig. 1.2, the techniques get gradually mature and some success are achieved under certain working conditions (simple uniform background in this case) [97, 130].

- **Compression, transmission, and real-time rendering.** 4D models contain explicit information on the motion of objects, and therefore on the temporal redundancy in the data. This exposed redundancy can apparently be compressed, yielding much more compact descriptions of dynamic scenes. If a 4D model is available, one can encode the shape information once, and only transmit one rigid transformation per frame. Such concise representations are of evident interest for storage, transmission over networks and real-time rendering.

- **Human-Machine interaction.** Capturing movement at interactive frame rates opens possibilities for human machine interaction. Measured body poses can be used as user input to provide intuitive manipulation of virtual objects or natural interaction with virtual agents. Beyond simple measurement, computed motion cues allow to recognize body gestures or complex actions and in some sense to use temporal information to evaluate the context and semantical content of 4D data. For example, tracking human facial deformations is crucial to understand non verbal communication.

- **Medical applications.** Recovering shape and motion can potentially be applied to medical field. For example, measuring the 3D motion of athletes allows for the bio-mechanical analysis of skilled movement such as a golf swing. It is also useful to measure the evolution of a gait pattern during physiotherapy or to automatically detect, record and document the evolution of patients for pathologies such as epileptic seizures. Motion capture also has applications for the observation of medical personnel. It allows to automatically document procedures and compare them to established work flow. This can save precious time when writing reports and help with the training of new staff. Furthermore, knowing the current stage of a procedure can help predict the end of a surgical intervention and get the next patient ready so that the usage of the operating room can be maximized. Other applications of interest include collision avoidance [85] or the estimation of the cumulated exposure to radiation of personnel in contact with x-ray sources [86].

1.2 Problem Definition and Challenges

Some progresses have been made recently to simultaneously digitize shape and motion that lay the foundation of this thesis. In the following, we briefly go through the standard pipeline, list the state-of-the-art approaches for each step and define our problem.

Starting from the observed multi-view videos, the objects of interest are first segmented to yield binary images, usually termed *silhouettes*, indicating foreground or background [14, 24, 91]. As depicted in Fig. 1.3, given the silhouettes from several views, 3D reconstruction methods such as [55, 100] are able to recover a coarse 3D shape, called *visual hulls* throughout this thesis. Such spatial reconstruction methods treat every frame independently, without taking into account the dynamic nature of moving objects. Thus, there is no temporal coherency on these raw 3D shapes and it is hard to analyze motion with them.

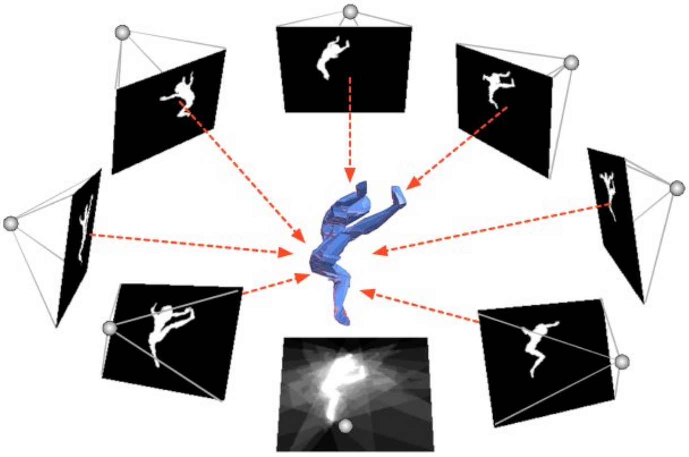


Figure 1.3: **Foreground segmentation [91] and 3D reconstruction [55].**

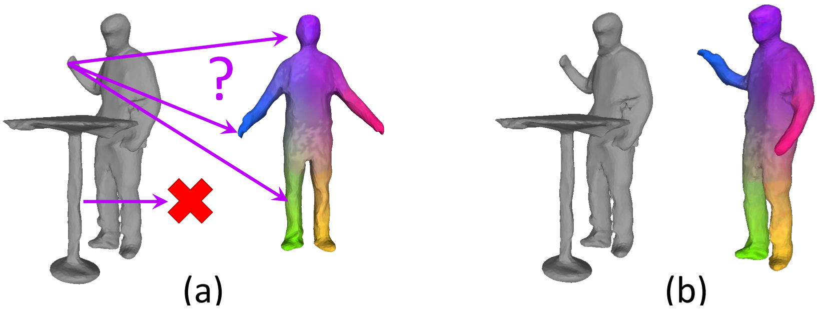


Figure 1.4: **Two fundamental steps in 3D human shape tracking**. Gray: input data; color: template. (a) data-model association step associates each input primitive to those of the template (Chapter 3-4); (b) optimization step deforms the template so that it looks similar to the observed visual hull (Chapter 5-6).

For this reason, people take these raw visual hulls as input and perform additional model-based tracking to recover the trajectories of each vertex on the template. It is sometimes also referred to as the *top-down* approach for human motion analysis because it already assumes a pre-defined 3D reference surface and guides its deformation with either 3D visual data[2] [35, 128], the processed 2D images (silhouettes) [61, 97] or directly with raw RGB images [115, 130]. With 3D visual hulls as input, the basic process of template tracking is illustrated in Fig. 1.4. One first associate points of visual hulls to those of the template (*association* step) and then deform the reference surface so that it resembles the input as much as possible (*tracking*, or *optimization* step). This procedure is repeated in a frame-by-frame manner. Many existing approaches [35, 43] base themselves on the well-known Iterative closest point (ICP) algorithm [37, 17, 158], which actually alternates between these two steps. Several challenges remain to be solved and questions to be answered in this ICP framework:

1. **Initialization**: the reference surface has to be aligned properly to the input of the first frame.

2. **Outlier rejection**: how to identify *outliers*? Namely, input points that cannot be explained by the template should not be associated to any reference vertices, like the ball in Fig. 1.1 or the desk in Fig. 1.4?

3. **Resilience from tracking error**: as ICP relies heavily on the previous tracked outcome to built data-model associations, errors from either of the two steps are prone to get accumulated and eventually break the tracking.

4. **Motion parameterization**: what is the accurate and yet meaningful representation of motion, surface deformation? skeletal poses? or volumes?

This dissertation concerns itself with these challenges. We build on top of the state-of-the-art non-rigid EM-ICP [35] and propose answers to the questions above. Generally speaking, we deal with the first three issues by introducing the

[2] In contrast, bottom-up approaches like [13] start with detecting human body parts in each 2D view and assemble them in 3D space afterwards.

concept of *discriminative associations*. Exploiting the state-of-the-art machine learning paradigm *random forests*, this strategy strives to decrease or even eliminate the dependence on previous frames when establishing new correspondences. These discriminative associations are in turn used to initialize the reference surface, identify outliers and get recovered from tracking drifting. Discovering associations in a frame-wise manner is analogous to detecting shapes in 2D images. Hence, we hereinafter refer to our approach as *tracking-by-detection of 3D human shapes*, which, to the best of our knowledge, does not exist in the literature of human shape tracking in multi-camera environments. Secondly, we proposed a simultaneous optimization framework that estimates shape deformations and human skeletal poses jointly. Such a versatile motion parameterization is not only accurate but also delivers richer semantic information.

1.3 Contributions

To achieve our objectives, we introduce a number of novel algorithms for 3D human motion tracking. The primary contributions of the dissertation are summarized in below:

- We mitigate the problem of error accumulations by introducing the concept of discriminative associations. Although the similar idea has been deployed in human pose estimation with RGBD data [112, 140], we are the first ones who utilize this concept in human shape tracking with full 3D data. With the help of regression forests, every input visual hull is warped back to the rest pose defined by the template. Such a warping is frame-independent and is learned offline.

- We discuss and analyze the interactions between two distinct motion parameterizations: skeletal poses vs. surface deformations and thereby introduce a joint optimization framework that estimates both simultaneously. The concept of joint optimization is first introduced in [132] but only used for refinement. We are the first ones who apply it directly to tracking and look at the optimization problem from the perspective of Bayesian network.

- The two aforementioned contributions comprise our tracking-by-detection of 3D human shapes framework. Besides surfaces, the advantage of this approach is also verified with another frequently-used shape parameterizations: volumes. We present a fully volumetric tracking-by-detection pipeline, which, to the best of our knowledge, is the first one of its kind.

- In order to validate the benefit of our methods, we record some multi-view datasets that contains outlying geometry. These outliers (specifically, furniture) are kept in silhouettes after background subtraction and stay consequently in the reconstructed visual hulls. To the best of our knowledge, none of the online-available datasets present such features.

These contributions will be presented and elaborated in detail in the following chapters, which are outlined in below:

1.4 Dissertation Outline

We provide a short overview for each chapter of the thesis. Most of the methods and material are published or under submission for a major conference or journal. Therefore, we additionally provide the related work for each chapter and encourage the readers to check the material online.

Chapter 2. We review the theoretical background of the thesis. In particular, we go through the deformable EM-ICP [35] and random forests [27, 45] which form the basis of our 3D tracking-by-detection model. The terminology and notations are also defined here for the later chapters.

- Cagniart, C., Boyer, E., Ilic, S.: Free-form mesh tracking: a patch-based approach. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 1339–1346. IEEE (2010)
- Cagniart, C., Boyer, E., Ilic, S.: Probabilistic deformable surface tracking from multiple videos. In: European Conference on Computer Vision (ECCV), pp. 326–339. Springer (2010)
- Criminisi, A., Shotton, J.: Decision forests for computer vision and medical image analysis. Springer (2013)

Chapter 3. In this chapter, a discriminative model for 3D human body correspondence estimation is introduced. We build our method on a random forest that regresses an input 3D point on a mesh to the corresponding location on the template. We describe a 3D surface in an implicit form with volumetric fields. The advantage of this approach is that one has organized data where features can be designed much easier than meshed indexed points.
Related work:

- Huang, C.H., Boyer, E., do Canto Angonese, B., Navab, N., Ilic, S.: Toward user-specific tracking by detection of human shapes in multi-cameras. In: Computer Vision and Pattern Recognition (CVPR), pp. 4027–4035. IEEE (2015)

Chapter 4. Continuing on this direction, in this chapter, we utilize different 3D shape parameterizations and formulate 3D tracking-by-detection in a fully volumetric manner. Instead of triangular meshes, centroidal Voronoi tessellation (CVT) is chosen to represent a 3D shape as uniform and anisotropic cells. The shape representation, deformation model, feature description, and primitive association are all built on the CVT volumetric representation.
Related work:

- Huang, C.H., Allain, B., Franco, J.S., Navab, N., Ilic, S., Boyer, E.: Volumetric 3d tracking by detection. In: CVPR 2016-IEEE Conference on Computer Vision and Pattern Recognition (2016)

- Allain, B., Franco, J.S., Boyer, E.: An efficient volumetric framework for shape tracking. In: Computer Vision and Pattern Recognition (CVPR). IEEE (2015). URL `https://hal.inria.fr/hal-01141207`

Chapter 5. We introduce a simultaneous optimization scheme that jointly recovers surface deformations and skeletal poses. This is a more versatile deformation framework, offering expressive motion parameterizations for various applications. This joint optimization scheme and the discriminative data-model associations estimated above form our complete 3D tracking-by-detection of human shapes pipeline.
Related work:

- Huang, C.H., Boyer, E., Ilic, S.: Robust human body shape and pose tracking. In: 3DV. IEEE (2013)
- Huang, C.H., Cagniart, C., Boyer, E., Ilic, S.: A bayesian approach to multi-view 4d modeling. International Journal of Computer Vision **116**(2), 115–135 (2016)

Chapter 6. We step back from the problem of discriminative data-model association and concentrate on generic temporal tracking method. Specifically, we argue that always sticking to the reference model in the same pose can be inaccurate and, for that reason, we propose a keyframe-based tracking strategy that changes the referenced pose of the template to improve the robustness.
Related work:

- Huang, C.H., Boyer, E., Navab, N., Ilic, S.: Human shape and pose tracking using keyframes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3446–3453. IEEE (2014)

Chapter 7. We conclude our work by presenting our findings, the limitations of the proposed methods and our directions for future work.

Appendix A. We present our comparison study on three different local-coordinate-frame approaches and conclude that the one based on signed distance field leads to best repeatability and it is therefore used in Chapter 3.
Related work:

- Huang, C.H., Tombari, F., Navab, N.: Repeatable local coordinate frames for 3d human motion tracking: From rigid to non-rigid. In: 3D Vision (3DV), 2015 International Conference on, pp. 371–379. IEEE (2015)
- Petrelli, A., Di Stefano, L.: On the repeatability of the local reference frame for partial shape matching. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 2244–2251. IEEE (2011)

Appendix B. We investigate an alternative output label space to learn the correspondences, where data are more naturally aligned. The key advantage is that one can learn across multi-view datasets despite the distinct mesh connectivity.

Appendix C. In order to bridge the gap between 2D and 3D, we take one step backward and study the common ground among the state-of-the-art methods in 2D parameter estimation, *e.g*. homography and fundamental matrix. We investigate whether these techniques are used in 3D registrations and try to provide a unified view on these problems.

- Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)

2 Background

The theoretical background of the thesis stems from *Iterative Closest Point* (ICP) algorithm [17, 38, 158], or more specifically, its non-rigid and probabilistic variant [34, 35], whose principles are briefly reviewed in this chapter for the sake of completeness. We first define two different shape parameterizations in 3D and elaborate the ICP procedure using one of them. Meanwhile, as the readers shall see later in Chapter 3 and 4, we apply a well known machine learning paradigm *Random Forests* [45] to draw discriminative predictions. Thus, the theory of forests are also described here. Despite the rich literature in both topics, we only outline each method briefly, provide the definition and lay the needed foundations for this thesis. Table 2.1 and Table 2.2 at the end of this chapter list a quick reference to the notations and the terminology we use.

2.1 3D Shape Parameterizations

A shape in 3D Euclidean space defines a continuous volumetric domain $\Omega \subset \mathbb{R}^3$, whose border, $\partial\Omega$, defines a 2-manifold surface. As shown in Fig. 2.1 on the right, the shape parameterization $\mathcal{M}$ depends on where the discretization takes place, on $\partial\Omega$ or on Ω. In both cases, it can be parameterized explicitly as mesh representations, which are pairs of point locations $\mathbf{M}$ and their connectivities $\mathcal{T}$, $\mathcal{M} = (\mathbf{M}, \mathcal{T})$. Throughout this thesis, we refer to a point on surfaces as a *vertex*, denoted as $v \in \mathcal{V}$, where $\mathcal{V}$ is the set of vertex indices with triangular connectivities. A volumetric sample point, on the other hand, is referred to as a *cell* $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of cell indices drawn by centroidal Voronoi tessellation (CVT) [50]. Without the lost of generality, in the following section we choose

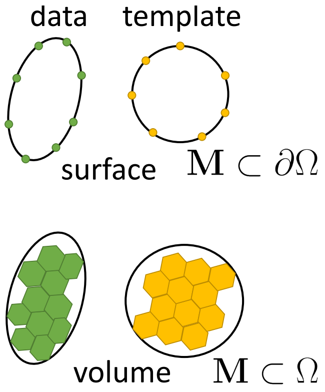

data
template
surface
$\mathbf{M} \subset \partial\Omega$
volume
$\mathbf{M} \subset \Omega$

Figure 2.1: **Two major 3D shape representations**.

surface representations to elaborate the principles of deformable ICP. Readers are referred to [5] for an example of volumetric deformable ICP applied in human shape tracking.

2.2 Surface-based Deformable Probabilistic ICP

A 3D reference surface is represented as $\mathcal{X} = (\mathbf{X}, \mathcal{T}_X)$, where $\mathbf{X} = \{\mathbf{x}_v\}_{v=1}^{V} \subset \mathbb{R}^3$ are the locations of vertices v, and $\mathcal{T}_X$ defines the triangles. The essential task of top-down human motion tracking is to register a template $\mathcal{X}$ (source) to the observed data $\mathcal{Y}$ (target), such as fitting the mesh in Fig. 2.2(a) to the observations in Fig. 2.2(d). Evolving $\mathcal{X}$ amounts to parameterizing $\mathbf{X}$ as a function of deformation parameters $\boldsymbol{\Theta}$. As $\boldsymbol{\Theta}$ changes over time, so are the vertex positions, *i.e.* $\mathbf{X}^t = \mathbf{X}(\boldsymbol{\Theta}^t)$. $\mathbf{X}^0$ refers to template vertex positions in the rest pose. This also implies that the triangles $\mathcal{T}_X$ do not change over time, which is often regarded as *no topological changes* in the literature and is the assumption that we follow throughout the thesis.

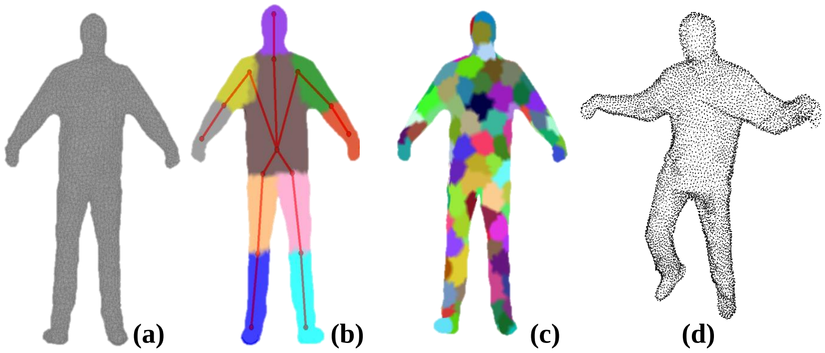


Figure 2.2: **Distinct mesh representations**. (a) is a watertight reference mesh with V vertices. (b-c) visualize different deformation structures. (b) is the standard skeleton-based animation [43, 62, 148], while in (c), the mesh is decomposed into K patches [34]. (d) is the observed point cloud with O points.

2.2.1 Patch-based Mesh Deformation Framework

The choice of deformation parameter $\boldsymbol{\Theta}$ varies from application to application. It can be a simple global rigid-body motion as in the original ICP papers [17, 37, 158] or the raw vertex positions (namely, $\mathbf{X}$ itself) as in Laplacian surface editing [125]. To characterize the intrinsic nature of human motions, many authors [62, 98, 140] choose an articulated graph structure that resembles the human *skeleton*, as visualized in Fig. 2.2(b). Unlike these approaches, our interest lies in recovering not only skeletal poses but also details of the shape. We opt for a surface deformation framework [34] that groups vertices into K patches (Fig. 2.2(c)) and assign each of them a rigid body motion $\boldsymbol{\theta}$ with 6 degree

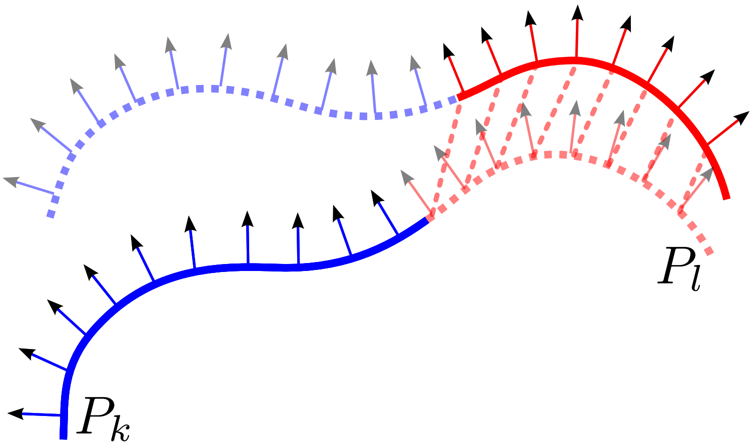


Figure 2.3: **Illustrations of patch-based deformation model [34].** P_k and P_l have their own rigid transform and predict positions for their own vertices (solid curves) and the vertices of the neighboring patches (dotted curves). The rigidity energy penalizes the discrepancies in these predictions (dotted lines).

of freedom (dof). Thus, $\boldsymbol{\Theta}$ is the collection of all $\boldsymbol{\theta}$, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_k\}_{k=1}^{K}$, encoding compactly the global *shape pose*[1] of the surface in a $6K$-dimensional space.

More formally, a rigid transformation w.r.t. the world coordinates is associated with each patch P_k. It is parameterized by the position of the patch center $\mathbf{c}_k$ and a rotation matrix $\mathbf{R}_k \in SO(3)$ (or equivalently by a unit-length quaternion $\mathbf{q}_k$). This rigid transform yields for every vertex of the mesh a predicted position $\mathbf{x}_k$ (dotted lines in Fig. 2.3):

$$\mathbf{x}_k = \mathbf{R}_k(\mathbf{x}^0 - \mathbf{c}_k^0) + \mathbf{c}_k, \tag{2.1}$$

where the superscript 0 denotes the corresponding variable in the reference pose. The mesh is deformed by linearly blending the predictions made by different neighboring patches for each vertex. The weighting functions α_k are Gaussians of the Euclidean distance to the center of the patch $\mathbf{c}_k$. As for the support for blending, we consider only the patch P_k itself and its direct neighbors, *i.e.* $k \cup \mathcal{N}_k$:

$$\mathbf{x} = \sum_{s \in k \cup \mathcal{N}_k} \alpha_s \mathbf{x}_s. \tag{2.2}$$

Blending weights α_s are normalized to add up to 1 and is computed only once on the template before tracking starts. The deformation parameter $\boldsymbol{\Theta}$ is thus $\{(\mathbf{R}_k, \mathbf{c}_k)\}_{k=1}^{K}$, where K is the total number of patches.

Ideally, patching should follow the intrinsic nature of the shape, *e.g.* its rigid parts, which can be learned using the approach like [56]. However, in the absence of prior knowledge on this structure, they are preferably regularly

[1] With a slight abuse of terminology, we sometimes abbreviate shape poses to only ‘shapes’ and the term ‘poses’ refers particularly to human skeletal poses. See Table 2.2 for comparison.

distributed over the surface as those in Fig. 2.2(c), with a parameter that controls the maximum radius of the patch and consequently the total number of patches K. Empirically we found that for human motion, $K \approx 150$ yields fairly good results and the resulting dimension of the search space, $150 \times 6 \approx 10^3$, lies in between skeletal-based animation (typically 10^2) and raw vertex locations (greater than 10^4). Note that multi-resolution patching is always possible.

2.2.2 Rigidity constraints

Given this patch-based control structure, we define an energy that penalizes non-rigid surface deformations w.r.t. its original status. This is inspired by the theory of elasticity in which deformable objects are defined by their material properties and a rest configuration. Many works simulate this behavior by constraining the displacement field [59, 95, 125, 124]. They operate on raw vertex positions and discourage the displacement fields from changing after local transformations, which are either expressed as a linear function of vertex positions [59, 95, 125], or iteratively estimated [124]. On the other hand, [22] and [135] optimize local transformations rather than vertex positions, and define elastic constraints between the transformed vertices themselves. Although this is not as physically accurate as the proper computation of the strain energy, we pursue this path because it naturally integrates to our patch-based representation. Our rigidity energy, as defined in Eq. (2.3) and shown in Fig. 2.3, simply enforces the predictions $\mathbf{x}_k(v)$ and $\mathbf{x}_l(v)$ of a vertex v by two neighboring patches, k and $l \in \mathcal{N}_k$, to be consistent:

$$E_r(\mathbf{\Theta}) = \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_k} \sum_{v \in P_k \cup P_l} w_{kl}(v) \left\| \mathbf{x}_k(v) - \mathbf{x}_l(v) \right\|^2 . \tag{2.3}$$

The choice of the weights $w_{kl}(v)$ is of importance as it allows to encode material properties. In all of our experiments, they are proportional to the sum of the blending weights: $\alpha_k(v) + \alpha_l(v)$ and is normalized such that all the $w_{kl}(v)$ depending on the same vertex v sum up to 1, simulating therefore uniform stiffness. Both Eq. 2.3 and [124] aim to minimize the non-rigidness to achieve as-rigid-as-possible property. The former defines it in absolute vertex positions while the latter in the displacement field.

Optimization

Evolving a mesh can then be viewed as an optimization problem balancing the rigidity energy E_r (with a weight λ_r) and data terms E_{data}:

$$\underset{\mathbf{\Theta}}{\operatorname{argmin}}\ \lambda_r E_r(\mathbf{\Theta}) + E_{data}\left(\mathbf{X}(\mathbf{\Theta})\right), \tag{2.4}$$

where E_{data} can be manually specified constraints or more sophisticated probabilistic likelihoods [35]. Eq. 2.4 is a non-linear least-squares problem, since $\mathbf{\Theta}$ involves rotations. We employ an iterative Gauss-Newton method [40] to find the minimizer. Instead of directly using elements in the rotation matrix as parameters with additional soft constrains for matrix-orthogonality, as in [135], we optimize the energy function w.r.t. small affine updates, $\boldsymbol{\theta}_k = [\mathbf{u}_k, \mathbf{v}_k] \in \mathbb{R}^6$. Specifically, the update in rotation, $\hat{\mathbf{R}}_k$, is approximated by the first-order expansion of the exponential mapping of $[\mathbf{u}_k]_\times$, namely, $\mathbf{I} + [\mathbf{u}_k]_\times$, and $\mathbf{v}_k$ is the displacement of $\mathbf{c}_k$. As shown in Eq. 2.5, this formulation allows to write the update of coordinates $\mathbf{x}_k$ linearly in $\boldsymbol{\theta}_k$, and thus of $\mathbf{x}$ linearly in $\{\boldsymbol{\theta}_k\}_{k=1}^K$.

$$\begin{aligned}\mathbf{x}_k \mapsto \mathbf{x}'_k &= \hat{\mathbf{R}}_k(\mathbf{x}_k - \mathbf{c}_k) + \mathbf{c}'_k \\ &= \mathbf{x}_k + [\mathbf{u}_k]_\times(\mathbf{x}_k - \mathbf{c}_k) + \mathbf{v}_k = \mathbf{x}_k + \mathbf{K}_k(\mathbf{x}_k)\,\boldsymbol{\theta}_k, \\ &\text{with } \mathbf{K}_k(\mathbf{x}_k) = \left[[\mathbf{c}_k - \mathbf{x}_k]_\times \quad \mathbf{I}\right]. \end{aligned} \tag{2.5}$$

The first order approximation of Eq. 2.3 then yields a simple quadratic form in the update parameters.

$$E_r(\boldsymbol{\Theta}) \simeq \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_k} \sum_{v \in P_k \cup P_l} w_{kl} \parallel (\mathbf{x}_k + \mathbf{K}_k(\mathbf{x}_k)\,\boldsymbol{\theta}_k) - (\mathbf{x}_l + \mathbf{K}_l(\mathbf{x}_l)\,\boldsymbol{\theta}_l) \parallel^2. \tag{2.6}$$

Knowing that $\mathbf{K}_k(\mathbf{x}_k)$ is actually the Jacobian of $\mathbf{x}_k$ w.r.t. $\boldsymbol{\theta}_k$, the gradient of Eq. 2.6 w.r.t. $\boldsymbol{\theta}_k$ can be expressed using the chain rule:

$$\left[\frac{\partial E_r}{\partial \boldsymbol{\theta}_k}\right] = \left[\frac{\partial E_r}{\partial \mathbf{x}_k}\right]\left[\frac{\partial \mathbf{x}_k}{\partial \boldsymbol{\theta}_k}\right] = \left[\frac{\partial E_r}{\partial \mathbf{x}_k}\right]\mathbf{K}_k(\mathbf{x}_k). \tag{2.7}$$

These first order approximations can be used to compute the gradient and the minimum of the quadratic approximation of the energy in the tangent space but the actual energy must be evaluated on the updated $\boldsymbol{\Theta}$. To recover the $\mathbf{R}_k$'s we actually perform the update of rotations in quaternion representation and normalize the result to limit the accumulation of numerical error. The cost function from Eq. 2.4 is in practice minimized by performing a line search in tangent space and making sure that the corresponding step taken on the parameter manifold actually decreases the energy.

Numerical Considerations

In the Gauss-Newton algorithm, the Hessian matrix is approximated by $\mathbf{G}^\top\mathbf{G}$ where $\mathbf{G}$ is the Jacobian matrix. In our case, $\mathbf{G}^\top\mathbf{G}$ is a $6K \times 6K$ sparse matrix, with only a few 6×6 non-zero blocks. This structure is fixed and reflect the connectivity in the graph of patches. Since Eq. 2.7 offers an analytic formulation, it is more practical to compute directly the matrix, rather than computing first $\mathbf{G}$ and then $\mathbf{G}^\top\mathbf{G}$. This computation can be mostly parallelized thanks to the sparse blocky structure. For example, the off-diagonal 6×6 blocks come from the rigidity terms of Eq. 2.6 and require for block (k, l) to accumulate gradient terms over $P_k \cup P_l$. These operations can be distributed on multiple processors as they access different parts of the memory. Finding a minimizer of the quadratic approximation at each step of the Gauss-Newton algorithm can then be tackled by any available sparse solver, either direct or iterative, as discussed in [20]. We use sparse Cholesky factorization [89] in `Ceres` [1].

2.2.3 Iterative Closest Point Search

So far we have introduced how the template should evolve according to the deformation parameter $\mathbf{\Theta}$. To take one step further, one would like to make it data-driven. Given an observed visual hull $\mathcal{Y} = (\mathbf{Y}, \mathcal{T}_Y)$, where $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{O} \subset \mathbb{R}^3$, the goal is to register $\mathcal{X}$ non-rigidly to $\mathcal{Y}$. In other words, one would like to estimate the optimal $\hat{\mathbf{\Theta}}$ such that $\mathbf{X}(\hat{\mathbf{\Theta}})$ resembles $\mathbf{Y}$ as much as possible. It requires a good data term E_{data} in Eq. 2.4 to account for automatic data explanations. It generally boils down to two sub-problems:

1. $Association$: matching each point in $\mathcal{Y}$ to the points in $\mathcal{X}$ to build the correspondence set $\mathcal{C}$.

2. $Tracking/Optimization$: estimating $\hat{\mathbf{\Theta}}$ by minimizing an energy E that describes the discrepancies between pairs in $\mathcal{C}$: $\hat{\mathbf{\Theta}} = \mathrm{argmin}_{\mathbf{\Theta}}\, E(\mathbf{\Theta}; \mathcal{C})$.

More formally, let $\mathcal{V}_Y$ and $\mathcal{V}_X$ denote respectively the set of point index in data and the template, *i.e.* $i \in \mathcal{V}_Y$ and $v \in \mathcal{V}_X$. The association step aims at finding the matched pairs (i, v) between the $\mathcal{V}_Y$ and $\mathcal{V}_X$: $\mathcal{C} = \{(i, v)\} \subset \mathcal{V}_Y \times \mathcal{V}_X$. Relaxing the hard correspondences to soft assignments extends the algorithm to its probabilistic counterpart.

Matching two point sets accurately and reliably in the presence of noise is still an on-going challenge. If $\mathbf{Y}$ and $\mathbf{X}(\mathbf{\Theta})$ lie in vicinity, correspondences $\mathcal{C}$ can be built directly by local proximity search. The deformation parameter $\mathbf{\Theta}$ is initialized with such coarse associations and $\mathbf{X}$ is evolved accordingly. Given the new $\mathbf{X}(\mathbf{\Theta})$, correspondences $\mathcal{C}$ are refined by a new nearest neighbor search. One thus keeps refining $\mathcal{C}$ and $\mathbf{\Theta}$ iteratively till convergence. Applying simple proximity search for $\mathcal{C}$ and alternating between the association and optimization steps is actually the trait of ICP. In the context of human motion tracking, this strategy has been adopted in many existing approaches [35, 43, 62]. In particular, [35] explains the association and optimization steps together in an *Expectation and Maximization* (EM) fashion. To meet the requirement that $\mathbf{Y}$ and $\mathbf{X}(\mathbf{\Theta})$ lie in vicinity, the nearest neighbor search is always performed between $\mathbf{Y}^t$ and $\mathbf{X}^{t-1} = \mathbf{X}(\mathbf{\Theta}^{t-1})$. Since the changes between successive frames are generally not too large, this usually leads to satisfactory results even when fast motions happen,

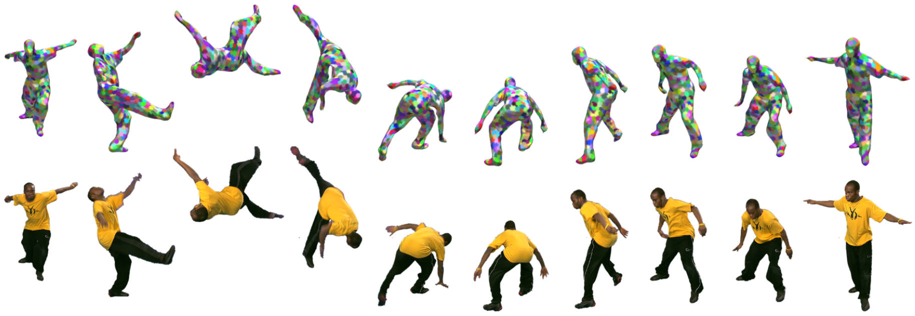


Figure 2.4: **Results of [35] on *Flashkick* sequence**. The kickflip itself consists of extremely fast motion as it spans over 10 frames.

as shown in Fig. 2.4. However, as the construction of correspondences $\mathcal{C}^t$ relies heavily on the previous result $\mathbf{X}^{t-1}$, the error in $\mathbf{\Theta}^{t-1}$ will propagate to $\mathcal{C}^t$ and in turn influence all the subsequent estimations. In the standard ICP procedure, there is no mechanism to recover from this vicious circle. Furthermore, ICP is also notorious for its slow convergence.

2.3 Random Forests

Given the aforementioned limitations, this thesis strives to provide an alternative for the association step. It does not rely on a proximity search on $\mathbf{X}^{t-1}$ to construct correspondence $\mathcal{C}^t$ as in ICP framework. Instead, it estimates $\mathcal{C}^t$ directly using random forests. For this reason, we also describe the basic principles of random forests and define the terminology that will be used in the later chapters.

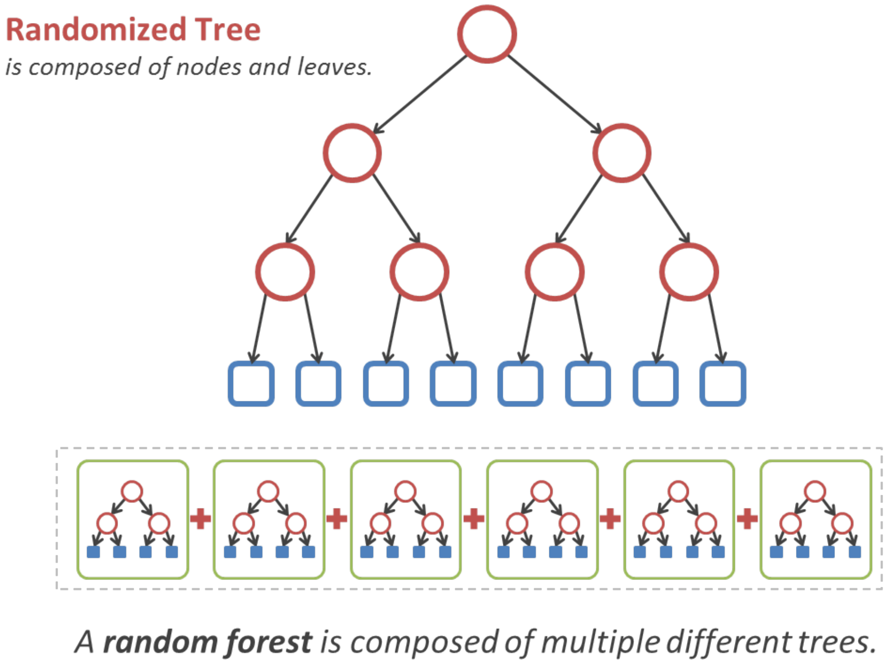

Randomized Tree
is composed of nodes and leaves.
A random forest is composed of multiple different trees.

Figure 2.5: **An illustration of random forests**.

As implied by the name, a forest consists of multiple trees. Decision trees have been around for a number of years [27]. Their recent revival is mostly due to the discovery that ensembles of slightly different trees tend to produce higher accuracy on previously unseen data, a property known as *generalization*. An ensemble of trees is called a forest. A *randomized* forest comprise trees that grown differently with some randomness injected, *e.g*. random bootstrap samples [27] , or randomized node optimization [68]. The former strategy actually leads to an alias of random forests called *tree bagging* .

A decision tree is a special type of graph. It is a data structure made of a collection of nodes and edges organized in a hierarchical fashion (Fig. 2.5). Nodes are divided into internal (or split) nodes and terminal (or leaf) nodes. We denote internal nodes with circles and terminal ones with squares. All nodes have exactly one incoming edge. In contrast to general graphs a tree cannot contain loops. In this thesis, we particularly concentrate on binary trees for its simplicity, where each internal node has exactly two outgoing edges.

The functioning of decision trees can be separated into an offline phase (training) and an online one (testing). Here we describe these two phases as well as the other components of forests used in these phases. We start with the concept of features, introduce the functionality of one single tree and finally discuss how to aggregate hypotheses from multiple trees. Since forests are deployed to draw predictions on 3D surfaces in this thesis, we hereinafter denote a data point as v, representing a vertex on a mesh. To balance between being general and specific, we take an intermediate approach that on one hand keeps definitions and explanations at an abstract level yet on the other hand elaborate only the concepts and techniques applied in the following chapters. For instance, we do not assume any particular type of feature at this point, keeping the descriptions generic. Yet, we will explain only one type of splitting functions at branch nodes considered in this thesis for its simplicity to train, which is axis-aligned thresholding. Readers are referred to [45] for more sophisticated techniques.

Features. A data point v is described as a high-dimensional vector in a feature space, $\mathbf{f}(v) = \big(f_1, \cdots, f_\kappa, \cdots, f_d\big) \in \mathcal{F}$ [2]. Each component κ represents some attributes of the data point v. The number of features naturally depends on the type of the data point as well as the application. In theory, the dimensionality of the feature space $\mathcal{F}$, d, can be very large, even infinite. In practice, it is often not possible, and further not necessary, to extract all d dimensions of $\mathbf{f}$ ahead of time. Instead, we extract only a small portion of d as needed basis. Based on this let us formulate the features of interest that are computed at any single time to be a subset selected from the set of all possible features. In most applications, d can be very large but the dimension of the subspace $\mathcal{F}^{d'} \subset \mathcal{F}$ is much smaller $d' \ll d$. It is therefore of crucial importance that the construction of feature vector $\mathbf{f}$ is dimension-wise, namely, the computation of each attribute f_κ is independent from each other. For this reason, many well-known histogram-based 3D descriptors that require normalization after construction, *e.g.* SHOT [142] and FPFH [118], can still be applied for random forests but do not take advantages of its virtue.

Online tree testing. The basic principle of tree testing is simple. The goal is to route the data point to traverse down the tree. Given a previously unseen data point $\mathbf{f}$, a decision tree hierarchically applies a number of predefined tests (see Fig. 2.6). Since each channel of $\mathbf{f}$ can be computed individually, this test is often chosen as simple axis-aligned thresholding [121, 137, 138]. The parameters of test functions, denoted as ϕ, are thus the selected feature attribute κ and the

[2] Whenever it is not crucial or it is clear from the context, we drop the dependency of v, or use it as a suffix $\mathbf{f}_v$, in order to keep notations uncluttered.

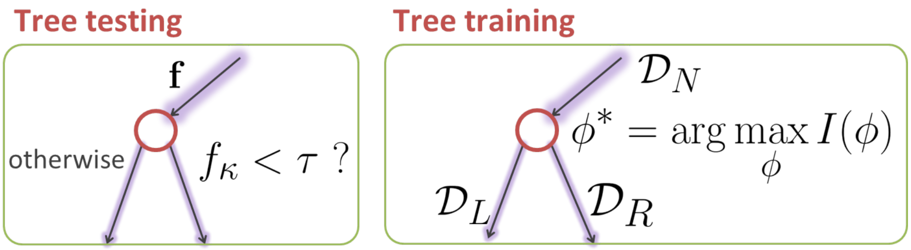


Figure 2.6: **Illustrations of the functionality in tree testing and training**. During testing a data point is routed through the tree by simple axis-aligned thresholding, whose parameters are learned during training.

applied thresholding value τ, *i.e.* $\phi = (\kappa, \tau)$, that have to be determined during training. Starting at the root, each split node applies its associated test function to $\mathbf{f}$. Depending on the result of this binary test the data is sent to the right or left child. This process is repeated recursively until the data point reaches a leaf node. The leaf nodes contain a predictor/estimator (*e.g.* a classifier or a regressor) which associates an output (*e.g.* a class label or a continuous value) with the input $\mathbf{f}$.

Offline tree training. Three things have to be determined during training:

- The weak learners (split functions) in each branch node.
- The tree structure.
- The statistics stored at each leaf node.

We follow this order to discuss each of them in below.

The split functions stored at the internal nodes are key for the functioning of the tree. One may think of designing these functions manually. However, this approach would only be possible for very simple problems. For more realistic problems the test functions need to be learned automatically, from exemplar data. Thus, the training phase takes care of selecting the type and parameters of the test function associated with each split node by optimizing a chosen objective function defined on an available training set. The optimization of the split functions proceeds in a greedy manner. At each node, depending on the subset of the incoming training set $\mathcal{D}_N$ we learn the function that "best" splits $\mathcal{D}_N$ into $\mathcal{D}_R$ and $\mathcal{D}_L$:

$$\mathcal{D}_L = \{v \in \mathcal{D}_N | f_\kappa(v) \geq \tau\} \tag{2.8a}$$

$$\mathcal{D}_R = \{v \in \mathcal{D}_N | f_\kappa(v) < \tau\}. \tag{2.8b}$$

This problem is formulated as the maximization of an objective function at that node. A commonly used objective function in supervised learning is *information gain* I :

$$\phi^* = \underset{\phi}{\operatorname{argmax}}\, I(\phi), \tag{2.9}$$

with

$$I(\phi) = H(\mathcal{D}_N) - \sum_{i \in \{L,R\}} \frac{|\mathcal{D}_i(\phi)|}{|\mathcal{D}_N|} H(\mathcal{D}_i(\phi)). \tag{2.10}$$

The *entropy* H can be viewed as a measure of uncertainty. The choice of H depends on the label space $\mathcal{L}$. For discrete probability distributions (classification tasks) one uses the Shannon entropy:

$$H(\mathcal{D}) = -\sum_{c} P(c) \log \left(P(c)\right). \tag{2.11}$$

The letter c indicates the class label and $P(c)$ indicates the empirical distribution extracted from the training points within the set $\mathcal{D}$. When it comes to continuous-valued labels and continuous distributions (regression tasks), the differential entropy is instead used:

$$H(\mathcal{D}) = -\int_{y} P(y) \log \left(P(y)\right) dy. \tag{2.12}$$

Here y is a continuous label of interest and p is the probability density function estimated from the training points in the set $\mathcal{D}$. It can be estimated either using parametric distributions or non-parametric methods such as Parsen density estimator as in [64]. One of the most popular parametric choice is to use Gaussian-based models to approximate the density $P(y)$ due to their simplicity. The differential entropy of a d-variate Gaussian is defined analytically as:

$$H(\mathcal{D}) = \frac{1}{2} \log \left((2\pi e)^d \left|\Lambda(\mathcal{D})\right| \right). \tag{2.13}$$

Eq. 2.13 is monotonically increasing with respect to the determinant of covariance matrix Λ of $\mathcal{D}$. When assuming isotropic Gaussian distributions, it is proportional to the variance of continuous labels, σ^2. Thus, many works [63, 71, 137], including this thesis, apply only the variance σ^2 as the measure of uncertainty to save some computation overhead.

Besides weak learners, one also need to optimize the tree structure (shape). Training starts at the root node, where the optimum split parameters ϕ are found. One then construct two child nodes, each receiving a different disjoint subset of the training set. This procedure is then applied to all the newly constructed nodes and the training phase continues. The structure of the tree depends on how and when we decide to stop growing various branches of the tree. Diverse stopping criteria can be applied. For example it is common to stop the tree when a maximum number of levels has been reached. Alternatively, one can impose a minimum value of the maximum $\max_{\phi} I$, in other words one stops when the seeked-for attributes of the training points within the leaf node are similar to one another. Tree growing may also be stopped when a node contains too few training points. Avoiding growing full trees has been demonstrated to have positive effects in terms of generalization. In this survey we avoid further post-hoc operations such as tree pruning [57, 114] to keep the training process as simple as possible. At the end of the training phase we obtain: (i) the (greedily) optimum weak learners associated with each node, (ii) a learned tree structure, and (iii) a different set of training points at each leaf.

Last but not least, during training we also need to learn how to make predictions. After training, each leaf node remains associated with a subset of (labeled) training data. During testing, a previously unseen point traverses the tree until it reaches a leaf. Since the split nodes act on features, the input test point is likely to end up in a leaf associated with training points which are all similar to itself. Thus, it is reasonable to assume that the associated label must also be similar to that of the training points in that leaf. This justifies using the label statistics gathered in that leaf to predict the label associated with the input test point. In the most general sense, the leaf statistics have to make summarizations on all the labels that arrive at this leaf, captured by the posterior distributions. One usually store empirical class distributions for classification tasks $P(c|\mathbf{f})$, while kernel density estimators, *e.g.* mean-shift [39, 42] are used for regression to approximate $P(y|\mathbf{f})$ as a set of confidence-weighted modes $\mathcal{H} = \{(\mathbf{h}, \omega)\}$, where $\mathbf{h}$ is the mode location in the label space $\mathcal{L}$ and ω is a scalar weight.

Prediction aggregation. A random decision forest is an ensemble of randomly trained decision trees. The key aspect of the forest model is the fact that its component trees are all randomly different from one another. This leads to decorrelation between the individual tree predictions and, in turn, results in improved generalization and robustness. All trees are trained independently (and possibly in parallel). During testing, each test point is simultaneously pushed through all trees (starting at the root) until it reaches the corresponding leaves. Tree testing can also often be done in parallel, thus achieving high computational efficiency on modern parallel CPU or GPU hardware. Combining all tree predictions into a single forest prediction may be done by a simple averaging operation. For instance, in classification:

$$P(c|\mathbf{f}) = \frac{1}{T}\sum_{\iota=1}^{T} p_{\iota}(c|\mathbf{f}), \tag{2.14}$$

where $P_{\iota}(c|\mathbf{f})$ denotes the posterior distribution obtained by the ιth tree. In regression, a popular choice is performing again mean-shift [64] on all the $P(y|\mathbf{f})$ from the reached leaves.

2.4 Data and metric of evaluations

As this thesis addresses the problem of recovering temporal human shapes and poses, we consider sequences recorded from multiple-view studios, *e.g.* Fig 2.7, for evaluation purposes. The profiles of these sequences are largely summarized in Appendix D. More detailed information will be provided whenever we choose the subsets of them in the experiments of each following chapter.

Depending on the provided ground truths, four different following error metrics are considered. The first two measure the goodness of surface deformations while the latter two validate human skeletal poses:

1. **2D silhouette overlap error**: this is the most widely-used error metric in the community. Given the estimated surfaces projected back to image domains, it measures the discrepancies between the rendered silhouettes and the observed ones by simple XOR operations. Despite the popularity, note that silhouette images are nevertheless the results of background-subtraction algorithms, which are not perfect. The magnitude of overlap error hence does not always reflect the quality of 3D shape estimation.

2. **3D surface registration error**: currently the temporal evolutions of infrared markers can be tracked efficiently with high fidelity using motion capture systems, such as `Vicon-Peak`[3]. When subjects wear suits with markers sparsely attached, this error metric measures how much is the estimated surface deviated from these markers. It is a very faithful error metric but the tight-suit requirement limits its applicability.

3. **3D error of human poses**: if the infrared markers are attached on some semantic-meaningful locations, the 3D positions of joints can be calculated and considered as ground truths. This metric therefore measures the discrepancies between the recorded joint positions and the estimated ones.

4. **2D error of human poses**: this metric measures also the pose errors, with the ground truths manually annotated in 2D images.

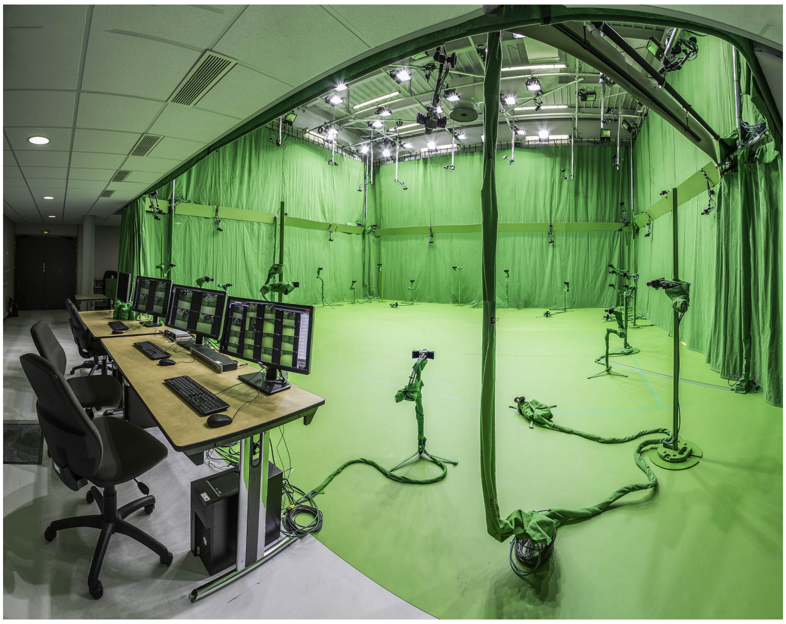


Figure 2.7: **Multi-camera studio** Kinovis. ©INRIA Grenoble Rhône-Alpes

[3] `http://www.vicon.com/`

Data notations	Descriptions
Ω	A continuous volumetric domain defined by a shape in 3D. A surface is expressed as the border of Ω, *i.e.* $\partial\Omega$.
$\mathcal{S}$	The index set for points (cells) sampled from a domain Ω. $\mathcal{S} = \{1, \cdots, s, \cdots, \lvert\mathcal{S}\rvert\}$.
$\mathcal{V}$	The index set for points (vertices) sampled from a surface $\partial\Omega$. $\mathcal{V} = \{1, \cdots, v, \cdots, \lvert\mathcal{V}\rvert\}$.
$\mathcal{M}$	A mesh/CVT is a pair of vertex/cell positions $\mathbf{M} \subset \mathbb{R}^3$ and their connectivity $\mathcal{T}$, *i.e.* $\mathcal{M} = (\mathbf{M}, \mathcal{T})$.
$(\cdot)_X, (\cdot)_Y$	Subscripts representing entities of templates $\mathcal{X}$ and observations $\mathcal{Y}$ respectively.
$\mathcal{X}$	The reference entity, or template. $\mathcal{X} = (\mathbf{X}, \mathcal{T}_X)$, where $\mathbf{X} = \{\mathbf{x}_v\}_{v \in \mathcal{V}_X}$ if it is a surface; $\mathbf{X} = \{\mathbf{x}_s\}_{s \in \mathcal{S}_X}$ if it is a volume.
$\mathcal{Y}$	Observations, input data. $\mathcal{Y} = (\mathbf{Y}, \mathcal{T}_Y)$, where $\mathbf{Y} = \{\mathbf{y}_i\}_{i \in \mathcal{V}_Y}$ if it is a surface; $\mathbf{Y} = \{\mathbf{y}_i\}_{i \in \mathcal{S}_Y}$ if it is a volume.
V	The number of vertices, *i.e.* $V = \lvert\mathcal{V}_X\rvert$.
O	The number of observed points, *i.e.* $O = \lvert\mathcal{V}_Y\rvert$.
$\mathcal{N}$	Neighbors of graph structure, *e.g.* $\mathcal{N}_i$ denotes the set of 1-ring neighborhood for input point i.
$\mathcal{C}$	The set of correspondences (i, p) associating each input point i to point p on the template. p is either a surface vertex v or a CVT cell s.
$\boldsymbol{\Theta}$	A set of rigid body motions $\boldsymbol{\theta}$ of each deformation primitives, *e.g.* patches of vertices, clusters of cells.
K	The number of deformation primitives, *e.g.* the number of patches for surfaces, the number of clusters for CVT cells.
$\mathbf{T}$	Same as $\boldsymbol{\Theta}$ but expressed in matrices, *i.e.* $\mathbf{T}_k \in SE(3)$.
$\mathbf{J}$	The poses of human skeletons, $\mathbf{J} = \{\mathbf{x}_j\}_{j=1:J} \subset \mathbb{R}^3$, where J is the number of joints.
$\mathcal{F}$	A feature space where data points are described.
$\mathbf{f}$	A vector describing a data point in a feature space. $\mathbf{f} = \big(f_1, \cdots, f_\kappa, \cdots, f_d\big) \in \mathcal{F}$.
$\mathcal{L}$	The label space of forests.
I	Information gain.
H	Entropy.
τ	Axis-aligned thresholding values.
ϕ	Parameters of testing functions, or called split candidates.
T	The number of trees for each forest.

Table 2.1: **Notations**.

Our primary term	Other terms in use
surface	visual hull (refers only to observations)
correspondence	association
reference	template
shape	shape poses, surface deformations, encoded by $\mathbf{\Theta}$.
pose	skeletal poses, encoded by $\mathbf{J}$.
random forest	decision trees, randomized forest
split function	weak learner, testing function
dimension	channel, attribute, component

Table 2.2: **Common terms used interchangeably in this thesis**.

Part II

Discriminative Associations

3 Mesh-based Discriminative Associations

The first step towards 3D tracking-by-detection of human shapes is discriminative data-model associations. In this chapter, we address the problem of estimating frame-wise discriminative correspondences. We aim to associate each vertex of input visual hulls to the vertex of template.

This is essentially a 3D shape matching problem and has long been studied in graphic community [28, 88]. Our problem is nevertheless, more challenging because significant topological noise is usually presented in visual hulls. We adopt a machine learning approach to learn from previously observed deformed meshes. Following the notations in Sect. 2.3, a number of questions need to be answered to fully characterize a machine-learning approach. 1) Which learning technique to use? 2) What is the feature? Or say, how to express data points v as vectors $\mathbf{f}$ in the feature space $\mathcal{F}$? 3) Last but not least, what is the output label space $\mathcal{L}$? We choose an ensemble learning method, random forest [45], as our learning algorithm. We follow the *Vitruvian* framework [112, 140] that considers a continuous 2-manifold surface domain $\partial\Omega$ defined by the template as a output space. The forest therefore performs a regression task. A different label space will be investigated in Appendix B. During training, a mapping that returns every vertices on the deformed meshes back to their original locations on the template is learned. During prediction, points on input visual hulls are thereby mapped to the locations of potential matches. In the following sections of this chapter, we detail our algorithm and the principles of regression forests.

At first, the literature is reviewed and the related work on 3D human shape tracking is presented. We discuss about detection-based (*i.e.* discriminative) and temporal-tracking-based (*i.e.* generative) approaches. Then we present our algorithm which is robust to outlying geometries or noise, as demonstrated in the evaluation part. We further compare our discriminative model with related work on publicly available datasets. At the last part of the chapter, we summarize the advantages and disadvantages of our model and discuss aspects that can be further addressed.

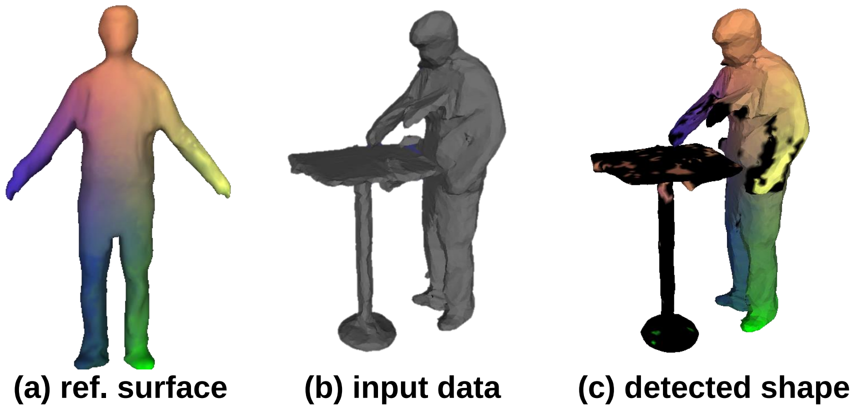


Figure 3.1: **Discriminative associations**. Given a reference surface (a), our method discovers reliable data-model correspondences by random forests, color-coded in (c). Unlike temporal tracking, this strategy detects user-specific shapes in a frame-wise manner, preventing erroneous associations from accumulations.

3.1 Introduction

Visual shape tracking is the process of recovering temporal evolutions of a template shape using visual information, such as image silhouettes or 3D points. It finds applications in several domains including computer vision, graphics and medical imaging. In particular, it has recently demonstrated a good success in marker-less human motion capture (mocap). Numerous approaches assume a user-specific reference surface, and the objective is to recover the skeletal poses [148], surface shapes [35], or both simultaneously [73].

Most of these model-based methods [35, 62, 98, 73, 148] can be viewed as extensions of Iterative-Closest-Point (ICP) framework [17, 38, 158], which attempts to explain newly observed data using the previous outcomes. As long as the initialization is close to the optimum solution, it is able to produce outstanding results. However, they also suffer from inherent weaknesses of *generative* strategies, *e.g.* the slow convergence. Moreover, when large deformations or outliers occur, discovering associations between data and models is particularly difficult. Unreliable associations result in ambiguous situations that yield erroneous numerical solutions and, consequently, break the tracking process.

In contrast, *discriminative* approaches that ‘detect’ rather than track models have shown better robustness over the past decade, for instance, in human pose estimation with Kinect [121, 140]. These approaches operate frame-independently, and are generally drift free. In this chapter, we explore this direction in order to get robust observation-model associations, regardless of the results from previous frames. We further present a discriminative ‘tracking-by-detection’ human mocap framework, as in Fig. 3.1. Inspired by Taylor *et al.* [140], we make use of regression forests to improve the associations. Shape geometries are characterized by volumetric representations, and are fed into user-specific forests to predict correspondences in *one shot*. Contrary to generative methods, this prediction does not require close initializations from a nearby frame. In addition, it allows a single model to be used as a reference surface for several different sequences, again even if large deformations or outliers exist. We combine this strategy with a generative tracking approach that takes our one-shot associations as input. Experiments demonstrate that this *hybrid* discriminative-generative framework leads to better, or comparable results than purely generative approaches, *e.g.* [35, 43], reducing error accumulations and hence increasing the stability.

The contribution of this chapter is a one-shot correspondence inference with complete 3D input (rather than 2.5D as in [121, 140]) that unlike other ICP-like methods [35, 62, 98, 73]), relies little on former results.

3.2 Related work

Among the vast literature on human pose estimation [102], we focus on top-down approaches that assume a 3D model and deform it according to input data, either directly with pixels as in [62, 98, 132], or with 3D points as in [35, 43, 73]. These methods typically decompose into two main steps: (i) data association, where observations are associated to the model, and (ii) deformation estimation, where deformation parameters are estimated given the associations. Our primary objective in this chapter is to improve the first part. Existing approaches for this issue are discussed below.

Generative approaches. Methods in this category do not require any training. They follow the association strategy in ICP while extending the motion model to more general deformations than the one in the original method [17]. Associations are addressed by searching for closest points, with various distance measures such as point-to-point [35], point-to-plane [37], or Mahalanobis distances [132]. This strategy heavily relies on the fact that observations in consecutive frames are in vicinity. Klaudiny *et al.* [84] generalize the idea from the previous frame to a certain frame in the considered sequences, finding the best non-sequential order to track, but the proximity assumption remains. Gall *et al.* [62] establish 3D-2D correspondences by considering both texture in images, and contours in silhouettes. Later, Liu *et al.* [98] include image segmentation information in order to differentiate multiple interacting subjects. These approaches implicitly assume that observations only describe the tracking subjects, which does not necessarily hold in 3D data that often contain fake geometries. Cagniart *et al.* [35] introduce an additional outlier class to reject associations with noisy observations. Data is explained by Gaussian Mixture Models (GMM) [18] in an Expectation-Maximization (EM) manner [18]. Huang *et al.* [73] follow a similar concept, but aggregate the outlier likelihood over every Gaussian component and offer better robustness. All these generative methods are highly likely to fail in large deformations. Furthermore, they are prone to error accumulations and, as a result of matching several successive frames wrongly (whether sequentially or not), they are prone to drift.

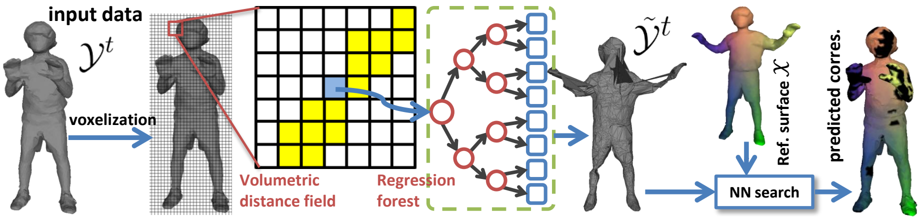


Figure 3.2: **Pipeline of prediction**. Correspondences are visualized in the same color. Black means no correspondence found for that data point.

Discriminative approaches. Recently, discriminative approaches have demonstrated their strengths in tracking human poses with depth images [10, 140]. Taylor *et al.* [140] propose a single-frame, or so called one-shot strategy, which yields decent dense correspondences without iterative refinements. With the help of regression forests, they map each foreground pixel to a weighted point in 3D, and thereby search the closest point within a predefined surface. Later, Pons-Moll *et al.* [112] train forests with a new objective in metric space, and couple the one-shot strategy with ICP. In the case of full 3D, Kanaujia *et al.* [81] use shape context histograms as descriptors, segment visual hulls into body parts with a pre-trained Support Vector Machine (SVM) [44], and build the skeletons with the method similar to [121]. Starka *et al.* [127] formulate the matching problem as the inference of Markov random field (MRF). Rodola *et al.* [117] apply forests to learn the parameters of wave kernel signatures [9] during training, and facilitate dense matching between two meshes.

To avoid computation overhead, we develop volumetric normal fields to describe meshes in a discretized volume Ω_3, and extend the comparison features from 2.5D [121, 140] to full 3D data. Our method can be viewed as a tracking-by-detection approach for human shape tracking.

3.3 Method Overview

In order to build the correspondences (step 1 in Chapter 2), standard ICP-based approaches [35, 43, 62] consider neighbors in vicinity and alternate between the association and optimization steps, refining $\mathcal{C}^t$ and $\mathbf{\Theta}^t$ iteratively. This generative strategy requires $\mathbf{X}^{t-1}$ to be sufficiently close to $\mathbf{Y}^t$ and is usually slow to converge. Furthermore, when the error occurs inevitably in either $\mathcal{C}^t$ or $\mathbf{\Theta}^t$, it influences all subsequent procedures, gets accumulated and eventually breaks down tracking.

We develop a different strategy that *warps* the input data $\mathcal{Y}$ to the reference mesh $\mathcal{X}$, denoted as $\tilde{\mathcal{Y}} = (\tilde{\mathbf{Y}}, \mathcal{T}_Y)$ and visualized as a triangular mesh in Fig. 3.2. If the warping is perfect, this mesh will look clean and resemble $\mathcal{X}$ as much as possible. Incorrect mapping, on the other hand, can be told from huge edges. Vertex positions $\tilde{\mathbf{Y}}$ represent the locations of potential matches between $\mathcal{Y}$ and $\mathcal{X}$. Thus, $\mathcal{C}$ can be built directly by doing nearest neighbor search between $\tilde{\mathbf{Y}}$ and $\mathbf{X}$ *just once*, as illustrated in Fig. 3.2.

Specifically, we consider this $\mathbb{R}^3 \to \mathbb{R}^3$ mapping as a composite one: $\mathbb{R}^3 \to$

$\mathbb{N}^3 \to \mathbb{R}^3$. The former mapping is voxelization (Sect. 3.4.1), while the latter is regression (Sect. 3.5). A forest is trained with many voxelized meshes off-line (Sect. 3.5.1). During runtime, $\mathbf{y}_i$ is first mapped to a voxel $\mathbf{v}_i$, and then regressed to a 3D point $\tilde{\mathbf{y}_i} \in \tilde{\mathbf{Y}}$, where it is matched to a vertex $\mathbf{x}_v$ by nearest neighbor search. Fig. 3.2 illustrates the whole matching pipeline.

3.4 Implicit surfaces and voxel-based features

We cast each mesh $\mathcal{M}$ into a volumetric scalar field $D : \mathbb{N}^3 \subset \mathbb{R}^3 \to \mathbb{R}$, and design features to describe surface geometries in volumes.

3.4.1 Truncated Signed Distance Transform (TSDT)

Voxelizing a mesh in general consists of two parts: (1) finding voxel positions for every vertex and (2) testing the overlap between triangles and voxels. The first part can be viewed as a quantization mapping from Euclidean space to a discretized space $\mathbf{v} : \mathbb{R}^3 \to \mathbb{N}^3$:

$$\mathbf{v}(\mathbf{T}_{\mathcal{M}}(\mathbf{x})) = \left\lfloor \begin{bmatrix} \mathbf{I} & -\boldsymbol{\mu} \\ \mathbf{0}^\top & s \end{bmatrix} \begin{bmatrix} \mathbf{T}_{\mathcal{M}}(\mathbf{x}) \\ 1 \end{bmatrix} \right\rfloor, \tag{3.1}$$

where s is the voxel size and $\boldsymbol{\mu}$ is the volume origin in $\mathbb{R}^3$. $\mathbf{T}_{\mathcal{M}} \in SE(3)$ represents the rigid transformation from world coordinate to PCA-like coordinate of $\mathcal{M}$; see the paragraph **Subject coordinate frame** below. The size of the volume is large enough to include all possible pose variations, and its center is aligned with the mean of the surfaces. The voxel size is chosen to be close to the average edge length of meshes, so that a single voxel is not mapped by too many vertices. To check the intersection of triangles with voxels, we apply *separating axis theorem* which is known to be efficient for collision detection in graphic community [3].

After determining voxels occupied by the surface, referred to as $\mathbf{v}_{\text{suf}}$, we further identify voxels located inside and outside the surface, denoted respectively as $\mathbf{v}_{\text{in}}$ and $\mathbf{v}_{\text{out}}$. Together they define a truncated signed distance transform:

$$D(\mathbf{v}) = \begin{cases} +\epsilon & \text{if } \mathbf{v}_{\text{out}} \text{ and } d(\mathbf{v}, \mathcal{M}) > \epsilon. \\ +d(\mathbf{v}, \mathcal{M}) & \text{if } \mathbf{v}_{\text{out}} \text{ and } d(\mathbf{v}, \mathcal{M}) \leq \epsilon. \\ 0 & \text{if } \mathbf{v}_{\text{suf}} \\ -d(\mathbf{v}, \mathcal{M}) & \text{if } \mathbf{v}_{\text{in}} \text{ and } d(\mathbf{v}, \mathcal{M}) \leq \epsilon. \\ -\epsilon & \text{if } \mathbf{v}_{\text{in}} \text{ and } d(\mathbf{v}, \mathcal{M}) > \epsilon. \end{cases} \tag{3.2}$$

$d(\mathbf{v}, \mathcal{M})$ denotes the shortest Euclidean distance from the voxel center to the mesh, which can be computed efficiently via AABB trees using CGAL library [141]. If the distance is bigger than a threshold ϵ, we store only $\pm\epsilon$ to indicate the inside/outside information. It is empirically set to be three times the physical length of diagonal of voxels. In the earlier version of this work [71], we store averaged surface normals at each $\mathbf{v}_{\text{suf}}$. However, such representations yield high memory footprint and thereby limit the amount of training meshes we

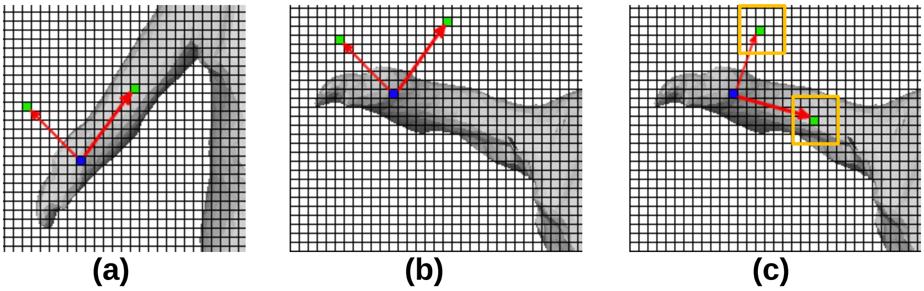


Figure 3.3: **The intuition behind adjusting offset vectors**. (a) original offset pair ψ. (b) $\eta = 0$ results in ψ without re-orientation, *i.e.* $\mathbf{R} = \mathbf{I}$. (c) $\eta = 1$. ψ is re-orientated by a rotation matrix $\mathbf{R} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$ characterized by a LCF.

can incorporate later in Sect. 3.5.1. The TSDT representation naturally encodes the spatial occupancies of a mesh and the required memory footprint is only one-third of the former (each voxel now stores just a scalar rather than a vector). It shares a similar spirit with implicit surface representations, *e.g.* level-set, and has been widely employed in RGBD-based tracking or reconstruction [83, 105].

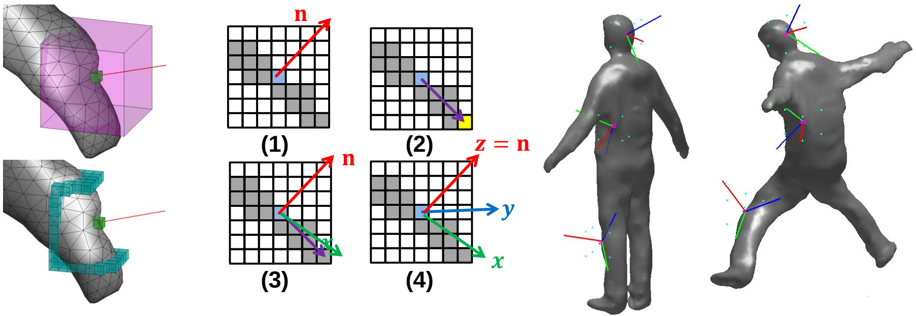
3.4.2 Voxel-based features

Next, we present the features $\mathbf{f}$ for describing the above TSDT, which are later used to train the forests. Since we are interested in predicting correspondences for vertices not triangles, from now on we concentrate only those surface voxels $\mathbf{v}_{\text{suf}}$ that are occupied by mesh vertices v, denoted as $\mathbf{v}_v$.

As mentioned previously in Sect. 2.3 and we shall see later in Sect. 3.5.1, conventionally, only one feature dimension κ is selected to separate data at each branch node of the tree (axis-aligned thresholding) [45]. A great advantage of forests is that features can be computed *on the fly*, *i.e.* while traversing trees. One does not have to prepare the whole high-dimensional feature vector for predictions, because only a few dimensions are needed. To make use of such property, the calculation of $\mathbf{f}$ is suggested to be dimensionally independent. We therefore avoid descriptors that requires normalization, like MeshHOG [157], FPFH [118], or SHOT [142], and resort to comparison features used in [46, 121].

As depicted in Fig. 3.3, for each surface voxel $\mathbf{v}_v$ (blue), we shoot two offsets (red vectors) $\psi = (\mathbf{o}_1, \mathbf{o}_2) \in \mathbb{N}^3 \times \mathbb{N}^3$, reaching two neighboring voxels (green). To describe the local geometry, we take the TSDT values within a cuboid around two respective voxels (yellow squares in Fig. 3.3(c)), perform element-wise subtractions and sum them up. Let ε denotes this sum-of-difference operation. By definition, ε of different offsets ψ can be evaluated independently and thus fully parallelizable, which is an useful trait since this computation will be carried out multiple times with thousands of randomly generated ψ for the same $\mathbf{v}_v$.

The feature vector $\mathbf{f}$ consist of ε resulted from many offset pairs ψ. More precisely, it is a function of $\mathbf{v}_v$ but takes an offset pair ψ, a binary variable η (whether to use *Local Coordinate Frame* (LCF) or not), and a rotational matrix $\mathbf{R} \in SO(3)$ (the orientation of LCF) as parameters. Every possible combination



(a) local support **(b) LCF procedure** **(c) Example results**

Figure 3.4: **Local coordinate frame**. Our method leads to quasi pose-covariant LCFs.

of offset pairs ψ and binary variables η results in one independent feature attribute κ, in notations: $f_\kappa(\mathbf{v}_v) = \varepsilon(\mathbf{v}_v; \mathbf{R}^\eta(\psi))$. The dimensionality of $\mathbf{f}$ is virtually infinite. Binary variables η determines the alignment of the offset ψ with respect to a LCF, whose transformation is specified by $\mathbf{R}$. The intuition behind this adjustment is to make features $\mathbf{f}$ invariant to poses, c.f. Fig. 3.3(b) and (c). Without re-orientations, ψ might land on different types of voxel pairs, c.f. Fig. 3.3(a) and (b), and hence cause different feature responses ε, despite the fact that the current voxels are located on the same position on the body. Both offset pairs ψ and binary variables η are learned during forest training, while the rotational matrix $\mathbf{R}$ is characterized by a LCF obtained as follows.

3.4.3 Local coordinate frame

Defining local coordinate frames for 3D primitives (voxels, vertices, points) has long been studied and usually comes with their 3D descriptor counterparts, see [109] for a comprehensive review. An ideal LCF is supposed to follow whatever transformations the meshes undergo, namely, as *co-variant* as possible, such that the consequent feature representations are as *invariant* as possible. Constructing a LCF consists in defining three orthonormal vectors as $[x, y, z]$ axes. To do that, the state-of-the-art methods in the field of LCFs for rigid matching of 3D meshes and point clouds mainly rely on the neighboring points within a *local support* [41, 101, 107, 110, 142]. The way they leverage spatial distributions can in general be classified into two categories: (1) EigenValue-Decomposition (EVD) [101, 107, 142], and (2) signed distance (SignDist.) [41, 110]. Since it is impractical to repeat EVD process for all surface voxels $\mathbf{v}_v$, in the following, we propose an adaptation of SignDist. approach to our volumetric representations [75]. This conclusion is drawn after an extensive study and comparison of three LCF approaches presented in Appendix A.

Specifically, for each $\mathbf{v}_v$, we consider its surface normals $\mathbf{n}_v$ as z axis, and obtain y axis by $z \times x$. The task remained is to identify a repeatable x axis. To this end, the class of SignDist. approaches look for a discerning point within the support (yellow voxel in Fig. 3.4(b)). We first open an local cuboid support (pink) around each $\mathbf{v}_v$ (green) as visualized in Fig. 3.4(a). The search involves only the peripheral voxels $\tilde{\mathbf{v}}$ (cyan) lying on the intersection of support borders and the surface. The discernibility is defined as the maximum signed distance to the tangent plane [41]:

$$\hat{\mathbf{v}} = \arg\max_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \left((\tilde{\mathbf{v}} - \mathbf{v}_v)^\top \mathbf{n}_v \right), \tag{3.3}$$

where $\tilde{\mathcal{S}}$ is the intersection of support borders and the surface. The x axis is the projection of the vector directed from $\mathbf{v}_v$ towards $\hat{\mathbf{v}}$. Fig. 3.4(b) illustrates the full procedure. Note that there is no guarantee that the discerning point $\hat{\mathbf{v}}$ from Eq. 3.3 is always repeatable: in particular, if different directions yield similar values of the signed distance, the x axis will be ambiguous, hence the resulting LCFs could rotate about the z axis. Therefore, as shown in Fig. 3.4(c), this approach produces LCFs quasi-covariant to pose changes, and as a result, only quasi-pose-invariant features $\mathbf{f}$. We leave such noise for forests to take care of during learning.

3.5 Correspondence inference

Now we proceed to the second mapping: $\mathbb{N}^3 \to \mathbb{R}^3$, where the surface voxel $\mathbf{v}_v$ traverses a pre-trained regression forest according to the feature responses $\mathbf{f}$ above, and attains a point in $\mathbb{R}^3$ that lies on the surface embedding defined by the vertices of the reference model $\mathbf{X}$.

3.5.1 Random forest

As briefly reviewed in Sect. 2.3, a forest is an ensemble of T binary decision trees, each separating data with split functions at branch nodes and storing statistic models at leaf nodes. The training objectives, split functions, and statistical models vary from task to task. We refer readers to [45] for a comprehensive review.

Forest training

To incorporate abundant training variations, we animate $\mathcal{X}$ to a variety of poses with a method similar to [134]. After voxelizing all animated meshes, we associate each surface voxel to their locations at rest pose, and obtain a pool of sample-label pairs $\mathcal{D} = \{(\mathbf{v}_v, \mathbf{x}_v)\}$. Each tree is trained with a random subset of $\mathcal{D}$. Our splitting candidate ϕ is the pair of testing channels κ and thresholds τ, $\phi = (\kappa, \tau)$, where κ is represented by offset pairs ψ and binary variables η. Let $\mathcal{D}_N$ denotes the samples arriving at a certain branch node. The training process is to partition $\mathcal{D}_N$ recursively into two subsets $\mathcal{D}_L$ and $\mathcal{D}_R$, based on randomly generated ϕ:

$$\mathcal{D}_L(\phi) = \{\mathbf{v}_v \in \mathcal{D}_N | f_\kappa(\mathbf{v}_v) = \varepsilon(\mathbf{v}_v; \mathbf{R}^\eta(\psi)) \geq \tau\} \tag{3.4a}$$

$$\mathcal{D}_R(\phi) = \{\mathbf{v}_v \in \mathcal{D}_N | f_\kappa(\mathbf{v}_v) = \varepsilon(\mathbf{v}_v; \mathbf{R}^\eta(\psi)) < \tau\}. \tag{3.4b}$$

Whether ϕ is a good split or not depends on whether it produces more homogeneous subsets. In supervised learning, this is often measured by information gain:

$$I(\phi) = H(\mathcal{D}_N) - \sum_{i \in \{L,R\}} \frac{|\mathcal{D}_i(\phi)|}{|\mathcal{D}_N|} H(\mathcal{D}_i(\phi)), \tag{3.5}$$

where $H = \sigma^2(\cdot)$ is the entropy, measured by the variance of all $\mathbf{x}_v$ in the sample set as in [63] . The split that maximizes the information gain, $\phi^* = \mathrm{argmax}_\phi I(\phi)$, is stored for the later prediction use. The tree recursively splits samples and grows until one of the following stopping criteria is true: (1) it reaches the maximum depth, or (2) the number of samples $|\mathcal{D}_N|$ is too small. Once a leaf node is reached, we perform mean-shift clustering [39] to represent the distributions of $\mathbf{x}_v$ as a set of confidence-weighted modes $\mathcal{H} = \{(\mathbf{h}, \omega)\}$. $\mathbf{h} \in \mathbb{R}^3$ is the mode location and ω is a scalar weight.

Hough forest. Outliers such as false geometries, or un-removed background often exist in visual hulls, drastically deteriorating tracking results. If their models are available, we also include them in the training process, so that forests can identify and reject them online. In this case, the goodness of a split ϕ should be evaluated in terms of both classification and regression. We follow Fanelli *et al.* [52] and adapt the entropy to be:

$$H(\mathcal{D}) = -\sum_{c} p(c|\mathcal{D}) \log p(c|\mathcal{D}) + (1 - e^{\frac{\delta}{\alpha}}) \sigma^2(\mathcal{D}), \tag{3.6}$$

where $p(c|\mathcal{D})$ is the class probability of being foreground or background. It is the weighted sum of the aforementioned regression measure σ^2 and the classification entropy measure. Forests trained with Eq. 3.6 as entropy measure are often referred to as *Hough forests*. During training it learns two task simultaneously:

1. how to distinguish between valid and invalid samples (outliers).

2. how to match valid samples to the template.

The regression part gets increasing emphasis when the current depth δ gets larger (*i.e.* the tree grows deeper), and the steepness is controlled by the parameter α.

Subject coordinate frame. To achieve global rotation invariance, we always rotate meshes into a canonical orientation before the voxelization. In the training phase, since each animated mesh is accompanied with a skeletal pose, we compute two unit-length vectors from the skeleton, and align them to x- and z-axis respectively. The first one is the common perpendicular vector of bone *RShoulder-Torso* and bone *LShoulder-Torso*, while the second one is the sum of them. See Fig. 3.5 for illustrations.

Further recall that the volume center is aligned with the barycenter of meshes, which brings translation invariance. Together these two steps characterize a subject coordinate system, simulating the PCA process on a whole mesh. This strategy substantially reduce the needed amount of training surfaces because the

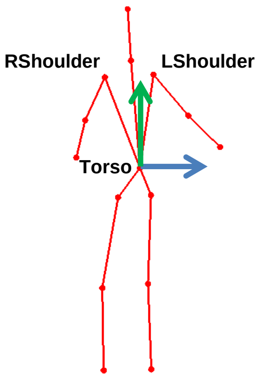


Figure 3.5: **Illustrations of two reference vectors from the skeleton.**

global rotations are canceled. Nevertheless, in order to repeat this process during tracking, we approximate the skeletal pose using results of the last frame. Forest prediction is therefore not completely frame-independent but subject to the previous outcome. We anyway point out that as tracking drifting often happens at the limbs not at the torso, this approach still manage to align the orientations and draw reasonable forest predictions.

Forest prediction

In the prediction phase, a voxel lands on T leaves containing different collections of modes, denoted together as $\mathcal{G} = \{\mathcal{H}_1 \cdots \mathcal{H}_T\}$. A standard way of aggregation is doing mean-shift clustering [39] and keeping the cluster with largest weight. This is usually done for each input point independently. For instance, in Fig. 3.6(a), the green vertex aggregates with only green leaves, and black vertices aggregate with their respective black leaves as well. The consequent $\tilde{\mathcal{Y}}$ is however, often noisy, where vertices of same triangles are mapped to locations distant from each other as in Fig. 3.6(b).

We utilize the triangles $\mathcal{T}_Y$ to ameliorate this problem. All input vertices $\mathbf{y}_i$ are first mapped to voxels, sent into the forest and return in total $O \times T$ predictions. For each vertex i, we consider not only its own predictions, but also those from the neighbors $\mathcal{N}_i$, *e.g.* all green and black leaves in Fig. 3.6(a). We sort the union set $\mathcal{G}_i \cup \mathcal{G}_{\mathcal{N}_i}$ descendingly according to their weights ω, and do mean-shift only with the first half of them. The new mode location with highest confidence is the final output $\tilde{\mathbf{y}}_i$. This strategy respects the mesh connectivity and results in more structured forest predictions. Compare Fig. 3.6(c) to (b) to observe the improvements. To further exploit the connectivity $\mathcal{T}_Y$, spatial mesh smoothing techniques like Taubin smoothing [139] are also possible options. In the next subsection, we will demonstrate how to aggregate leaves more contextually with the well-known *Markov Random Field* (MRF) machinery [94]. However, due the computational overhead, this strategy is not deployed in the final tracking-by-detection pipeline.

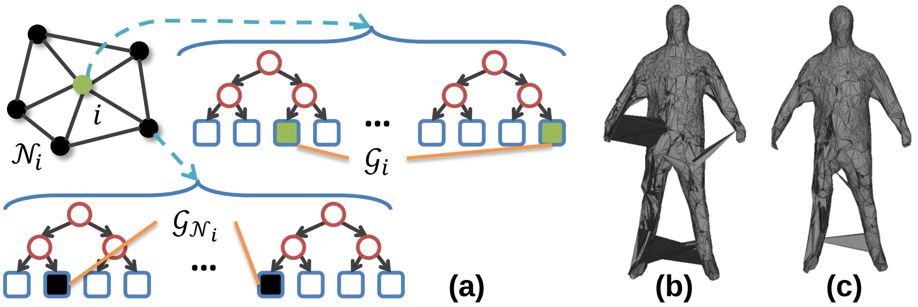


Figure 3.6: **Comparisons of different local aggregation approaches**. (a) Traditional approach aggregates leaves of each vertex (green) independently, while we also take the 1-ring neighbors (black) into account. (b) Forest output $\tilde{\mathcal{Y}}$ by aggregating each $\mathcal{G}_i$ separately. (c) Forest output $\tilde{\mathcal{Y}}$ by aggregating $\mathcal{G}_i \cup \mathcal{G}_{\mathcal{N}_i}$.

Nearest neighbor search. Given the regression results $\tilde{\mathbf{y}}_i$, each vertex i in input gets a closest vertex p in the reference vertex set $\mathcal{V}$:

$$p = \operatorname*{argmin}_{v \in \mathcal{V}_X} \|\tilde{\mathbf{y}}_i - \mathbf{x}_v\|_2. \tag{3.7}$$

Similar to [35], we reject the searched correspondences if their normals differ from each other too much. Our advantage over [35] is that $\mathbf{X}$ and its normals are fixed throughout tracking. One does not have to re-compute normals online, and other speed up algorithms like kd-tree [16] are also feasible. Each correspondence pair (i, p) is associated to a weight $w_{ip} = \exp(-d^2(i,p)/2l^2)$, where $d(\cdot)$ stands for Euclidean distance, and l is the averaged edge length on $\mathcal{M}$. w_{ip} can be used to weight each residual differently in the least-square-based data terms, such as Eq. 5.22. Since the forest gives relatively good initial guess of the matches. It is much easier to find the right matches in this way.

MRF-based global aggregation

As readers shall see later in the experiment section, the TSDT-forest combination proposed above already predict sufficiently decent correspondences $\mathcal{C}$. In this subsection, however, we would like to shortly deviate ourselves from the pursuit of full 3D tracking-by-detection and continue investigating how to draw contextual predictions. Note that due to the computation overhead, even though the method presented in this subsection yield smoother data-model associations, we do not include it in the final tracking-by-detection pipeline.

One fundamental issue in machine learning is: how to make the predictions transit smoothly according to the structures of data. In other words, how to respect the structures of data to predict. When operating on images, this structure is the geometry represented by the changes in intensity values; when it comes to 3D shapes, it is the triangle connectivity. This is usually known as *structural learning* and has a enormous body of literature; see [108] for a complete review.

Generally speaking, vertices that are close to each other should obtain *closer* predictions. The spatial closeness is measured more accurately by *geodesic*

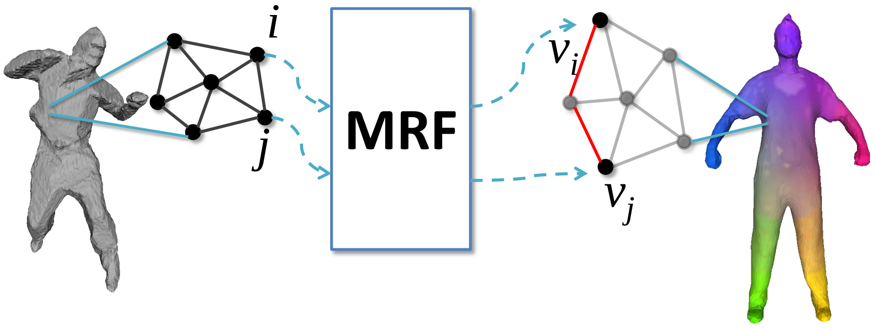


Figure 3.7: **Illustrations of our global aggregation concept**. We apply MRF to preserve neighborhood relationships, *i.e.* geodesic distances between vertices. The left side depicts an input frame, the right side shows the reference frame.

distances, a metric operating on surface manifolds rather than in Euclidean space. Ideally, a pair of neighboring points on the input mesh i and j should be matched to a pair of neighboring vertices on the template, v_i and v_j as well, yielding small geodesic distance. The geodesic distances can be involved in both learning and prediction phases. Usually the former increases training time significantly and hence we use it only in the prediction phase. Since the tree structure and the split functions are already determined and are difficult to change, geodesic information is incorporated only at the leaf-node aggregation stage, where we formulate the inference as an MRF problem which has been well studied in computer vision [94], as illustrated in Fig. 3.7.

Given O input points $\mathbf{y}_i$, the forest returns $O \times T$ leaf predictions $\mathcal{H}$. As described in Fig. 3.6, the standard way of aggregation clusters each T-leaf group parallelly using mean-shift [42, 121], followed by an NN search to assign a corresponding vertex for the most dominant cluster. We move the NN search prior to aggregations, yielding $O \times T$ corresponding vertices. Each point i hence has T candidates v_i to select from. This selection is highly interdependent among neighboring points. To decide which one is the best candidate v_i, one should consider not only its own leaves but also those from the 1-ring neighbors $\mathcal{N}_i$. Taking this into account, we find the overall best v_i by solving the Markov Random Field problem below:

$$\operatorname{argmin}\left(\underbrace{\sum_i -\ln(\omega_{v_i})}_{\text{unary}} + \underbrace{\sum_i \sum_{j\in\mathcal{N}_i} geoDist(v_i, v_j)}_{\text{pairwise}}\right), \tag{3.8}$$

where ω_{v_i} are the scalar weights of the most dominant modes in the reached leaves. Eq. 3.8 encourages candidates v_i with high confidence (unary term) and yet prevents two neighboring points i, j from being mapped too far away in the geodesic sense (pairwise term). It can be solved either with *Simulated Annealing* (SA) [30], or more efficiently with *Loopy Belief Propagation* (LBP) [53].

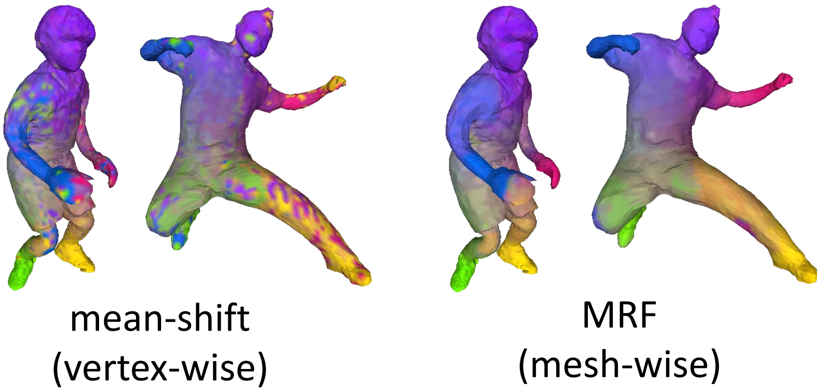


Figure 3.8: **Comparisons of mean-shift and MRF-based global aggregation**.

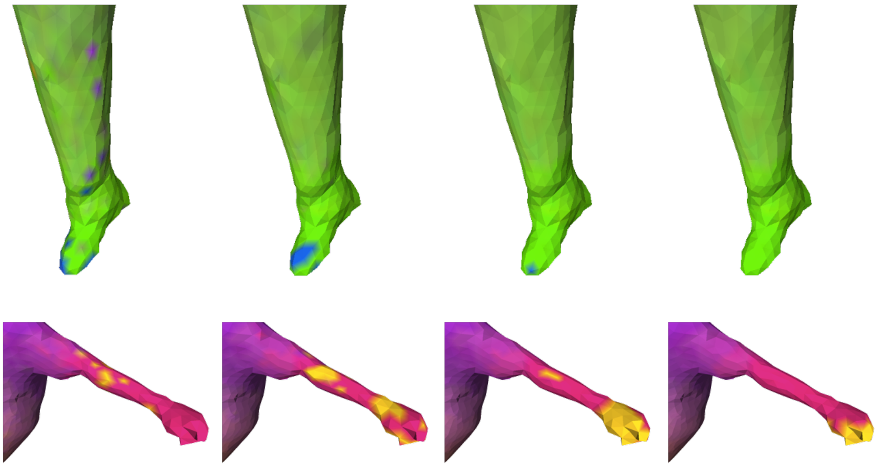


Figure 3.9: **Results for frame 81 of *Bouncing* during different stages of LBP optimization**. From left to right: iterations 1, 2, 4, and 10.

Global vs. local aggregation. Unlike methods in Fig. 3.6 which aggregate leaves more or less locally, Eq. 3.8 is a global aggregation approach that takes all $O \times T$ leaves as input and finds the overall best correspondences $\mathcal{C}$ for the whole shape. We contrast the improvement in Fig. 3.8 and also visualize in Fig. 3.9 the behavior of Eq. 3.8 in each LBP iteration. One can see that erroneous matches first get accumulated and then eliminated altogether. Quantitatively, Eq. 3.8 also lead to substantial improvement against mean-shift, as depicted in Fig. 3.10. Please refer to the experiment section for the error metric of this figure.

Note that this approach actually behalves similar to image de-noising, where MRF has been widely used. One critical assumption is that the majority of points in the local region are matched correctly. Otherwise, Eq. 3.8 tends to dilate the wrong matches and deteriorate the final results. See the right knee/hand of the person on the left in Fig. 3.8 for example.

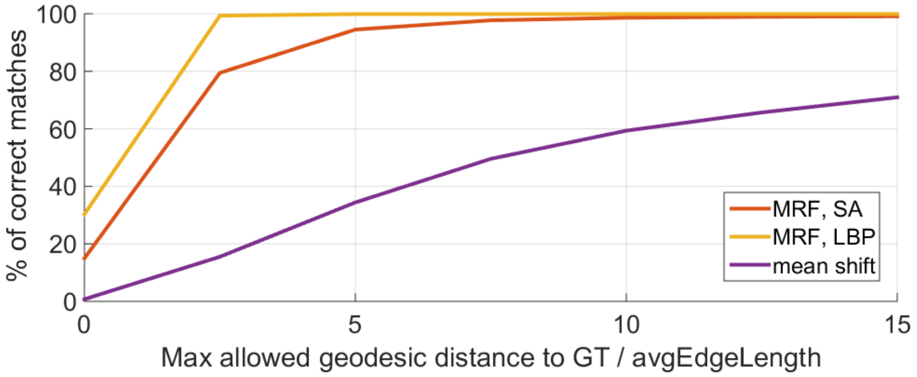


Figure 3.10: **Accuracy comparison of MRF with mean-shift**.

3.6 Experiments

The proposed discriminative model is evaluated briefly in this section. Since data-model associations are not the final goals but the intermediate results of our 3D tracking-by-detection pipeline, we only aim to show that the TSDT-forest combination is more or at least equally descriptive than other well-known 3D descriptors, *e.g*. Heat Kernel Signature (HKS) [29, 136], Wave Kernel Signature (WKS) [9] and MeshHOG [157]. The whole pipeline will be evaluated extensively later in Chapter 5. We describe each vertex with these descriptors and match them either using forests (HKS and WKS) or NN search (MeshHOG). We demonstrate, both visually and quantitatively, that the presented TSDT-based feature is indeed more informative than the existing state-of-the-arts.

Two datasets are chosen for the evaluation purpose: *FAUST* [19] and *Crane* sequence [148]. The former contains 100 static 3D scans from various subjects, while the latter is a temporal sequence of 173-frame from one single subject, reconstructed in multiple-view studios. The accuracies on these two datasets tell us how well the TSDT-forest framework deals with motion sequences and human shape variations, respectively. Given one input mesh, the task is finding correspondences on the reference surface. Due to the availability of ground-truth vertex indices, we consider the registrations of the template as input. To this end, we apply an non-rigid ICP-based method to register the reference surface to each frame in *Crane*, while registrations are already provided in *FAUST*. For each dataset, we train one forest consisting of $T = 50$ trees. At branch nodes, 5000 splitting candidates ϕ are randomly generated and the best one is stored. For *Crane*, we animate the reference into 140 meshes as training data and test on the 173 registrations. In *FAUST*, half the registrations (50 meshes) are chosen for training and the other half for testing. The error measure is the geodesic distance between predicted vertices and ground-truths. If the error is smaller than a certain threshold, we consider the point matched correctly. The percentage of correct matches in varying thresholds fully characterizes the performance of one algorithm and is commonly used in many matching papers [36, 154].

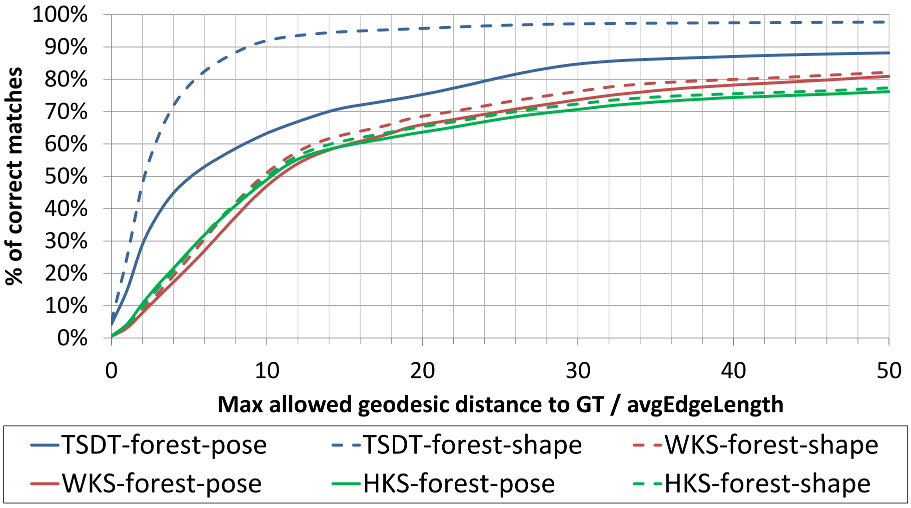


Figure 3.11: **Cumulative errors on *FAUST* [19].**

FAUST. The 100 scans in *FAUST* comprise 10 subjects in 10 poses. One can partition them in different ways to test the generalization to unseen shapes or poses. The results are shown in Fig. 3.11, with x-axis normalized by the averaged edge length of the template. The keyword *pose* means that the forest is trained with all 10 subjects but with only 5 poses, whereas *shape* represents the other way around. To compare with other methods fairly, we keep the Vitruvian-manifold label space unchanged while replacing the TSDT-based features with 30-dimensional scale-invariant HKS [29] or WKS feature vectors [9].

In Fig. 3.11, we see that the proposed TSDT-forest combination yields overall best accuracy. It suggests that the volumetric TSDT-based feature is indeed more informative than H/WKS. Comparing the blue solid curve to the dashed one, we notice that our approach handles unseen shapes better than unseen poses. This is most likely due to the fact that our feature relies mainly on 3D geometry and pose variations lead to significant changes in 3D space more than shape variations. Although this is not observed in the curves of H/WKS because they exploit the spectral domain for better pose invariance, they suffer from the confusion between symmetric parts as visualized in Fig. 3.16. Nevertheless, we would like to point out that as 5000 different splitting candidates are randomly generated at each branch node, our feature vector has a dimensionality virtually longer than 5000, while it is only 30 for H/WKS. To conclude that our TSDT-based feature is certainly a better 3D descriptor than HKS and WKS requires more fair and thorough comparisons but is not the main goal of this thesis.

Crane. We compare the results of *Crane* with MeshHOG [157] which is the extension of image-based histogram of oriented gradient (HOG) to surface manifolds. The correspondences are predicted by searching for the closest point in the MeshHOG feature space.

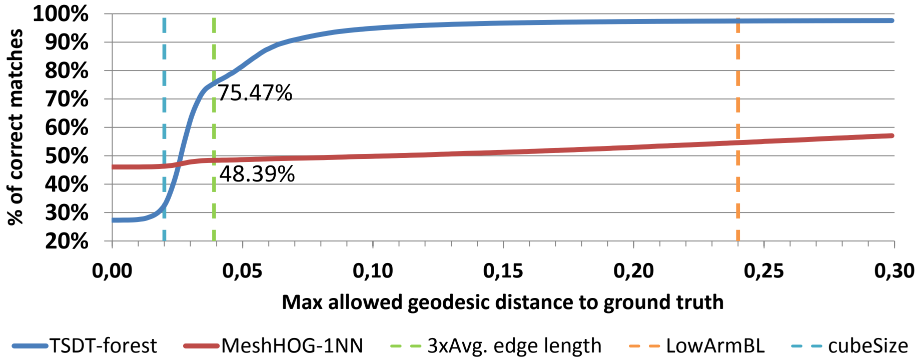


Figure 3.12: **Matching error comparison of our method and MeshHOG of *Crane*.** We yields more locally consistent correspondences.

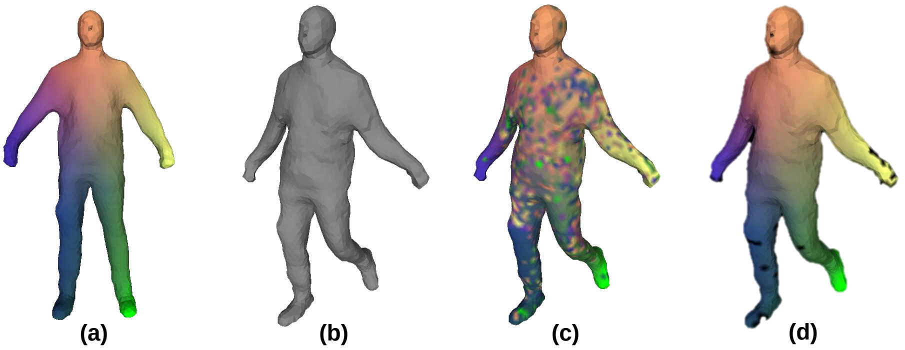


Figure 3.13: **Visual comparisons of MeshHoG and our approach**. (a) references surface of *Crane* (vertex indices encoded in colors). (b) testing mesh. (c-d) matching results from MeshHOG and the proposed TSDT-forest framework, respectively. Our framework attains apparently visually smoother results.

Fig. 3.12 depicts the proportion of vertices whose errors are less than a certain distance. The TSDT-forest framework gives overall 29.34% exact matches, while searching the vertex with closest MeshHOG response gives 46.07%. However, when increasing the tolerance of correctness, our method presents a significant improvement, and yet MeshHOG gains only a little. In order to have more informative analysis, the x-axis is not normalized as in Fig. 3.11. We observe that the accuracy starts increasing when the tolerance reaches approximately the size of voxels (cyan dashed line). With our method, 75.47% of vertices obtain correspondences no farther than three times averaged edge length (green dashed line), and 48.39% for MeshHOG. Moreover, almost half of matches from MeshHOG lie outside the range of roughly the length of lower arms (orange dashed line) from the ground truths. Such a phenomenon is confirmed in Fig. 3.13. Taking Fig. 3.13(b) as input, MeshHOG returns results as in Fig. 3.13(c), whereas our method gives visually smoother matches like Fig. 3.13(d). The proposed

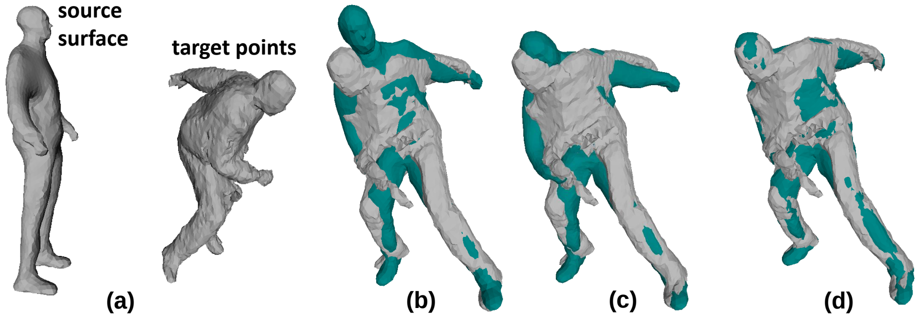


Figure 3.14: **Our discriminative associations bring better initializations**. (a) the source surface and target points. (b) registration results (green) of EM-ICP [35] after 30 iterations. (c) results of TSDT-forest framework (no EM-ICP applied). (d) results of TSDT-forest framework plus 14 iterations of EM-ICP.

TSDT-forest framework certainly provides more locally consistent associations, which, as we will show later in Chapter 5, are sufficiently accurate to guide plausible deformations.

To support this claim, we further visualize in Fig. 3.17 the predicted data-model accusations on noisy reconstructed visual hulls, where no ground truths are available. Black colors means that the predicted correspondences are rejected by simple normal compatibility check [35]. As one can see, our approach is capable of predicting reasonable associations even for noisy meshes. This is of importance since noisy visual hulls are the real input data of the final tracking-by-detection pipeline. HKS and WKS are known for their sensitivity to topological noises, *e.g.* the merging of arms and torso, hence working only for clean meshes. In Fig. 3.17(b), we also shortly demonstrate the efficacy of Eq. 3.6 on outlier rejection. Chair meshes are included into training data and the forest is grown with Eq. 3.6 as the entropy measure. As a result, we can identify observations on the chair online and remove them, so that they do not affect the subsequent tracking stage. In this scenario, the task of forests is throwing away the points of known outlier classes and in the meanwhile also predicting correspondences for the remaining points.

Registration. Next, we shortly demonstrate the capacity of the proposed method on registration tasks. With the source surface and distant target points as in Fig. 3.14(a), ICP-like approach only produces results like Fig. 3.14(b), where many discrepancies still remain after 30 iterations. Instead, our TSDT-forest framework brings the source mesh closer to the target points in one shot as in Fig. 3.14(c). After 14 ICP-iterations it refines the results as in Fig. 3.14(d). Later in the experiments of Chapter 5, if reference surfaces are not aligned with the first frames, we also register them like this before the tracking starts. Namely, one does not have to manually align the template and the input of first frame for initializations. The whole pipeline becomes fully automatic thanks to the discriminative correspondences presented in this chapter.

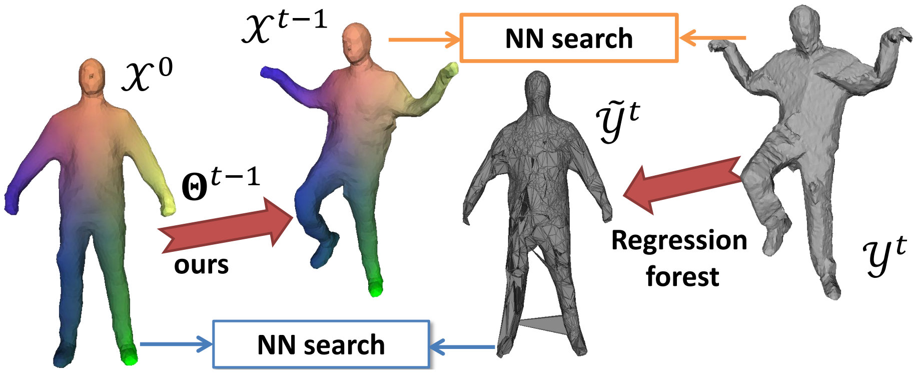


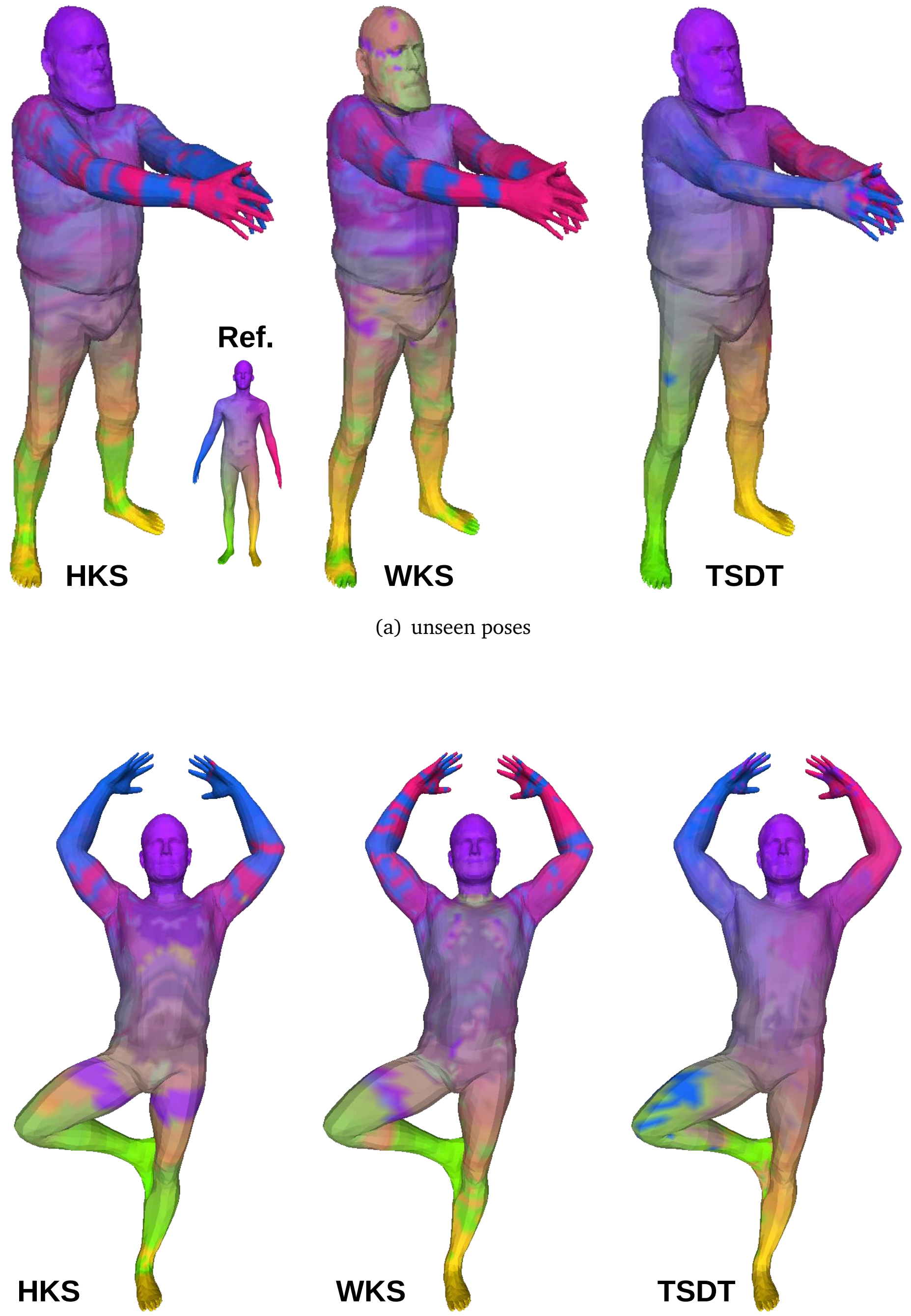
Figure 3.15: **The conceptual comparison between discriminative associations and ICP-based tracking strategies**.

Application in tracking. In Fig. 3.15, we again highlight the difference between two correspondence search strategies in tracking. The goal is to associate primitives in $\mathcal{Y}^t$ to those in $\mathcal{X}^0$. As they are too far away from each other, it is infeasible to get decent correspondences by a simple local proximity search (NN) on them. However, during tracking one always has the previous-frame results $\mathcal{X}^{t-1}$ which is closer to $\mathcal{Y}^t$. Initializing correspondences $\mathcal{C}$ by a NN search on $\mathcal{X}^{t-1}$ (orange block) and refining it while evolving $\mathcal{X}^{t-1}$ amount to the standard deformable ICP framework. Our approach presents an alternative, where a NN search is done between $\mathcal{X}^0$ and $\tilde{\mathcal{Y}}^t$ (blue block), which is the output of regression forests. It has two advantages over the previous approach, both of which result from the fact that $\mathcal{X}^0$ stays fixed throughout tracking. First, as forests draw predictions quasi frame-independently, $\mathcal{X}^0$ and $\tilde{\mathcal{Y}}^t$ are less affected by the tracking errors, while they directly corrupt $\mathcal{X}^{t-1}$ and in turn the subsequent correspondences. In other words, our approach is more robust for its resilience to error and its capacity to recover from drifting. Second, one can speed up the NN search by learning a kd tree [16] on $\mathcal{X}^0$ offline. This is, however, impractical for ICP-based strategy because both $\mathcal{X}^{t-1}$ and $\mathcal{Y}^t$ change each frame. The kd tree has to be re-learned online which costs more computation than it saves.

Limitation. One clear limitation of the presented approach is inherited from the voxel-based framework: high memory footprints. This in turn limits the number of training meshes and consequently the accuracy of forest prediction. Later in Chapter 4, we will present a more memory-efficient volumetric discriminative association approach based on centroidal Voronoi tessellation (CVT) [50].

3.7 Conclusion

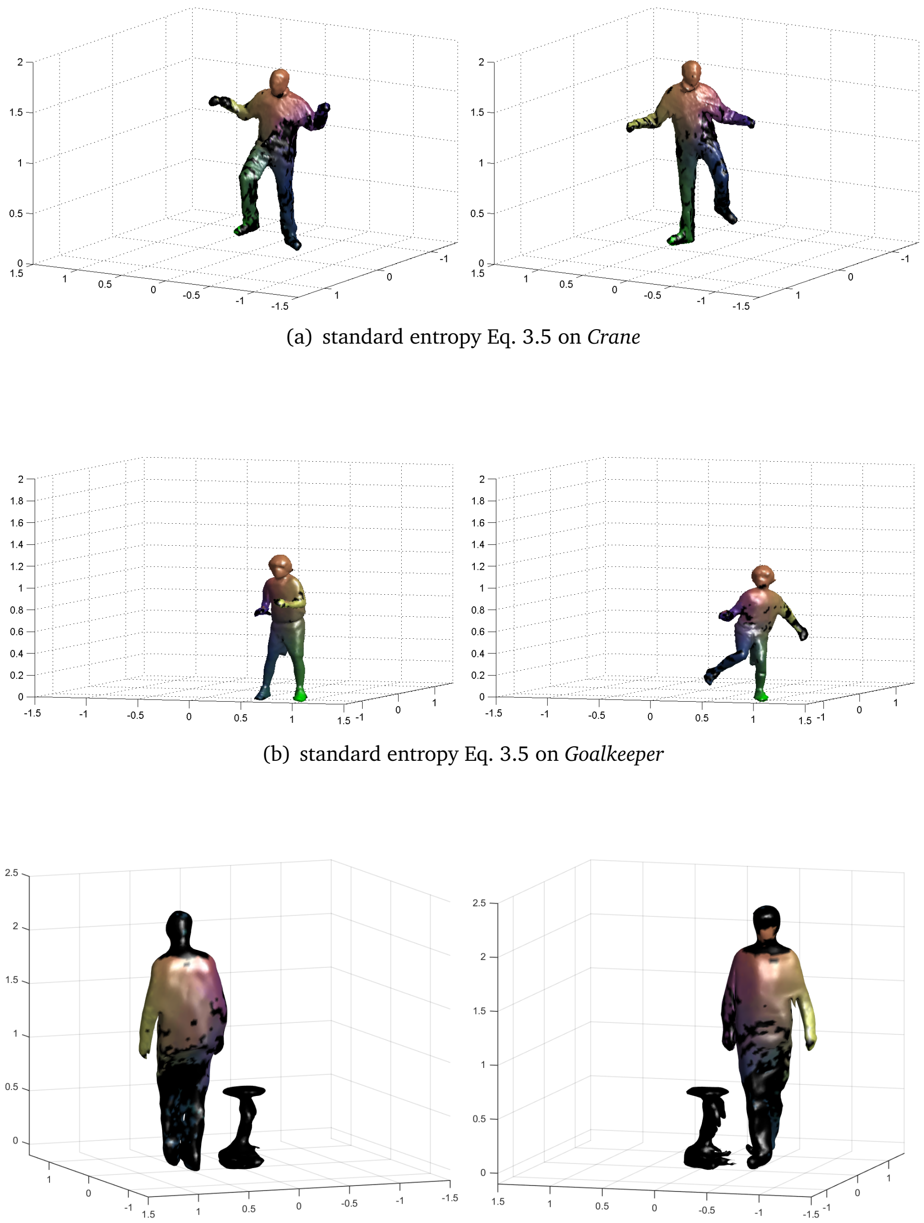
We present an alternative data-model association strategy. With the help of regression forests, we learn from pre-animated/pre-registered meshes to discover better correspondences on input visual hulls. We devise features that exploit directional distance fields to describe 3D geometry. Our learning framework contributes to locally consistent correspondences, which speeds up the convergence of ICP when used as the initializations. The reliability of the proposed method is confirmed by the experiments on numerous public datasets. Future directions include alleviating problems of topological changing and incorporating photometric information.



(a) unseen poses

(b) unseen shapes

Figure 3.16: **Qualitative matching results on *FAUST*.**



(a) standard entropy Eq. 3.5 on *Crane*

(b) standard entropy Eq. 3.5 on *Goalkeeper*

(c) joint classification and regression entropy Eq. 3.6 (Hough forest)

Figure 3.17: **Predicted data-model associations on noisy visual hulls**. Block color means that the points are either outliers, or the inferred correspondences are rejected due to incompatible normals.

4 Volumetric Discriminative Associations

In the previous chapter, we present how to predict surface associations discriminatively with regression forests, where triangular meshes are represented as implicit surfaces with regular voxel grid. This opens the question how solid volumetric representations may help matching hollowed surfaces.

Indeed, surface-based models are dominant in the vision and graphics communities to represent and track shapes for several reasons, primarily due to the fact that visual observations generally lie on the shape surface, but also to the availability of efficient tools to manipulate them. Yet, it has been observed that certain forms of volume-preserving properties are beneficial to model shape deformations in applications [4, 21], or more of our interest, in capturing actors' motions [47] since human movements are largely volume-preserving. Facilitating volumetric discriminative correspondences and eventually a volumetric tracking-by-detection pipeline can be favorable for 3D human shape tracking.

In this chapter, we opt for centroidal Voronoi tessellations (CVTs) to represent volumes and contrive an algorithm that applies again regression forests to match CVT cells of input to those of the reference. The difference of CVTs from voxels is contrasted in Fig. 4.1 and its advantages will be shown in the experiments.

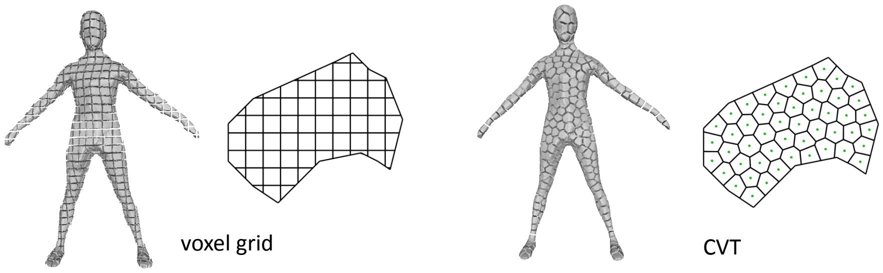


Figure 4.1: **Voxels vs. centroidal Voronoi tessellations**. Image courtesy: [5]

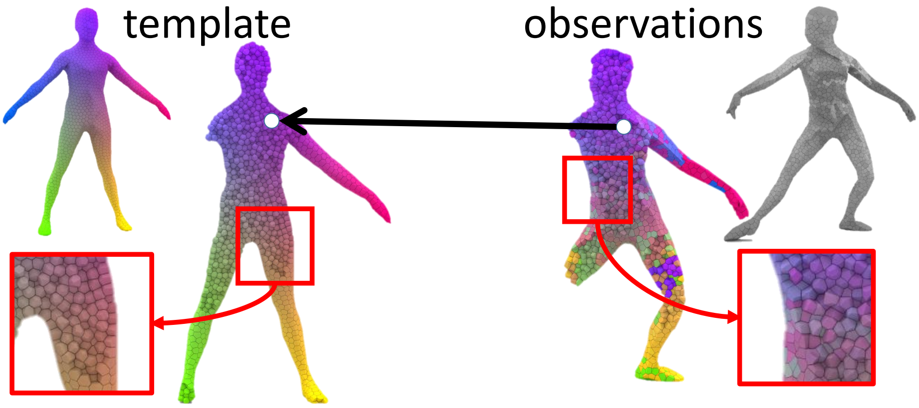


Figure 4.2: **Volumetric discriminative associations**. We represent 3D shapes using centroidal Voronoi tessellations. The goal of this chapter is to match volumetric cells of the observations to those of the template.

4.1 Introduction

3D visual shape tracking aims to recover the temporal evolution of a 3D template shape using visual information. It finds applications in many domains including computer vision, graphics, medical imaging, and has proven successful for marker-less motion capture in recent years. A standard tracking process consists in an alternation of the following two steps. First, finding associations from each primitive of the observed data, *e.g*. 3D points acquired from camera systems, to corresponding primitives of the template 3D surface, typically based on the proximity in Euclidean space (ICP) [17] or a feature space. Second, given such associations, recompute the pose of the template under the constraint of a deformation model, typically based on kinematic skeletons [62, 112, 140, 148], or the piecewise-rigid assumption [6, 35], among others.

Recently, a number of alternative approaches and enhancements have been explored for both stages independently. On one hand, progress has been made in the deformation stage by introducing volumetric deformation models instead of purely surface-based ones. Thanks to its inherent local volume preservation property, this strategy has shown significantly improved robustness to various tracking situations, such as shape folding and volume bias of observed shapes. On the other hand, alternatives have also been proposed for the association problem by discovering them discriminatively using machine learning techniques [117, 140]. This in turn yields the possibility for 3D tracking techniques that are robust to partial tracking failure, while also improving the rate of convergence. Although surface-based features are used in many cases to describe local shapes and construct the associations, volumetric features have proven to be a promising direction for 3D shape description with surface-based templates [71], which we generalize to a fully volumetric pipeline.

In this chapter, we propose a unified volumetric pipeline, where the shape representation, deformation model, feature description, and primitive association are all built on a single volumetric representation, the centroidal Voronoi tessellation (CVT) [50]. Specifically, the observed and template shapes are all tessellated as a set of uniform and anisotropic cells (see Fig. 4.2), which bring benefits at all stages and yield a volumetric representation of regular cell shape and connectivity with controllable cell complexity.

While benefiting from local volume preservation properties inherent to this representation and the associated deformation model, we leverage the configurations of cells to build volumetric distance fields which we use to construct our volumetric feature space. On this basis, we propose a full framework to register a template shape to an observed shape, as two generic CVT cell sets. Because features are expressed in the volume, the proposed method is well suited to obtain fully volumetric detections, in turn helping the volumetric template tracking to be more robust. Thanks to its significantly low memory footprint, we use the representation to propose a multi-template learning framework, where large training sets can be assembled from multiple tracked action sequences for several human subjects. Specifically, every different subject's template is mapped to a generic, subject-agnostic template where the actual learning takes place, to benefit all subsequent tracked subjects. This framework consequently yields better or comparable detection and tracking performance than current state of the art 3D temporal tracking or tracking by detection methods.

4.2 Related Work

3D tracking by detection. The tracking by detection strategy applied to human skeletal poses estimation (Kinect) [121, 155] has shown robustness to tracking failure and reasonable convergence efficiency in real-world applications. It was first transposed to the problem of 3D shape tracking through the work of Taylor *et al.* [140] and presented similar targeted benefits, with the initial intention to substitute ICP-based optimization. The method achieves this goal by learning the mapping from input 3D points from depth sensors, to the human template surface domain, termed the Vitruvian manifold. This yields discriminative associations that replace the step of proximity search in ICP-based tracking methods. Variants of this work have explored changing the entropy function used to train random forests from the body-part classification entropy to the variance on surface embeddings for better data separation [112], or replacing surface-based features with 3D volume features computed on a voxel grid in a local coordinate frame [71]. Both increase the precision by finishing convergence with an ICP-based loop after the discriminative association stage. All these methods are nevertheless based on surface points, thus relying on heterogeneous shape representations, deformation models, target primitives and feature spaces. Our proposal builds a unified framework for all these purposes and takes advantage of volumetric tracking strategies as described below. Also, we introduce a multi-template strategy, where a template is assigned to each subject and mapped to a generic template, allowing to learn from all subject motions sequences for the benefit of any subsequent subject tracking task.

3D features. In many cases, surface-based features are used for recognition or shape retrieval, such as heat kernel signatures (HKS) [136] and wave kernel signatures (WKS) [9] . Both exploit the Laplacian-Beltrami operator, the extension of the Laplacian operator to surface embeddings. These features are nonetheless known for their lack of resilience to artifacts present in noisy surface acquisitions, especially significant topology changes. MeshHoG [157] and SHOT [142] attach a local coordinate frame at each point to achieve invariant representations and reach better performance for noisy surfaces. More detailed reviews can be found in [25] and [65] for triangular surfaces and point clouds, respectively. In the context of discriminative 3D tracking, depth difference features have been used to build random forests on depth data [121, 140]. One common trait of the aforementioned features is that the computation involves only surface points. Huang *et al.* [71] show that features can be built based on local coordinate frames in a regular-grid volume. However, these features are only computed on surface vertices and do not address the need for fully volumetric correspondences as proposed in our work.

4.3 Preliminaries and Method Overview

Given a volumetric domain Ω defined by a shape in $\mathbb{R}^3$, CVT is a clipped Voronoi tessellation of Ω which holds the property that the seed location of each cell coincides with its center of mass [50]. Cells are of regular size and shapes as in Fig. 4.1. A surface is expressed as the border of Ω, *i.e.* $\partial\Omega$.

Let $\mathcal{S}$ denote the set of all cell centroids. Both the template shape Ω_X and the observed data Ω_Y are expressed by their CVT samplings, $\mathcal{S}_X$ and $\mathcal{S}_Y$ with locations $\mathbf{X} \subset \Omega_X$ and $\mathbf{Y} \subset \Omega_Y$ using the method in [150]. We adopt a volumetric deformation framework [5] that groups cells into K clusters, each having a rigid transformation $\mathbf{T}_k \in SE(3)$. The collection of all transformations, $\mathbf{T} = \{\mathbf{T}_k\}_{k=1}^{K}$, encodes the *pose* of the shape. As a result, the problem amounts to estimating the best $\hat{\mathbf{T}}$ such that the deformed template cells $\mathbf{X}(\hat{\mathbf{T}})$ resembles $\mathbf{Y}$ as much as possible. Matching cells $i \in \mathcal{S}_Y$ with cells $s \in \mathcal{S}_X$ is therefore an indispensable task. To this end, each point in $\mathbf{Y}$ is first mapped to the template domain Ω_X, where the closest point in M is sought as the correspondence (as represented by the green line in Fig. 4.5). This mapping $\mathbf{r} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is accounted for by a regression forest which is learned off-line with many pre-tracked CVTs (Sect. 4.5). Given the detected associations, the best pose $\hat{\mathbf{T}}$ is estimated using an EM-ICP algorithm in Sect. 5.5.

4.4 Features

The feature $\mathbf{f}$ proposed in the last chapter cannot be adopted here since it relies on regular voxel grids. We design our new feature $\mathbf{f}$ with several principles in mind. In order to be discriminative for shape matching, our feature should be able to characterize the local neighborhood of any point of the *volumetric* shape. This rules out the descriptors that rely on surface normals such as SHOT [142]. For time and memory efficiency of forest training and prediction, we want our feature vector coefficients to be computable separately. This requirement is not met by the descriptors that rely on unit length normalization. In order to be able

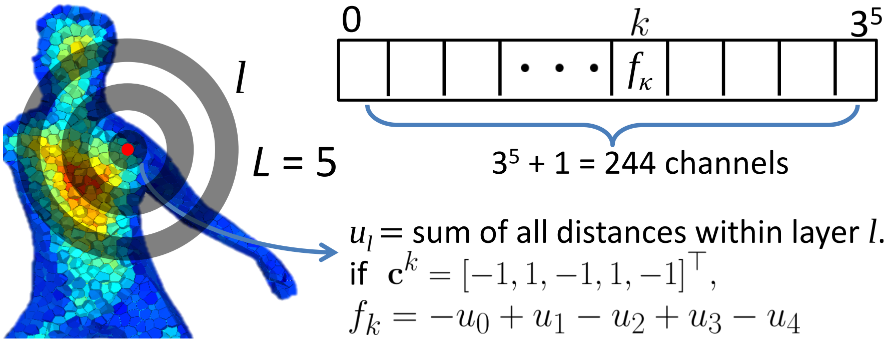

0
k
3^5
f_κ
l
$L = 5$
$3^5 + 1 = 244$ channels
$u_l =$ sum of all distances within layer l.
if $\mathbf{c}^k = [-1, 1, -1, 1, -1]^\top$,
$f_k = -u_0 + u_1 - u_2 + u_3 - u_4$

Figure 4.3: **Illustrations of our feature**. Left: the distance field defined by a CVT sampling $\mathcal{S}$, where each cell stores the distance $d(s, \partial\Omega)$. Blue to red colors means from close to far. Red dot: cell center s to be described. Right: illustration of our feature $\mathbf{f}$. $L = 5$ in this toy example. See text for more explanations.

to match any deformed pose with the template, we would like our feature to be pose-invariant. Therefore, we build it on the Euclidean distance from cell centroids s to the surface $\partial\Omega$: $d(s, \partial\Omega) = \min_{p \in \partial\Omega} d(s, p)$ because it naturally encodes the relative location with respect to the surface and it is invariant to rotations, translations and quasi-invariant to changes of poses. Finally, our feature needs to be robust to the topological noise present in the input data.

Given a distance field defined by a CVT sampling $\mathcal{S}$, our feature is similar in spirit to Haar feature in the Viola-Jones face detector [147], except that the rectangular neighborhood is replaced with a sphere. As visualized in Fig. 4.3, we open an L-layer spherical support region in the Euclidean space around each cell. An L-dimensional vector $\mathbf{u}$ is defined accordingly, where each element u_l is the sum of the distances of all cells falling within layer l. The feature value is the linear combination of all u_l, with coefficients c_l chosen from a set $\Upsilon = \{-1, 0, 1\}$. Formally, suppose $\mathbf{c}$ are L-dimensional vectors whose elements are the bootstrap samples of Υ . Let $\mathbf{c}^\kappa$ denote one particular instance of $\mathbf{c}$, *i.e.* , $\mathbf{c}^\kappa \in \Upsilon^L$. The feature value is then expressed as an inner product: $\mathbf{u}^\top \mathbf{c}^\kappa$, corresponding to one feature attribute κ. We consider all possible $\mathbf{c}^\kappa$ and also take the distance d itself into account. $\mathbf{f}$ is hence a vector of $(3^L + 1)$ dimensions, where 3^L is the cardinality of Υ^L and each element f_κ is defined as:

$$f_\kappa \triangleq \begin{cases} \mathbf{u}^\top \mathbf{c}^\kappa = \sum_l c_l^\kappa u_l, & \kappa < 3^L,\ c_l^\kappa \in \{-1, 0, 1\} \\ d(s, \partial\Omega), & \kappa = 3^L \end{cases} . \tag{4.1}$$

Since each dimension f_κ is computation-wise independent, $\mathbf{f}$ is suitable for decision forests, which select feature channels κ randomly to split the data during training. Being derived from $d(s, \partial\Omega)$, $\mathbf{f}$ inherits the invariance to rigid-body motions. In addition, we normalize distances by their standard deviation in one surface, achieving scale invariance to a certain extent. However, $\mathbf{f}$ is not invariant to pose changes as the contained cells in each layer vary with poses. Although considering geodesic spherical supports instead of Euclidean ones would overcome this issue and yield quasi-invariance to pose changes, the

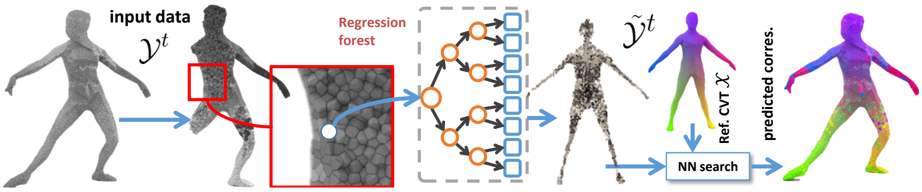


Figure 4.4: **Pipeline of volumetric association prediction**. Correspondences are visualized in the same color. It is essentially the same as Fig. 3.2 but operating on CVT data.

resulting feature would be highly sensitive to topological noise. Thus, we keep the Euclidean supports and let forests take care of the variations caused by pose changes in learning.

4.5 Correspondences inference

We explain in this section how to learn the mapping $\mathbf{r} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ from the observation domain to the template domain with a regression forest [45], which is a set of T binary decision trees. An input cell is first described as a feature vector $\mathbf{f}$ in Sect. 4.4. Taking $\mathbf{f}$ as input, during training each tree learns the split functions that best separate data recursively at branch nodes, while during testing the cell is routed through each tree, reaching T leaves that store statistics (a mode in $\mathbb{R}^3$ in our case) as predictions (Sect. 4.5.1). We first discuss the scenario with one single template and then generalize to multiple ones in Sect. 4.5.2.

4.5.1 Training and prediction

The aim of forests is to map an observed cell to the template domain Ω_X, typically chosen to be in the rest pose. Given a set of CVTs corresponding to the template Ω_X deformed in various poses, we associate each cell $s \in \mathcal{S}_X$ to its locations at the rest pose, denoted as $\mathbf{x}_s^0 \in \mathbf{X}^0$, forming a pool of sample-label pairs $\{(s, \mathbf{x}_s^0)\}$ as the dataset. Suppose $\mathcal{D}_N$ is the set of samples arriving at a certain branch node. The training process is to partition $\mathcal{D}_N$ recursively into two subsets $\mathcal{D}_L$ and $\mathcal{D}_R$ by simple thresholding on a chosen feature channel. Our splitting candidate $\phi = (\kappa, \tau)$ is therefore the pair of thresholds τ and feature attribute indices κ in Eq. 4.1. In branch nodes, many candidates ϕ are randomly generated and the one that maximizes the information gain I, $\phi^* = \operatorname{argmax}_\phi I(\phi)$, is stored for the later prediction use.

We use the typical definition of information gain:

$$I(\phi) = H(\mathcal{D}_N) - \sum_{i \in \{L,R\}} \frac{|\mathcal{D}_i(\phi)|}{|\mathcal{D}_N|} H(\mathcal{D}_i(\phi)), \qquad (4.2)$$

where H is the entropy , measured as the variance in Euclidean space, *i.e.* $H = \sigma^2$. We do not apply the more sophisticated measure [112] because (1) our continuous labels $\mathbf{x}_s^0$ lie in a volumetric domain Ω and (2) templates are

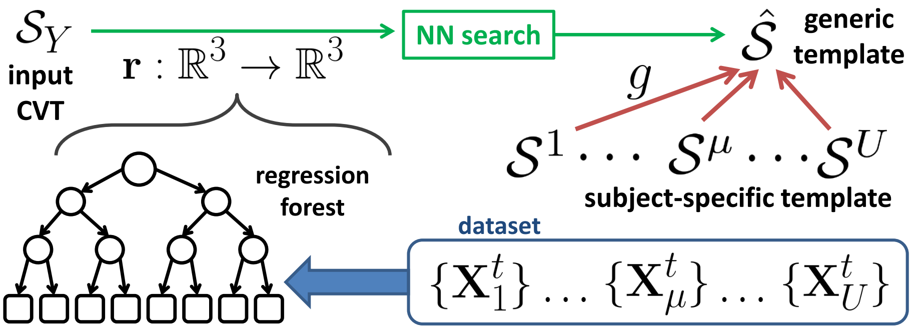

$\mathcal{S}_Y$
input CVT
$\mathbf{r}:\mathbb{R}^3 \to \mathbb{R}^3$
NN search
$\hat{\mathcal{S}}$
generic template
g
$\mathcal{S}^1 \dots \mathcal{S}^\mu \dots \mathcal{S}^U$
subject-specific template
regression forest
dataset
$\{\mathbf{X}_1^t\} \dots \{\mathbf{X}_\mu^t\} \dots \{\mathbf{X}_U^t\}$

Figure 4.5: **The schematic flowchart of the multi-template learning framework.** Red arrows: mappings g^μ that associate the indices from each subject-specific template $\mathcal{S}^\mu$ to the common template $\hat{\mathcal{S}}$. $\mathbf{X}_\mu^t$ are the temporal evolutions of each template. Blue: training; green: prediction.

usually chosen in canonical T or A poses. The Euclidean approximation holds more naturally here than in [71, 112], where the regression is performed along the surface manifold. The tree recursively splits samples and grows until one of the following stopping criteria is met: (1) it reaches the maximum depth, or (2) the number of samples $|\mathcal{D}_N|$ is too small. A mean-shift clustering is performed in a leaf node to represent the distributions of $\mathbf{x}_s^0$ as a set of confidence-weighted modes $\mathcal{H} = \{(\mathbf{h}, \omega)\}$. $\mathbf{h} \in \mathbb{R}^3$ is the mode location and ω is a scalar weight.

In the prediction phase, a cell $i \in \mathcal{S}_Y$ traverses down the trees and lands on T leaves containing different collections of modes: $\{\mathcal{H}_1 \cdots \mathcal{H}_T\}$. The final regression output $\mathbf{r}_i$ is the cluster centroid with largest weight obtained by performing mean-shift on them. Each observed cell then gets a closest cell p in the reference $\mathcal{S}_X$: $p = \operatorname{argmin}_{s \in \mathcal{S}_X} \left\|\mathbf{r}_i - \mathbf{x}_s^0\right\|_2$. The correspondence pair (i, p) serves as input to the volumetric deformation framework described in Sect. 5.5.

4.5.2 Multi-template learning

The above training scenario requires deformed CVTs of consistent topology such that one can easily assign each cell sample s a continuous label which is its rest-pose position $\mathbf{x}_s^0$. It hence applies only to one template. However, the amount of training data for one single template is often limited because a fully volumetric shape and pose modeling framework is still an open challenge. To avoid overfitting, the rule of thumb is to incorporate as much variation as possible into training. This motivates us to devise an alternative strategy that learns across different CVT topologies.

Given U distinct CVT templates: $\{\mathcal{S}^\mu\}_{\mu=1}^U$[1], whose temporal evolutions are recovered with the method in [5], resulting in a collection of different templates deformed in various poses: $\{\{\mathbf{X}_1^t\} \cdots \{\mathbf{X}_U^t\}\}$ as our dataset. To include all of them into training, we take one generic template $\hat{\mathcal{S}}$ as the reference. Intuitively, if there exists a mapping g that brings each cell $s \in \mathcal{S}^\mu$ to a new cell $g(s) = \hat{s} \in \hat{\mathcal{S}}$,

[1] The template suffix X is dropped to keep notations uncluttered.

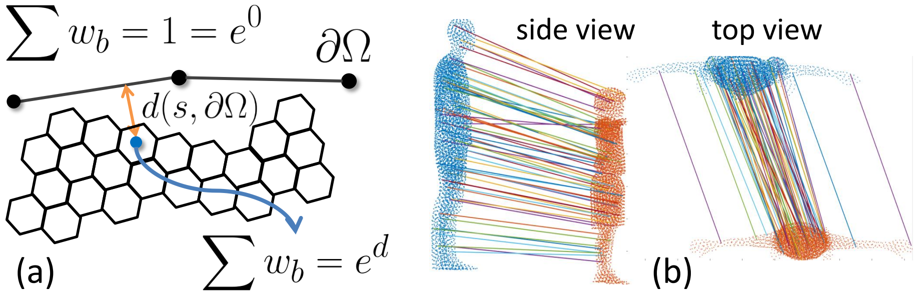


Figure 4.6: **Generalized skinning weights and matching results**. (a): illustration of our strategy adapting skinning weights to CVT cells. Distances $d(s, \partial\Omega)$ are reflected in normalizations. (b): result of matching two templates.

one only needs to change the template-specific labels $\mathbf{x}_s^0$ to the corresponding $\mathbf{x}_{\hat{s}}^0$, which are common to all templates, and the training process in Sect. 4.5.1 can again be applied. In other words, we align topologies by matching every template $\mathcal{S}^\mu$ to $\hat{\mathcal{S}}$. Fig. 4.5 depicts this multi-template learning scheme.

Although various approaches for matching surface vertices exist, only a handful of works discuss matching voxels/cells. Taking *skinning weights* as an example, we demonstrate in the following how to adapt a surface descriptor to CVTs. Note that our goal is not to propose a robust local 3D descriptor. With proper modifications, other descriptors can be used as well for shape matching.

Generalized skinning weights. Skinning weights are originally used for skeleton-based animations, aiming to blend the transformations of body parts (bones). Usually coming as a side product of the skeleton-rigging process [11], it is a vector $\mathbf{w}$ of B-dimensions, each corresponding to a human bone b and B is the number of bones. The non-negative weight w_b indicates the dependency on that part and is normalized to sum up to one, *i.e.* $\sum_b w_b = 1$. As such, a skinning weight vector $\mathbf{w}$ is actually a probability mass function of body parts, offering rich information about vertex locations. To extend it from surface vertices to CVT cells, we first relax the unity-summation constraint as $\mathbf{w}$ is not used to average transformations of bones but only as a descriptor here. The intuition behind the adaptation is that, a CVT cell should have bone dependencies similar to the closest surface point. Therefore, for a cell whose distance to the surface is d, its skinning weight is simply the one of its closest surface point, scaled by e^d. When the shortest distance does not exactly correspond to a vertex but to a point in the middle of a triangle, we use barycentric coordinates as the coefficients to linearly combine the skinning weights of the three vertices. Note that this does not violate the unity-summation constraint for surface vertices as their distance d is still zero. We illustrate this concept in Fig. 4.6(a). The mapping g is then determined by searching for the nearest neighbor in the skinning weight space: $g(s) = \arg\min_{\hat{s}\in\hat{\mathcal{S}}} \|\mathbf{w}_{\hat{s}} - \mathbf{w}_s\|_2$.

In practice, we use `Pinocchio` [11] to computes skinning weights, extend them from surface vertices to CVT cells, and match all cells to those of the common template $\hat{\mathcal{S}}$. The resulting skeletons are not used in our method. Fig. 4.6(b) visualizes one example of matching results. Our approach yields reasonable matches, regardless of the difference in body sizes. Due to the descriptiveness of skinning weights, symmetric limbs are not confused. Note that this computation is performed only between user-specific templates $\mathcal{S}^{\mu}$ and the generic one $\hat{\mathcal{S}}$ off-line once. Input data $\mathcal{S}_Y$ cannot be matched this way, because rigging a skeleton for shapes in arbitrary poses remains a challenging task.

Probabilistic inverse mapping g^{-1}. In Sect. 4.5.2, we align the topologies of different templates so that forests are learned based in a topology-consistent domain. To this end, we utilize skinning weights $\mathbf{w}$ and develop a mapping g that maps each cell s on a subject-specific template $\mathcal{S}^{\mu}$ to a cell $\hat{s}$ on the common template $\hat{\mathcal{S}}$:

$$g^{\mu}(s) = \arg\min_{\hat{s}\in\hat{\mathcal{S}}} \|\mathbf{w}_{\hat{s}} - \mathbf{w}_s\|_2, \forall s \in \mathcal{S}^{\mu}. \tag{4.3}$$

During tracking, forests predict the correspondences that lie on the generic template $\hat{\mathcal{S}}$ and one has to revert it back to the cell index on the subject-specific template $\mathcal{S}^{\mu}$.

We assume users know the tracking subject, *i.e.* μ is known. Since g^{μ} is constructed by nearest neighbor search, leading to a many-to-one function, the inverse mapping $(g^{\mu})^{-1}$ is by nature ill-defined. We therefore resort to a probabilistic formulation. Specifically, given a predicted cell $\hat{s}$ on the common template $\hat{\mathcal{S}}$, all the possible $s \in \mathcal{S}^{\mu}$ being mapped to $\hat{s}$ are taken into account. When they are used to construct the least-square-based energy formulations, we weight them differently according to their distances to $\hat{s}$ in the skinning-weight space. This strategy fits naturally to the EM-ICP framework presented later in Sect. 5.5.

4.6 Experiments

We validate our approach with numerous multi-view sequences, whose profiles are summarized in Table 4.1. For each frame, a coarse visual hull is reconstructed

Template / #Vertex / #Cell	Sequence	Frames
Ballet / 6844 / 5000	*Seq1* [5]	500
	Seq2 [69]	936
Goalkeeper / 5009 / 5000	*SideJump* [5]	150
	UpJump [6]	239
Thomas / 5000 / 4998	*Seq1* [69]	1500
	Seq2 [69]	1400

Table 4.1: **Sequences used in the experiments in Sect. 4.6**. For each subjects, the training set is the random 250 tracked CVTs sampled from first sequences and testing on the unseen second sequence.

Sect.	Forest	T	Testing data
4.6.1	template-specific[2]	20	1. tracked CVTs of seq1 (Tr) 2. unseen tracked CVTs of seq2 (Te)
5.6.3	multi-template	50	unseen raw CVTs of seq2

Table 4.2: **Different experimental settings in two sections**.

by a shape-from-silhouette method [55], followed by [150] to draw CVT samplings (raw CVTs). Given a CVT template, we then perform an EM-ICP based method [5] on the raw CVTs to recover temporal coherent volumetric deformations (tracked CVTs). We evaluate our method in two aspects: detection accuracy (Sect. 4.6.1) and tracking results (Sect. 5.6.3). For the sake of completeness and clarity, we describe the setting of two experiments here together. Unless otherwise specified, we follow the experimental protocol below.

Experimental protocol. We first explain the settings common to two experiments. For each subject, up to 250 tracked CVTs are randomly chosen from the first sequence as the training dataset, while the second sequences are completely left out for testing. We open $L = 8$ sphere layers for the feature computation. Each tree is grown with 30% bootstrap samples randomly chosen from the dataset and trees are grown up to depth 20.

Two experiments, however, differ in the input data for testing. To evaluate the quality of estimated associations, we feed the tracked CVTs into forests due to the availability of ground truth indices (Sect. 4.6.1), whereas raw CVTs are used as the input for tracking experiments in Sect. 5.6.3. Some distinct experimental settings of the two are exposed in Table 4.2.

4.6.1 Single-template learning

We draw a careful and thorough comparison to the discriminative associations presented in [71], which is the early version of the approach presented in Chapter 3. The contributions of CVT on improving the correspondences detection are evaluated with two folds.

First, we follow the learning framework in [71] but replace their voxel-based features with ours in Sect. 4.4, denoted as *CVTfeature*. Next, we further change the regression domain from surfaces to volumes, as described in Sect. 4.5.1 (*fullCVT*). We test on the tracked CVTs and report the results on all frames of training sequences (Tr) and testing ones (Te). The drop between them is a natural phenomenon for every machine learning algorithm and indicates the ability to generalize. If the Euclidean distances between the predicted cell index and the ground truth are smaller than a certain threshold, it is considered as correct. To align the experimental setting, here the regression forests are subject-specific and consist of only $T = 20$ trees.

[2] More precisely, forests in Sect. 4.6.1 are all single-template based except for the one in "multi-template learning" paragraph.

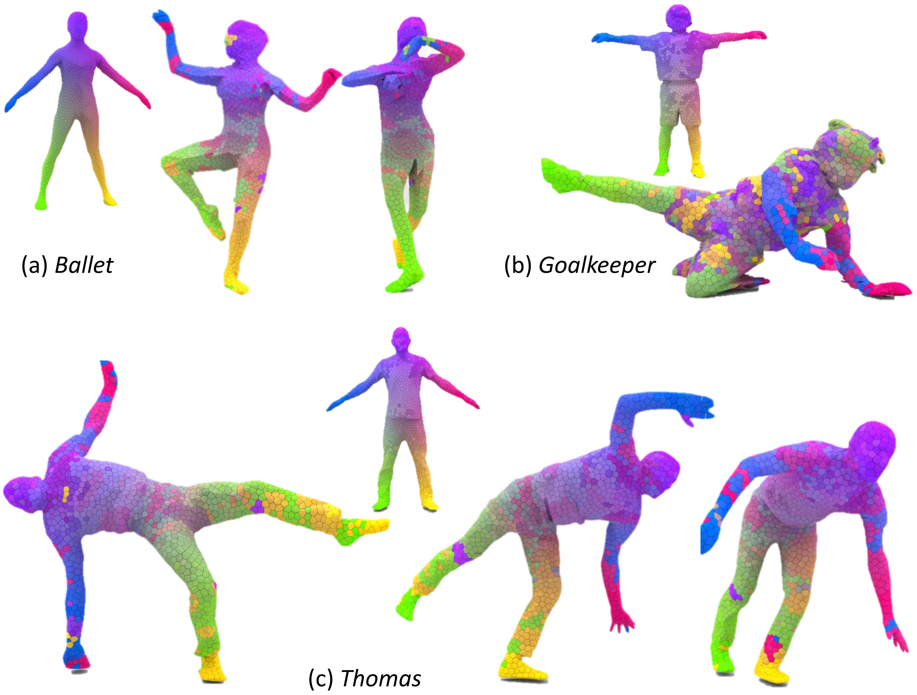
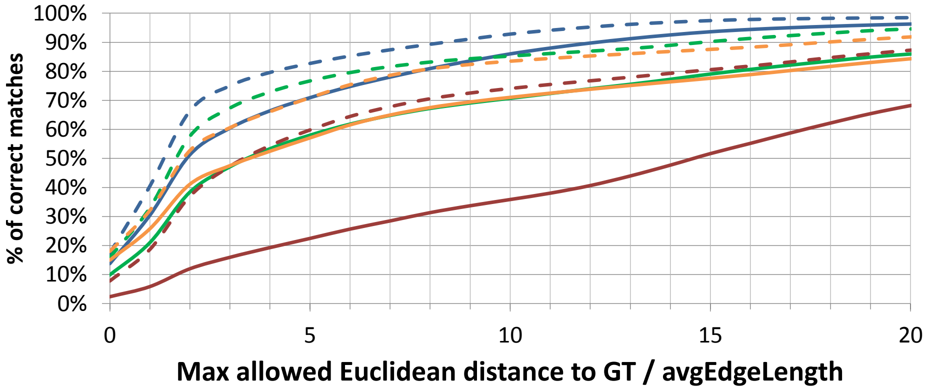


Figure 4.7: **Qualitative matching results on the raw CVTs**. Best viewed in pdf.

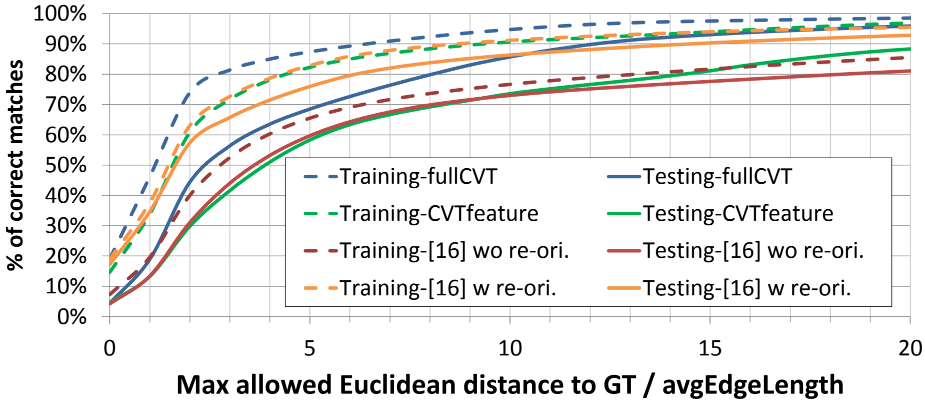
Some visual results of the *fullCVT* approach on raw CVT input are shown in Fig. 4.7. Fig. 4.8 shows the percentage of correct matches in varying thresholds for *Thomas* and *Ballet*. Since *CVTfeature* and [71] are regressing to surfaces whereas *fullCVT* regresses to volumes, we normalize the x-axis by the average edge length of templates to yield fair comparisons. While the results of *CVTfeature* are comparable to [71] (green vs. red or orange), *fullCVT* attains the improved accuracies (blue vs. red or green), demonstrating the advantages of our fully volumetric framework.

Discussion. It is worth a closer analysis to compare our approach against [71]. Compared to volumes of regular grids, CVT is certainly a more memory-efficient way to describe 3D shapes. In practice, [71] describes each mesh with 150^3 voxels, while we need only 5k cells[3]. Consequently, [71] is not able to include a sufficient amount of training shapes, leading to a major drawback that forests are limited to one single subject and learn merely pose variations. To further decrease the needed number of training meshes, [71] exploits skeletal poses to cancel the global orientation. This in turn makes every mesh in the training dataset face the same direction. It follows that during tracking the input data has to be re-oriented likewise using the estimated skeletal poses from the last frame. Our approach, on the other hand, considers distance fields of CVTs which

[3] Further note that [71] stores a 3D vector in each voxel, whereas we store a scalar in each CVT cell. So the ratio is 3×150^3 to 5k.



(a) *Thomas*



(b) *Ballet*

Figure 4.8: **Cumulative matching accuracy of different approaches**. The x-axis is normalized with respect to the average edge length of the templates. The number of trees T is 20 in this experiment. Dashed and solid lines are accuracies on training (Tr) and testing (Te) sequences respectively.

is naturally invariant to rotations and hence does not require re-orientations. We anyway compare to [71] in both settings. Orange curves in Fig. 4.8 shows the results with re-orientation, which is better than the proposed strategy in *Ballet*. Nonetheless, without re-orienting data, the accuracy drops substantially during testing (compare red to orange). The efficiency on memory and the invariance of our features are two determining reasons why the presented method is better than [71] and needs just one forest for different subjects in the following experiment.

4.6.2 Multi-template learning

We use the sequences of *Goalkeeper* to verify the advantages of the multi-template learning strategy in Sect. 4.5.2. It is a particularly difficult dataset because motions in the testing sequence *UpJump* have little overlap with those in the

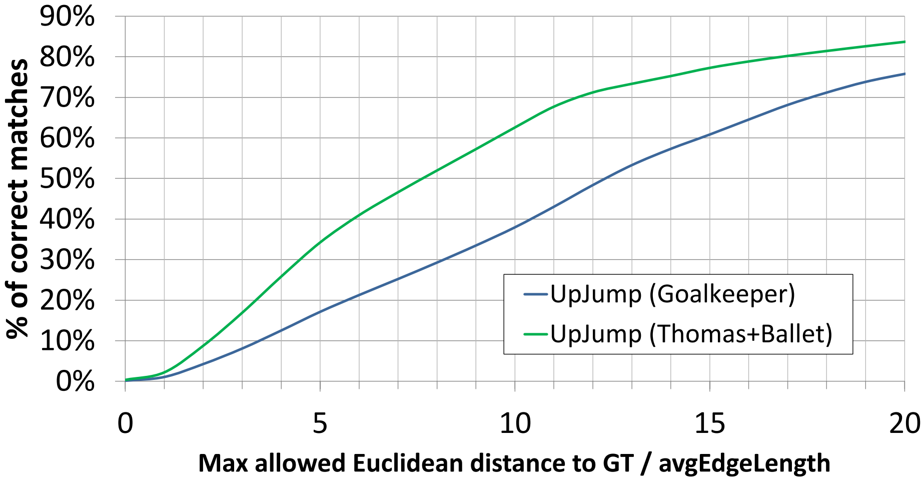


Figure 4.9: **Cumulative matching accuracy of single and multi-template strategy on *Goalkeeper*.**

training *SideJump*. We report in Fig. 4.9 the correctness of correspondences in *fullCVT* setting. Both curves represent the accuracy on testing *UpJump* sequence. The blue curve corresponds to a forest only trained with *Goalkeeper* tracked CVTs, whereas the green curve corresponds to a forest trained with tracked CVTs of *Ballet* and *Thomas*. For both forests, *UpJump* sequence is unseen during training. Compared with the forest of the blue curve, the one of the green curve is trained with twice the amount of meshes from different subjects, and yet it leads to better prediction accuracy on unseen testing poses. This suggests that including more variation of motions indeed results in better generalization to unseen data. It also confirms the necessity and efficacy of our multi-template strategy. We anyway point out that due to the lack of adequate amount of training data, these encouraging preliminary results need to be confirmed on datasets consisting of more subjects and sequences.

4.7 Conclusion

In this chapter, we present a fully volumetric approach for discriminative associations. Centroidal Voronoi tessellation is chosen to be the unified representation used in feature computations, predicting domains, and deformation models (later in Sect. 5.5). Such informative and consistent representations have shown better detected correspondences than other discriminative strategies. We further devise a multi-template learning strategy to enrich the training variation. This leads to one single forest for different subjects and yields cross-subject learning of discriminative associations. The method opens several research directions, and thanks to low memory-footprint characteristics, it can be tested on much larger training sets for discriminative 3D tracking in the future. The methodology can easily be transposed to other volumetric features emphasizing other discriminative characteristics.

notations	descriptions
Ω	A volumetric domain; $\Omega \subset \mathbb{R}^3$.
$\partial\Omega$	A surface (the border of a volume).
$\mathcal{S}$	A set of indices representing CVT cells/centroids.
$(\cdot)_X, (\cdot)_Y$	Subscripts representing variables of templates (X) and observations (Y) respectively.
s, i	Indices of CVT cells on templates (s) and observations (i); $s \in \mathcal{S}_X, i \in \mathcal{S}_Y$.
$\mathbf{X}, \mathbf{Y}$	Sets of 3D locations of CVT centroids; $\mathbf{X} \subset \Omega_X, \mathbf{Y} \subset \Omega_Y$. Note that locations $\mathbf{X}, \mathbf{Y}$ and index sets $\mathcal{S}_Y$ are time dependent variables, while index set $\mathcal{S}_X$ is constant during tracking.
$\mathbf{x}$	3D locations of CVT centroids on the template; $\mathbf{x}_s \in \mathbf{X}, s \in \mathcal{S}_X$.
K	Number of clusters of CVT cells; $K = 150$ for *Ballet* and *Goalkeeper*; $K = 250$ for *Thomas*.
L	Number of layers clusters for feature computation; $L = 8$.
B	Number of bones for skinning-weight computation; $B = 17$.
μ	Denoting $\mathcal{S}$ of different templates; $\mu = 1 \cdots U$.

Table 4.3: **Notations and the setting of parameters in Chapter 4.**

Part III

Tracking

5 Energy Minimization Framework

In the previous chapters, we have addressed the problem of discriminative correspondences in different parameterizations. This chapter continues the quest on tracking-by-detection of 3D human shapes. Now that the data-model associations have been estimated, one needs to deform the template accordingly so that it resembles the observations. This requires the motion parameterization, namely, the search space to be determined first. On one hand, the skeleton-based parameterization has semantic meanings corresponding to the human anatomical structure and is therefore more preferable in many applications; on the other hand, the pure surface-based parameterization holds the ability to explain highly non-rigid deformations, usually leading to more accurate results.

In this regard, we propose to infer skeletal information from surface deformations. This is a complete opposite concept compared to skeleton-based animation, which controls surfaces by the underlying kinematic chain. Instead, we advocate guiding the skeletons by surface deformations. The skeletal pose comes as a by-product of the deformation framework in [35] and fits naturally to its probabilistic formulation. Together they form a simultaneous optimization framework, which can also be interpreted properly from a Bayesian network perspective. Unlike [132], who use this idea only in refinement, we directly use it for tracking [72]. To the best of our knowledge, in the context of 3D human tracking, this is the first work which recovers both human shapes and poses with a single optimization objective. In extension to the work in [72], this chapter provides additional analysis on the simultaneous optimization strategy and presents an improved solution for the pose estimation from surface shapes.

We first derive the formulation of joint optimization in this chapter and verify its strength with experiments. The mesh-based discriminative associations from Chapter 3 is then integrated into it to complete the full tracking-by-detection pipeline. Last but not least, we also elaborate a CVT-based deformation model [5] that takes discriminative correspondences from Chapter 4 as input and realize the tracking-by-detection concept in a fully volumetric fashion.

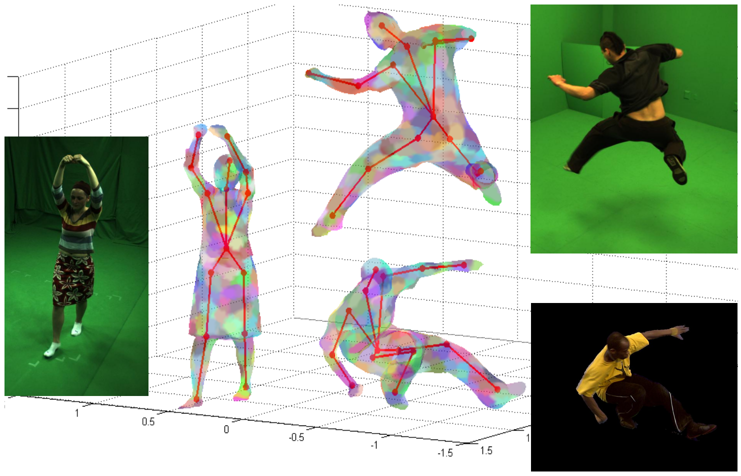


Figure 5.1: **Our approach tracks both the shape and the pose of humans simultaneously**. Results with three different datasets are shown above. Left: Skirt in [62]. Top right: Bouncing in [148]. Bottom right: Free in [128].

5.1 Introduction

Marker-less human motion capture from multiple camera videos is a fundamental task in many applications including sport science, movie industry, and medical diagnostics. Since human motion is defined by both articulated motion and surface deformation they should ideally be estimated simultaneously. However, this requires sophisticated physics-based models that capture the real relationships between pose and shape. Since such models are hard to build and also involve complex parametrization, researchers often decouple them and treat each problem separately. One line of approaches considers only the estimation of surface deformations by fitting a reference model to the incoming image observations, *e.g.* [34, 35, 47, 48]. Another line of approaches parameterizes the model deformations with an articulated human skeleton represented as a kinematic chain [62, 92, 129, 148]. While the latter are less generic and strongly depend on the skeleton parametrization, the former are more generic and require less priors, hence allowing for larger classes of model deformations. Since the human anatomical structure can not be perceived by traditional visual sensors such as color cameras, approaches that model and track shapes instead of internal and unobserved skeletons tend to give more reliable results with visual data. Nevertheless, in many graphical applications that involve human body models, the pose is required as much or more than the shape surface[1]. To this objective, we introduce a method that simultaneous recovers both the shape surface, in the form of a mesh, and its pose with articulated skeleton parameters. This method builds on two related works. First, the patch-based deformable surface registration framework proposed in [35] that relies on soft observation assignments and handles outliers. Second, the bone binding energy presented

[1] Recall that with a slight abuse of terminology, 'shape' refers to the shape pose of surfaces while 'pose' only to skeletal pose. See Table 2.2.

in [132] that forces the skeleton model to stay inside the deformed human body shape. The combination of these two strategies allows us to devise an approach that benefits from a robust surface registration when recovering human body pose and without the need for complex inverse kinematic parametrizations.

This chapter has several contributions. Different from [35] or [62], pose and shape are recovered at the same time. The rest of this chapter is organized as follows. In Sect. 5.2 we review the most relevant related work. Details of the proposed method are described in Sect. 5.3 to 5.5. Validation experiments and results are provided in Sect. 5.6, and we conclude this chapter in Sect. 5.7.

5.2 Related Work

Human motion tracking/capturing has been long studied in both computer vision and graphic communities. Based on the way of parameterizing motion, existing works can be categorized into three classes:

Mesh-based approaches. In this class of methods, motion is solely parameterized on the humanoid surface which evolves in time, without incorporation of a skeleton model. Authors usually introduce some constraints among vertices such that implausible deformations are avoided. Aguiar *et al.* [48] propose a scene-flow-based deformation scheme. To overcome the accumulated flow estimation error, they utilize Laplacian deformation framework [23] as a refinement step. In their follow-up work [47], they first deform a low-resolution tetrahedral mesh to roughly estimate the pose, and then transfer it to a high-resolution scanned model. Surface details are again preserved by Laplacian constraint. Cagniart *et al.* [34] advocate to divide the mesh into small cells called *patches*. A rigidity constraint is imposed among neighboring patches which smooths model deformation. In [35], they further improve the data term and the whole deformation framework acts like a probabilistic iterative closest point (ICP) approach. The advantage of these purely-mesh-based methods is that they can generalize to non-humanoid surface tracking, and they better handle non-rigid deformation such as loose apparel.

Skeleton-based approaches. Since human motion is highly articulated, many authors use skeleton-based models. Motion is then parameterized in a low-dimensional pose parameter space. However, in the observations, whether 3D point clouds or silhouettes, one does not observe the skeleton directly. A mesh surface is still needed for the fitting purpose but it is controlled by the underlying skeleton. As a result, skeleton plays the role of prior deformation model. From this point of view it is actually much more constrained than purely-mesh-based methods. There are mainly two concerns in this family of work: first, how to parameterize motions in terms of the skeleton, and second, how this skeleton should interact with the reference mesh. Vlasic *et al.* [148] parameterize motions as transformation of local coordinate of each joint. Vertex transformation is computed by the linear combination of different joint transformations, known as linear blend skinning [92]. With similar parameterizations, Gall *et al.* [62] adopt quaternion blend skinning [82] which produces less artifacts. In both methods, the skeleton acts as a kinematic chain where local transformations are transferred from the parents to the children. Energies between mesh and observations are defined in pose parameter space, based on the simple assumption that surface deformation is explained only by the skeleton. A second stage of surface refinement is usually required.

Hybrid approaches. The first category of approaches emphasizes more on the surface consistency, whereas the second category of approaches focuses on the pose. Straka *et al.* [132] advocate the integration of both categories into one energy function. They introduce differential bone coordinates as an implicit skinning approach, and therewith they formulate a skeleton-binding energy term defined on the parameters of both mesh surface and skeleton. This allows them to jointly estimate pose and shape, and they show that optimizing in this coupled space results in more robustness. Moreover, skeletons are parameterized only in terms of joint location. Although losing some rotational degree of freedom (DoF) for each joint, this makes the energy term quadratic in terms of both, body joint positions and mesh vertex positions. Therefore, the optimal solution can be obtain via standard optimization method.

The difference of our approach compared to [132] is that we compute the bone energy per patch rather than per vertex. In addition, our observations are 3D visual hull reconstructions instead of 2D silhouettes. With 3D information, we are able to handle ambiguous situations that one cannot do with only image observations. Furthermore, we partition observations into body part regions according to learned partitioning in the previous frames. This allows us more efficient matching of the reference mesh and the input 3D observations, which combined with optimization with soft assignments from [35] makes it more robust to outliers and missing data.

3D volumetric tracking. While many visual tracking techniques employ skeletons [62, 148] or surface-based representations [6, 84], volume-based representations have also been proposed to address various issues. On one hand, topology changes or online surface reconstructions are better handled if surfaces are implicitly represented in volumes as *e.g.* truncated signed distance field (TSDF) [58, 105, 104], with high memory cost due to regular grids storing empty space information. On the other hand, volumetric techniques have also been devised for robustness in long term tracking, as a way to alleviate the so-called *candy-wrapper artifacts*, namely, collapsing surfaces in animations. Without explicitly tessellating surface interiors, Zhou *et al.* [161] introduce internal nodes to construct a volumetric graph and preserve the volumes by enforcing Laplacian constraints among them. Instead, Budd *et al.* [31] and De Aguiar *et al.* [47] perform a constrained tetrahedralization on surfaces to create interior edges. Allain *et al.* [5] generate internal points by CVT decomposition and thereby propose a generative tracking strategy that yields high quality performance. These techniques are nevertheless based on ICP-variants, whereas we aim at detecting associations discriminatively.

5.3 Skeleton

The patch-based deformation framework evolves a surface with a low dimensional set of parameters. It is not limited to humanoid surfaces, holding the possibility to handle complex scenes. Nonetheless, in the context of human motion tracking, many authors use an intrinsic data structure to guide the deformation in an even lower dimensional space [61, 148], followed by a shape refinement stage. This structure often resembles the skeleton of human bodies, and each node actually corresponds to one human body joint. Due to this anatomical meanings, an articulated skeletal structure is usually preferred than the 3D surface shape in many applications. Thus, in this section, we extend the pure-surface framework so that it retains the generality of surface-based methods, while providing skeletal poses as *side products*.

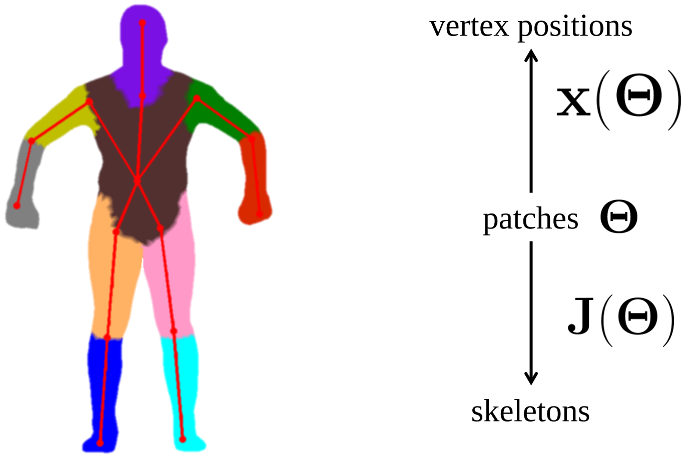
5.3.1 Initialization and Pose Parametrization

Our skeleton is a hierarchical tree structure consisting of J nodes (joints). It has to be placed properly inside the mesh such that the root is close to the pelvis of the body, and each vertex v is assigned to a branch-node joint, as in Fig. 5.2(a). This *rigging* process and vertex-joint associations are automatically accomplished off-line once on the reference surface, using the software `Pinocchio` [11] prior to the tracking process.

After rigging, many authors attach a local coordinate frame on each joint and parameterize the pose as the rigid transformations of these coordinates [43, 61, 148]. Meshes are thereby controlled by these transformations based on blend skinning techniques. However, surface deformation is by nature high dimensional and difficult to be fully characterized by only few rigid transformations. For humanoid surfaces, typically $K \approx 150$ yields plausible surface shapes, while J is usually less than 20. We therefore advocate for the reverse strategy: guiding the skeleton by the surface deformation, *i.e.* *inverse skinning*. Formally, the pose is represented as a set of positions for each joint j: $\mathbf{J} = \{\mathrm{x}_j\}_{j=1}^{J}$ and is parameterized as a function of shapes, $\mathbf{J}(\mathbf{\Theta})$. As illustrated in Fig. 5.2(b), patch transformations can be regarded as intermediate controlling primitives that lie in between complex vertex positions and overly simplified skeletons. Both high dimensional surface shapes and low dimensional skeletal poses are controlled by the transformations of patches. To this end, we associate each patch P_k to a joint $j(k)$ by a taking majority vote on vertex-joint assignments. Given the patch-joint associations, there are two ways to devise $\mathbf{J}(\mathbf{\Theta})$: either through bones, or directly through joints.

5.3.2 Inverse Skinning through Bones

On the reference mesh, we compute the beta-coordinate proposed by [132] in a per-patch manner. It represents the displacement between the bone and the patch center: $\boldsymbol{\beta}_k^0 = \boldsymbol{\delta}_k^0 - \mathbf{c}_k^0$, where $\boldsymbol{\delta}_k^0$ is the linear combination of the positions of joint $j(k)$ and its child:


vertex positions
$\mathbf{X}(\boldsymbol{\Theta})$
patches $\boldsymbol{\Theta}$
$\mathbf{J}(\boldsymbol{\Theta})$
skeletons

(a) rigged ref. surface **(b) controlling structure**

Figure 5.2: **Our skeleton (a) and deformation-controlling structure (b)**. Vertices associated to the same joint have the same color.

$$\boldsymbol{\delta}_k^0 = \gamma_k \mathbf{x}_j^0 + (1 - \gamma_k)\mathbf{x}_{\text{child}(j)}^0. \tag{5.1}$$

γ is chosen such that $\boldsymbol{\beta}$ is orthogonal to the bone, as shown in Fig. 5.3(a). We drop the dependency notation (k) in order to keep equations uncluttered.

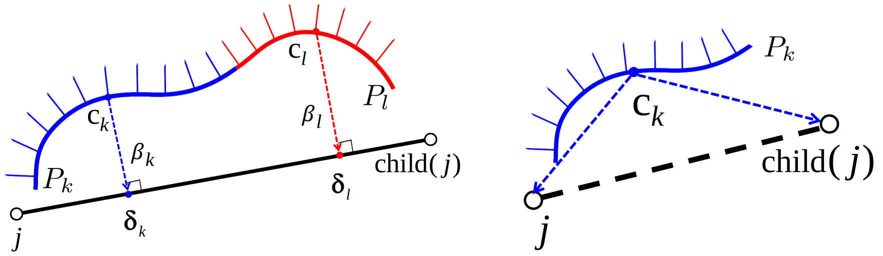
By preventing $\boldsymbol{\beta}$ from varying after deformation, we encourage bones to follow their corresponding patches. More specifically, if a patch undergoes a rigid transformation $\mathbf{T}_k$ during tracking, it predicts the new $\boldsymbol{\delta}_k$ as $\mathbf{T}_k(\boldsymbol{\delta}_k^0) = \mathbf{c}_k + \mathbf{R}_k\boldsymbol{\beta}_k^0$. Substituting into Eq. 5.1, one can formulate the new position of a bone as:

$$\gamma_k \mathbf{x}_j + (1 - \gamma_k)\mathbf{x}_{\text{child}(j)} = \mathbf{c}_k + \mathbf{R}_k\boldsymbol{\beta}_k^0. \tag{5.2}$$

Here γ stays fixed to prevent bones from sliding along the surface. Eq. 5.2 shows the linear relation between the transformation of a patch and the location of its associated joint. Since there are two unknown variables, $\mathbf{x}_j$ and $\mathbf{x}_{\text{child}(j)}$, we need at least two equations (*i.e.* two patches) to determine the new position of one bone. In practice, we stack this linear relation for every patch and form a linear system:

$$\underbrace{\begin{bmatrix} & \vdots & & \vdots & \\ \ldots & \gamma_k \mathbf{I} & \ldots & (1-\gamma_k)\mathbf{I} & \ldots \\ & \vdots & & \vdots & \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} \vdots \\ \mathbf{x}_j \\ \vdots \\ \mathbf{x}_{\text{child}(j)} \\ \vdots \end{bmatrix}}_{\mathbf{j}} = \underbrace{\begin{bmatrix} \vdots \\ \mathbf{c}_k + \mathbf{R}_k\boldsymbol{\beta}_k^0 \\ \vdots \end{bmatrix}}_{\boldsymbol{\Delta}}, \tag{5.3}$$

where $\mathbf{B}$ is a $3K \times 3J$ matrix, $\mathbf{j}$ is a $3J \times 1$ vector containing the positions of all joints, and $\boldsymbol{\Delta}$ is $3K \times 1$ vector containing all $\mathbf{T}_k(\boldsymbol{\delta}_k^0)$. In general, K is larger



(a) bone-based

(b) joint-based

Figure 5.3: **Two inverse skinning strategies**. In (b), the bone is shown in dashed line because there is no explicit concept of bone.

than J, so Eq. 5.3 is an over-determined system whose optimal solution can be obtained via pseudoinverse [89]:

$$\mathbf{j} = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{\Delta}. \tag{5.4}$$

Eq. 5.4 demonstrates that the pose of the subject can be parameterized as the function of the shape parameter (encoded in $\mathbf{\Delta}$), namely, $\mathbf{J}(\mathbf{\Theta})$. Thus, each time a when humanoid surface is deformed, the pose of the skeleton can be computed via Eq. 5.4 accordingly. Since pseudo-inverse solves a linear system in the least-square sense, one can also formulate an equivalent energy term:

$$E_{bone}(\mathbf{\Theta}, \mathbf{J}) = \sum_{k=1}^{K} \kappa_k \left\| \mathbf{T}_k(\boldsymbol{\delta}_k^0) - \boldsymbol{\delta}_k' \right\|_2^2, \tag{5.5}$$

where $\boldsymbol{\delta}_k' = \gamma_k \mathbf{x}_j + (1-\gamma_k)\mathbf{x}_{\text{child}(j)}$. It simply enforces the right-hand and the left-hand side of Eq. 5.2 to be consistent. This way we can also weight the contribution of each patch differently with κ_k.

5.3.3 Inverse Skinning through Joints

An alternative way to guide the skeleton is more straightforward: predicting the positions of joints by every associated patch, as illustrated in Fig. 5.3(b). When a patch k moves to a new position $\mathbf{c}_k$ with rotation $\mathbf{R}_k$, it assumes the associated joint moves together:

$$\mathbf{x}_k(j) = \mathbf{R}_k(\mathbf{x}_j^0 - \mathbf{c}_k^0) + \mathbf{c}_k. \tag{5.6}$$

The position of each joint is then recovered by linearly blending the predictions from different patches:

$$\mathbf{x}_j = \sum_k \tau_k \mathbf{x}_k(j), \tag{5.7}$$

where the weights τ_k are determined similarly as α in Eq. 2.2. This approach fits naturally into the patch-based deformation framework, and it requires at least only one patch to predict two joint positions.

It is informative to combine Eqs. 5.6 and 5.7 together in one formulation, as it emphasizes the difference of our approach from the others:

$$\mathbf{x}_j = \sum_k \tau_k \mathbf{T}_k (\mathbf{T}_k^0)^{-1} \mathbf{x}_j^0. \tag{5.8}$$

Here $(\mathbf{T}_k^0)^{-1}$ means $(\mathbf{x}_j^0 - \mathbf{c}_k^0)$ in Eq. 5.6, and $\mathbf{T}_k$ means rotating with $\mathbf{R}_k$ and moving to the new center $\mathbf{c}_k$. Eq. 5.8 mirrors the linear-blend skinning formulation:

$$\mathbf{x}_v = \sum_j w_j \mathbf{T}_j (\mathbf{T}_j^0)^{-1} \mathbf{x}_v^0, \tag{5.9}$$

where one first represents the vertices in local bone coordinate frames, applies new transformations, and blend the predictions from relevant joints. We clearly see that Eqs. 5.8 and 5.9 share the same mathematical computations, only the opposite operands. We argue that inferring skeletons from surfaces because it is a logical way to determine low dimensional representations (skeletal poses) from high dimensional deformations (surface shapes), instead of doing the other way around with unrealistic rigid-body-part assumptions.

Eq. 5.7 provides the second way to parameterize the pose $\mathbf{J}$ as a linear function of the shape parameter $\mathbf{\Theta}$, encoded in $\mathbf{x}_k(j)$. Let J' denotes the patches connected to joint j or the parent of j. Similar to the first approach, we also formulate an equivalent energy term:

$$E_{joint}(\mathbf{\Theta}, \mathbf{J}) = \sum_{j=1}^{J'} \sum_{k \in \mathcal{N}_j} \tau_{jk} \left\| \mathbf{x}_j - \mathbf{x}_k(j) \right\|_2^2. \tag{5.10}$$

5.3.4 Optimization

Given the two different ways to obtain the pose $\mathbf{J}$ from the shape $\mathbf{\Theta}$ (Eqs. 5.4 and 5.7), one can, during tracking, first get the optimal $\mathbf{\Theta}$ via Eq. 2.4 and then compute the pose $\mathbf{J}$ as a post-processing step. The alternative, as suggested by [132], is to simultaneously estimate the two, with Eq. 2.4 is augmented into:

$$\underset{\mathbf{\Theta}, \mathbf{J}}{\operatorname{argmin}} \; \lambda_r E_r(\mathbf{\Theta}) + \lambda_s E_{skl}(\mathbf{\Theta}, \mathbf{J}) + E_{data}\left(\mathbf{X}(\mathbf{\Theta})\right). \tag{5.11}$$

E_{skl} is either E_{bone} or E_{joint}, and λ_s is the balancing weight. This is again a non-linear least-squares problem. The quadratic approximation in Sect.. 2.2.2 applies here as well to both E_{bone} or E_{joint}, so the numerical considerations basically remain unchanged; only the approximated Hessian matrix $\mathbf{G}^\top \mathbf{G}$ becomes a $(6K + 3J) \times (6K + 3J)$ matrix. Although solving Eq. 5.11 recovers poses and shapes at once, we anyway stress that, due to the difference in the numeric scales, poses $\mathbf{J}$ behave like side products of shapes $\mathbf{\Theta}$. As discussed later in Sect. 5.6.1, it has negligible effects on shape deformations.

5.4 Simultaneous Surface and Skeleton Tracking

As discussed in Sects. 5.1 and 5.2, we deal with data-driven mesh deformation and cast the problem as a geometric registration of 3D point sets. For this purpose, the data term E_{data} in Eq. 5.11 has to be designed carefully. In [35], Cagniart *et al.* consider the optimization problem of Eq. 2.4 from Bayesian's point of view and model the data term (likelihood function) as Gaussian Mixture Model (GMM). The optimization process then behalves like the well-known Expectation-Maximization algorithm. Since our Eq. 5.11 is the extension of Eq. 2.4, it also fits naturally into the probabilistic framework, which amounts to the Bayesian network below.

We aim at estimating the shape $\mathbf{\Theta}$ of the surface and the pose $\mathbf{J}$ of the skeleton simultaneously. In a Bayesian context, this means that given a set of observed 3D points, the estimation of shape and pose is achieved by maximizing the a posteriori (MAP) probability:

$$\max_{\mathbf{\Theta},\mathbf{J}} P(\mathbf{\Theta},\mathbf{J}|\mathbf{Y}). \tag{5.12}$$

Considering $P(\mathbf{Y})$ as a constant, maximizing Eq. 5.12 is equivalent to maximize the joint distribution, which can be decomposed as follows:

$$\max_{\mathbf{\Theta},\mathbf{J}} P(\mathbf{Y},\mathbf{\Theta},\mathbf{J}) = P(\mathbf{Y}|\mathbf{\Theta},\mathbf{J}) \cdot P(\mathbf{J},\mathbf{\Theta}). \tag{5.13}$$

5.4.1 Bayesian Network Model

We employ two assumptions which further simplify Eq. 5.13:

1. surface-based approach:

$$P(\mathbf{J},\mathbf{\Theta}) = P(\mathbf{J}|\mathbf{\Theta}) \cdot P(\mathbf{\Theta}). \tag{5.14}$$

2. conditional independence between $\mathbf{J}$ and $\mathbf{Y}$:

$$P(\mathbf{Y}|\mathbf{\Theta},\mathbf{J}) = P(\mathbf{Y}|\mathbf{\Theta}). \tag{5.15}$$

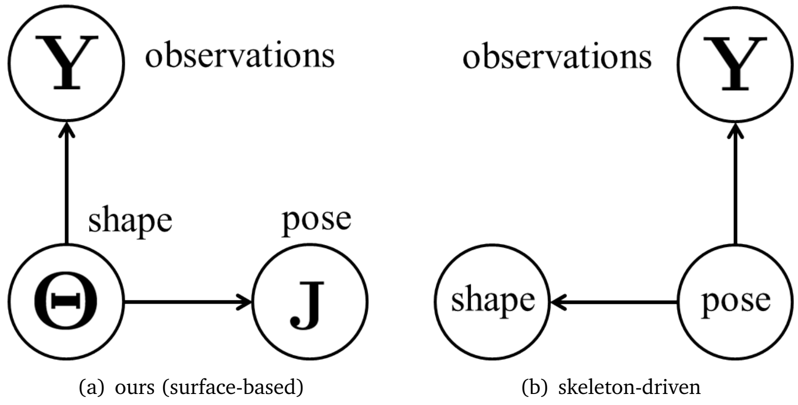
The first assumption comes from the fact that we rely on the shape parameter $\mathbf{\Theta}$ to determine the pose parameter $\mathbf{J}$, as described in Sect. 5.3. $P(\mathbf{J},\mathbf{\Theta})$ is thus factorized as in Eq. 5.14, not $P(\mathbf{\Theta}|\mathbf{J}) \cdot P(\mathbf{J})$. Here, $P(\mathbf{J}|\mathbf{\Theta})$ represents the probability of the skeleton pose given the shape, and $P(\mathbf{\Theta})$ is the prior knowledge on possible shape deformations. Secondly, we assume that $\mathbf{J}$ is conditionally independent of $\mathbf{Y}$ given $\mathbf{\Theta}$, *i.e.* $\mathbf{Y} \perp\!\!\!\perp \mathbf{J}|\mathbf{\Theta}$. It is a reasonable assumption, since from the input data perspective, one usually observes only the surface of human bodies (shape) rather than the anatomical structure (skeleton). It makes sense that when conditioning on the shape, the observations and the skeleton can no longer influence each other. Eq. 5.15 holds as the property of conditional independence [18], and $P(\mathbf{Y}|\mathbf{\Theta})$ is considered as the likelihood between observations and the shape.

Based on Eqs. 5.14 and 5.15, the joint distribution is:

$$P(\mathbf{Y},\mathbf{\Theta},\mathbf{J}) = P(\mathbf{Y}|\mathbf{\Theta}) \cdot P(\mathbf{J}|\mathbf{\Theta}) \cdot P(\mathbf{\Theta}). \tag{5.16}$$

Alternatively, one can decompose the joint distribution as follows:

$$P(\mathbf{Y},\mathbf{\Theta},\mathbf{J}) = P(\mathbf{Y},\mathbf{J}|\mathbf{\Theta}) \cdot P(\mathbf{\Theta}), \tag{5.17}$$



(a) ours (surface-based) (b) skeleton-driven

Figure 5.4: **Directed graphical models of two strategies**.

and see directly that the conditional independence between $\mathbf{J}$ and $\mathbf{Y}$ lead us to Eq. 5.16 by definition. Note that Eq. 5.16 is actually a Bayesian network model, whose directed graph is illustrated in Fig. 5.4(a). From the graph we see straightforwardly that the joint distribution is decomposed into three different terms as in Eq. 5.16, and the property of conditional independence is also demonstrated in the graph [18]. The figure also shows, in (b), the dependence graph corresponding to skeleton-based motion parameterizations. Next we proceed with each term in Eq. 5.16.

5.4.2 Shape Prior and Pose Posteriors

In the absence of knowledge on the nature of the shape, we model a probability distribution over the range of shape deformations by seeding patches on a reference surface and making the approximation:

$$P(\mathbf{\Theta}) \propto e^{-\lambda_r E_r(\mathbf{\Theta})}, \tag{5.18}$$

where $E_r(\mathbf{\Theta})$ is the rigidity energy defined in Eq. 2.3. This energy emulates elastic behavior with respect to the patched reference mesh. Because our patching approach infers the topology of the object from the vertex connectivity, this reference mesh has to be topologically suitable, that is it has to be split wherever the surface might split during the sequence. See Sect. 5.6.1 for more discussion on this aspect.

The pose posteriors measure the probability of a pose given a certain shape, approximated as:

$$P(\mathbf{J}|\mathbf{\Theta}) \propto e^{-\lambda_s E_{skl}(\mathbf{\Theta},\mathbf{J})}, \tag{5.19}$$

where $E_{skl}(\mathbf{\Theta}, \mathbf{J})$ is the skeleton energy in Eq. 5.11. This approximation assumes that the connectivity between the skeleton and the surface is stable, or say, the patch-joint association $j(k)$ does not change during tracking, which usually holds for human subjects.

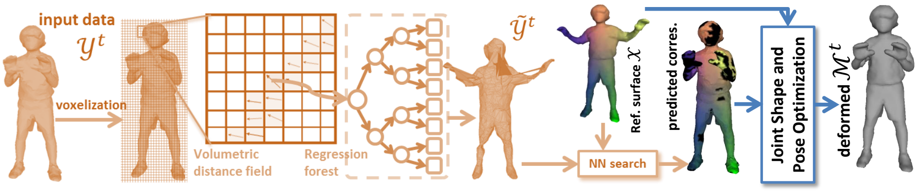


Figure 5.5: **Our full 3D tracking-by-detection pipeline**. The correspondence prediction part (Fig. 3.2) is colored in orange. We feed the predicted data-model associations into the simultaneous optimization framework in this chapter and deform the reference surface accordingly.

5.4.3 Likelihood

Recall that our goal is to estimate deformation parameter $\mathbf{\Theta}$, whose resulting $\mathbf{X}(\mathbf{\Theta})$ best explains $\mathbf{Y}$. The goodness of explanation is accounted for by the data term E_{data}, or in the perspective of probability, the likelihood function $P(\mathbf{Y}|\mathbf{\Theta})$:

$$P(\mathbf{Y}|\mathbf{\Theta}) \propto e^{-E_{data}(\mathbf{X}(\mathbf{\Theta}))}. \tag{5.20}$$

Omitting the vertex position $\mathbf{X}$, in the following discussion, we denote data term only as $E_{data}(\mathbf{\Theta})$ to keep notations uncluttered and highlight the dependency.

Generative EM-based Data Term

Cagniart *et al.* [35] adopt probabilistic ICP in patch-based deformation framework. Instead of a deterministic correspondence, each target point $\mathbf{y}_i$ has a soft assignment w_{ik} to every patch P_k. They define a data term as:

$$E_{data}(\mathbf{\Theta}) = \sum_{i=1}^{O} \sum_{k=1}^{K+1} w_{ik} \|\mathbf{y}_i - \mathbf{x}(v_{ik})\|_2^2. \tag{5.21}$$

They also introduce one additional outlier class which is modeled as a constant uniform distribution (therefore $K+1$). Since w_{ik} are obtained by simple nearest neighbor search, minimizing Eq. 5.21 directly with Gauss-Newton solver might lead to sub-optimal outcome due to erroneous associations. Instead, they perform just one Gauss-Newton iteration and update w_{ik} accordingly. The alternating process be viewed as Expectation-Maximization algorithm. In E-step, soft assignments w_{ik} are computed and in M-step, the energy is minimized in terms of model parameters $\mathbf{\Theta}$.

Discriminative Correspondences

Minimizing Eq. 5.21 in an EM manner is able to refine correspondences $\mathcal{C}$ and $\mathbf{\Theta}$ progressively. Still, it needs close initializations in order to reach the convergence, *i.e.* $\mathbf{Y}$ and the current $\mathbf{X}(\mathbf{\Theta})$ are supposed to be close enough to each other, which does not always hold. On the other hand, heavily relying on the previous status $\mathbf{X}(\mathbf{\Theta}^{t-1})$ also make the whole tracking pipeline apt to drifting.

Providing quasi frame-independent discriminative data-model associations, the TSDT-forest framework in Chapter 3 is devised to address this problem. Given correspondences $\mathcal{C}$ from the forest (Eq. 3.7), a data term is formulated as:

$$E_{data}(\mathbf{\Theta};\mathcal{C}) = \sum_{(i,p)\in\mathcal{C}} w_{ip}\|\mathbf{y}_i - \mathbf{x}_p(\mathbf{\Theta})\|_2^2, \tag{5.22}$$

Given an input $\mathbf{Y}$ the regression forest returns a fixed response $\tilde{\mathbf{Y}}$, and hence a fixed $\mathcal{C}$. We therefore apply standard Gauss-Newton method to directly find the minimizer of Eq. 5.22. There is no need to alternate between optimizing $\mathcal{C}$ and $\mathbf{\Theta}$ as in EM-ICP [35]. Note anyway that refining $\mathcal{C}$ afterwards with ICP is always possible. Two techniques are actually complementary to each other, leading us to the '*discriminatively initialized, generatively refined*' paradigm. In this case, the regression forest provides better initializations than using last frame results, reducing the number of needed ICP-iterations. Later on our experiments show that, using our results as initializations rather than the results in the previous frame, methods like [35] yield comparable accuracy but less iterations to converge.

5.5 Volumetric Tracking

As the surface-based discriminative correspondences from chapter 3 are fed into the joint shape and pose optimization framework introduced above, so are the volumetric discriminative associations from chapter 4 fed into a volumetric deformation model [5]. Although it is not the contribution of this thesis, we anyway briefly explain in this section on the basic principles of [5] and how to apply the predictions from chapter 4 to track a sequence of temporally inconsistent observations.

In short, unlike the current state-of-the-art 3D shape tracking methods [6] that employ non-rigid ICP algorithms to search extensively over all possible associations, we directly use the correspondence pair (i,p) detected by the forest as initializations. This results in a faster shape-pose estimation. Although we adopt the CVT-based deformation framework proposed in [5], the approach described here can easily be adapted to other ICP variants.

5.5.1 Bayesian Tracking

Bayesian tracking such as [6] consists in maximizing the *a posteriori* probability $P(\mathbf{\Theta}|\mathbf{Y})$ of the deformation parameters $\mathbf{\Theta}$ given the observations $\mathbf{Y}$. It can be further simplified as $P(\mathbf{\Theta}|\mathbf{Y}) \propto P(\mathbf{\Theta},\mathbf{Y}) = P(\mathbf{\Theta})\cdot P(\mathbf{Y}|\mathbf{\Theta})$, where the deformation prior $P(\mathbf{\Theta})$ discourages the implausible shape poses [5] and the likelihood term $P(\mathbf{Y}|\mathbf{\Theta})$ expresses the compatibility between the observations and the shape estimate $\mathbf{\Theta}$. Since maximizing a probability $P(\cdot)$ is equivalent to minimizing $-\log P$, it leads us to the following problem:

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmin}}\left(-\log P(\mathbf{\Theta}) - \log P(\mathbf{Y}|\mathbf{\Theta})\right). \tag{5.23}$$

In EM-ICP algorithms [35], the conditional likelihood $P(\mathbf{Y}|\mathbf{\Theta})$ is expressed by introducing a set of latent selection variables $\{k_i\}_i$ that explicitly associate the cell k_i of the deformed template model to the observed cell i.The prior on the latent association variables is usually uniform, which means that an observed point can be associated to any template point with the same probability. This leads to a long exhaustive search among all possible associations and produces a high number of residuals, slowing down the EM-ICP algorithms. Moreover, it is the source of wrong associations that guides the optimization to suboptimal local minimum.

EM-ICP with forest predictions

With a small number of possible associations provided by forests, our algorithm averts the need for an exhaustive search, and therefore highly decreases the running time of each optimization iteration. Moreover, it removes a lot of wrong association hypotheses. We integrate the predictions from the forest as a prior on the selection variable k. The selection variable k_i (for the observed cell i) follows a probability distribution where only the cell predicted by the forest has a non-zero probability.

Usually the forest outputs only one prediction per cell, which is at the mode with higher weight resulting from the mean-shift algorithm. However, because of the symmetry, the good match is often not the mode with highest weight. Thus, it makes sense to consider several modes instead of one in the prediction phase. The robust scheme described in the next section will usually select the good one.

5.6 Experiments

So far we have introduced our full 3D tracking-by-detection pipeline in two distinct representations, which are surface-based (Chapter 3 + Section 5.4) and volume-based (Chapter 4 + Section 5.5). Since we propose new methods in different stages of the pipeline, we employ three experiment settings to highlight their advantages respectively:

1. **Generative ICP-based surface tracking (Sect. 5.6.1)**. This setting is to demonstrate the benefit of simultaneous skeleton-and-surface optimization framework in Sect. 5.4. We assume that the template is closely aligned to the first frame and apply Eq. 5.21 as the data term.

2. **Discriminative surface-based tracking (Sect. 5.6.2)**. This is our full surface-based tracking-by-detection setting. For each frame, the template is initialized with correspondences $\mathcal{C}$ predicted by forests in Chapter 3 and we consider Eq. 5.22 as the data term. Results are refined by the ICP above.

3. **Volumetric tracking-by-detection (Sect. 5.6.3)**. This is the same tracking-by-detection setting but in CVT representations. Predictions are drawn by forests in Chapter 4 and we follow Eq. 5.23 to estimate the deformations. Please refer to Table 4.2 for the setting of this experiment.

Each experimental setting considers different sets of sequences that contrast the advantages of the proposed methods most. We will summarize their profiles in the beginning of the subsections.

5.6.1 Generative ICP-based Surface Tracking

In this subsection, we present the results of our algorithm on 8 publicly available multi-view sequences, ranging from rapid motions, *e.g. free* [128] to non-rigid deformations, *e.g. Samba* [148]. We evaluate the shapes and the poses separately and analyze the results both qualitatively and quantitatively. Table 5.1 lists a overview of these sequences and the corresponding error measure. As input, we used the resulting visual hulls from a rudimentary shape from silhouette method [55]. The resulting visual hulls, although only a coarse approximation of the true shape, are enough to drive the deformation of the provided template mesh through the sequences.

Sequence	Views	Frames	Metric
Samba [148]	8	175	A
Fighting [98]	12	500	B
S4_walking [122]	4	350	C & D
Free [128]	8	200	A
Skirt [62]	8	720	A
Dance [62]	8	573	A
Wheel [62]	8	280	A
Handstand2 [62]	8	400	A

Table 5.1: **Sequences used for evaluation in Sect. 5.6.1**. Several error measures are applied depending on the provided ground truths. A: silhouette overlap error. B: distances in $\mathbb{R}^3$ between markers and associated vertices. C: 3D error on joints positions. D: 2D pixel error on re-projected joint positions. A and B are metrics for shapes while C and D are for poses.

Evaluation on human shapes

First we evaluate our algorithm on shape estimation. The metric is either silhouette overlap error in 2D, or distance to markers in 3D. For sequences evaluated with 2D error, we follow [62, 148] who directly use silhouettes as input data and minimizes the re-projection error as a refinement step. This procedure relies on the same optimization framework defined in Sect. 5.3.4, uses considerably small patches and minimizes the residual error in silhouette overlap. The gradient of this energy is approximated by guessing from the current pose estimation which vertices are on occluding contours and pulls their projections towards the observed contours in the images. As shown in this section, compared with the state-of-the-art approaches, our algorithm provides satisfactory or even better results in both metrics.

We used the multi-view image data made public by MIT CSAIL group [148] and by MPI-Informatik [62] to run a simple shape from silhouette algorithm [55].

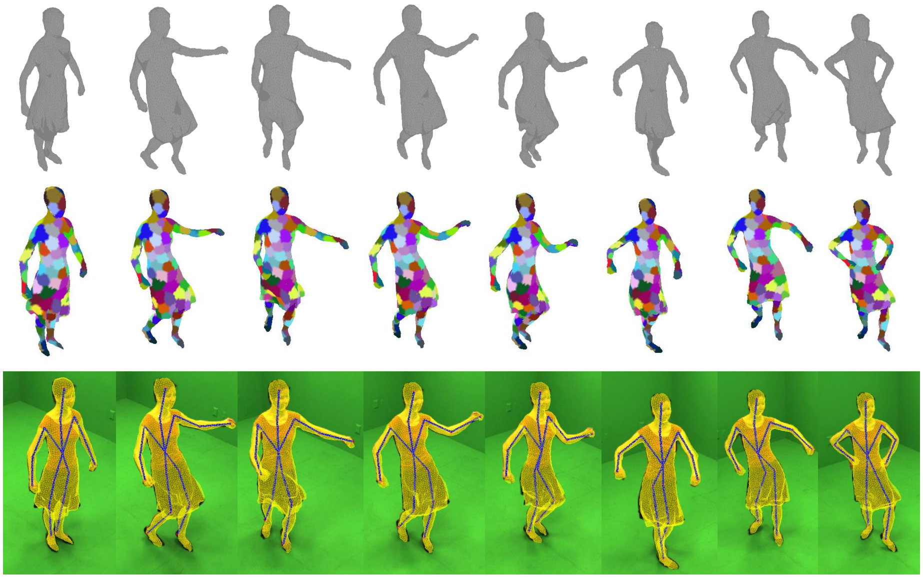


Figure 5.6: **Results on *Samba* sequence**. Input visual hulls (1st row) and results (2nd and 3rd rows) on frame $10 - 38 - 68 - 78 - 84 - 100 - 118 - 124$ of *Samba* sequence. Here the skeletons are obtained via E_{joint}. Our approach yields visually convincing results on the tracking of a skirt.

In *Samba* sequence, skirts are difficult to handle for methods deforming a reference mesh as the interpolated surface between the bottom of the skirt and the legs has to undergo severe compression and stretching. We show in Fig. 5.6 that our approaches produces visually convincing results. We report better results than [62] and [132] in Table 5.2.

	Wheel	*Dance*	*Skirt*	*Handstand2*
ours	**3961**	**3780**	**3413**	**4573**
Gall *et al.* [62]	4168	5098	3678	5028
Straka *et al.* [132]	4300	4100	4100	4900

Table 5.2: **Silhouette overlap errors (pix) for *Wheel*, *Dance*, and *Handstand2* sequences**. Image resolution: 1004×1004.

Temporal Consistency. In addition to silhouette overlap error which measures the discrepancies on 2D images, we also evaluate shapes in 3D. In *Fighting* sequence, the raw marker positions are provided for almost 500 frames. We follow [98] who associated each marker to the closet vertex on the reference surface at frame zero and obtain $7.98mm$ initial distance. After tracking the whole sequence, the average distance between markers and the corresponding vertices becomes $38.49mm \pm 32.39mm$, which is reasonably small. Note that this includes measurement errors introduced by the marker-based system. It demonstrates the strength of our method on recovering temporal consistent

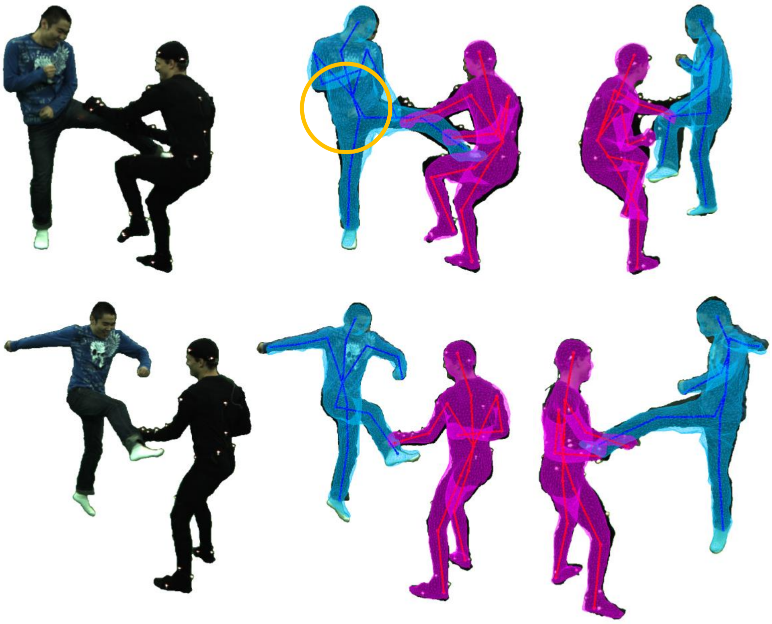


Figure 5.7: **Exemplar visual results of *Fighting* sequence**. Frames 190 and 250 in two views. Skeletons are obtained simultaneously via E_{joint}. Our approach applies well to multiple human subjects. Yellow circle: tracking is sometimes affected by close interaction, but is soon recovered.

meshes. The average time per frame for this sequence is 7 seconds. Compared to [98] who attain more accurate results ($29.61mm \pm 25.50mm$), but require several minutes per frame for tracking, and to [130] which is fast (around 6 frames per second), but less accurate ($44.93mm \pm 27.16mm$), our approach certainly offers a good compromise between performance and accuracy. In Fig. 5.7, we overlay the estimated meshes and skeletons on images. We see that close interaction between subjects does not effect the results too much, which demonstrates that our method generalizes well to multiple humans.

Evaluation on human poses

It is also crucial to evaluate skeletal poses in the context of human tracking. The widely-used benchmark `HumanEva-II` [122] is challenging for 4D modeling because it contains too few cameras. We anyway test our algorithm on *S4* sequence *walking* section. Frames $298 - 335$ are excluded due to the reported ground truth corruption. For the remaining frames, our method presents errors around $65mm$ in average (Table 5.3). According to [123], errors smaller than $80mm$ typically correspond to correct poses, which verifies the reliability of our method in terms of human pose estimation. Compared with [43] who use visual hulls as well and report $80mm \pm 13mm$ errors, our approach is certainly more accurate and stable. Note that their approach is articulated ICP where deformations are guided by the underlying skeleton. This confirms the advantage

Inverse Skinning	Bone (simul.)	Bone (post)	Joint (simul.)	Joint (post)
3D error (*mm*)	65.48 ± 7.74	65.78 ± 7.52	$\mathbf{64.45 \pm 7.14}$	65.06 ± 7.10
Cam. 1 (pix.)	9.28 ± 1.31	9.31 ± 1.31	$\mathbf{9.14 \pm 1.19}$	9.21 ± 1.23
Cam. 2 (pix.)	8.32 ± 1.34	8.38 ± 1.29	$\mathbf{8.26 \pm 1.23}$	8.36 ± 1.22
Cam. 3 (pix.)	8.33 ± 1.36	8.35 ± 1.36	$\mathbf{8.21 \pm 1.31}$	8.26 ± 1.34
Cam. 4 (pix.)	9.99 ± 2.43	10.08 ± 2.39	$\mathbf{9.96 \pm 2.27}$	10.07 ± 2.29

Table 5.3: **3D error in millimeter and 2D errors in pixel number for *S4_walking*.** Image resolution: 656×490.

of our inverse skinning strategy over conventional skeleton-based methods: when observations are noisy, a generic but robust surface-based approach offers better estimates on poses than approaches that constrain the search space with object-specific intrinsic deformation model. Still, we would like to stress that our goal is to track arbitrary objects and simultaneously provide a low-dimensional motion parametrization (which are skeletal poses) when the subjects are humans. We do not aim to estimate precise human joint locations since modeling a real human joint as a single 3D point is anyway an over-simplified assumption. The numerical error here is only a coarse measure of how well the pose is estimated. Further optimizing on this error does not necessarily improve the estimation.

Lastly, more qualitative results are shown in Fig. 5.8. Even in the challenging *Free* sequence, the skeletal poses are tracked properly. Our method is able to produce convincing results in terms of both shape and pose.

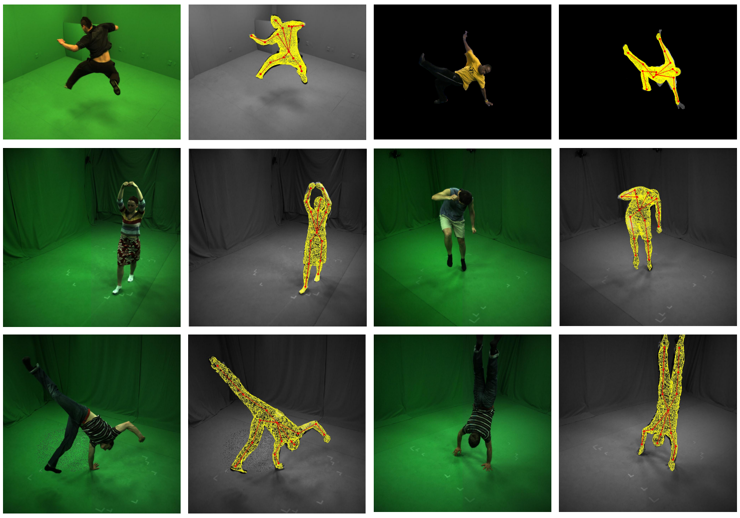


Figure 5.8: **Example frames of input videos and overlaid results.**

Discussions

The role of E_{skl} in optimization. In Sect. 5.3.4 and 5.4.1, we emphasize that skeletal poses are side products of surface shapes. We verify this numerically by reporting again the raw silhouette overlap errors on *Crane* sequence in Fig. 5.9.

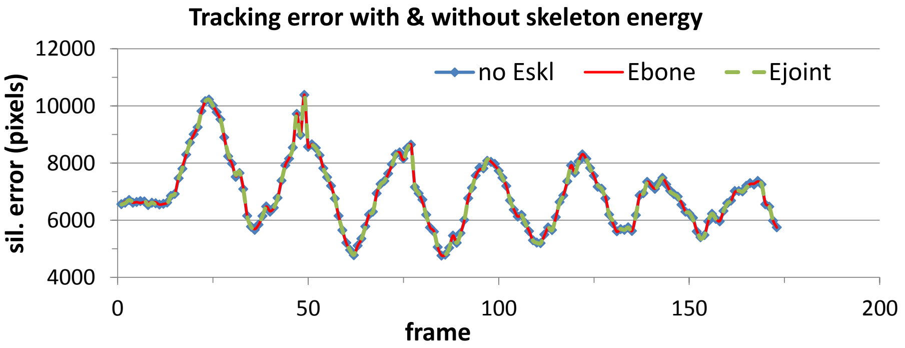


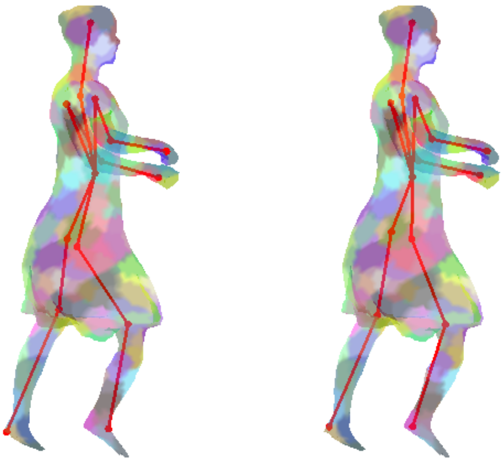
Figure 5.9: **Sil. overlap error with different skeleton energies**. Three strategies yield almost same errors in terms of shape. Image resolution: 1600×1200.

Without the skeleton energy terms E_{skl} ($\lambda_s = 0$ in Eq. 5.11), the averaged error of surface shapes is 6938.73, whereas E_{bone} and E_{joint} get 6939.65 and 6938.69, respectively. Such a negligible difference confirms that in our inference framework, skeletal poses contribute little to surface shapes. E_{skl} plays the role as an augmentation term that recovers poses $\mathbf{J}$ in no time during optimizations. Recall that in Table 5.3, $\lambda_s = 1$ already yields good numeric solutions for poses $\mathbf{J}$. In Eq. 5.11, the magnitude of E_{skl} is therefore relatively small compared to the other two terms and hence contribute little to the gradient of shapes $\mathbf{\Theta}$.

E_{bone} vs. E_{joint}. To further select between E_{bone} and E_{joint}, we mark two observations in Table 5.3:

1. Simultaneous optimization yields equivalent or even slightly better results than post processing, which confirms our descriptions in Sect. 5.3.
2. Inverse skinning via joints attains consistently better and more stable results than via bones.

It is worth a closer look to contrast two approaches. We particularly choose *Samba* sequence which contains both rigid (arms) and non-rigid deformations (skirts). Results of frame 70 are shown in Fig. 5.10. Firstly, two surface shapes look visually the same, again confirming the above discussions that E_{skl} contribute little to the gradient of the shape $\mathbf{\Theta}$. To analyze how different deformations affect the poses, one can see that two strategies present similar poses on the arms but different behaviors on the legs. Since there is no ground truth for poses provided in this sequence, we check how the bone lengths of six body parts vary during tracking: the right hip bone (RHip), the right upper and lower legs and arms (RULeg, RLLeg, RUArm, and RLArm), and the torso spine. The results are shown in Table 5.4 and Fig. 5.11. In general, bone lengths in E_{joint} strategy are more correct (smaller bias) and more stable (smaller oscillations). For rigid body parts like arms and torso, variations are rather small and the differences are still negligible. For non-rigid body parts *e.g.* RULeg, the margins become significant (see the fluctuations of red and green solid curves in Fig. 5.11). We therefore conclude that E_{joint} is a more effective way to realize inverse skinning.



(a) E_{bone}

(b) E_{joint}

Figure 5.10: **Visual comparison of two different inverse skinning strategies**. Results are from frame 70 in *Samba* sequence.

Body part	Initial bone length (m)	Bias (mm)		Standard deviation (mm)	
		E_{bone}	E_{joint}	E_{bone}	E_{joint}
RHip	0.224	12.60	**1.75**	20.91	**7.01**
RULeg	0.377	7.13	**4.84**	27.92	**9.98**
RLLeg	0.454	19.16	**6.45**	21.97	**13.99**
RUArm	0.268	2.49	**1.74**	14.48	**9.18**
RLArm	0.249	5.90	**4.26**	9.33	**4.29**
Torso	0.322	**5.47**	7.22	6.40	**5.19**

Table 5.4: **Bone length variation on** 6 **body parts of *Samba* sequence**. Bias means the absolute difference between average length during tracking and its original length.

Comments on the approach. One fundamental premise of our approach is that no drastic variations are presented in the topology. The reference frame has to be topologically suitable, that is, it has to be split wherever the surface might split in the sequence. In other words, a moderate amount of disappearing geometry, *e.g.* self-intersections or close interactions of different subjects as in Fig. 5.7 can be handled, but it is highly likely to fail when any creation of new geometry occurs. Moreover, when the arms and body are merged, the local density of points in the input data does not double, which clearly indicates that the data generation by two overlapping patches on the arm and the body is not independent. In that sense, our method, which implicitly assume *independent and identically distributed* (i.i.d.) observed data, are only approximations. Since both E_{bone} and E_{joint} behave like auxiliary energy terms to compute poses from the given shapes, we set $\lambda_s = 1$ throughout all experiments.

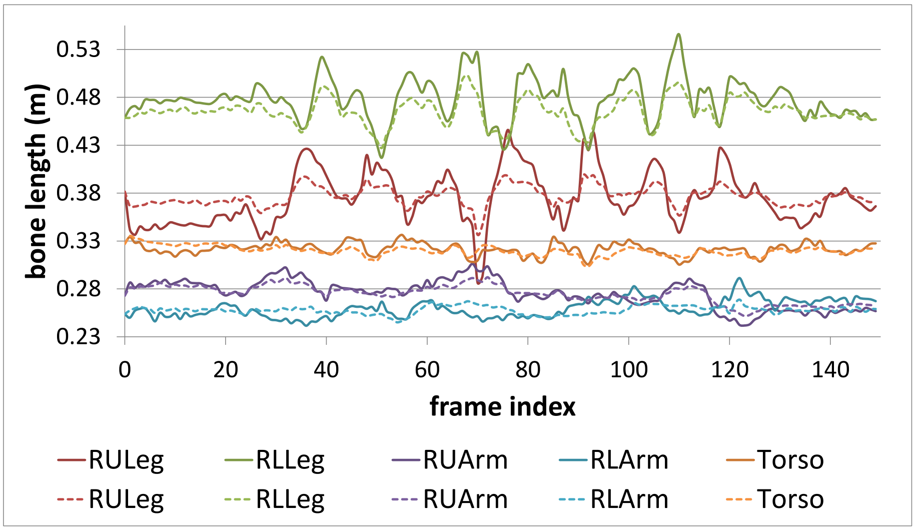


Figure 5.11: **Bone length variation on 6 body parts of *Samba* sequence.** Solid curve: E_{bone}. Dotted curve: E_{joint}.

5.6.2 Initialized with Discriminative Associations

Now since the behavior of the proposed simultaneous optimization is discussed and analyzed, we take one step further to evaluate the full tracking-by-detection pipeline. Namely, we take the predicted correspondences $\mathcal{C}$ from Chapter refchapDiscSurf and consider Eq. 5.22 as the data term. The full tracking-by-detection pipeline is evaluated extensively on 9 sequences, whose profiles are summarized in Table 5.5. An individual forest is trained for each subject with up to 200 meshes, depending on the number of vertices per mesh. For S$1-3$ we train standard regression forest; for S4 & S5 we apply the adaptation in Eq. 3.6 due to the un-properly segmented chairs and tables in input data ($\alpha = 2$). Growing $T = 20$ trees to depth 25 with 15000 testing offset pairs ψ takes about 8 hours. The performance of our method is analyzed in different aspects, both qualitatively and quantitatively. We evaluate shapes with the widely-used silhouette overlap error, and evaluate poses with the 2D distances between ground truths and the projected joint locations.

Tracking without outliers. For sequences without outliers, we compare with surface-based ICP (surICP) [35] and articulated ICP (artICP) [43], both of which explain data with GMM using the Expectation-Maximization algorithm. We run an additional ICP step to reduce the errors (ours + ICP) for all testing sequences. The averaged silhouettes overlap errors are shown in Fig. 5.18(a-d). In general, our method performs much better than artICP, and yields comparable results with surICP. However, our method requires less ICP-iterations to converge, as in Table 5.7. This demonstrates that, compared with using results of previous frames as initializations, our method is capable of providing better ones. It follows that, compared with using results of previous frames as initializations, our method is

Sequence	Views	Frames	Outliers	Err. metric	Compared approaches	Subject / # Vertices
Goalkeeper [6]	48	176	-	-	-	S1 / 4980
Crane [148]	8	173	-	A	surICP [35], artICP [43]	S2 / 3407
Jumping [148]	8	149	-	A		
Handstand [148]	8	173	-	A		S3 / 3848
Bouncing [148]	8	174	-	A		
Cutting	9	81	√	A	fixOL [35], bpSVM [72], patchedOL [73]	S4 / 5211
WalkChair1	9	130	√	A		
HammerTable [73]	9	93	√	A & B	[35], [72], [73], [131] + [132]	
WalkChair2 [73]	9	148	√	A & B		S5 / 5233

Table 5.5: **Sequences used for evaluation in Sect. 5.6.2**. We apply two different error measures, depending on the provided ground truths. A: silhouette overlap errors averaged over all views. B: distances to annotated joint positions in pixels.

	ours + ICP		[131] + [132]		bpSVM [72]		patchedOL [73]		fixOL [35]
	sil. error		sil. error		sil. error		sil. error		sil. error
WalkChair1	**7432**		-		fail		8931		24976
Cutting	**4048**		-		fail		20385		fail
	sil. error	joint error	sil. error	joint error	sil. error	joint error	sil. error	joint error	sil. error
HammerTable	4019	13.3 ± 6.9	17285	64.2 ± 53.9	fail	fail	**3593**	$\mathbf{10.1 \pm 3.0}$	fail
WalkChair2	**6144**	$\mathbf{15.8 \pm 6.1}$	12219	20.6 ± 22.0	18063	24.6 ± 10.7	6803	15.9 ± 6.3	18482

Table 5.6: **Pixel overlap error of** 4 **sequences over all frames and all cameras**. Image resolution: 1000×1000.

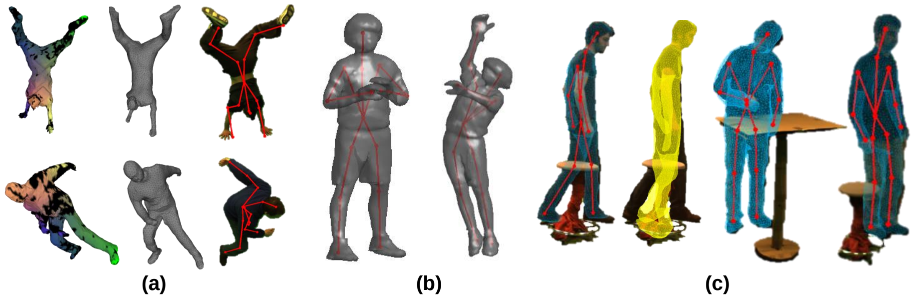


Figure 5.12: **Qualitative results**. (a) input visual hulls and the estimated correspondences in colors. Black color means no correspondence found for that vertex. Top: *Handstand*; bottom: *jumping*. (b) estimated shapes and poses of *Goalkeeper*. (c) estimated shapes and poses of *Cutting*, *WalkChair2*, and *WalkChair1*, overlayed on images. Blue: our results. Yellow: Cagniart *et al.* [35].

capable of providing better ones. Examples of the estimated correspondences, the deformed surfaces and the skeletons are shown in Fig. 5.12(a).

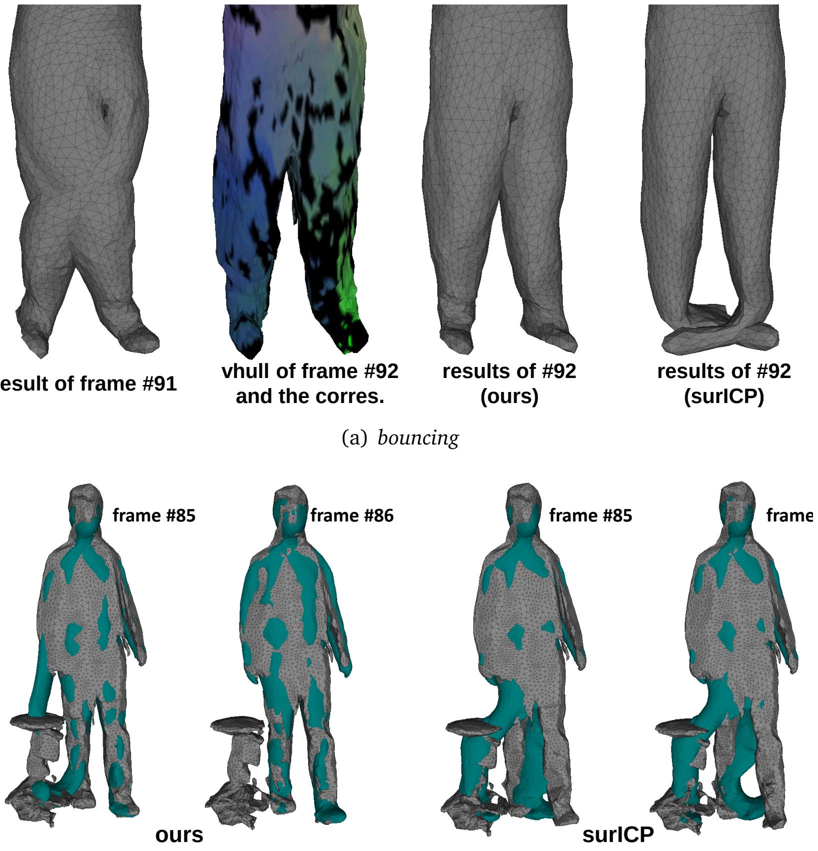
	ours + ICP		surICP [35]	
	error	# ICP-itr.	error	# ICP-itr.
Crane	8015	17	8138	20
Jumping	7976	16	7648	21
Bouncing	7569	34	7826	44
Handstand	9767	27	9963	60

Table 5.7: **Average silhouette overlap error in pixels, and the average ICP-iterations (itr.) of** 4 **sequences**. Image resolution: 1920×1080.

We further investigate what happens when there are tracking errors in previous frames. Fig. 5.13(a) shows the results of frame 91 in sequence *bouncing*. Note that left/right legs are crossed due to their close interaction with each other. In the next frame, when they are separate in the visual hull, the forest discovers correspondences more correctly, as in Fig. 5.13(b) (c.f. the colors of $\mathcal{M}$ in Fig. 3.2), and lead us to results in Fig. 5.13(c) without self-intersections. ICP strategy discover wrong associations and gives results in Fig. 5.13(d) under the same softness parameter λ. Errors propagate to the next frame, and gradually deteriorate the tracking, unless luckily regularization terms implies the opposite.

Tracking with outliers. Four of our testing sequences, *Cutting*, *WalkChair1*, *HammerTable*, and *WalkChair2* contain tables or chairs in observations, which play the roles as static outliers. We compare with other outlier rejection strategies such as, fixed outlier proportion (fixOL) [35], removing outliers by body-part classifications with SVM (bpSVM) [72], and modeling outlier likelihood dynamically by aggregating over all patches (patchedOL) [73].

As shown in Fig. 5.18(e-h), conventional outlier strategy fixOL drifts quickly and soon fail to track (green curves). ICP with robust outlier treatment, patche-



(a) *bouncing*

(b) *WalkChair1*

Figure 5.13: **Recovery from tracking error**. Our method is capable of recovering from tracking error while surICP [35] cannot.

dOL, is able to sustain noisy input to a certain extent. Once it starts drifting, the error only gets higher due to its ICP nature (yellow curves). When subjects and outliers are sperate components in visual hulls, we cast them into VNF, and feed them separately into the joint classification-regression forest. If they are connected to each other, forests inevitably associate some outliers to vertices on the model, and cause undesirable deformations, as the spike in blue curves in Fig. 5.18(d). Nonetheless, since we rely less on previous frames for data associations, the results can always get recovered when they are separated again. In average, we still yield low errors throughout the whole sequences, as in Table 5.6. We remark that such ability to recover is the essence of our discriminative approach, which is the biggest advantages over the existing generative methods.

We also verify the efficacy on pose estimations and compare to another frame-wise approach from Straka *et al.* [131, 132]. The error metric is the discrepancy between estimated joint positions and the annotated ground truths. Fig. 5.17 plots the results of *HammerTable* and *WalkChair2*, where we confirm again considerably better accuracy than bpSVM [72] (red curves), Straka *et* *al.* [131, 132] (green), and comparable results with patchedOL [73]. The recovered shapes and poses are also presented in Fig. 5.12(c), superimposed on original images.

5.6.3 Discriminative Volumetric Tracking

After evaluating the surface-based tracking-by-detection framework, now we turn to evaluate the volumetric one. We compare in two quantitative metrics against the whole pipeline in [71], which is the early version of our surface-based tracking-by-detection approach. We also show its resilience to large pose changes and its generalization capacities on an unknown subject.

Unlike the matching experiment in Sect. 4.6.1, here we apply the multi-template strategy in Sect. 4.5.2 to train one universal regression forest, with *Goalkeeper* chosen as the common template $\hat{\mathcal{S}}$. Training $T = 50$ trees up to depth 20 where each one is grown with around 200 CVTs (approximately one million samples) takes about 15 hours on a 24-core Intel Xeon CPU machine. For each subject, we track the testing sequence, which is not part of the training set. Tracking inputs are raw CVTs which have no temporal coherence. Correspondences are predicted by the forest and fed into the volumetric deformation framework described in Sect. 5.5. The number of clusters K is 250 for *Ballet* and *Goalkeeper* and 150 for *Thomas*.

Some visual results are shown in Fig. 5.14 and in the supplemental video[2]. With the help of regression forests, our approach is able to discover volumetric associations even in challenging poses found in *Thomas* and deform the templates successfully.

Quantitative evaluation and comparison

We also evaluate our tracking approach with two different metrics. On one hand, evaluation with marker-based motion capture evaluates the correctness of the surface pose, but only for a sparse set of surface points. On the other hand, the silhouette overlap error evaluates the shape estimate, but it does not evaluate the estimated pose. Hence these metrics are complementary.

Silhouette overlap error. We evaluate the tracking approach by computing the overlap error between the ground truth silhouette and the projection of the estimated surface. The metric we use is the pixel error (number of pixels that differ). Statistics are computed on all frames of all cameras.

Marker-based motion capture. The *Ballet/Seq2* sequence has marker-based motion capture data: fifty markers were attached to the body of the subject. The 3D tracking of the markers provides a sparse ground truth for surface tracking. First, each marker is associated to a surface vertex of the template. Then, for each marker, in each temporal frame, we measure the distance between the marker location and the estimated vertex location. Statistics on the distance are reported on Table 5.11. We observe that our approach gives slightly better

[2] `https://hal.inria.fr/hal-01300191`

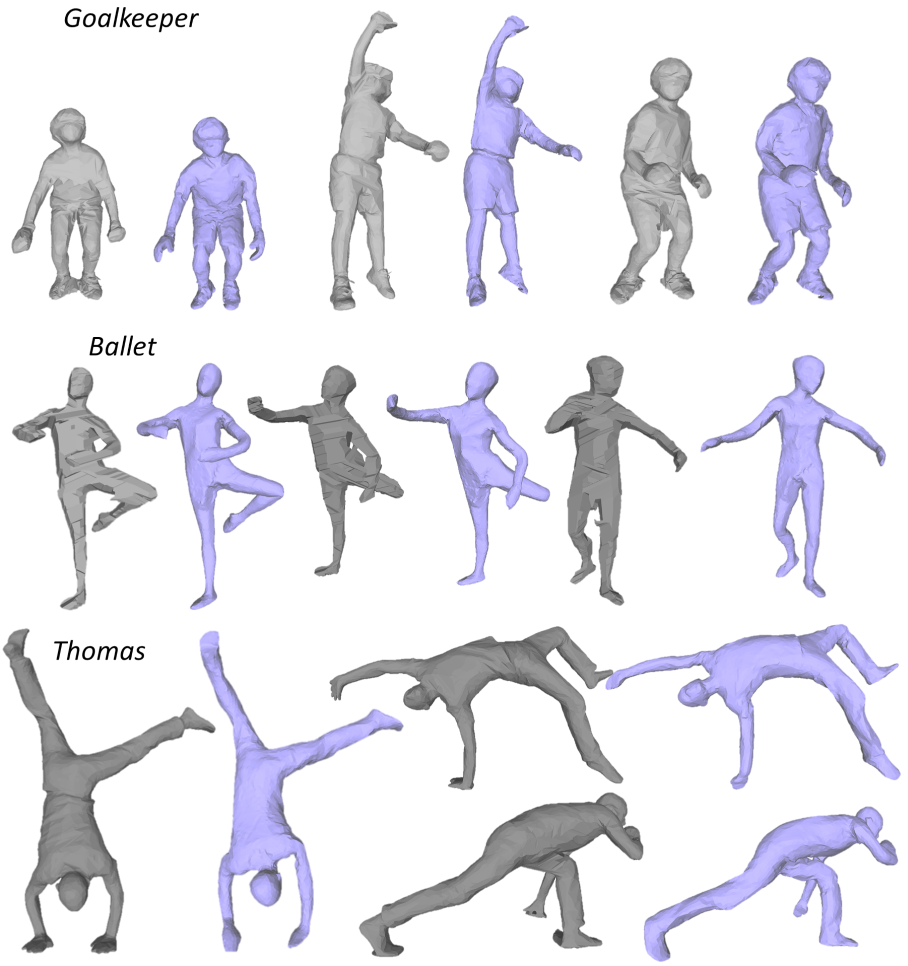


Figure 5.14: **Qualitative tracking results on three sequences**. Gray: input observed visual hulls; purple: deformed templates.

performances than a state of the art ICP-based approach, and outperforms a learning-based tracking approach which mostly fails to correctly register the legs of the subject.

Discussion. As discussed above, the high memory footprint of voxel-based volume in Chapter 3 limits the allowed training variations. Consequently, we choose to align the orientations for both training and input data such that forests only need to learn the pose variations of one single subject. In [71], we rely on the skeletal poses of previous frames to re-orient the input data of the current frame (Sect. 3.5.1). This leads to not fully frame-independent forest predictions and makes tracking subject to the potential risk of drifting. On the other hand, the approach presented in Chapter 4 attempts to incorporate rotational, pose, and even shape variations during training, yielding completely

method	mean	stddev.	median	max
Proposed	15221	6843	14754	57748
Huang *et al.* [71]	19838	14260	15607	109428
Allain *et al.* [5]	14773	6378	14355	43359

Table 5.8: **Silhouette pixel error on sequence *Goalkeeper/UpJump*.** Image size is 2048×2048.

method	mean	stddev.	median	max
Proposed	2620	1041	2557	8967
Huang *et al.* [71]	5427	2809	4863	39559
Allain *et al.* [5]	2606	1008	2571	7642

Table 5.9: **Silhouette pixel error on sequence *Ballet/Seq2*.** Image size is 1920×1080.

method	mean	stddev.	median	max
Proposed	9991	7089	7968	78242
Huang *et al.* [71]	28731	23421	22991	354293
Allain *et al.* [5]	10199	7379	8022	81649

Table 5.10: **Silhouette pixel error on sequence *Thomas/Seq2*.** Image size is 2048×2048.

method	mean (mm)	stddev. (mm)
Proposed	26.37	16.67
Huang *et al.* [71]	124.02	200.16
Allain *et al.* [5]	27.82	18.39

Table 5.11: **Statistics of surface registration error at marker locations, on the *Ballet/Seq2* sequence.**

frame-wise forest predictions. To facilitate a fully 3D tracking-by-detection framework, the information of previous frames is preferred no to participate in the discriminative correspondence estimation.

Therefore, to yield a fair experimental setting, we do not align the orientations of meshes when implementing our method and [71] here. As reported in Fig. 4.8, without canceling rotational variations, the accuracies of correspondences drop substantially on the testing sequences for the method in [71]. This means that voxel-based framework and the corresponding features do not generalize well to unseen rotations. When deployed in tracking applications, such unreliable associations eventually result in tracking failure. In particular, one can observe in Table 5.10 that [71] attains really high silhouette overlap discrepancy, most likely due to the fact that the subject rotates himself in many different orientations and thus confuses the forest.

Tracking at low frame rate. One of the expected benefits of our framework over purely ICP-based methods is improved resilience with large pose changes. We test this assertion by tracking the *Thomas* sequence at low frame rate (5fps). Figure 5.15 shows how our method recovers from tracking failures while [5] does not. This improvement is confirmed by the median silhouette overlap pixel error, which we found to be twice lower with our method (10054 pixels compared to 19998 pixels).

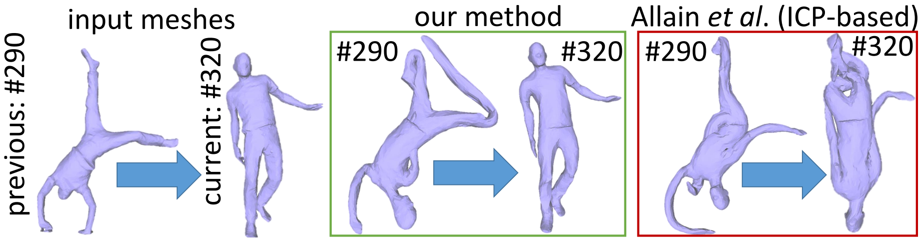


Figure 5.15: **Tracking results of *Thomas* dataset at low frame rate.**

Testing with a new subject. We tested the generalization capacities of our framework with a subject (*Dancer* dataset [6]) which is not in the training data. For this purpose, one can either select an existing template from the training sequences, or build a template model by matching one of the samples from the test sequence to the common reference model using skinning weights, as we do in multi-template training. We use the latter, which is more subject specific and can be expected to yield better results. Most poses are correctly tracked in our experiment (see Fig. 5.16). Not unexpectedly for this type of approach, some failures occur on more complex poses unseen in training data and would probably be improved with a larger training set.

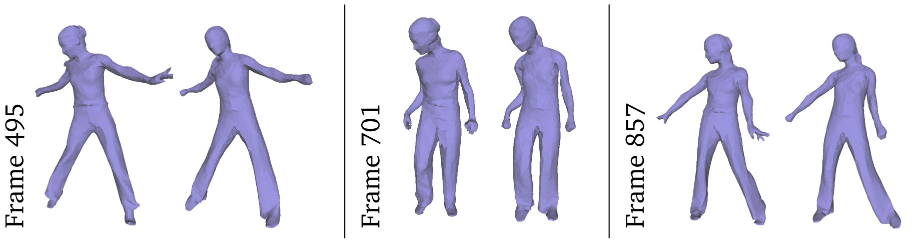
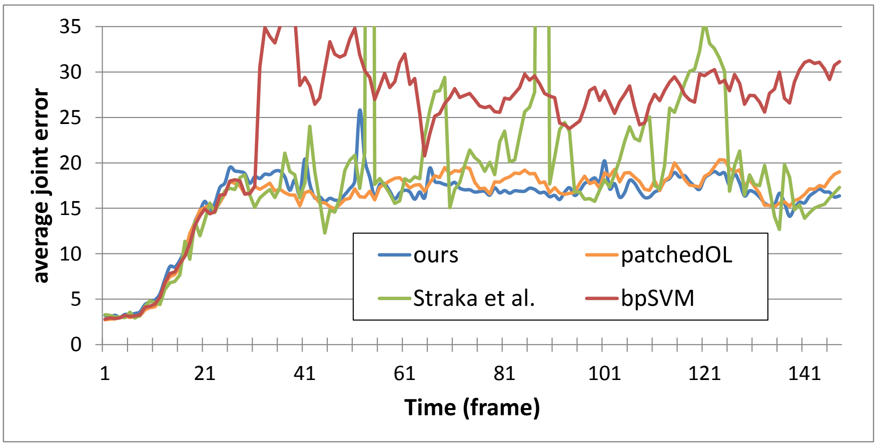


Figure 5.16: **Tracking results with a new subject, *Dancer* dataset**. Input mesh (left) and tracked mesh (right).

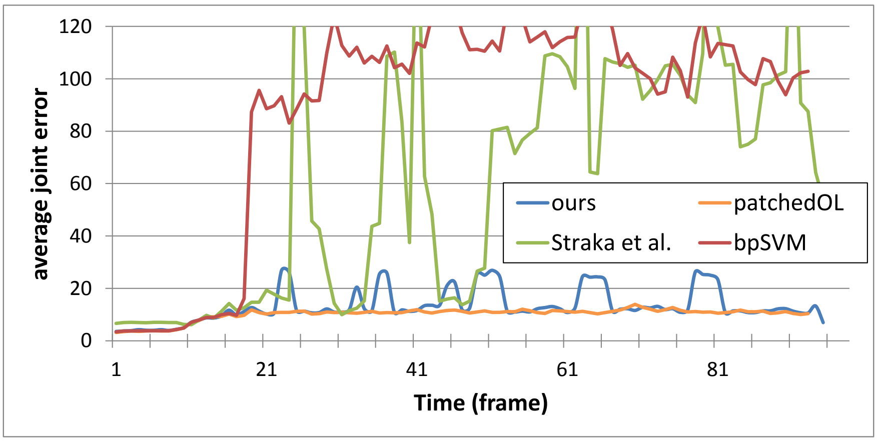
5.7 Conclusion

In this chapter, we first present, for the optimization step, an approach that jointly estimates poses and shapes of the human body. To this end, we augment the probabilistic deformable surface registration approach based on patched representation of the reference human body model with a skeleton energy. We demonstrate two approaches to estimate skeletal poses from surface deformations, which fits naturally to the Bayesian formulation of mesh registration.

Next, when combined with the discriminative associations, our full tracking-by-detection of 3D human shape pipeline is finally complete. This is a hybrid human shape tracking approach. The one-shot property of associations effectively prevents errors from accumulating, yielding more stability compared to other generative ICP extensions. We present numerous qualitative and quantitative analysis, confirming that we can recover meaningful deformations in spite of fast motions, large deformations and significant reconstruction artifacts. Such benefits are also confirmed in volumetric shape representations.

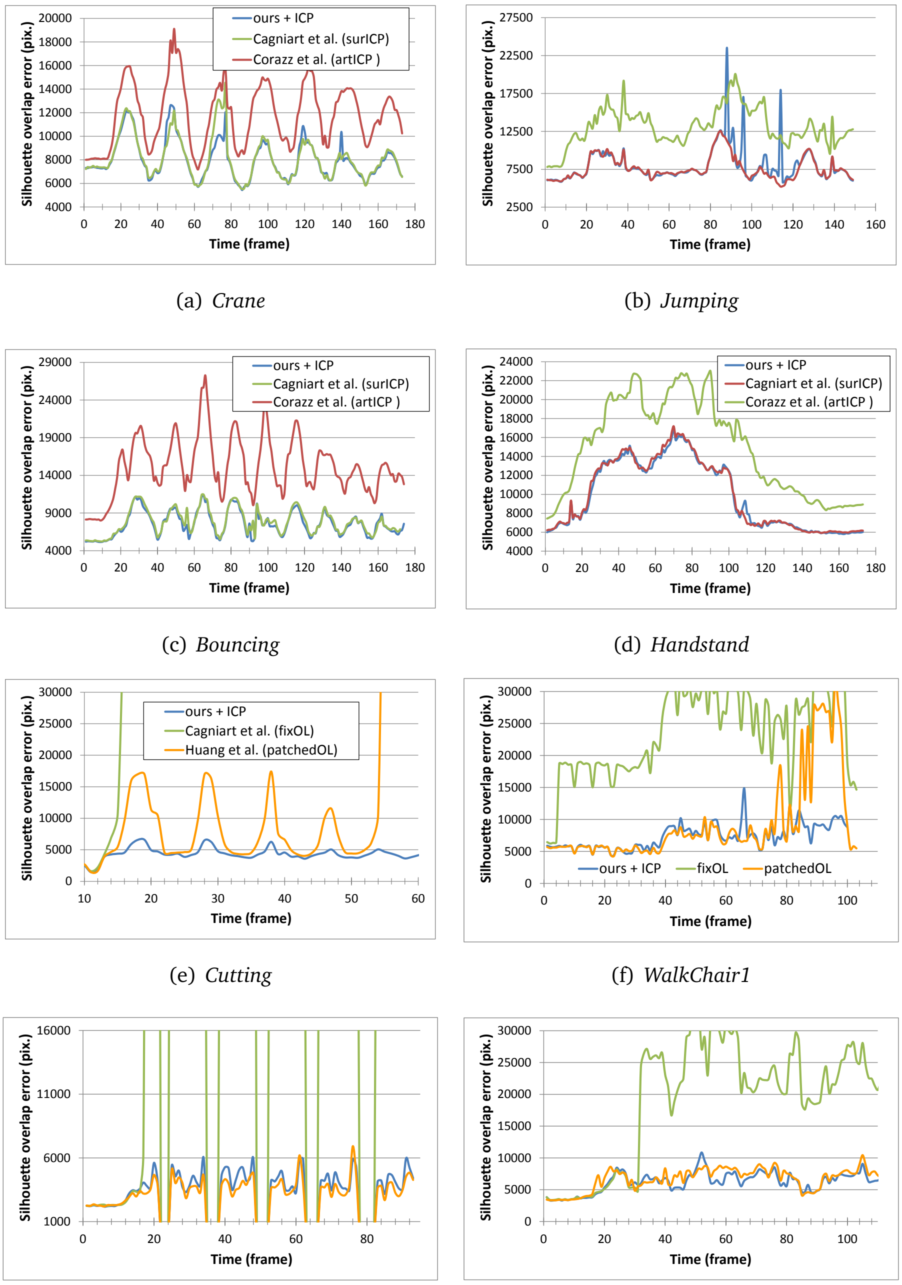


(a) *WalkChair2*



(b) *HammerTable*

Figure 5.17: **Averaged joint errors of four different approaches**: ours (blue), bpSVM [72] (red), Straka *et al.* [131, 132] (green), and patchedOL [73] (orange). Image resolution: 1000×1000.



(a) *Crane*

(b) *Jumping*

(c) *Bouncing*

(d) *Handstand*

(e) *Cutting*

(f) *WalkChair1*

(g) *HammerTable*

(h) *WalkChair2*

Figure 5.18: **Pixel overlap error of** 8 **sequences, averaged over all cameras**. Image resolution: (a-d): 1920 × 1080. (e-h): 1000 × 1000

6 Keyframe-based Tracking

The main scope of the thesis is to address the problem of 3D human shape tracking in multi-camera environments. Up to this point, we have approached the problem from a discriminative perspective and from a generative points of view. The former aims to discover the deformed template in observations using random forests; the latter assumes close initializations and fits the reference surface to the input with two motion parameterizations. In both cases, it has been assumed that the pose of the template that we refer to, called *rest pose* (typically in T or A poses) is the same during tracking. In other words, $\mathbf{\Theta}$ is always relative to $\mathbf{X}^0$ where the regularization is imposed upon as well. Since the local configurations of patches vary with poses, sticking to solely one pose for regularizing deformation is clearly a suboptimal option. In this Chapter, we explore the possibility to change the rest poses of the reference surface. We term the chosen poses as *keyframes* and contrive algorithms to identify them online during tracking phase, without any offline process. We follow the simpler generative experiment setting in order to contrast the strength of this concept.

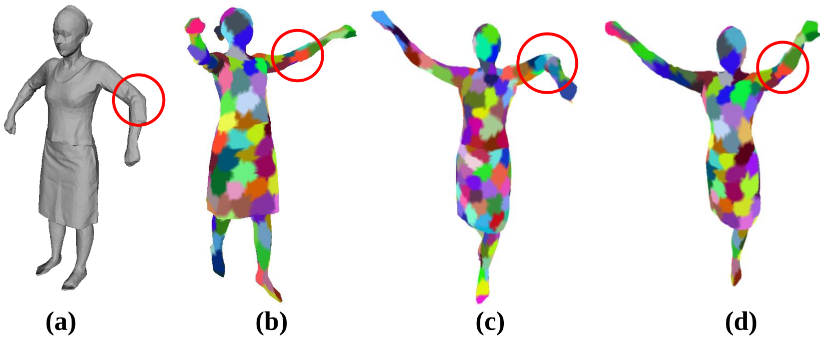


Figure 6.1: **Illustrations of the advantages of multi-keyframe strategy**. (a) reference surface of *Skirt* [62] as the first keyframe. (b) second keyframe identified at $t = 95$. At $t = 102$, the left arm is missing in the observations, as in Fig. 6.2. Using (a) as the reference yields (c) while using (b) yields (d).

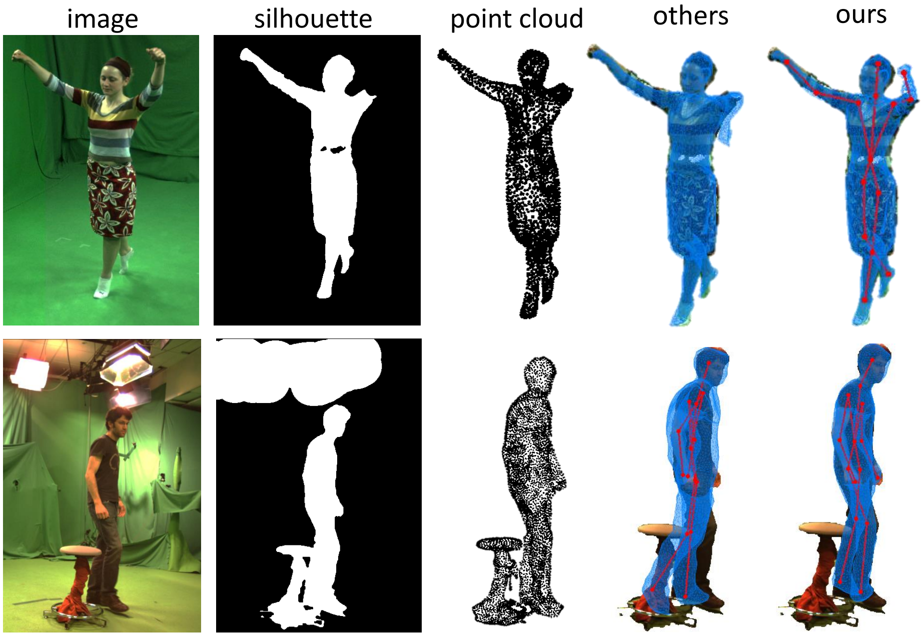


Figure 6.2: **Comparisons of various approaches**. Our approach recovers the shape and the pose despite missing data (top row) and outliers (bottom row) whereas other approaches (top: [35] and bottom: [131] + [132]) fail.

6.1 Introduction

Marker-less human motion capture consists in tracking human shape and pose using visual information. This has become an important research area with many applications in motion analysis or digital content production. Perhaps the most widespread approach to solve this problem is to deform a pre-defined reference surface so as to fit data derived from image observations, *e.g.* silhouettes or 3D points. This model-based strategy has demonstrated a good success over the past few years [35, 62, 98, 72, 132, 148], because of its ability to enforce strong consistencies over time through the prior models of shape and deformation.

However, this strategy still relies on the assumption that image observations are complete and relevant, *i.e.* they do not describe another shape. In practice, it appears to be difficult to maintain such assumption when considering more realistic dynamic scenarios and with fewer constraints on the capture environment. As shown in Fig. 6.2, top row, background subtraction is often erroneous with the consequence of missing data, *e.g.* the missing arm. Another type of errors occurs when image observations describe a shape that is not in consideration, and can therefore mislead the surface as well, *e.g.* the chair in Fig. 6.2 bottom row. Our aim is to propose an alternative strategy that better handles such situations and hence contributes to the general objective of unconstrained human motion tracking in real environments.

Our framework is inspired by keyframe-based methods, *e.g.* [146] in camera/object tracking, and by non-sequential surface registration methods [32, 84].

In these works the tracking task is eased by reducing the discrepancy between the model and the input data to be matched. We exploit a similar idea that consists in having multiple reference shape models that can be fit to the observations. Numerous existing approaches rely on a one-fit-all strategy where a single reference model is deformed to fit to all observations. This strategy is likely to fail when observations describe a shape significantly different from the model, due to the presence of missing data or outliers. Instead, we propose to build a set of reference models called *keyframes*, which correspond to several representative shapes and poses that have been explored during tracking. They are identified online using mean-shift clustering and without the need for offline pre-processing. At each frame, the *best* keyframe is chosen as the reference model. We combine this strategy with a robust surface deformation method. Comparisons with the state-of-the-art confirm the advantages of this approach with real and inaccurate data.

This chapter has several contributions. First we introduce the notion of keyframes in 3D human motion tracking, and further propose a keyframe-based tracking framework that updates the keyframe pool incrementally. Second, a new outlier rejection method is presented with the benefit of high integrability into previous probabilistic surface deformation framework. Both contributions increase the robustness and significantly limit the impact of missing data and outliers. To evaluate our method, we recorded new sequences that include static outliers. To the best of our knowledge, none of the current public dataset presents such feature.

6.2 Related work

Existing methods that track human poses and shapes generally express the problem as maximum *a posteriori* (MAP) estimation which involves a data term modeling the likelihood of the estimation and a regularization term modeling the adequacy to the prior information. Methods differ then by the input data and the assumed prior knowledge.

6.2.1 Data term

Data terms measure how well the model explains the observations. In general, silhouettes, point clouds, and photometric information are considered for this purpose.

Silhouettes. Many approaches deform the model such that the contour of the projected surface coincides with the contour of the observed silhouettes, *e.g.* [47, 148, 62, 132]. In [62], Gall *et al.* consider also photometric information to establish 3D-2D model-data correspondences. Later in [98], additional image segmentation information are used to differentiate multiple interacting subjects. In these works, silhouette overlap error is often regarded as a standard error measure, which is sensible only when silhouettes are accurate and fully describe shapes. Also note that distances in 2D images do not necessarily reflect distances in 3D and small errors along the silhouette contour can correspond to large distances in 3D. As a result, some authors, *e.g.* [35, 72] advocate for considering 3D points as input data.

Point clouds. Given a set of points reconstructed from multiple silhouettes, *i.e.* [54, 60], some authors first estimate correspondences between the model and the 3D observations, and then deform the model accordingly. Although the reconstruction suppresses artifacts resulting from 2D noise, it also introduces new errors, such as missing body parts or fake geometry elements like ghost limbs. To robustly track in the presence of outliers, Huang *et al.* [72] train a linear support vector machine (SVM) that classifies the input data into different body parts. Outliers are then rejected based on the posteriors given by the SVM classifier. This approach depends heavily on the classifier, and is time-consuming since the SVM must be trained at each frame. In [35], Cagniart *et al.* model outliers as an additional component of a Gaussian Mixture Model (GMM) equipped with a uniform distribution defined *a priori*. The adjustment of this distribution is however difficult and has a strong influence on the results. We propose a more robust outlier rejection that does not depend on user defined parameters.

6.2.2 Regularization term

Evolving a surface with discrete observations is ambiguous by nature and some prior information on the model is usually required. This information varies from generative spatial shape models to discriminative models that are learned from already known shapes and poses.

Spatial shape models. Several works employ Laplacian coordinates [125] to preserve local shapes, *e.g.* [62, 132], while others define a rigidity term that serves similar purposes, *e.g.* [34]. Note that all these methods refer to a single static reference shape to constrain local deformations. This reference shape model is usually in rest pose and built prior to the tracking [35, 47, 148]. However, the observations can significantly deviate from the reference model along time. The shapes and poses that were already recovered during tracking can help in that respect, which motivates the multiple-keyframe concept presented in this chapter. During tracking this knowledge might change as the surface evolves. It is more beneficial if we can utilize the new prior knowledge, which is achieved by our multiple-keyframe-based tracking framework.

Learned deformation models. A few works also make use of pre-collected information to help the tracking. They seek to learn the possible deformations in advance to regularize the results. In a non-sequential strategy, Budd *et al.* [32] and Klaudiny *et al.* [84] assume that the complete input sequence is available beforehand and they find the best order to traverse it using a minimum spanning tree algorithm. They further propose a cluster tree to balance between drift and jumps [84]. Duveau *et al.* [51] propose a supervised learning strategy that regularizes the results based on the learned distribution in a latent parameter space. These methods require a pre-processing step either to build a shape-similarity tree from the input sequence [32, 84], or to learn a low-dimensional representation from the gathered motion training data [51, 145].

Our multiple-keyframe approach also exploits temporal information. As opposed to the mentioned approaches, we learn the reference models online in an unsupervised manner and do not require any preliminary step. Note anyway that our framework could also take advantage of already tracked sequences and generate the keyframe pool offline.

6.3 Overview

In this section we state our problem and give an overview of the proposed method. At every time frame t, a point cloud $\mathbf{Y}^t \subset \mathbb{R}^3$ is reconstructed from silhouettes using `EPVH` [54]. The goal is to deform a reference model such that it fits the observations $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1:O}$[1]. Our model comprises a reference triangle surface and an intrinsic skeleton. We adopt the patch-based model proposed in [34], where vertices are grouped into K patches. Without the prior knowledge of motion, patches are better to distribute uniformly on the surface as shown in Fig. 2.2(b). Our skeleton is a tree structure of J nodes (3D joints) and the root is set at the pelvis, as shown in Fig. 2.2(c). The skeleton is rigged into the mesh using `Pinocchio` [11], which gives the associations between vertices v and joints j. Deformations are parameterized with respect to: (i) the shape of the surface; (ii) the pose of the skeleton. The shape parameters $\mathbf{\Theta} = \{(\mathbf{R}_k, \mathbf{c}_k)\}_{k=1}^{K}$ are the orientation and position pair for each patch k and encode the deformation of the reference mesh model. The pose parameters $\mathbf{J} = \{\mathbf{x}_j\}_{j=1}^{J}$ are the 3D joint positions of the skeleton. Although loosing some rotational degree of freedom, parameterizing directly on joint position leads to a quadratic energy term that keeps optimization feasible [132]. The problem is then formulated as the maximization of the joint probability distribution of the data and model:

$$\max_{\mathbf{\Theta},\mathbf{J}} P(\mathbf{Y}, \mathbf{\Theta}, \mathbf{J}). \tag{6.1}$$

This above distribution can be decomposed into $P(\mathbf{Y}|\mathbf{\Theta}) \cdot P(\mathbf{J}|\mathbf{\Theta}) \cdot P(\mathbf{\Theta})$, which represents respectively the likelihood of the shape given the observations, the probability of the pose given the shape, and the prior knowledge on shape deformations. Hence Eq. 6.1 can be rewritten as:

$$\min_{\mathbf{\Theta},\mathbf{J}} \left[E_r(\mathbf{\Theta}) + E_{skl}(\mathbf{\Theta}, \mathbf{J}) - \ln\ P(\mathbf{Y}|\mathbf{\Theta})\right], \tag{6.2}$$

where $E_r(\mathbf{\Theta}) = -\ln P(\mathbf{\Theta})$ is the rigidity energy in Eq. 2.3 and $E_{skl}(\mathbf{\Theta}, \mathbf{J}) = -\ln P(\mathbf{J}|\mathbf{\Theta})$ is the skeleton energy proposed in Sect. 5.3, both of which behave like regularization terms. The likelihood $P(\mathbf{Y}|\mathbf{\Theta})$ is similar to [35] and uses Gaussian Mixture Model (GMM) where every patch explains every observations

[1] The superscript variable t is dropped whenever it is clear from the context.

$\mathbf{y}_i$ according to:

$$P(\mathbf{y}_i|\boldsymbol{\Theta}) = \sum_{k=1}^{K+1} \Pi_k P(\mathbf{y}_i|z_i = k, \boldsymbol{\Theta}). \tag{6.3}$$

z_i is the latent variable for each $\mathbf{y}_i$: $z_i = k$ means that $\mathbf{y}_i$ is generated by the mixture component associated with patch k. $\Pi_k = P(z_i = k|\mathbf{y}_i, \boldsymbol{\Theta})$ represents the probability that patch k explains observation $\mathbf{y}_i$. For each patch, when the closest vertex v_i^k with a compatible normal vector exists, the likelihood that $\mathbf{y}_i$ is generated by the k-th component is modeled as a multivariate Gaussian with a mean located at the position $\mathbf{x}_{v_i^k}$ of v_i^k and isotropic variance; otherwise the likelihood is a negligible number ϵ:

$$P(\mathbf{y}_i|z_i = k, \boldsymbol{\Theta}) = \begin{cases} \mathcal{N}(\mathbf{y}_i|\mathbf{x}_{v_i^k}, \sigma^2) & \text{if } v_i^k \text{ exists} \\ \epsilon & \text{otherwise.} \end{cases} \tag{6.4}$$

Solving Eq. 6.2 yields both the pose and the shape. In [35], the tracking over a complete sequence is achieved by deforming the model on a frame-by-frame basis. That is, using $(\boldsymbol{\Theta}^{t-1}, \mathbf{J}^{t-1})$ as the initialization to solve Eq. 6.2 at frame t. To increase the robustness of this framework with respect to missing data and outliers better, we propose two methods that improve the deformation prior $P(\boldsymbol{\Theta})$ (Sect. 6.4) and the likelihood $P(\mathbf{Y}|\boldsymbol{\Theta})$ (Sect. 6.5), respectively.

6.4 Multiple-Keyframe Tracking Framework

The rigidity energy $E_r(\boldsymbol{\Theta})$ enforces neighboring patches to keep the original local configurations they have on the reference model, usually the shape at a given time t (*e.g.* $t = 0$). However, such local configurations do not always match with the current frame. This is particularly critical with missing data since patches without close observations tend to keep a *possibly* wrong prior reference configuration, as illustrated in Fig. 6.1.

Therefore, to effectively handle missing data, we introduce a framework that exploits multiple reference models or keyframes. While already used in image

Algorithm 1 Keyframe-based human motion tracking

```
$\mathcal{Q} \leftarrow \{0\}$, $\Psi \leftarrow \{(\boldsymbol{\Theta}_0^0, \mathbf{J}^0)\}$
Overall shape parameters $\hat{\boldsymbol{\Theta}}t_0 \leftarrow \{(\mathbf{I}, \mathbf{c}_k^0)\}_{k=1:K}$.
for $t$ in timeFrames do
    Choose the reference model $f_{\text{ref}}$ based on $\mathbf{Y}^t$.
    $\boldsymbol{\Theta}_{f_{\text{ref}}}^{t-1} \leftarrow \boldsymbol{\Theta}_0^{t-1} * (\boldsymbol{\Theta}_0^{f_{\text{ref}}})^{-1}$
    With $(\boldsymbol{\Theta}_{f_{\text{ref}}}^{t-1}, \mathbf{J}^{t-1})$ as initialization, solve Eq. 6.2 to obtain $(\boldsymbol{\Theta}_{f_{\text{ref}}}^{t}, \mathbf{J}^{t})$.
    $\boldsymbol{\Theta}_0^{t} \leftarrow \boldsymbol{\Theta}_{f_{\text{ref}}}^{t} * \boldsymbol{\Theta}_0^{f_{\text{ref}}}$
    if new keyframe detected then
        Update $\mathcal{Q}$ and $\Psi$.
    end if
end for
```

tracking [146], keyframes have not yet, as far as we know, been applied to 3D human motion tracking problems. Multiple keyframes correspond to different instances of a shape that better represent the shape variability than a single pose at a given frame t. Our framework learns online keyframes and, at each frame, selects the best one as the reference model to be fitted to the observations. Let $\mathcal{Q} = \{f_q\}_{q=1:Q}$ denote the keyframe pool where f_q is the frame index and Q is the total keyframe number, and let $\Psi = \{(\mathbf{\Theta}_0^{f_q}, \mathbf{J}^{f_q})\}_{q=1:Q}$ denote the corresponding parameter set. Our keyframe-based tracking method is summarized in Alg. 1, where $\mathbf{\Theta}_0^f$ corresponds to the accumulated rotation and translation from $t = 0$ to $t = f$, and $(\mathbf{\Theta}_0^f)^{-1}$ represents the inverse transformation. Allowing for different reference models enables different prior knowledge to be taken into account in the rigidity energy $E_r(\mathbf{\Theta})$. Two crucial steps in Alg. 1 are:

- *Line 4*: how to select the best reference model from the keyframe pool?
- *Line 8*: when to add a new keyframe in the pool?

We tackle the former issue with shape dissimilarity and the latter with key pose detection. More details are elaborated in the following two subsections.

6.4.1 Keyframe detection

We first explain how new keyframes are added (*i.e.* Line 8 in Alg. 1). The essence of the multiple keyframe strategy lies in its ability to identify and record, during tracking, new local patch configurations. When the observed shape takes a pose at t that is very different from the reference pose, it is worth considering adding a new keyframe corresponding to frame t. Such analysis can be performed offline if knowledge on the shape poses is available prior to tracking. However, we consider here the more generic situation with little prior knowledge and where keyframes are detected online during tracking.

Pose descriptor. In order to incrementally identify distinctive poses, we explore the previously-obtained skeleton poses $\mathbf{J}$. For each new pose $\mathbf{J}^t$, the pelvis of the skeleton is aligned to the global origin. The 3D coordinates of the remaining 14 joints are then concatenated to form a 42-dimensional human body pose vector $\mathbf{v}$. Aligning the pelvis to the origin cancels the global position and similar poses at different locations get similar representations. Here the skeleton is not rotated into a canonical direction and $\mathbf{v}$ still encodes orientation, which may not be a desirable attribute. We address this issue later. Similar poses yield $\mathbf{v}$s that are close in $\mathbb{R}^{42}$ whereas different poses correspond to $\mathbf{v}$s that are distant. Using this descriptor, we cast the key pose detection as a clustering problem in $\mathbb{R}^{42}$. In this scenario, the number of clusters is supposed to be the output of the clustering technique and not a prior knowledge. Hence mean-shift clustering naturally appears as a well adapted solution.

Mean-shift based key pose detector. Assume Q keyframes are already identified. Intuitively, there should be also Q clusters of poses. When a new pose vector $\mathbf{v}$ is obtained, mean-shift is performed and returns a number of clusters

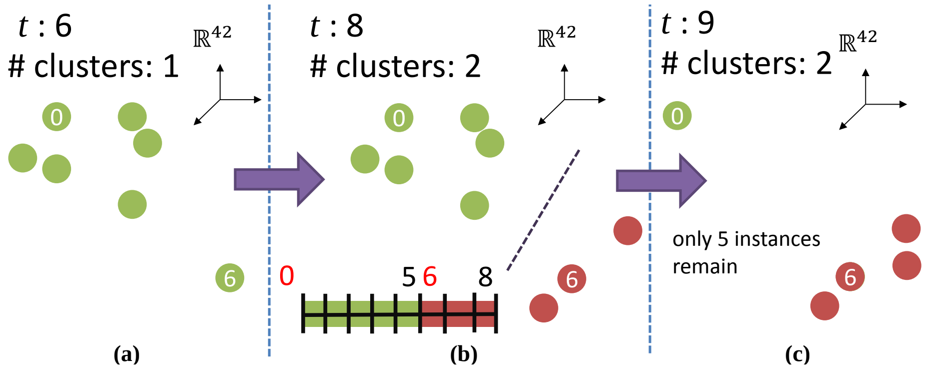


Figure 6.3: **Illustrations of Algorithm 2.** (a): time frame $t = 6$, $\mathcal{Q} = \{0\}$, and $f_{\text{last}} = 0$. (b): $t = 8$, $\mathcal{Q}$ becomes $\{0, 6\}$, and f_{last} becomes 6. (c): $t = 9$, $\mathcal{Q} = \{0, 6\}$, and $f_{\text{last}} = 6$. Only 5 instances $(9 - 6 + 2)$ are left for mean-shift.

n_c. If $n_c = Q$, the number of clusters has not changed and we proceed to the next frame. If $n_c = Q + 1$, it means that the shape pose has changed enough to justify a new cluster of poses. In general, poses in the same cluster also distribute closely in the time domain. Hence, starting from the current frame and going backward, the transition frame $\tilde{t}$ where the new cluster starts is determined (*e.g.* $\tilde{t} = 6$ in Fig. 6.3(b)) and added as a new keyframe to $\mathcal{Q}$; its pose $(\mathbf{\Theta}_0^{\tilde{t}}, \mathbf{J}^{\tilde{t}})$ being added to Ψ. Fig. 6.3 illustrates this principle, where f_{last} is the last element included in the keyframe set. The algorithm is sketched in Alg. 2, where Line 3 to 9 correspond to Line 8 to 10 in Alg. 1.

Algorithm 2 Mean-shift-based key pose detector

```
 1: Compute the bandwidth. Last keyframe $f_{\text{last}} \leftarrow 0$
 2: for each new incoming pose $\mathbf{J}^t$ do
 3:     De-pelvis the skeleton to obtain $\mathbf{v}^t \in \mathbb{R}^{42}$.
 4:     De-pelvis all $\mathbf{J}^f$ in $\Psi$ and obtain $\mathbf{V}_{\mathcal{Q}} = \{\mathbf{v}^f\}$.
 5:     Do mean-shift clustering on $\{\mathbf{v}^{f_{\text{last}}+1} \ldots \mathbf{v}^t\} \cup \mathbf{V}_{\mathcal{Q}}$.
 6:     if the number of clusters $= Q + 1$ then
 7:         Add transition frame $\tilde{t}$ to $\mathcal{Q}$ and $(\mathbf{\Theta}_0^{\tilde{t}}, \mathbf{J}^{\tilde{t}})$ to $\Psi$.
 8:         $f_{\text{last}} \leftarrow \tilde{t}$
 9:     end if
10: end for
```

Note that, in general, $\tilde{t} \neq t$ but $\tilde{t} < t$, which means that new keyframes are created online with some delay. For example, in Fig. 6.3(b), frame at $t = 6$ is detected as a keyframe with a 2 frame delay since in Fig. 6.3(a), $\mathbf{v}^6$ is still in cluster no. 1. When a new keyframe at $\tilde{t}$ is added, all the pose vectors before $\tilde{t}$, except the existing keyposes, are left out for further clustering (*e.g.* see Fig. 6.3(c)). This brings two advantages: first, the number of poses considered for clustering, *i.e.* $(t - f_{\text{last}} + Q)$, is significantly reduced compared to the full set of poses; second, if a pose re-appears during tracking, the new collected $\mathbf{v}^t$ is very likely to be clustered with existing keyframes, avoiding this way the occurrence of duplicated keyframes.

Mean-shift bandwidth. One concern with Alg. 2 is the bandwidth of mean-shift. A small bandwidth leads to many clusters while a large bandwidth gives few clusters. Since the intrinsic scale of the pose variation varies among sequences, an automatic way to determine this bandwidth is desirable. We achieve this using virtual pose vectors. Specifically, the de-pelvised skeleton model at $t = 0$ is rotated for $360°$ with steps of $10°$, creating $N_s = 36$ virtual pose vectors accordingly. Although they map to different points in $\mathbb{R}^{42}$, it is reasonable to cluster them together since they actually correspond to a single pose. We thus compute all $\binom{N_s}{2}$ pairwise distances and set the bandwidth as the half of their maximum. This way we ensure that they converge to the same mode with mean-shift. Recall that when a pose vector $\mathbf{v}$ is built from the estimated skeleton $\mathbf{J}$, only the position is canceled but not the orientation. Using the above bandwidth, we expect pose vectors that differ only by a rotation to be clustered together hence canceling the rotation as well. More analysis on the influence of the bandwidth are presented in the experiment section.

6.4.2 Choosing the best keyframe

Given a new set of observations $\mathbf{Y}$, the problem is now to determine the best keypose in $\mathcal{Q}$ to be matched to these observations (Line 4 in Alg. 1). When the shape associated to such a keypose is close to the observed shape $\mathbf{Y}$, it simplifies the estimation of the shape parameters and reduces the chances to fall in local minima. Therefore, we apply a shape similarity criterion to select the best keypose. This criterion uses shape histograms [8] to describe 3D shapes and the L^2 distance between normalized histograms as the dissimilarity measure [76]. The keyframe that presents the smallest dissimilarity with $\mathbf{Y}$ is chosen as the reference frame.

6.5 Patch-based Outlier Modeling

Besides missing data, sometimes point clouds contain false segmented foreground as the chair in Fig. 6.2 bottom row. In order to be robust to such outliers, care must be taken when designing the likelihood function $P(\mathbf{Y}|\mathbf{\Theta})$. Note that the association of an observation $\mathbf{y}_i$ to a patch, *i.e.* Eq. 6.4, applies only for $z_i = k \in [1, K]$ and that $z_i = K + 1$ is a special case introduced to model the outliers $\mathbf{y}_i$ that are not explained by any patch. However, there is no physical outlier patch in the model to be associated to. In [35], Cagniart *et al.* use a uniform distribution to model $P(\mathbf{y}_i|z_i = K + 1, \mathbf{\Theta})$ which basically assumes a certain proportion of the observations to be outliers and requires therefore some ad-hoc knowledge. Here we present a patch-based outlier modeling technique that takes into account spatial information and is based on the fact that the observations at frame t usually lie in the vicinity of the estimated surface at frame $t - 1$.

Before modeling the general outlier event $z_i = K + 1$, we first consider the outlier event just for patch k, denoted as W_k. The likelihood between $\mathbf{y}_i$ and W_k can be interpreted as how “bad" $\mathbf{y}_i$ is explained by patch k. Since Eq. 6.4

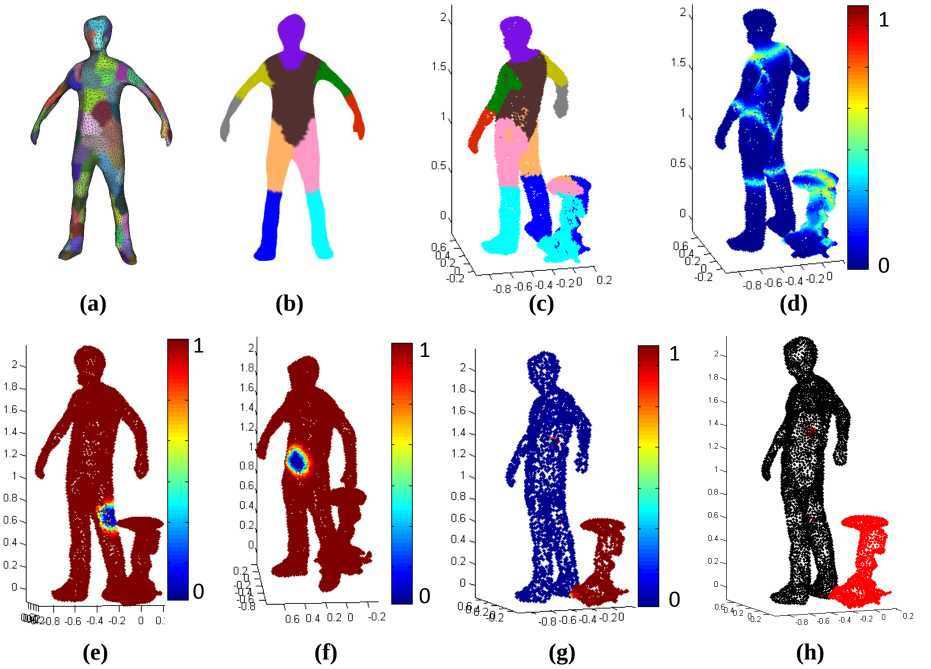


Figure 6.4: **Comparison of outlier rejection in [72] (c-d) and ours (g-h)**. (a-b): reference surface colored in patches and body parts, respectively. (c): point cloud classified into body parts using the SVM trained on (b) from the previous frame, as suggested in [72]. (d): corresponding outlier likelihood from SVM. (e-f): two examples of Eq. 6.5. (g): point cloud colored based on Eq. 6.6. (h): points with outlier likelihood higher than 0.5 are colored in red.

expresses how well patch k explains $\mathbf{y}_i$ under shape parameter $\boldsymbol{\Theta}$, we define the likelihood $P(\mathbf{y}_i|W_k,\boldsymbol{\Theta})$ as:

$$P(\mathbf{y}_i|W_k,\boldsymbol{\Theta}) \equiv 1-P(\mathbf{y}_i|z_i=k,\boldsymbol{\Theta}) = \begin{cases} 1-\mathcal{N}(\mathbf{y}_i|\mathbf{x}_{v_i^k},\sigma^2) & \text{if } v_i^k \text{ exists} \\ 1-\epsilon & \text{otherwise.} \end{cases} \tag{6.5}$$

Eq. 6.5 also expresses, given shape parameter $\boldsymbol{\Theta}$ and from the point of view of patch k, how likely $\mathbf{y}_i$ is to be an outlier. Checking over all patches how poorly they explain $\mathbf{y}_i$, and assuming independence between W_k, we can approximate the overall outlier likelihood as:

$$P(\mathbf{y}_i|z_i=K+1,\boldsymbol{\Theta}) \approx \prod_{k=1}^{K} P(\mathbf{y}_i|W_k,\boldsymbol{\Theta}). \tag{6.6}$$

From Eq. 6.6, we see that observations that are well explained by patches can not be outliers, where [35] considers equal chances for each observation to be an outlier. Fig. 6.4 illustrates this strategy. Fig. 6.4(a) and (b) shows the reference model represented in patches and body parts respectively. In [72], Huang *et al.* train a linear SVM on Fig. 6.4(b) obtained at $t-1$ to classify $\mathbf{Y}^t$ into different rigid body parts, as shown in Fig. 6.4(c). Chair observations are classified as body parts because of the linear assumption of SVM. If we consider the SVM output for the outlier likelihood, as in Fig. 6.4(d), we cannot distinguish between chair and human observations. However, our patch-based outlier modeling is able to assign high values to chairs, as in Fig. 6.4(g), and to identify them by simple thresholding.

Optimization We follow the optimization framework in [35, 72] which alternates between estimating associations, *i.e.* Eq. 6.3, and solving for the model parameters, *i.e.* Eq. 6.2. These two steps correspond to the E-step and the M-step in the Expectation-Maximization framework [18] . The advantage of our outlier strategy is that it easily integrates into this method. In practice, outliers are not removed once and for all with hard thresholding, but Eq. 6.6 is substituted in Eq. 6.3. This means that outliers are estimated during the EM optimization, and that there is no need to add any other sophisticated learning-based method to improve outlier rejection.

6.6 Experiments

The method was evaluated on 4 publicly available sequences as well as on 3 new sequences: *WalkChair*, *HammerTable*, and *SideSit* that contain static occlusions. These sequences were recorded with 9 cameras at 1000×1000 resolution. The occlusion objects are considered as foreground by the background subtraction and they remain in silhouettes and hence appear in the resulting point cloud observations, as shown in Fig. 6.12(c) and (d). We manually annotate the joint positions in 5 cameras to evaluate the poses of the skeleton. Due to the lack of realistic dynamic 3D surface ground truth, we use silhouette overlap error to evaluate the shape parameter estimation. If the occlusion objects are separated from the human body in silhouettes, we manually remove them and consider only the human parts as ground truth. These sequences are available at the *4D Repository*[2]. To draw fair comparisons with other approaches, we do not refine the surfaces with silhouettes after tracking. These 7 sequences serve different purposes in the experiments and we summarize them in Table 6.1. Results are analyzed with respect to missing data and outliers, both qualitatively and quantitatively. In all the presented experiments, both the multiple keyframes strategy and the outlier rejection mechanism were used.

6.6.1 Robustness to missing data

Skirt and *Dance* demonstrate the effectiveness of the multiple keyframe strategy. The average silhouette overlap error for these sequences is shown in Table 6.2. In [62], Gall *et al.* refine the shape using silhouettes as a second stage of their method. Such refinement could fail if occlusion objects are close to the subject and appear in the silhouettes (*e.g.* Fig. 6.2 bottom row). We thus compare to their first stage results only. In *WalkChair*, missing data can be observed when the arms are too close to the torso. With the shape at $t = 0$ as the reference

[2] `http://4drepository.inrialpes.fr/`

Sequence	Views	Frames	Outlier	Mis. data	Err. metric	Keyframe pool	Bandwidth	Compared approaches
Skirt [62]	8	720	-	$\surd$	A	$0, 95, 198$	0.31	[35], [62] (1st stage)
Dance [62]	8	574	-	$\surd$	A	$0, 201$	0.41	[35], [62] (1st stage)
Basketball [35]	8	1330	dynamic	$\surd$	-	$0, 29$	0.41	-
Fighting [98]	12	500	dynamic	$\surd$	B	$0, 20, 59, 74$	330.25	-
WalkChair	9	148	static	$\surd$	A & C	$0, 32, 54$	0.50	[35], [72], [131] + [132]
HammerTable	9	93	static	-	A & C	$0, 21$	0.44	[35], [72], [131] + [132]
SideSit	9	97	static	-	C	$0, 21$	0.50	[35], [72], [131] + [132]

Table 6.1: **Sequences used for evaluation in Scet. 6.6**. We apply three different error measures, depending on the provided ground truth. A: silhouette overlap error. B: distances in $\mathbb{R}^3$ between makers and associated vertices. C: distances in pixels with annotated joint positions.

	Cagniart [35]	Gall [62]	Prev.	ours
Skirt	7283	6900	7466	**6715**
Dance	7881	7600	fail	**6940**

Table 6.2: **Average silhouette overlap error with different approaches**. Image resolution: 1004×1004. Note that comparisons with [62] concern only their first stage results. Prev. are the results obtained when using the previous frame as the reference model. See Sect. 6.6.3 for more discussion.

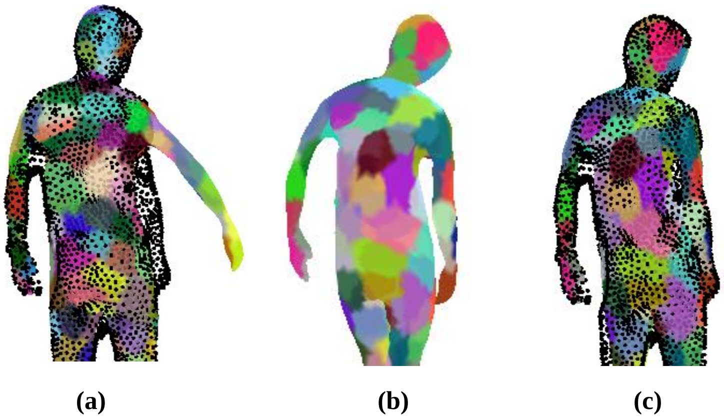


Figure 6.5: **Benefit of keyframe strategy**. Results of frame $t = 119$ in *WalkChair*, with and without multiple keyframe strategy. Black dots are the observed point cloud where right arm gets merged into the torso. (a): estimated shape using surface at $t = 0$. (b): estimated shape at $t = 32$. (c): estimated shape using (b) as the reference surface.

model result in Fig. 6.5(a) are obtained. The right arm stays in its configuration at $t = 0$ since not enough observations support the patches on the arm. Using another keyframe, as in Fig. 6.5(b), as the reference yields better result as shown in Fig. 6.5(c).

The influence of the mean-shift bandwidth. We report here on the influence of the mean-shift bandwidth on key pose detection and on the final tracking results. Tests were conducted on *Skirt*, *HammerTable*, and *Fighting* with varying bandwidths. Results on the keyframe numbers as well as on the corresponding errors are depicted in Fig. 6.6(a-c). The first two sequences were chosen for the repeating actions and the third one for its different numerical scale, see Fig.6.6(d). In general, small bandwidths lead to more keyframes (green curves) and it appears that the errors (blue curves) also decline as the bandwidth decreases. However, small bandwidths have higher chances to identify tracking failures as keyframes and therefore accumulate errors. This explains why the error slightly rises when the bandwidth gets really small. We noticed that due to the different numerical scales, resulting from different recording setups, there is no fixed bandwidth that guarantees the best performance over the three sequences. This indicates that manually fixing the bandwidth is difficult. However, our

	WalkChair	*HammerTable*
Cagniart [35]	18482	fail
Huang [72]	18063	fail
Straka [131, 132]	12219	17285
ours	**6803**	**3593**

Table 6.3: **Average silhouette overlap error of *WalkChair* and *HammerTable* from different approaches**. Silhouette resolution: 1000×1000.

	WalkChair	*HammerTable*	*SideSit*
Huang [72]	24.6 ± 10.7	fail	75 ± 40
Straka [131, 132]	20.6 ± 22.0	64.2 ± 53.9	84.4 ± 59.3
ours	$\mathbf{15.9 \pm 6.3}$	$\mathbf{10.1 \pm 3.0}$	$\mathbf{19.3 \pm 7.9}$

Table 6.4: **Average joint 2D re-projection error in pixels of *WalkChair*, *HammerTable* and *SideSit***. Image resolution: 1000×1000. We attain overall lowest error.

strategy considers the numerical scale of the sequence and adjusts the bandwidth accordingly. This provides optimal or close to optimal performance (red dots). Adding more keyframes in the keyframe pool does not bring much improvement in terms of error, sometimes even worse. Therefore we argue that the way we estimate the bandwidth is sufficient to discover distinguished poses and produce decent results, which is automatic and requires no tweaking. Note also that in *Skirt*, the subject raises up both arms and slowly rotates herself for a while, which leads to many similar poses only differing in orientations. As a result of our approach to cancel rotations, and the way we perform mean-shift with $\mathbf{V}_Q$, duplicate key poses do not occur. Keyframe pools of all testing sequences are shown in Fig 6.10.

6.6.2 Robustness to outliers

Static outliers. *WalkChair*, *HammerTable* and *SideSit* were used to demonstrate the robustness to outliers. Methods using silhouettes as input like [62, 132], are prone to be confused by the close occluding objects. Our approach was compared with [35], [72] and the tracking framework proposed by Straka *et al.* ([131]+[132]). The silhouette overlap error as well as the discrepancies between the projected skeletons and the manually annotated joint positions in 5 cameras were measured. As reported in Table 6.3 and Table 6.4, our method attains consistently lowest errors in both pose and shape estimations. It is worth noting that in *HammerTable*, around 41% of the observations are not from the human subject, but we still get decent results (see Fig. 6.12(c) and (d)). We notice also that when the occlusions are closely touching the subject, it confuses the method in [131] that deforms the skeleton model according to the observed skeletal graph in the point cloud, and thus the shape adaptation [132] cannot improve significantly the results. Please refer to Fig. 6.11 and the accompanying video of [73] for more comparisons.

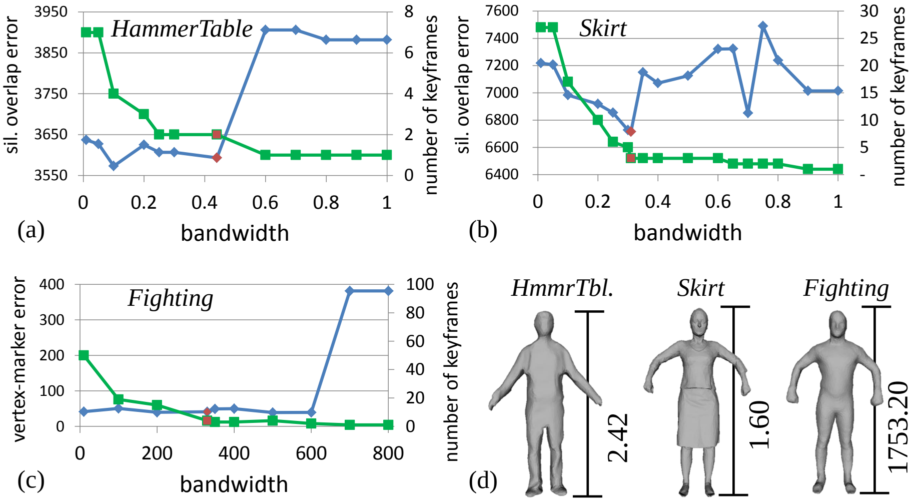


Figure 6.6: **Accuracy and the number of keyframes v.s. bandwidth; various dataset scale**. (a-c): Blue curves are the corresponding error (left y-axis) while green curves represent the number of keyframes (right y-axis). (a): *HammerTable*. (b): *Skirt*. (c): *Fighting*. (d): the heights of the subjects in 3 sequences.

Dynamic outliers. Due to the lack of public datasets with dynamic outliers, we evaluate our approach with two multi-subject sequences where we track only one subject and consider the others as outlier observations. For *Basketball*, the human subject is tracked against the ball observations. For *Fighting*, the subject with markers is tracked and the observations from the other subject are considered as outliers. The results are shown in Fig. 6.12(a) and (b). First we confirm that without the outlier rejection strategy presented in Sec. 6.5, the method in [35] fails with this experimental setting, which is expected since it also fails to track against static outliers as in Table 6.3. Following the metric in [98], our approach attains 40.95 *mm* of average vertex position error with a standard deviation of 15.34 *mm* over 500 frames. Although the corresponding error in [98] is 29.61 *mm* and 25.50 *mm*, respectively, we would like to point out that their task is different from ours. Our objective is to robustly track in noisy environment whereas [98] simultaneously track two surfaces. Every observations is, in this case, associated to a patch or a vertex, and thus outliers are not to be considered. We observe anyway that, despite the presence of strong dynamic outliers (*i.e.* the ball and the second subject), our approach still provide reasonable results.

6.6.3 Discussions

Influence of mean-shift bandwidth. In Fig. 6.7 we visualize the generated keyframe pools of *Skirt* and *HammerTable* in different bandwidths. Two sequences are chosen because the subjects repeat the actions. With small bandwidths, we observe many similar key poses, which however does not guarantee smaller error. With the estimated bandwidths we not only obtain distinctive key poses but also provide comparable performance.

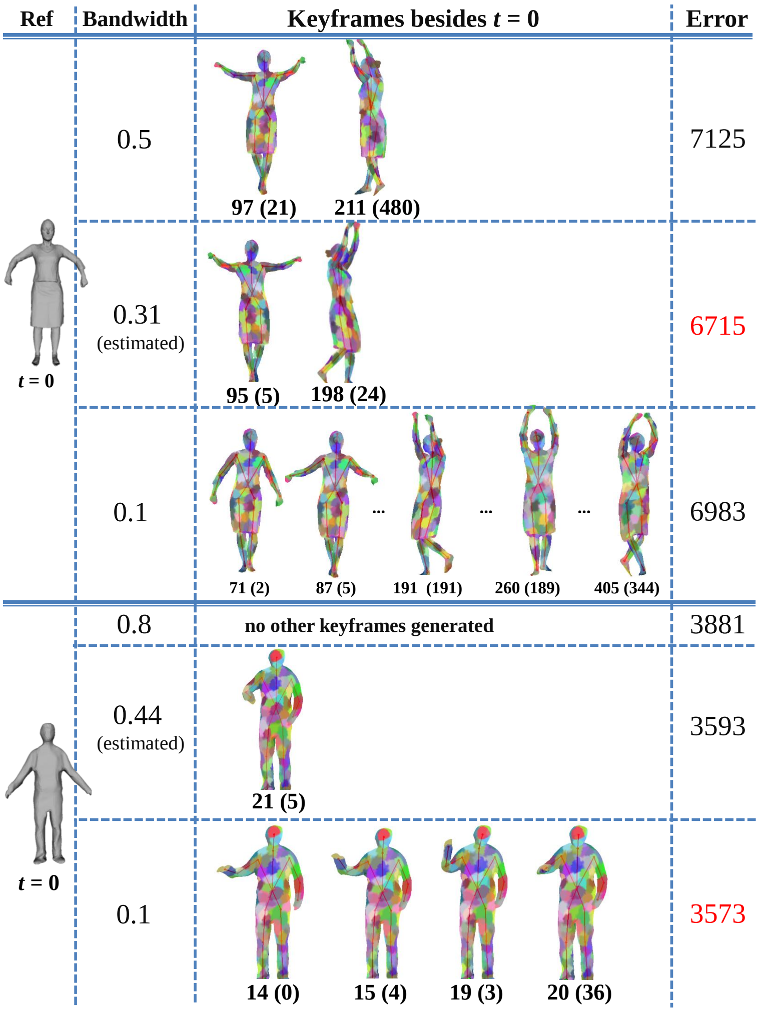


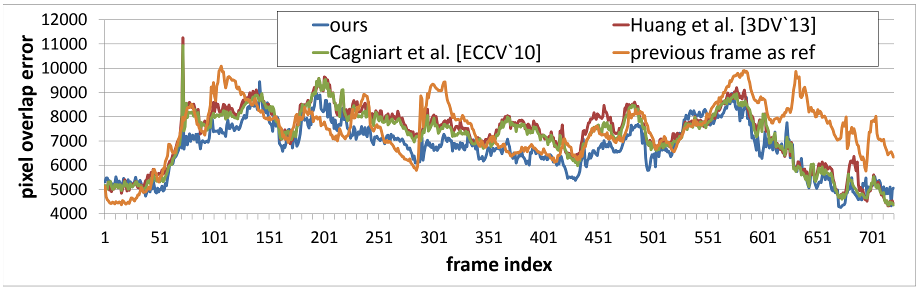
Figure 6.7: **Generated keyframe pool of *Skirt* [62] (top) and *HammerTable* (bottom) in varying mean-shift bandwidths.**

Further quantitative analysis. Table 6.2 shows the overall average pixel overlap error of *Dance* and *Skirt*. In Fig. 6.8, we report the error in each frame. Broadly speaking, our approach attains smaller error over the whole sequences, compared with Cagniart *et al.* [35] and Huang *et al.* [72]. In Fig. 6.11, we further report the 2D joint error of *WalkChair*, *HammerTable*, and *SideSit*. We see that while [72] fails to track at a certain point, and Straka *et al.* [131] + [132] produces sporadic high errors, our approach obtains consistent low error over sequences.

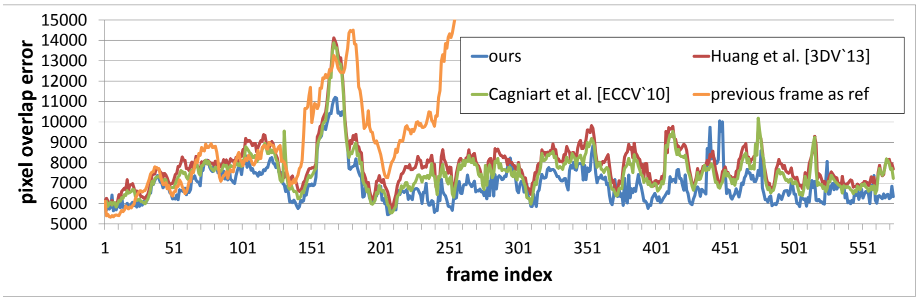
To further justify the advantage of our keyframe-based framework, we make a comparison with following two strategies:

1. Adhering to $t = 0$ as the reference model.

2. Adhering to previous frame as the reference model.



(a) *Skirt*



(b) *Dance*

Figure 6.8: **Pixel overlap error of *Dance* and *Skirt* [62] in each frame, averaged over 8 cameras**. Image resolution: 1004×1004. Blue: ours. Green: Cagniart *et al.* [35]. Red: Huang *et al.* [72]. Orange: using the previous frame as the reference model.

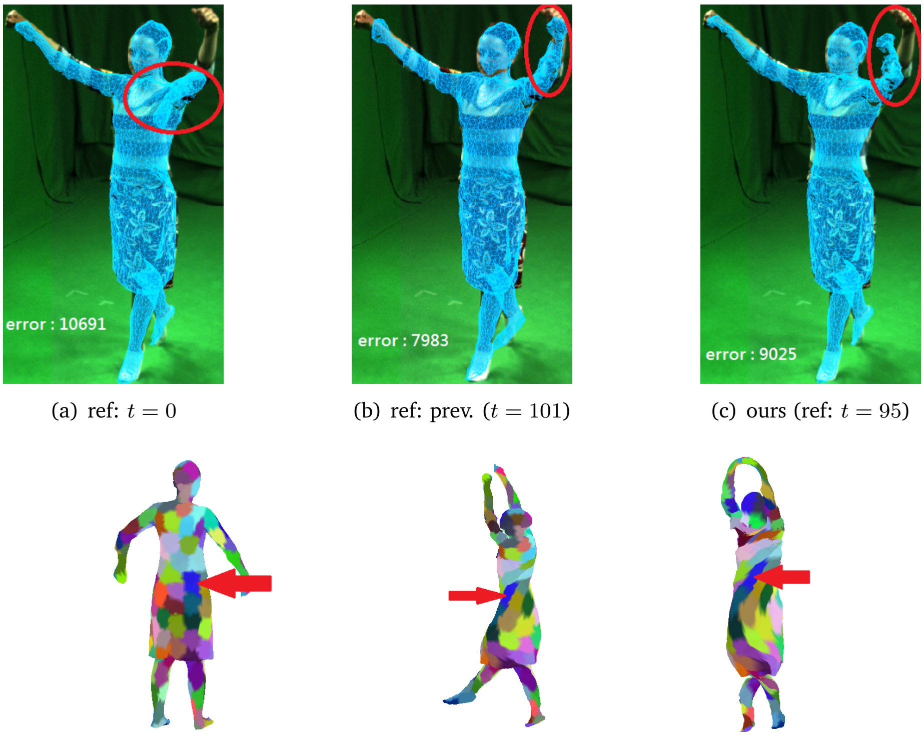
The benefit of our approach over the first strategy (*i.e.* ref: $t = 0$) is already presented in Fig. 6.1, Fig. 6.5 and the corresponding text above. Here we concentrate on comparing with the 2^{nd} strategy, which always uses the tracked result of previous frame as the reference model for the current frame. In Fig. 6.9(a-c), we overlay the corresponding results of $t = 102$ in *Skirt* sequence. For this frame only, using the previous frame result as reference actually yields smallest error. We demonstrate in Fig. 6.9(d-f) the potential drawback of this strategy: drifting. We see that the blue patch is supposed to be at the back side of the subject ($t = 31$), but it moves *along* the surface embedding during tracking, and ends up at the front side of the body ($t = 462$). In the very beginning of the tracking, drifting is difficult to be observed via overlap error because the silhouette does not differ too much (orange curves in Fig. 6.8). However, as the errors accumulates, drifting gradually deteriorate the results, and eventually leads to noticeably large errors (*Skirt*), or even a tracking failure (*Dance*).

Generated keyframe pool. We show the identified keyframes of all testing sequences and the associated estimated bandwidth (BW) in Fig. 6.10. Thanks to the way we create virtual samples, we do not observe duplicate keyframes in the same sequences, and the delay time are all within acceptable range.

Further qualitative results. In Fig. 6.13, we further demonstrate the effectiveness of our approach on taking care of outliers and missing data. In Fig. 6.13(b), we observe that the hand of the subject is connected to the table in both the silhouette and the point cloud. Such observations confuse methods like [131] which results in the high peak error in Fig. 6.11, whereas our method still estimates the pose and the shape successfully. In Fig. 6.13(c), despite that the ball observations have close interaction with the subject, we still obtain correct shape around the right leg. In Fig. 6.13(d), we see that our method properly handles merging body parts (the right hand), and excludes outliers, while [35] does not manage to do so.

6.7 Conclusion

We present an approach that captures human performances from multi-view video without markers. Considering realistic cases, we propose a multiple-keyframe-based tracking framework that uses mean-shift clustering to update a keyframe set online. A patch-based outlier modeling method is also presented to identify outliers more efficiently and effectively. Combining these two techniques into a surface deformation framework increases the robustness and enables the estimation of human shape and poses against missing data and outliers. The reliability of the proposed method is confirmed by the experiments on various public sequences as well as newly recorded sequences. Future directions include alleviating the requirement for background subtraction by considering photometric information.



(a) ref: $t = 0$ (b) ref: prev. ($t = 101$) (c) ours (ref: $t = 95$)

(d) prev. $t = 31$ (e) $t = 256$ (f) $t = 462$

Figure 6.9: **Comparison of three different strategies (a-c), and the drawbacks of sticking to the previous frame result as the reference model (d-f).**

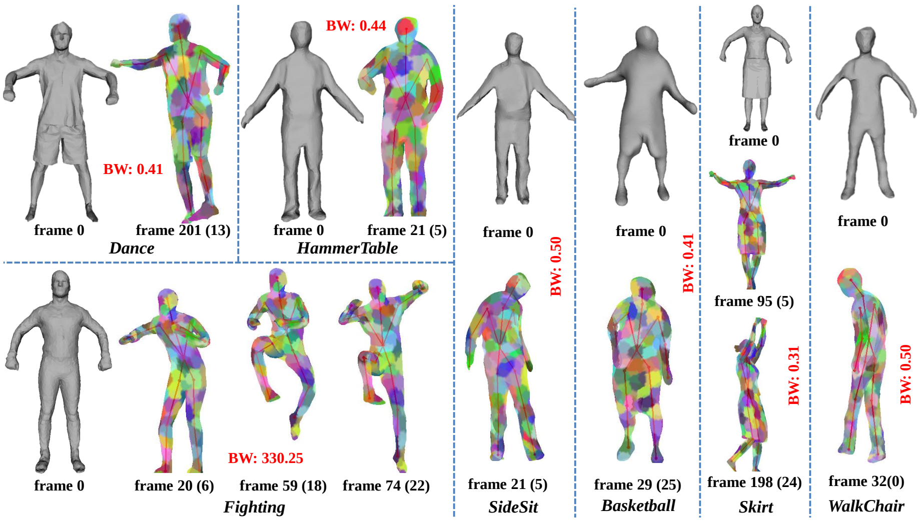
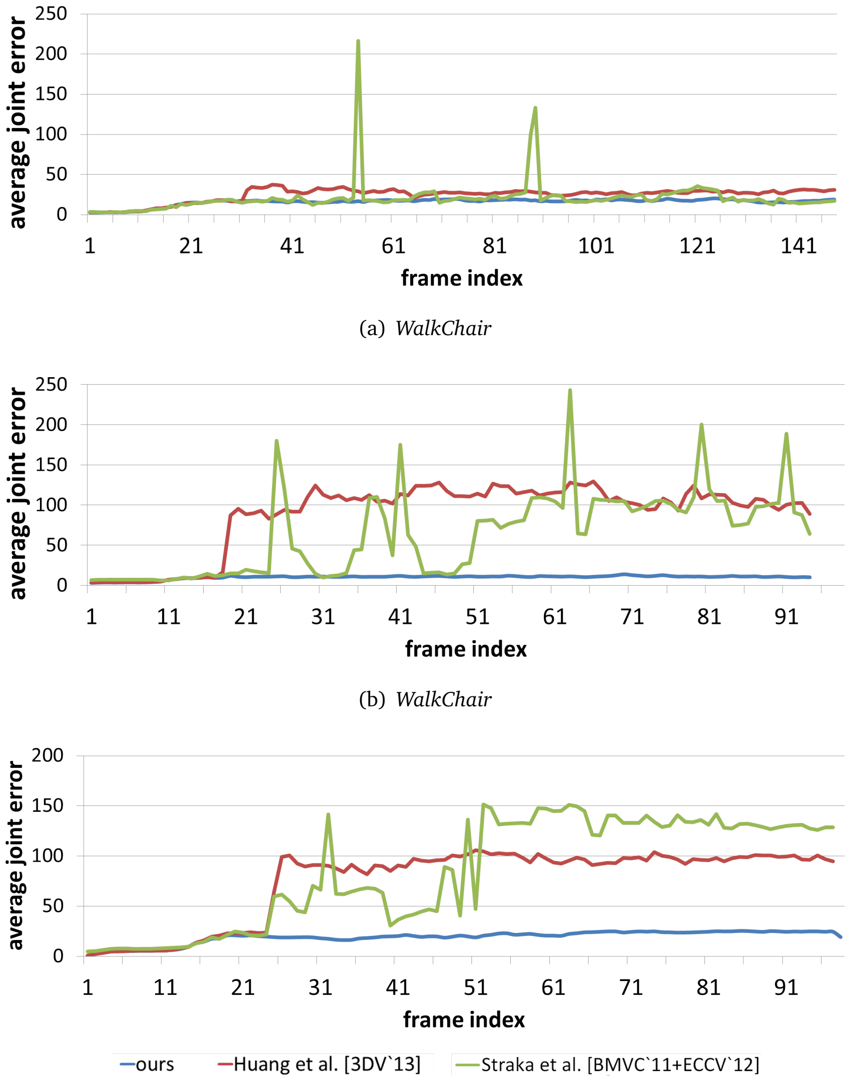


Figure 6.10: **Generated keyframe pool of all testing sequences**. Numbers in the parenthesis are the delay time.



(a) *WalkChair*

(b) *WalkChair*

(c) *WalkChair*

Figure 6.11: **The curves of 2D joint error of three newly recorded sequences**. Image resolution: 1000×1000. **Blue: ours**. Green: Straka *et al.* [131] + [132]. Red: Huang *et al.* [72].

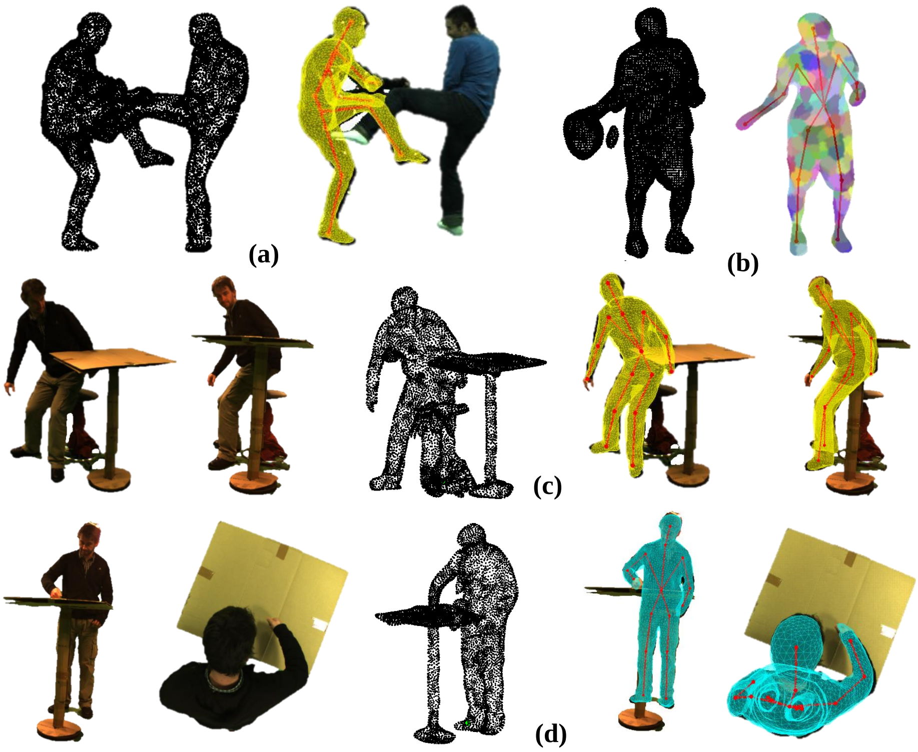


Figure 6.12: **Visual results of (a) *Fighting*, (b) *Basketball*, (c) *SideSit*, and (d) *HammerTable*.** Black dots represent the observed point clouds.

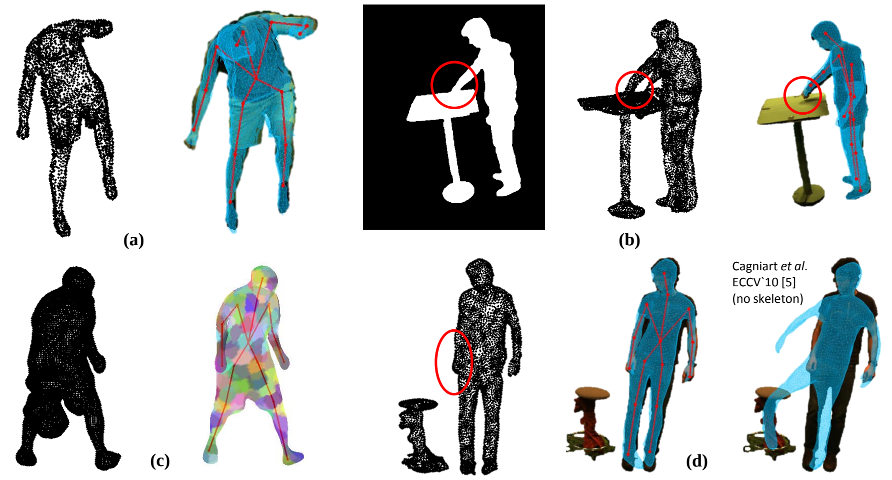


Figure 6.13: **Qualitative results of (a) *Dance*, (b) *HammerTable*, (c) *Basketball*, and (d) *WalkChair*.** Black dots are the point clouds.

7 Conclusion and Outlook

7.1 Summary and Findings

This thesis is devoted to the study of 3D human shape tracking in multi-camera environments. We follow the traditional ICP paradigm and contribute to the two stages of it: *association* and *optimization* steps respectively. We advocate using regression forests to draw discriminative associations, both in surface parameterizations (meshes) and volumetric ones (CVTs). In the optimization step, we do not rely on one particular motion representations but propose a joint optimization framework that simultaneously recovers both surface deformations and skeletal poses. For all these problems, we propose novel algorithms and also provide theoretical derivations. To evaluate our methods, we rely not only on standard multi-view datasets but also introduce a number of challenging sequences that contain outlying geometries. Furthermore, we dedicate a chapter to study the usage of keyframes, which could be eventually employed in the human shape tracking algorithms.

First of all, we commence with the quest for discriminative associations. In Chapter 3, we work with regression forests, introducing a novel discriminative model for 3D human shape correspondences. To profit the most from the strengths of forests, we avoid pure hand-crafted features but resort to a less engineered one, voxel-based offset feature. Each pair of offsets corresponds to one individual feature attribute whose computation is independent from each other. Using such features, the regression forest learns to map every vertex on the input visual hull to the location of its potential matches on the template. In Chapter 4, we argue that the same concept applies to another shape parameterization, volumetric centroidal Vonoroi tessellation as well. For that reason, we devise a Haar-like distance-field-based feature and a multi-template learning strategy to learn from abundance of shapes and motions. We point out the following observations empirically from our experiments: 1) our voxel-based feature performs better than pure surface-based descriptors like Heat/Wave Kernel Signatures; 2) regressing to the whole shape volume yields better accuracy than only to surface domains; 3) when features are 3D-geometry-based, *e.g.* our offset-pair features and the distance-field-based features, unseen poses are more difficult to generalize than unseen shapes due to the drastic changes in geometry. One should hence give pose variations higher priority than shape variations when arranging training data.

In Chapter 5, we orient our research towards the optimization step, where motion parameterizations play a crucial role. The most prevalent choices are non-rigid deformations and articulated human poses. Instead of selecting one from the two, we propose to estimate them jointly in one single energy objective by parameterizing skeletal poses as the function of surface deformations. Thanks to the explanatory power of surface-based approaches, this strategy is sufficiently accurate and yet provides an informative skeletal poses that resembles human anatomical structure. When integrated with the surface-based discriminative associations, they amount to a hybrid human shape tracking approach, which we term tracking-by-detection of human shapes throughout this thesis. It effectively prevents tracking errors from accumulating, yielding more stability compared to other generative ICP extensions. The volumetric discriminative associations also contribute to the volumetric shape tracking in the similar fashion. We thereby present a fully volumetric tracking-by-detection framework. Centroidal Voronoi tessellation is chosen to be the unified representation used in feature computations, predicting domains, and deformation models. Such informative and consistent representations have shown better detected correspondences than other discriminative strategies.

In Chapter 6, we propose a multiple-keyframe-based tracking framework that uses mean-shift clustering to update a keyframe set online. We argue against adhering to the same rest pose throughout tracking and show that exploiting the already-tracked poses of the template is beneficial because the regularization could in turn be enforced upon various patch configurations. The reliability of the proposed method is confirmed by the experiments on various public sequences as well as newly recorded sequences.

In the thesis, all proposed models offer an algorithmic solution to a certain problem under some assumptions. Next, we discuss the limitations of our methods and afterwards we will propose a number of directions for future work.

7.2 Limitations

We work on the problem of 3D human shape tracking in multi-camera environments and deliver a number of methods for addressing it. In Chapter 3 and 4, we apply discriminative models for one-shot 3D human shape correspondences. Avoiding spectral descriptors that are sensitive to topological noise, we directly describe the 3D geometry. We also rule out the histogram-based features to take the most advantages of forests. Our first attempt is voxelizing meshes and transpose the offset-pair features in human pose estimation with RGBD data [121] to our problem. It requires 3D shapes to be close and watertight in order to distinguish internals from externals, which does not always hold. Storing those empty space information also consumes considerable amount of memory, in turn limiting the training data it can incorporate. To overcome this limitation, we then resort to a more memory-efficient volumetric representation, CVT. This indeed significantly reduces the required memory and is definitely a better parameterization to tackle the problem of human shape correspondence estimation. As it allows including more training datasets, we further devise a multi-template learning strategy to learn from a variety of shapes and poses. Nevertheless, a coherent mesh connectivity is always required to yield consistent output labels (3D points in our case), which is in general the limitation of Vitruvian framework. We propose to replace the labels by matching each template to the generic one, which is a workaround but not a solution in the long run.

Putting it in a more general way, since we adopt machine-learning techniques, generalization is inevitably an inherited problem. Namely, the algorithm works reasonably well on the observed training data but maybe not on the unseen testing samples. For example, when the input visual hull is way too different, forests might yield undesirable correspondences that even deteriorate tracking rather than gain robustness. This is the reason why we always advocate combining discriminative associations with the generative ICP-based tracking (initialize discriminatively, refine generatively). As our problem concerns itself with human shapes, the variations of data one should consider range from positions, rotations, poses, scales, to shapes of the subjects. One usually strives to achieve some invariance property at the feature level and leaves the rest to be learned by the learning algorithms. Thinking from the perspective of features and training variations is certainly an informative way to analyze the limitations of models.

Beside the learning algorithms, a more fundamental limitation stems from the observations. As we perform 3D reconstructions from silhouettes to obtain input visual hulls, foreground is supposed to be segmented properly from the background. Currently this only holds in studio environments with uniform and static background, which for sure limits the applicability of our method.

7.3 Future Work

To address the aforementioned limitations, we propose a number of research ideas for future directions.

- **Exploiting photometric information.** To get away from the strong requirement of background subtractions, one has to reason with color information to track. A first attempt could be assuming close initializations, devising a generative ICP-based method (namely, no machine-learning techniques involved) that guides the deformations with only colors. *Blobby man* representations [116, 126, 130] pioneer in this direction and are certainly worth a look.

- **Learned feature.** When it comes to pure geometry-based approach, we believe discriminative associations are definitely one direction to proceed. Our experiences in the former chapters show that, features are critical to the performance. Recently the vision community has witness the success of *deep learning* that learns the features from data instead of engineering them [15]. Some work have been proposed in this direction for shape retrieval and recognition [133, 156].

- **Better output label space.** On the other hand, an uniform and coherent label space where points from all datasets are naturally aligned is also favorable for learning human body correspondences. In this case, our preliminary results in Appendix B suggest that considering skinning weights is one direction to pursue.

7.4 Epilogue

It has been a long, exciting and inspiring way to reach these last lines of the thesis. We have not yet solved the problem of 3D human shape tracking, but we have set the milestone for the future research on this topic and substantially contributed to the computer vision community. We focused on discovering data-model associations discriminatively by utilizing machine learning models. In this thesis, we present models that can be applied to different shape parameterizations and distinct motion parameterizations.

Part IV

Appendix

A

A Comparison Study on Three LCF Approaches

A.1 Introduction

Local coordinate frame (LCF) is a key component deployed in most 3D descriptors for invariant representations of 3D surfaces. Attaching a LCF has been long employed in matching rigid objects but less discussed when it comes to non-rigidly deforming ones. In this appendix we addresses this problem, in particular focusing on humanoid surfaces. We facilitate this by (1) consider the LCF approach in our previous work [71] and (2) extending two current LCF paradigms for rigid surface matching to the non-rigid case. Such an adaptation is motivated by the assumption that interpolating locally rigid movements often amounts to smooth globally non-rigid deformations. All three approaches leverage spatial distributions, based on normals, signed distance and principal component analysis, respectively. To perform a thorough comparison study, ground truth for non-rigid LCFs are synthetically generated by interpolating locally-rigidly transformed LCFs. Therefore, three LCF proposals are evaluated extensively in terms of repeatability, robustness on estimating correspondences, and accuracy of final tracking results. All the experiments demonstrate the benefits of the proposed methods with respect to the state-of-the-art.

A.2 Related Work

This appendix aims at developing a LCF-based approach used for correspondence estimation in 3D human motion tracking. In the following, the previous work is therefore briefly reviewed and discussed from the two different perspectives of LCFs and 3D human motion tracking.

LCFs are usually proposed with their 3D descriptor counterparts. Here, we provide an overview, whereas a comprehensive review and evaluation is available in [109]. Constructing a LCF consists in defining three orthonormal vectors as $[x, y, z]$ axes. To this end, the local geometry has to be taken into account, involving all neighboring points $\mathbf{p}_i$ (hereinafter referred to as *support*) that lie within a sphere of a certain radius centered at the feature point $\mathbf{p}$. Current approaches can be broadly classified into eigenvalue-decomposition-based [101, 142], which establish three axes at once, and methods that identify them one by one separately [41, 71, 109, 110, 157].

The first category relies on computing the Principal Component Analysis (PCA) [79] or EigenValue Decomposition (EVD) [89] of the local spatial distribution of points. In [101], a covariance (or *scatter*) matrix is constructed as:

$$\mathbf{\Sigma} = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{p}_i - \mathbf{p}')(\mathbf{p}_i - \mathbf{p}')^\top, \tag{A.1}$$

where $\mathbf{p}'$ is the centroid of $k+1$ support points $\mathbf{p}_i$. Later in [142], the centroid $\mathbf{p}'$ is replaced with the feature point $\mathbf{p}$ itself for higher efficiency. The contribution of each support point $\mathbf{p}_i$ to the covariance matrix is also weighted by its Euclidean distance to $\mathbf{p}$. The three axes are provided by the three normalized eigenvectors obtained from scatter matrix decomposition. Conventionally, the one with the largest eigenvalue (principal direction) is defined as the x axis, while the one with the smallest eigenvalue is considered as the z axis. One of the major issue of these methods is that EVD defines only the directions of the axes but not their signs, which have to be disambiguated with additional efforts. For instance, the sign of z axis is usually the one that yields positive inner product with the surface normal $\mathbf{n}$.

Another family of work defines three axes individually [41, 71, 109, 110, 157]. Typically, y axis is attained as $z \times x$ to keep the orthogonality constraint. z axis is either the surface normal vector itself [71], the averaged surface normal [157] across the 5-ring neighborhood, or the normal of a fitted plane within a smaller support [41, 109, 110], where the signs are again disambiguated by considering surface normals. More efforts are devoted to facilitate a stable x axis. Some approaches [71, 109, 157] rely on higher order information (normals or curvatures) to identify prominent geometry and determine the orientations of x axis. Instead of high order information which is prone to be noisy in visual 3D data, [41, 110] considers the signed distances of each support point to the tangent plane defined by z axis, and yields more repeatability as demonstrated in [110].

A.3 Method

A.3.1 Preliminaries and Overview

We first provide a overview of [71] and then outline our method. A human surface is denoted as $\mathcal{M} = (\mathbf{M}, \mathcal{T})$, where $\mathbf{M} = \{\mathbf{x}_v\}_{v=1}^{N_v} \subset \mathbb{R}^3$ are the locations of vertices v, and $\mathcal{T}$ defines the triangles. To attach a LCF for each vertex v, Huang *et al.* consider its normal $\mathbf{n}_v$ as z axis, and search for a reference vector in a local cuboid to establish x axis. Formally, the surface is first voxelized

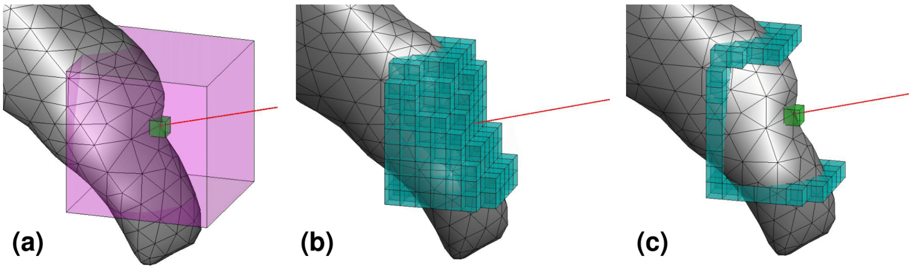


Figure A.1: **Cuboid neighborhood and support voxels.** (a) visualizes the volumetric framework. Green: voxel $\mathbf{v}_v$ mapped by the current vertex v; pink: cuboid neighborhood; red line: normal vector. (b) The LCF method in [71] and EVD involve all the surface voxels (cyan) within the cuboid, whereas (c) SignDist. considers only those lying on the border. Best viewed in colors and in pdf.

into a volumetric field $\mathbf{N} : \Omega_3 \subset \mathbb{R}^3 \to \mathbb{R}^3$, where each voxel $\mathbf{v}$ holds either a unit-length normal $\mathbf{n}$ averaged from the containing triangles, or a number indicating inside or outside:

$$\mathbf{N}(\mathbf{v}) = \begin{cases} +\boldsymbol{\epsilon} & \text{if lies outside surfaces} \\ \mathbf{n} \in [-1, +1]^3 & \text{if overlap with surfaces} \\ -\boldsymbol{\epsilon} & \text{if lies inside surfaces.} \end{cases} \tag{A.2}$$

A vertex v is first mapped to a voxel $\mathbf{v}_v$[1], by discretization of the space. We consider a cubic support of neighbors centered on the voxel $\mathbf{v}_v$, $\mathcal{S} \subset \Omega_3$, as depicted in Fig A.1(a). A surface voxel $\mathbf{v}$ is selected based on the following criteria:

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v} \in \mathcal{S}} \left(\hat{d} + \mathbf{N}(\mathbf{v}_v)^\top \mathbf{N}(\mathbf{v}) \right). \tag{A.3}$$

where $\hat{d}$ is the distance between $\mathbf{v}_v$ and $\mathbf{v}$, normalized with respect to the size of cuboids. The projection of $(\hat{\mathbf{v}} - \mathbf{v})$ onto the plane define by $\mathbf{n}_v$ is then taken as x axis of the LCF. Finally, the y axis is obtained as $z \times x$.

We retain the volumetric framework to keep the property of organized data, *i.e.* , accessing spatial neighbors simply by indexing without iteratively parsing the triangles like nearest neighbor search. We also consider $\mathbf{n}_v$ as z axis, and obtain y axis as $z \times x$. Differently, we pay more attention on the characteristic voxel/vector for x axis. Eq. A.3 favors the voxel that is far from the center voxel $\mathbf{v}_v$, and yet holds least normal changes. As an important trait of our approach, since it is well known that higher order information, *e.g.* normals and curvatures in visual 3D data, are particularly noise prone, we resort to rely on robust zeroth-order information such as spatial distributions.

[1] With a slight abuse of notations, in the remainder of this chapter, $\mathbf{v}$ refers only to voxels that overlap with meshes (intersected with either vertices or triangles), since the other two cases, inside and outside, are both not of our interest. In particular, $\mathbf{v}_v$ refers to voxels containing mesh vertices.

A.3.2 LCF Proposals

Analogous to what proposed in [71], the state-of-the-art methods in the field of LCFs for rigid matching of 3D meshes and point clouds mainly rely on the neighboring points within a local support [41, 101, 107, 110, 142]. As reviewed in Section 3.2, the way they leverage spatial distributions can in general be classified into two categories: (1) EigenValue-Decomposition (EVD) [101, 107, 142], and (2) signed distance (SignDist.) [41, 110]. In the following, we propose an adaptation of both classes of methods to volumetric representations, so to be able to use them within the human motion tracking framework.

EVD. Methods within this class define the LCF as the principal directions of the point distribution within the support. Since the z axis is already defined, we project all the $N_{\mathbf{v}}$ support surface voxels $\mathbf{v}$ onto the plane defined by $\mathbf{n}_v$, denoted as $\bar{\mathbf{v}}$. This way the resulting vectors defining the principal directions lie naturally on the xy plane. It is given by the normalized eigenvectors of the covariance matrix:

$$\mathbf{\Sigma}_{\mathcal{S}} = \frac{1}{N_{\mathbf{v}} - 1} \sum_{\bar{\mathbf{v}} \in \bar{\mathcal{S}}} (\bar{\mathbf{v}} - \mathbf{v}_v)(\bar{\mathbf{v}} - \mathbf{v}_v)^\top, \tag{A.4}$$

where $\bar{\mathcal{S}}$ is the projection of all surface voxels falling within the support. The centroid of $\bar{\mathcal{S}}$ is replaced with the voxel $\mathbf{v}_v$ itself to speed up the computation, without decreasing much repeatability as in [142]. The eigenvector of largest eigenvalue is chosen as x axis. Note that at this point, the sign of the x axis is not uniquely determined, due to the inherent ambiguity of the sign of the eigenvectors obtained from the EVD process [142]. Because of this, the computed LCF might flip 180° along the z axis. In our work [75], we propose a specific feature to tackle this undesired effect, so to make the overall approach invariant to such ambiguity.

SignDist. This class of approaches look for a discerning point within the support. As contrasted in Fig. A.1(b-c), the search involves typically only the peripheral points $\tilde{\mathbf{v}}$ lying on the intersection of the cuboid border and the surface, unlike EVD-based method where all points contribute to the covariance matrix. The discernibility is defined as the maximum signed distance to the tangent plane [41].

We propose to adapt this idea to volumetric representations as follows:

$$\hat{\mathbf{v}} = \arg\max_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \left((\tilde{\mathbf{v}} - \mathbf{v}_v)^\top \mathbf{n}_v \right). \tag{A.5}$$

where $\tilde{\mathcal{S}}$ is the intersection of the surface and the border of local cuboids. The x axis is the projection of the vector directed from $\mathbf{v}_v$ towards $\hat{\mathbf{v}}$. Note that there is no guarantee that the discerning point $\hat{\mathbf{v}}$ from Eq. A.5 is always repeatable: in particular, if different directions yield similar values of the signed distance, the x axis will be ambiguous, hence the resulting LCFs could rotate about the z axis. Purposely, we incorporate such an ambiguity in our new proposed features to make our method robust to this phenomenon, as explained in the next section.

Sequence	Frames	Metric	Subject / #Vertex
Crane [148]	173	A, B, C	S1 / 3407
Jumping [148]	149		
Handstand [148]	149		S2 / 3848
Bouncing [148]	173		
Cutting [73]	81		S3 / 5211
Hammer [73]	93		

Table A.1: **Sequences used in the experiments in Sect. A.4**. Depending on the evaluation tasks, we apply different error measures. A: deviated angle. B: repeatability score as defined in [110]. C: vertex index for correspondences.

A.4 Experimental Results

In this section, we evaluate our approach under two aspects. First, we verify the repeatability of the proposed LCF methods by measuring how much they deviate from the ground truth. Secondly, we demonstrate the benefits of averaged LCF features for 3D human motion tracking by measuring how much they improve the correspondence task between input data and the template surface. The profiles of our sequences are summarized in Table A.1.

A.4.1 Ground truth generation

Due to the lack of ground truth LCFs from real visual data, we resort to synthetic transformations. For each method to be evaluated, we first compute a LCF for each vertex on the reference mesh $\mathcal{M}^0$, denoted as LCF^0 and depicted in Fig. A.2(a). The mesh is then animated with a data-driven patch-based approach [35], and the goal is to see if the newly obtained LCFs follow such transformations. This deformation framework models global non-rigidness as a sparse set of control bases called *patches* that move locally rigidly. The animation is done by tracking with real visual hulls as input, so that we have realistic deformations.

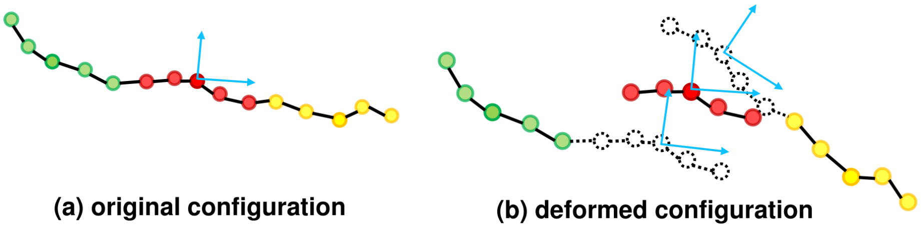


Figure A.2: **Synthetically-generated ground truth LCFs**. A LCF is first computed in the original configuration in (a). When the mesh deforms, the new LCF is the linear combination of the predicted LCFs from the neighboring patches.

Specifically, a mesh is decomposed into several small patches. Each patch k has a rigid body motion $(\mathbf{R}_k, \mathbf{t}_k)$. When the mesh $\mathcal{M}^0$ deforms into $\mathcal{M}^t$, the new vertex position is the linear combination of its transformed location and all the predictions from the neighboring patches $\mathcal{N}_k$, visualized as dot circles in Fig. A.2(b). The final position of each vertex is determined by interpolating their predicted locations from the neighboring patches, where the coefficients α_k encode the desired physical property and are normalized to sum up to 1. More details can be found in [35]. The attached LCF^0 follows the same operations:

$$LCF_k = \mathbf{R}_k LCF^0, \tag{A.6}$$

$$LCF' = \sum_{s \in k \cup \mathcal{N}_k} \alpha_s LCF_s. \tag{A.7}$$

In Eq. A.6, it is rotated according to either its own rigid body motion, or those from the neighboring patches. In Eq. A.7, multiple predictions, *e.g.* blue LCFs in Fig. A.2(b), are blended to yield the final local coordinate frame LCF' (all axes normalized to unit length), which is considered as ground truth in our experiments.

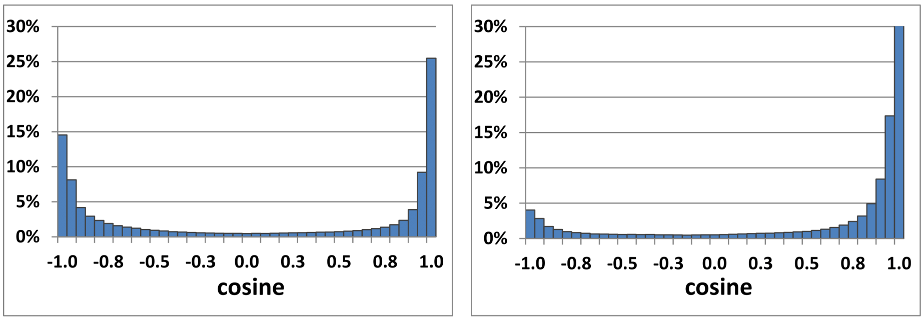
A.4.2 LCF Repeatability

We compute our LCF methods, *i.e.* EVD (Eq. A.4) and SignDist (Eq. A.5), for all vertices on $\mathcal{M}^t$, and check respectively their discrepancies to the ground truth LCF' by computing cosine scores as in [109]. Fig. A.5 shows the results in degrees for each sequences, aggregated by averaging across every frame. For z-axes, all the approaches yield consistently low errors regardless the radius of the cuboid. For x-axes, on the other hand, the differences are more obvious. We see that the error suggests a monotonous decreasing trend when the support gets larger. EVD has relatively poor performance with small radius, due to the fact that the support is insufficient to fully characterize the local point distributions. SignDist. shares the same concern in smallest cuboid size 7, but the error drops faster than EVD and attains most of the time the best results also with respect to [71]. In the remainder of this section, we always use the largest cuboid size 15 for comparisons and analysis.

We also apply the repeatability score $\bar{A}$ in [110], which is defined as the number of points whose cosine scores are higher than a certain threshold T_A, measured in percentage and averaged across every frames in the sequence. The scores in varying thresholds are presented in Fig. A.6. It actually reflects the angle errors in Fig. A.5 faithfully. SignDist. attains better (S1 and S2) or comparable (S3) repeatability compared with [71]. This confirms that spatial distributions are more reliable in 3D data than higher order information such as normals used in [71]. On the other hand, we notice that EVD does not always present such a merit (S3), which is worth further investigation.

To understand why EVD performs badly in S3, we look into the cosine scores for x-axes, and plot the histogram in Fig. A.3(a). One can clearly see that a considerable portion of vertices have scores less than -0.8, presenting a bimodal distribution. This indicates that EVD method indeed suffers from the sign-ambiguity and explains the low repeatability score. For SignDist, the similar problem also exists but, as indicated in Fig. A.3(b), is much less severe. In

Fig. A.4, we visualize for each vertex of a 3D mesh taken from the test sequences the angle error of x-axes in colors and show the distribution of cosine scores for each method. Overall, we can qualitatively observe how both SignDist. and EVD attain in average better cosine scores than [71]. Interestingly, while SignDist. has a more scattered error, EVD shows a more piece-wise distribution (compare, *e.g.* , the left leg with the right leg), this highlighting the previously discussed disambiguation problem affecting EVD-based LCFs. In addition, the plotted cosine distributions in Fig. A.4 again confirm the aforementioned discussions.



(a) EVD

(b) SignDist.

Figure A.3: **The distributions of cosine scores of x axes for two different methods on *Hammer* and *Cutting*.**

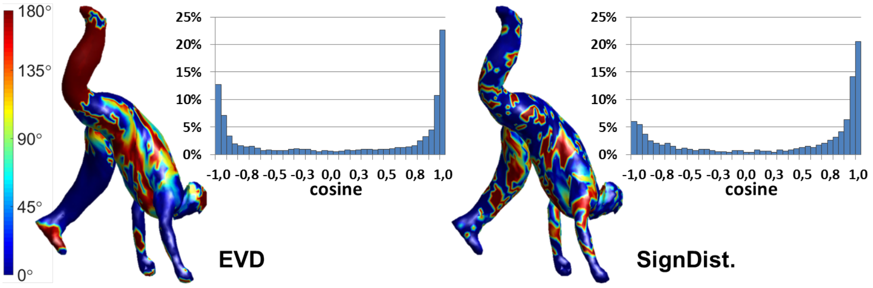


Figure A.4: **Qualitative results on *Handstand*.** From left to right, the averaged cosine score is 0.565 and 0.626, respectively.

A.4.3 Correspondences prediction

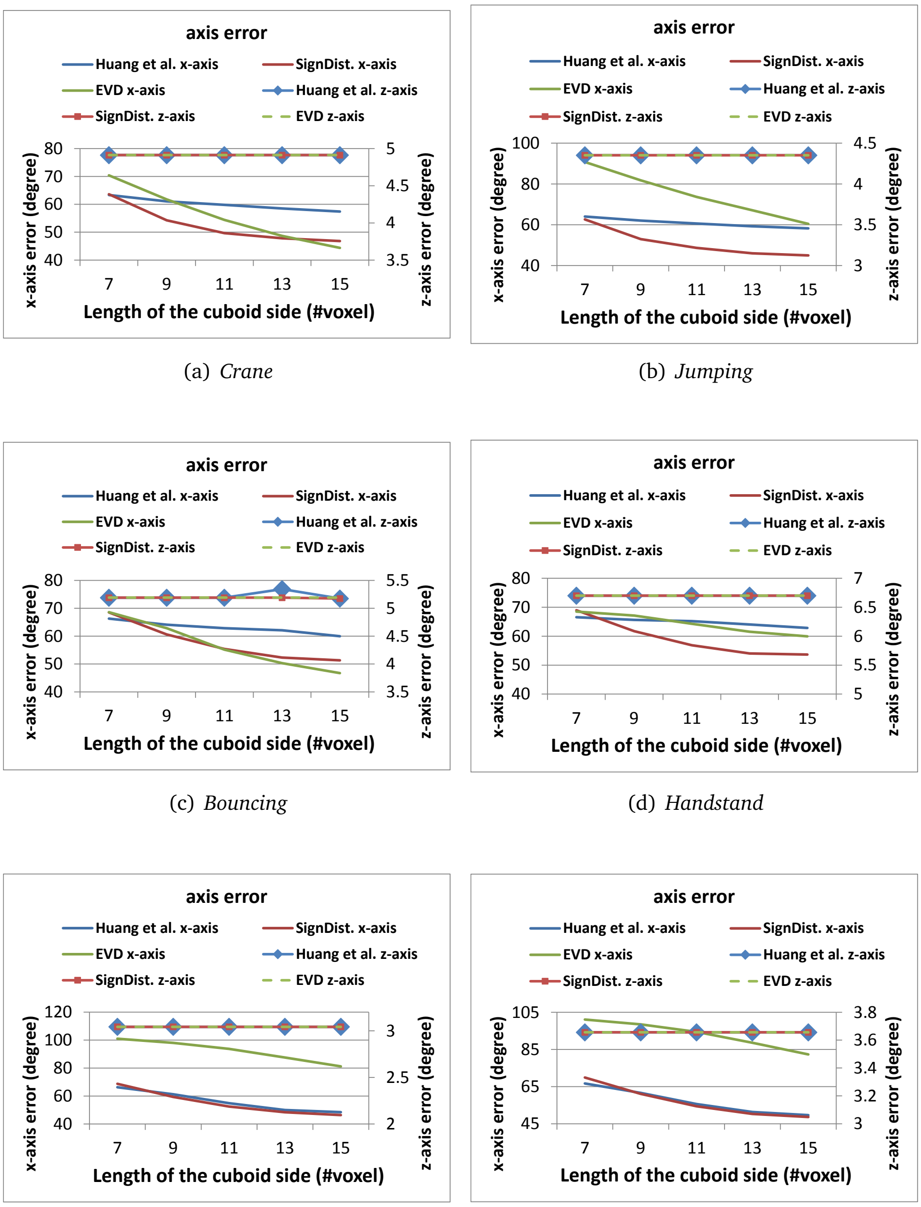
In addition to previous results, we investigate how different LCF strategies influence surface matching in terms of accuracy of the retrieved correspondences. The task is to find correspondences between input meshes and the reference template $\mathbf{X}^0$. For each subject, we learn separate regression forests using 3 LCF methods (length of cuboid side 15): Huang *et al.* [71], SignDist. (Eq. A.5), and EVD (Eq. A.4) respectively. For the latter two cases, we also consider the proposed extensions to averaged representation. To draw a fair comparison, all the other parameters remain the same: 20 trees, 15000 testing neighbor pairs ψ at each branch node, and maximum tree depth 20. We consider the animated meshes $\mathbf{X}^t$ as input data, where ground truth vertex indices are available. The error measure here is the geodesic distances on template surfaces $\mathbf{X}^0$.

If the geodesic distance between the estimated and ground truth vertex indices is lower than a certain threshold, we consider it as a correct match. Fig. A.7 shows the percentage of right matches in varying thresholds. We highlight half of the length of lower arms in orange dashed line for better interpretation of the estimated correspondences. Two observations can be remarked. First of all, among the 3 approaches, SignDist. always yields more than 90% of matches that falls within the range of half of lower arms and achieve best results. Secondly, the averaged representations are consistently better than their counterparts. These observations confirm that the proposed LCF methods improve the state-of-the-art and furthermore, the strategy of averaged LCF results in more invariant representations, and hence better matching accuracy.

Last but not least, since the end application is human mocap, we evaluate the tracking results as well. Given the data-model associations from the forests trained with different LCF approaches, we deform the template surfaces using the method similar to [72]. Differently, we do not run ICP refinements, so as to see the direct impact of correspondences in the end results. The metric is the silhouette overlap error that measures the discrepancy between the deformed reference surfaces and the silhouettes. As reported in Fig. A.8, the averaged LCF method always yield lower errors than their counterparts. Moreover, both presented LCF methods improve the tracking compared to the LCF in [72].

A.5 Conclusion

In this appendix, we study methods that attaches LCFs to non-rigidly deforming surfaces, with the goal to facilitate correspondence tasks in 3D human motion tracking. The non-rigidness of human motions is approximated as the interpolations of several locally-rigid motions. We then adapt two LCF paradigms for rigid surface matching to the non-rigid case. In addition, we incorporate the sources of unrepeatability in learning, and present a more invariant representation, sparing the efforts of devising robust LCFs and yet maintaining the descriptiveness of features. Ground truth LCFs are also produced locally rigidly. Our methods are thereby evaluated thoroughly, and the experiments suggest that the proposed LCFs attain higher repeatability than the state-of-the-art approaches, the new representations from multiple LCFs yield improved correspondences than their counterparts, and, in turn, better tracking results.



(a) *Crane*

(b) *Jumping*

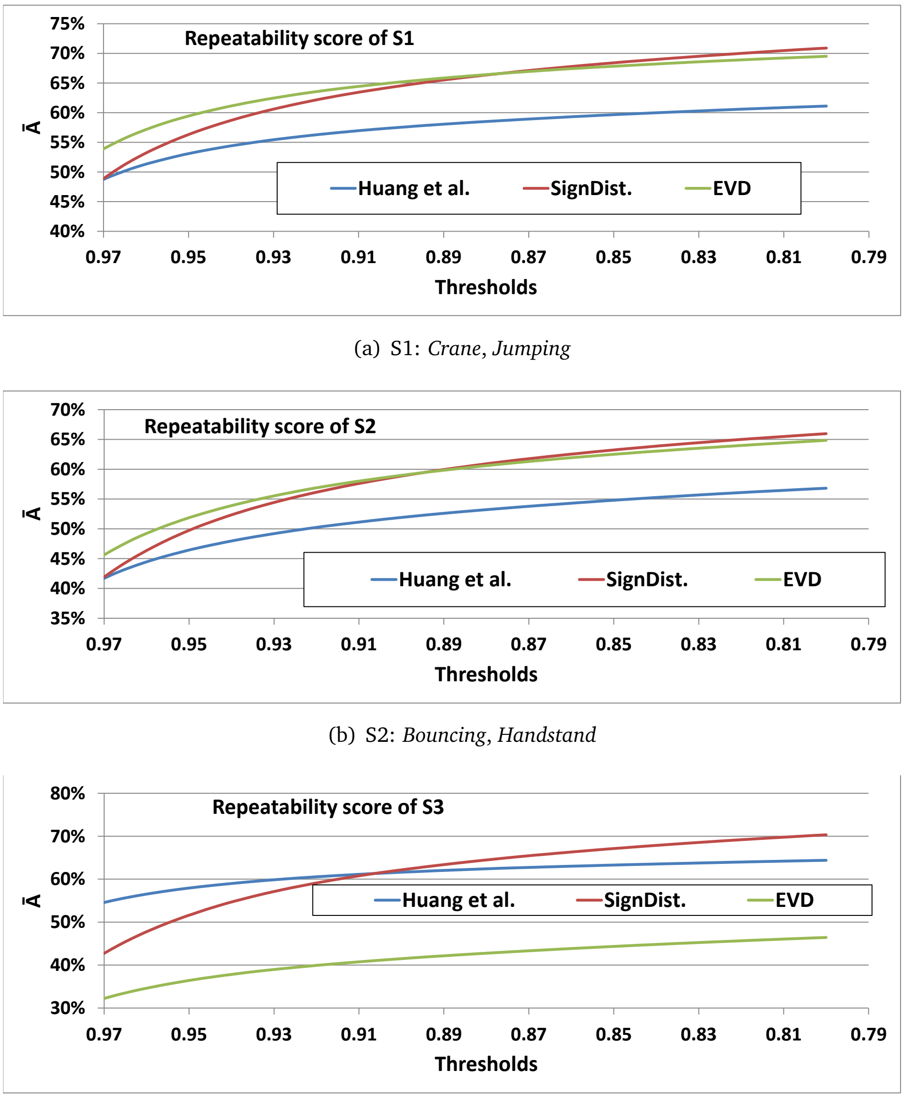
(c) *Bouncing*

(d) *Handstand*

(e) *Hammer*

(f) *Cutting*

Figure A.5: **Cosine error of six sequences**. Error reported by the three evaluated LCF methods with varying cuboid sizes, measured in degrees. Left vertical axes: error of x axes; right vertical axes: error of z axes. In general,

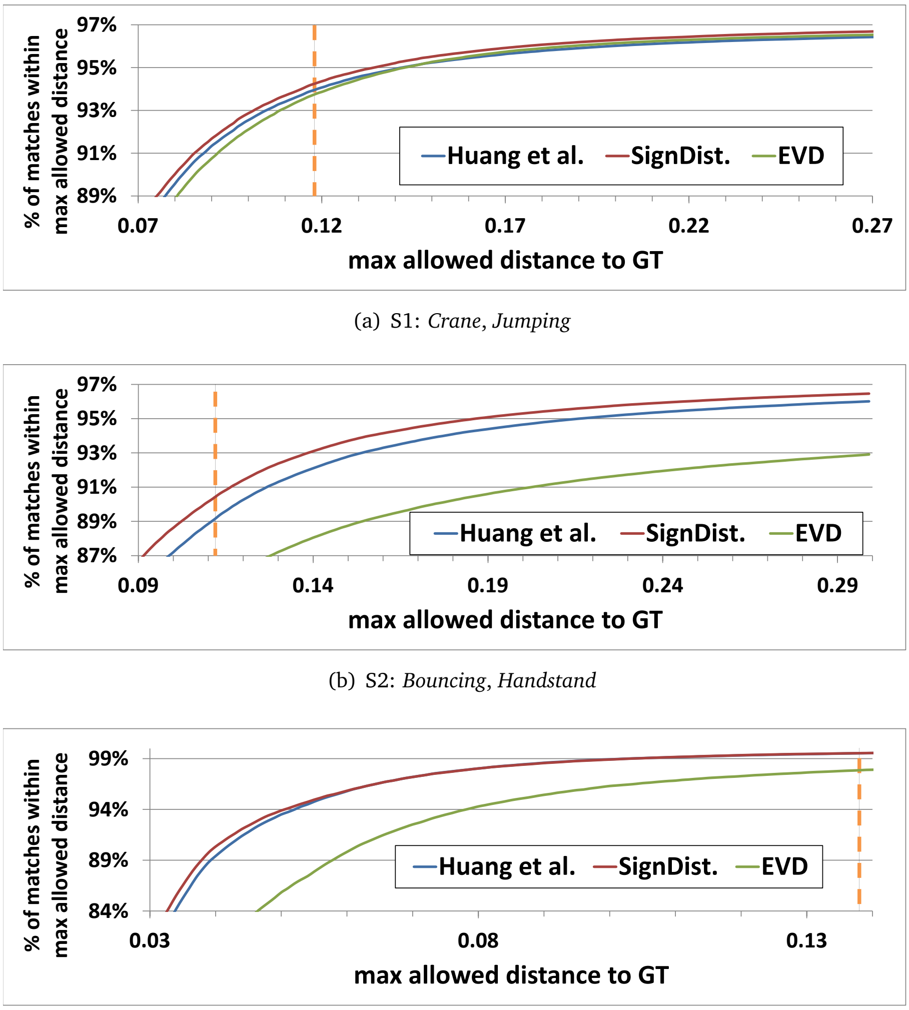


(a) S1: *Crane*, *Jumping*

(b) S2: *Bouncing*, *Handstand*

(c) S3: *HammerTable*, *Cutting*

Figure A.6: **Repeatability scores $\bar{A}$ on three subjects in varying thresholds** T_A. Cuboid size: 15.



(a) S1: *Crane*, *Jumping*

(b) S2: *Bouncing*, *Handstand*

(c) S3: *Hammer*, *Cutting*

Figure A.7: **Comparison of the correspondence error from different LCF strategies for all subjects.**

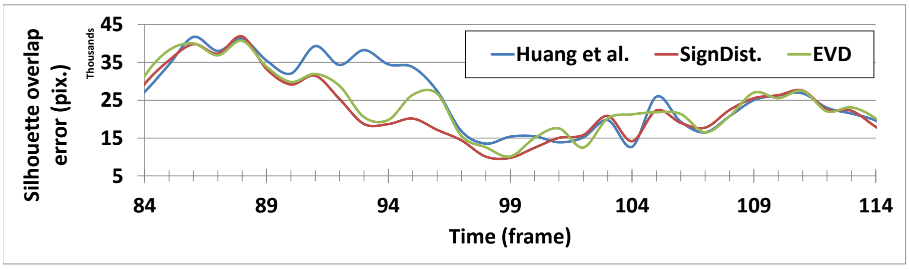


Figure A.8: **Comparisons of tracking error on *Jumping*.** Image resolution: 1920 × 1080.

B
Skinning Weights as the Label Space to Learn

B.1 Introduction

As demonstrated in Chapter 4 and Chapter 5, discriminative data-model associations indeed result in a more robust tracking pipeline, both in surface-based and volume-based representations. The biggest requirement of discriminative strategy is offline training. To learn the associations, several label spaces $\mathcal{L}$ have been considered in the literature. We briefly analyze their pros and cons in below.

On one hand, people cast matching as a classification problem [117], where learning machines output categorical labels that are actually the vertex indices on the reference surface: $\mathcal{L} = \{1 \cdots v \cdots V\}$. On the other hand, another family of work perform regression [71, 112, 140], where trained models predict points in continuous 3D space $\mathbb{R}^3$, or more specifically, points on a 2-manifold defined by the template, termed *Vitruvian manifold*: $\mathcal{L} = \partial\Omega_X \subset \mathbb{R}^3$. The associations are built by an additional nearest neighbor search (NN) on the template $\mathbf{X}^0$. During training, both approaches require a reference surface whose mesh connectivity, *i.e.* vertex index, is shared among all training meshes. The trained classifiers/regressors apparently can only be applied for matching against this reference surface, unless other approaches are devised to align the labels as we do in Section 4.5.2. For this reason, in order to use the abundant online-available multiple-view datasets [5, 62, 73, 128, 148], these approaches need to align distinct mesh topologies in advance to yield consistent labels. Instead, they often resort to human shape models such as SCAPE [7] or SMPL [99] that generate topology-consistent human surfaces in various shapes and poses. However, meshes generated by these models are usually naked or in tight suits [19, 111], whereas multiple-view sequences are recorded with subjects in real apparel.

In [154], Wei *et al.* take a hybrid approach. Although the learning machine perform classification tasks, the correspondences are actually established by a NN search in the feature space learned by convolutional neural networks (CNN) [90]. In both this work and Vitruvian-manifold methods, the aim of learning algorithms

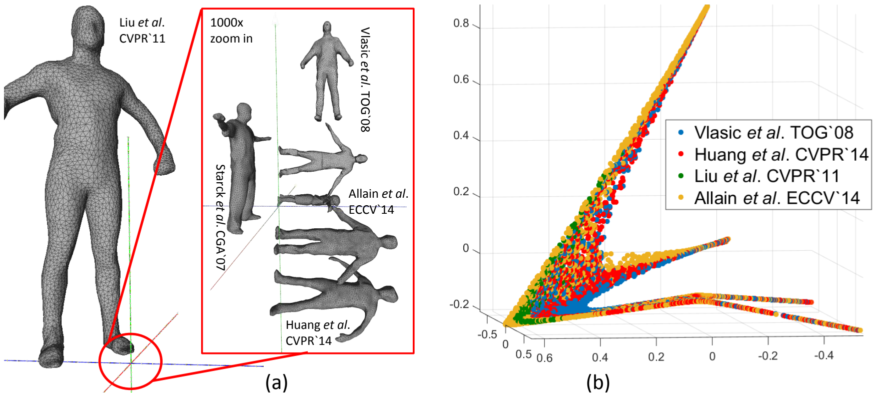


Figure B.1: **Intuition of the skinning-weight space regression approach**. (a) Templates of various multi-view datasets shown in the same $\mathbb{R}^3$ space. (b) The distributions of four templates in the skinning-weight space (pca is performed for visualization purposes only). We advocate to learn in skinning-weight space as vertices from different datasets are naturally aligned.

is not to match points directly but rather to conduct a transformation, mapping the input to a space where the actual matching procedure (a NN search) takes place. In the case of Vitruvian-manifold, this space is a 2-manifold $\partial\Omega_X$ in 3D defined by the reference surface $\mathbf{X}^0$ (the label space that forests regress to), while in [154], it is the feature space that CNNs learn from the data. To include different datasets into training, authors in [154] manually annotate several semantic salient points to roughly align the topologies. To benefit from as many captured realistic deformations as possible, one needs a more effortless way to learn across heterogeneous mesh connectivities. To this end, we follow the feature-transformation paradigm but advocate a space characterized by *skinning weights* [92]. When considering it as a label space $\mathcal{L}$ to learn, one does not have to take care of topology alignment because points from different templates are naturally aligned in this skinning weight space as shown in Fig. B.1.

In short summary, this work has the following contributions. To the best of our knowledge, we are the first ones that consider skinning weights as labels to learn. A great advantage is the spared effort on aligning distinct mesh connectivities. Thus, one can easily make use of numerous multi-view sequences to enrich the training set, thereby bringing dense human-surface correspondence learning to a larger scale. For this purpose, we propose a method that exploits random forests to estimate vertex-wise skinning weights for human meshes in arbitrary poses without the need to rig a skeleton.

B.2 Method

B.2.1 Skinning weights

Skinning weights usually come from skeleton-rigging process. Aiming to animate articulated characters such as virtual humans, it is a fundamental operation in computer graphics and interactive applications. Character skinning has a long history in graphics. Related techniques have been used extensively to provide intuitive animation controls. Readers are referred to the related work section of [92] for a comprehensive summary. Approaches for rigging character skins by weighting vertices to an associated skeleton, or by interpolating example deformations, are widely used in video games and the computer animation industry. There are several reasons for their popularity: most skinning approaches are conceptually easy to understand and apply; they are capable of approximating interesting character shapes; the complex mesh deformations are compressed into a pose space to avoid transmitting raw vertex positions; and skinning can be hardware-accelerated on almost every commodity graphics card. Among these reasons, data compression and real-time rendering have wide industrial applications and have therefore been long studied and investigated [66, 77, 96].

In principle, one attempts to answer the following two questions when devising rigging algorithms:

1. how to place joints j properly inside meshes?

2. how the transformations of joints j influence vertex v?

As the first question is rather ill-posed, many off-the-shore graphic softwares, such as `Maya` and `Blender`, still rely on users' manual assistance to determine the locations of joints inside human bodies. For the second question, a standard and straightforward approach is linear averaging, which leads to the well-known linear-blend skinning formulation briefly mentioned in Section 5.3 (Eq. 5.9):

$$\begin{gathered}\mathbf{x}_v = \sum_j w_{v,j} \mathbf{T}_j (\mathbf{T}_j^0)^{-1} \mathbf{x}_v^0, \qquad \text{(B.1)}\\ \forall w_{v,j} : w_{v,j} \geq 0, \ \sum_j w_{v,j} = 1.\end{gathered}$$

A vertex v is first represented in the local coordinate frame of joint j (multiplied by the inverse of $\mathbf{T}_j^0$) and then brought to a new position ($\mathbf{T}_j$). This is repeated for all joints and the final location is simply the linear combination of them. The *skinning weights* $w_{v,j}$ are supposed to place different emphasis on each joint j and are sum up to 1. Given B non-terminate-point joints that have transformations, one can define a B-vector $\mathbf{w}_v$ that contains all the weights for one single vertex v, *i.e.* $\mathbf{w}_v \subset [0,1]^B$. As depicted in Fig. B.2(a), vertex on the thigh has high weight on hip joint, while vertex around the neck has non-zero weights for both neck and shoulder. The vector $\mathbf{w}_v$ apparently encodes the dependency of v on human body parts and in turn offers a strong cue on its location along the human surface.

A closer look on the rigging process helps understanding some useful properties of skinning weights $\mathbf{w}_v$. Rigging is typically done offline just once because it is unnecessary and further, impractical to re-rig the skeleton for each frame during animation. We follow the approach in [11] to offline rig the skeleton. Users are required to rotate surfaces into one unified direction (facing outward the screen) to distinguish symmetric body parts. Meshes are normalized into a

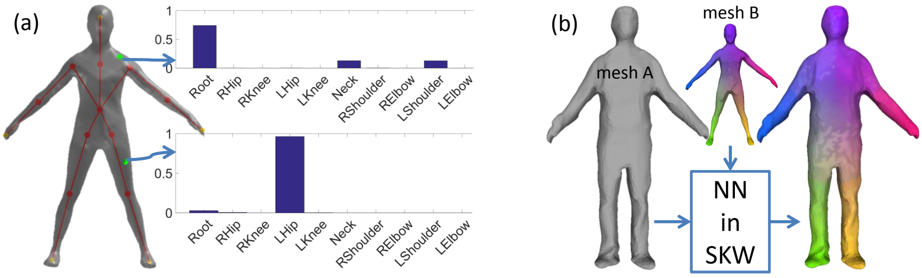


Figure B.2: **Visualization of skinning weights and its descriptiveness.** (a) The illustration of skeletons and skinning weight vector $\mathbf{w}_v$. The leaf-node joints (yellow) are excluded in $\mathbf{w}_v$. (b) A simple nearest neighbor search (NN) in skinning weight space can yield plausible matches between mesh A and B. Best viewed in pdf.

unit bounding box to yield scale invariance. Joints are placed by a maximum margin classifier [33] that learns from various pre-rigged human shapes for better generality. When the locations of joints are determined, skinning weights $w_{v,j}$ are computed by heat diffusion. All these contribute to an informative representation independent of the changes in scale, rotation and shape. See Fig. B.1(b) for a real visualization. When used as a descriptor, skinning weights offer satisfactory matches, as shown in Fig. B.2(b) and also confirmed in Section 4.5.2 above. Nevertheless, the rigging process in [11] applies only for meshes in the rest pose. This motivates us to contrive a method that estimates skinning weights in arbitrary human poses.

Generally speaking, if the skeleton does not slide inside meshes, the relative location of a vertex with respect the joints and consequently the skinning weight $\mathbf{w}_v$, can be considered unchanged. As such, it is theoretically also invariant to poses changes, orientations and translations, which are indispensable properties for a good descriptor. We exploit this assumption to prepare training meshes in all kinds of poses. Regression forests are again applied to learn from exemplar data and to predict skinning weights online for a mesh in an arbitrary pose, so that we can make use of them to estimate the correspondences $\mathcal{C}$.

B.2.2 Random forest

As briefly reviewed in Sect. 2.3, a forest is an ensemble of T binary decision trees, each separating data with split functions at branch nodes and storing statistic models at leaf nodes. The training objectives, split functions, and statistical models vary from task to task. Random forests are employed for regression tasks in a B-dimensional skinning-weight space. Instead of training one single forests for B-dimensional regression, which bears the concerns of curse of dimensionality, we train one forest $\mathcal{R}_j$ for each individual skinning-weight channel $w_{\cdot,j}$ and normalize the regression results to unit-sum. We explain in the following description only the training and prediction process of one skinning-weight attribute. It can be easily generalized to all the B forests. Training and prediction procedure is visualized in Fig B.3.

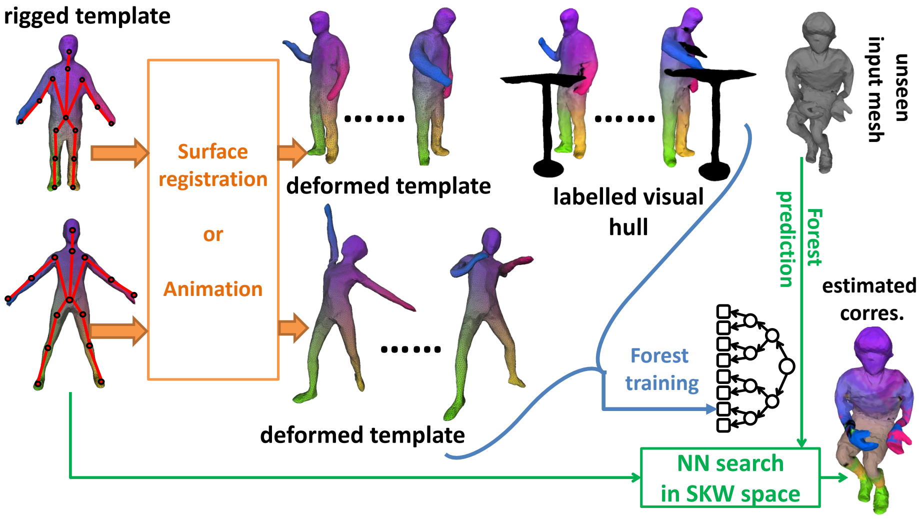


Figure B.3: **Pipeline of the training and prediction process**. This framework is able to integrate training meshes generated from templates with distinct mesh connectivity.

Feature and forest training

As visualized in Fig. B.4, we apply scale-invariant Heat Kernel Signature [29] and Wave Kernel Signature [9] to represent each vertex v as a 50-dimensional vector $\mathbf{f}$, each of which is also associated to a skinning weight label w_v[1], forming a pool of sample-label pairs $\mathcal{D} = \{(\mathbf{f}_v, w_v)\}$. Each tree is trained with a random subset of $\mathcal{D}$. Our splitting candidate ϕ is the pair of testing channels κ and thresholds τ, $\phi = (\kappa, \tau)$. Let $\mathcal{D}_N$ denotes the samples arriving at a certain branch node. The training process is to partition $\mathcal{D}_N$ recursively into two subsets $\mathcal{D}_L$ and $\mathcal{D}_R$, based on randomly generated ϕ:

$$\mathcal{D}_L(\phi) = \{v \in \mathcal{D}_N | f_{v,\kappa} \geq \tau\} \tag{B.2a}$$
$$\mathcal{D}_R(\phi) = \{v \in \mathcal{D}_N | f_{v,\kappa} < \tau\}. \tag{B.2b}$$

The tree recursively splits samples and grows until one of the following stopping criteria is true: (1) it reaches the maximum depth, or (2) the number of samples $|\mathcal{D}_N|$ is too small. Once a leaf node is reached, we perform mean-shift clustering [39] to represent the distributions of $\mathbf{x}_v$ as a set of confidence-weighted modes $\mathcal{H} = \{(\mathbf{h}, \omega)\}$. $\mathbf{h} \in \mathbb{R}^3$ is the mode location and ω is a scalar weight.

Forest prediction

In the prediction phase, a input point i lands on T leaves containing different collections of modes, denoted together as $\mathcal{G} = \{\mathcal{H}_1 \cdots \mathcal{H}_T\}$. We aggregation them by doing mean-shift clustering [39] and keeping the cluster with largest weight. This process is repeated for the forests $\mathcal{R}_j$ of all skinning-weight channel

[1] The suffix j is dropped because we describe the training process of one single forest.

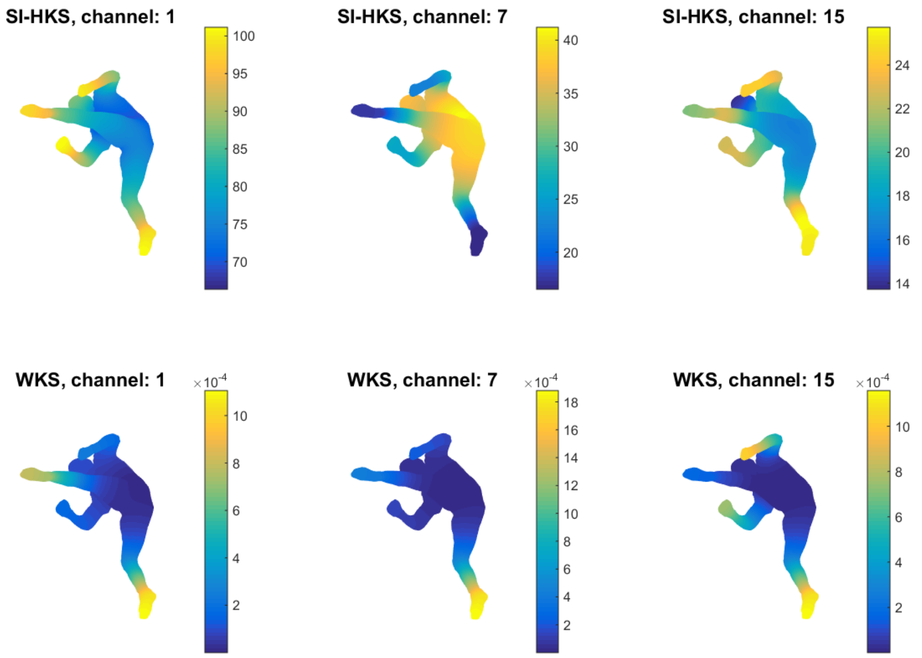


Figure B.4: **Visualization of extracted SI-HKS and WKS response**. One sees that different feature channels encode different body part information.

j and the results are stacked into a B-dimensional vector $\mathbf{w}_i$, followed by a unit-sum normalization.

Nearest neighbor search. Given the regression results $\mathbf{w}_i$, each point i in input gets a closest vertex p in the reference vertex set $\mathcal{V}$:

$$p = \operatorname*{argmin}_{v \in \mathcal{V}} \|\mathbf{w}_i - \mathbf{w}_v\|_2. \tag{B.3}$$

We apply one intuitive sanity check to reject the potential erroneous correspondences. Since the body-part label is actually the attribute with the highest skinning-weight, we reject p if it has different body-part label with i.

B.3 Preliminary Results and Conclusion

We brief validate the benefit of using skinning weight as label space with *Faust* [19]. As shown in Fig. B.5, it yields comparable accuracy (SKW) with using the 2-manifold defined by the template (Vitruvian). However, the forest trained with Vitruvian strategy apparently applies only for matching against the template, while our approach can match meshes with heterogeneous connectivities.

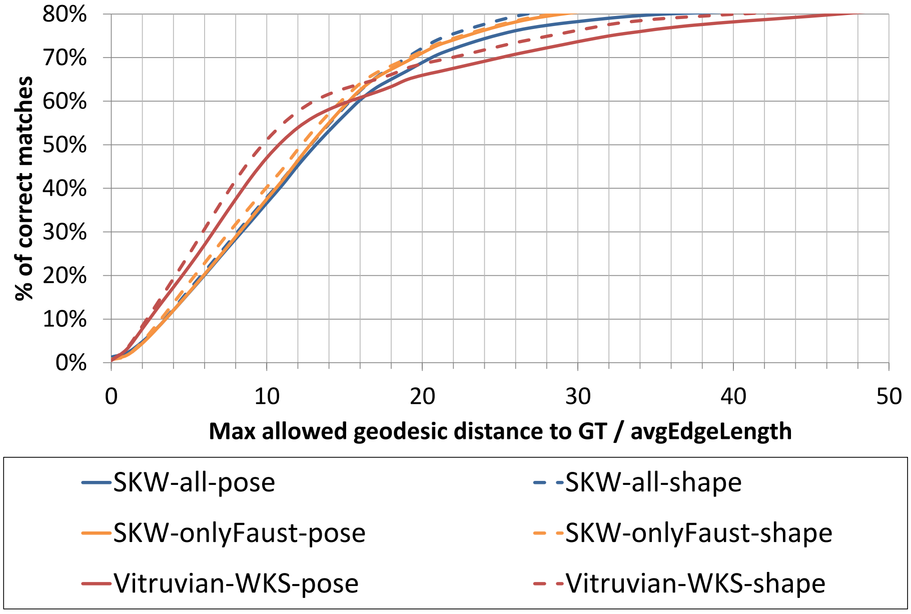


Figure B.5: **Cumulative errors on *FAUST*.**

C

A Unified View on Geometric Alignment in Vision Tasks

C.1 Introduction

A great part of work in Computer Vision has been dedicated to understanding the geometry of our surroundings. Mathematically speaking, by 'understanding' it means recovering from the observations a predefined parameterized model meaningful to machines. This is the very reason why Vision is often also referred to as 'inverse Graphics.' It deals with real-world raw observations and attempts to come up with some parameterized models that machines can interpret, while Graphics strives to computationally simulate real-world phenomena using these models. Among them, the biggest family is perhaps the geometric transformations that align one group of data $\{\mathbf{x}\}$ to the other $\{\mathbf{x}'\}$. Since the world we live in is essentially 3D while images in 2D, the case of alignment varies largely from 2D-2D, 3D-2D to 3D-3D, as visualized in Fig. C.1 below. The corresponding transformation is respectively known as homography $\mathbf{H}$, fundamental matrix $\mathbf{F}$, camera projection matrix $\mathbf{P}$ and rigid body motion $\mathbf{T} \in SE(3)$. Estimating them robustly plays a fundamental role in many advanced Vision applications, such as tracking and registration. The problem we have been tackled thus far in this dissertation actually belongs to the last category and the transformation is non-rigid (piecewise rigid), commonly denoted as a function $\mathbf{N}$.

Notably, the first two cases in Fig. C.1 have long been studied and the solutions that work reasonably well exist as described in [67], where several techniques haven been proposed and discussed to improve the accuracy, *e.g.* Gold Standard algorithm, virtual subsidiary correspondences, reprojection error and Sampson approximation. However, we notice that these techniques are seldom mentioned or considered (at least not explicitly) in the last case, the 3D registration task. In this chapter, we investigate the reasons by doing a comparison study. We first thoroughly review the state-of-the-art approaches in computing homography, camera matrix and fundamental matrix, in order to

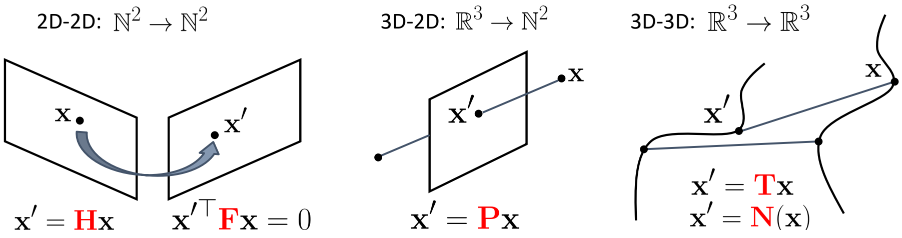

2D-2D: $\mathbb{N}^2 \to \mathbb{N}^2$
3D-2D: $\mathbb{R}^3 \to \mathbb{N}^2$
3D-3D: $\mathbb{R}^3 \to \mathbb{R}^3$
$\mathbf{x}$
$\mathbf{x}'$
$\mathbf{x}' = \mathbf{H}\mathbf{x}$
$\mathbf{x}'^\top \mathbf{F}\mathbf{x} = 0$
$\mathbf{x}' = \mathbf{P}\mathbf{x}$
$\mathbf{x}' = \mathbf{T}\mathbf{x}$
$\mathbf{x}' = \mathbf{N}(\mathbf{x})$

Figure C.1: **Three fundamental geometric alignment tasks in Vision**. These tasks are essentially finding the relations (red) between two sets of points.

come up with a common ground. We then seek to realize this in 3D registrations and analyze the prior arts from this point of view.

C.2 Literature Review

For the sake of consistency, we start with the ICP framework in chapter 2, where one takes care of the associations $\{(\mathbf{x}, \mathbf{x}')\}$ first and then computes the parameters $\mathbf{H}/\mathbf{F}/\mathbf{P}$. To simply the discussion, we assume 1-to-1 associations are given but not exact due to the measurement noise. The number of correspondence pairs is suggested to be more than it is really required and the parameters are computed in a least-square manner. Taking homography as example:

$$\hat{\mathbf{H}} = \min_{\mathbf{H}} \sum_i d(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2 = \min_{\mathbf{H}} \sum_i \|\mathbf{x}'_i - \mathbf{H}\mathbf{x}_i\|^2, \text{ where } \mathbf{x}'_i, \mathbf{x}_i \in \mathbb{N}^2, \quad \text{(C.1)}$$

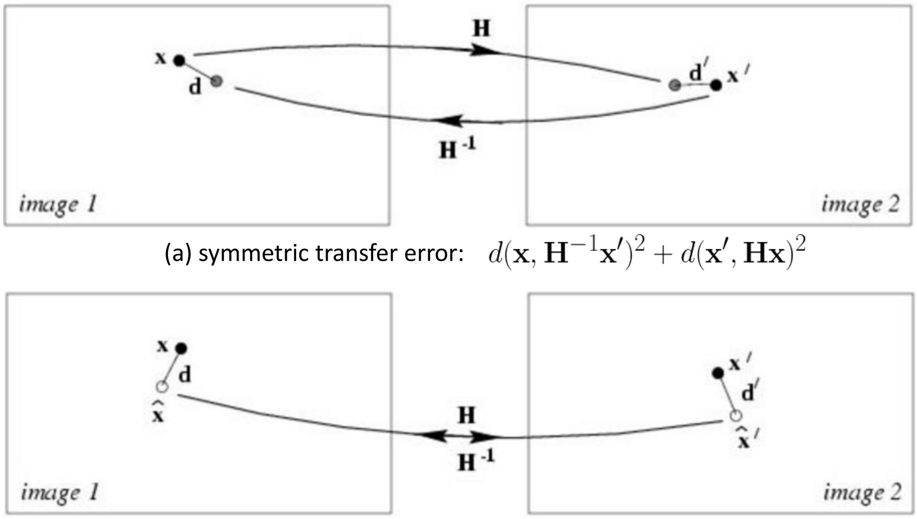
which is very similar to the data terms used in 3D deformable registrations. Nonetheless, since image observations are organized data, there is no need for an explicit closest-point search like in ICP. One simply considers the locations mapped by the current estimate $\mathbf{H}$ as new correspondences and this is already well embedded in Eq. C.1, explaining why ICP is seldom used in 2D tracking.

C.2.1 Reprojection error (Gold Standard error)

Besides Eq C.1, a few distance measures are also proposed to be minimized:

- Algebraic distance: $d_{\text{alg}}(\mathbf{x}', \mathbf{H}\mathbf{x})^2 = \|\mathbf{x}' - \mathbf{H}\mathbf{x}\|^2$.
- Symmetric transfer error: $d(\mathbf{x}', \mathbf{H}\mathbf{x})^2 + d(\mathbf{x}, \mathbf{H}^{-1}\mathbf{x}')^2$.
- Reprojection error: $d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$, s.t. $\hat{\mathbf{x}}' = \mathbf{H}\hat{\mathbf{x}}$.

The algebraic distance is the one used in Eq. C.1. With the constraint of $\|\mathbf{H}\|_F = 1$, the solution is unique and can be obtained in a closed-form manner without iterating, which is known as direct linear transformation (DLT) algorithm. In contrast to algebraic distances, the latter two are referred to as geometric distances and Figure C.2 illustrates the idea behind them. The symmetric transfer error augments algebraic distances by one error term in the other image, $d(\mathbf{x}, \mathbf{H}^{-1}\mathbf{x}')^2$, which is more thorough but the computation of inverse



(a) symmetric transfer error: $d(\mathbf{x}, \mathbf{H}^{-1}\mathbf{x}')^2 + d(\mathbf{x}', \mathbf{H}\mathbf{x})^2$

(b) re-projection error: $d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2 = d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \mathbf{H}\hat{\mathbf{x}})^2$

Figure C.2: **Two bi-directional error measures**. Image courtesy: [67]. Top:. Bottom:.

matrix is less desirable. The reprojection error follows this line of thought and take the measurement errors into account. It introduces the notion of exact correspondences that fulfill the homography relation perfectly and denotes them as $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$. The problem then boils down to minimizing the distances between exact correspondences and measured points.

Empirically, reprojection error demonstrates better accuracy than the other two [67] and is therefore also known as *Gold Standard* error. Here, we summarize the Gold Standard error for homography, projection matrix and fundamental matrix:

- Homography:

$$\hat{\mathbf{H}} = \min_{\mathbf{H}, \{\hat{\mathbf{x}}_i\}} \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \tag{C.2}$$

$$= \min_{\mathbf{H}, \{\hat{\mathbf{x}}_i\}} \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \mathbf{H}\hat{\mathbf{x}}_i)^2, \text{ where } \mathbf{x}'_i, \mathbf{x}_i, \hat{\mathbf{x}}_i \in \mathbb{N}^2. \tag{C.3}$$

  Degrees of freedom: $9 + 2n$. 9 for $\mathbf{H}$ and $2n$ for $\{\hat{\mathbf{x}}\}$, where $n = |\{\hat{\mathbf{x}}\}|$.

- Projection matrix:

$$\hat{\mathbf{P}} = \min_{\mathbf{P}, \{\hat{\mathbf{x}}_i\}} \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \tag{C.4}$$

$$= \min_{\mathbf{P}} \sum_i d(\mathbf{x}'_i, \mathbf{P}\mathbf{x}_i)^2, \text{ where } \mathbf{x}'_i \in \mathbb{N}^2, \mathbf{x}_i = \hat{\mathbf{x}}_i \in \mathbb{R}^3. \tag{C.5}$$

  Degrees of freedom: only 12 for $\mathbf{P}$. This special case results from the fact that when computing camera matrices, 3D points have to be given and are usually measured in high accuracy. It follows that $\hat{\mathbf{x}} = \mathbf{x}$, $d(\mathbf{x}, \hat{\mathbf{x}})^2$ vanishes and $\hat{\mathbf{x}}' = \mathbf{P}\hat{\mathbf{x}} = \mathbf{P}\mathbf{x}$.

- Fundamental matrix:

$$\hat{\mathbf{F}} = \min_{\mathbf{F}, \{\hat{\mathbf{x}}_i\}} \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \tag{C.6}$$

$$= \min_{\mathbf{P}', \{\hat{\mathbf{X}}_i\}} \sum_i d(\mathbf{x}_i, \mathbf{P}\hat{\mathbf{X}}_i)^2 + d(\mathbf{x}'_i, \mathbf{P}'\hat{\mathbf{X}}_i)^2, \text{ where } \mathbf{x}_i, \mathbf{x}'_i \in \mathbb{N}^2, \hat{\mathbf{X}}_i \in \mathbb{R}^3. \tag{C.7}$$

Degrees of freedom: $12 + 3n$. Here, a pair of camera matrices $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}' = [\mathbf{M}|\mathbf{t}]$ are defined and n subsidiary 3D points $\hat{\mathbf{X}}_i$ are introduced. The final fundamental matrix is computed as $\hat{\mathbf{F}} = [\mathbf{t}]_\times \mathbf{M}$.

We see that Gold Standard errors introduce auxiliary unknown variables as exact correspondences. When one follows the concept of ICP, it might be tempting to alternate between solving associations and parameters iteratively. As the matter of fact, this is not needed when one has organized data like 2D images. Hartley and Zisserman [67] describe methods thats solve them directly in the joint parameter-correspondence space using Levenberg-Marquardt optimization, which is the interpolation between Gauss-Newton and gradient-descent [40]. Similar ideas are actually used in [93, 162] for registering 3D range scans as well, in which case data is generated from organized 2D depth map. Note however that, despite the improved accuracy, the complexity increases linearly w.r.t. the number of correspondence pairs (except for projection matrix), which is in all respects an undesirable property. *Sampson error*, as described below, provides a workaround to alleviate this issue.

C.2.2 Sampson error

The Sampson approximation is originally derived in the conic-fitting problem [119] and is essentially Taylor first-order expansion under the hood. We explain its idea in homography estimation and then extend it to fundamental matrices.

Given a single pair of association $(\mathbf{x}, \mathbf{x}')$, the algebraic error minimizes the norm of a residual vector $\boldsymbol{\epsilon}$ of homography equation $\mathbf{x}' = \mathbf{H}\mathbf{x}$, *i.e.* $\boldsymbol{\epsilon} = \mathbf{x}' - \mathbf{H}\mathbf{x} \in \mathbb{R}^2$. Stacking the correspondence $(\mathbf{x}, \mathbf{x}')$ into a 4-vector $\mathbf{X} \in \mathbb{R}^4$, we may rewrite the residual vector as $\boldsymbol{\epsilon}(\mathbf{X})$ to emphasize the dependency on $\mathbf{X}$. To first order, this residual function may be approximated by a Taylor expansion:

$$\boldsymbol{\epsilon}(\mathbf{X} + \boldsymbol{\delta}_\mathbf{X}) = \boldsymbol{\epsilon}(\mathbf{X}) + \frac{\partial \boldsymbol{\epsilon}}{\partial \mathbf{X}} \boldsymbol{\delta}_\mathbf{X} = \boldsymbol{\epsilon}(\mathbf{X}) + \mathbf{J}\boldsymbol{\delta}_\mathbf{X}. \tag{C.8}$$

If we write $\boldsymbol{\delta}_\mathbf{X} = \hat{\mathbf{X}} - \mathbf{X}$, minimizing the reprojection error $d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$ is equivalent to minimizing $\|\boldsymbol{\delta}_\mathbf{X}\|^2$. Meanwhile, since $(\hat{\mathbf{x}}, \hat{\mathbf{x}}')$ is the subsidiary unknowns that really satisfy the homography constraint, namely, $\boldsymbol{\epsilon}(\hat{\mathbf{X}}) = \boldsymbol{\epsilon}(\mathbf{X} + \boldsymbol{\delta}_\mathbf{X}) = \mathbf{0}$, minimizing the Gold Standard error now becomes:

- *Given $\mathbf{X}$, find the vector $\boldsymbol{\delta}_\mathbf{X}$ that minimizes $\|\boldsymbol{\delta}_\mathbf{X}\|^2$ subject to $\mathbf{J}\boldsymbol{\delta}_\mathbf{X} = -\boldsymbol{\epsilon}(\mathbf{X})$.*

The standard way to solve this problem is using Lagrange multipliers [87], where a vector $\boldsymbol{\lambda}$ is introduced and the problem reduces to finding the extrema of the following:

$$\boldsymbol{\delta}_{\mathbf{X}}^{\top}\boldsymbol{\delta}_{\mathbf{X}} - 2\boldsymbol{\lambda}^{\top}(\mathbf{J}\boldsymbol{\delta}_{\mathbf{X}} + \boldsymbol{\epsilon}). \tag{C.9}$$

Taking derivatives with respect to $\boldsymbol{\delta}_{\mathbf{X}}$ and equating to zero results in:

$$2\boldsymbol{\delta}_{\mathbf{X}}^{\top} - 2\boldsymbol{\lambda}\mathbf{J}^{\top} = \mathbf{0}, \tag{C.10}$$

leading us to:

$$\boldsymbol{\delta}_{\mathbf{X}} = \mathbf{J}^{\top}\boldsymbol{\lambda}. \tag{C.11}$$

Likewise, the derivative with respect to $\boldsymbol{\lambda}$ equals to zero; we have:

$$\mathbf{J}\boldsymbol{\delta}_{\mathbf{X}} + \boldsymbol{\epsilon} = \mathbf{0}. \tag{C.12}$$

Putting Eq. C.11 and Eq. C.12 together, we finally reach our solution:

$$\boldsymbol{\delta}_{\mathbf{X}} = -\mathbf{J}^{\top}(\mathbf{J}\mathbf{J}^{\top})^{-1}\boldsymbol{\epsilon}., \tag{C.13}$$

which has the pseudo-inverse form of $\mathbf{J}$ [89]. The norm $\|\boldsymbol{\delta}_{\mathbf{X}}\|^2$ is the Sampson error:

$$\|\boldsymbol{\delta}_{\mathbf{X}}\|^2 = \boldsymbol{\epsilon}^{\top}(\mathbf{J}\mathbf{J}^{\top})^{-1}\boldsymbol{\epsilon}. \tag{C.14}$$

A few points to note:

1. Sampson error is an approximation of Gold Standard error.
2. Both $\boldsymbol{\epsilon}$ and $\mathbf{J}$ do not depend on auxiliary variables $(\hat{\mathrm{x}}, \hat{\mathrm{x}}')$. Consequently, minimizing Sampson error as in Eq. C.14 involves only the entries of $\mathbf{H}$.
3. The spirit of circumventing auxiliary variables $\hat{\mathbf{X}}$ is replacing it using Eq. C.13: $\hat{\mathbf{X}} = \mathbf{X} - \mathbf{J}^{\top}(\mathbf{J}\mathbf{J}^{\top})^{-1}\boldsymbol{\epsilon}$, which, as discussed below, is actually an approximation of true $\hat{\mathbf{X}}$.
4. The algebraic residual function $\boldsymbol{\epsilon}$ is typically multilinear in the entries of $\mathbf{X}$ (as the case of fundamental matrix). It is however, linear in the homography case, the first order approximation in Eq. C.8 is hence exact (higher order terms are zero), meaning that Sampson error is identical to Gold Standard error.

Newton-Raphson root finding analogue. All these remarks can also be properly visualized in Fig C.3, a Newton-Raphson procedure of root-finding [26]. The measured association $\mathbf{X}$ yields a non-zero residual $\boldsymbol{\epsilon}$ (dashed line) corresponding to the algebraic distance. The Gold Standard error to be minimized is the distance from zero-crossing $\hat{\mathbf{X}}$ (unknown) to $\mathbf{X}$. Sampson approaches $\hat{\mathbf{X}}$ (the green arrow) as Newton-Raphson does for roots. In each step, it takes the direction that compensates the residual most (the concept of steepest descend), where

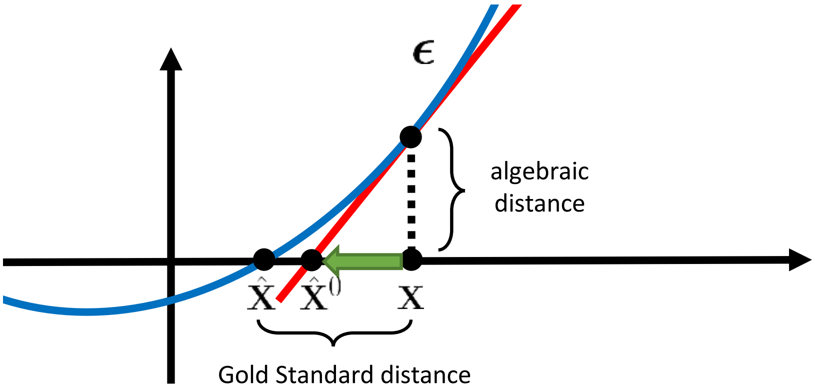


Figure C.3: **Root-finding analogy for Sampson approximation**. This figure illustrates how algebraic error is related to Gold Standard error.

the step size (Sampson error) is determined by the algebraic distance divided by the slope (Jacobian $\mathbf{J}$) and it approximates the Gold Standard distance. In this procedure, the residual vector ϵ and the Jacobian depend on the measured associations $\mathbf{X}$ or in later iterations on the estimated correspondences $\hat{\mathbf{X}}^0$. The subsidiary variables $\hat{\mathbf{X}}$ are never involved in the parameters being optimized, which is the key advantage of Sampson approximation.

For fundamental matrix, the Sampson approximation of Gold Standard error is similarly derived:

$$\|\delta\mathbf{x}\|^2 = \frac{\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon}}{\mathbf{J}\mathbf{J}^\top} = \frac{\left(\mathbf{x}'^\top\mathbf{F}\mathbf{x}\right)^2}{(\mathbf{F}\mathbf{x})_1^2 + (\mathbf{F}\mathbf{x})_2^2 + (\mathbf{F}^\top\mathbf{x}')_1^2 + (\mathbf{F}^\top\mathbf{x}')_2^2}, \tag{C.15}$$

where $(\mathbf{F}\mathbf{x})_j$ represents the j-th entry of the vector $\mathbf{F}\mathbf{x}$ and $\mathbf{J}\mathbf{J}^\top$ is a scalar. Again, this cost function involves only the entries of $\mathbf{F}$ so minimizing Gold Standard error in this way excludes the set of subsidiary variables and keeps the degree of freedom to only $\mathbf{F}$. It has been used successfully in estimation algorithms like [143, 159].

C.3 Gold Standard error in 3D

After reviewing these techniques for estimating 2D transformations, we turn to investigate how to deploy them in 3D, in particular, for deformable surface registrations. We focus on if and how the concepts of auxiliary correspondences and Sampson error are implemented in the literature.

C.3.1 Concept of subsidiary points in 3D

Unlike 2D images where one has organized data, currently the most prevalent data structure in 3D is indexed points, where one has only a set of point coordinates (meshed or not) and the rest is just empty space. Such fundamental difference explains why one needs to search for associations in 3D but not in

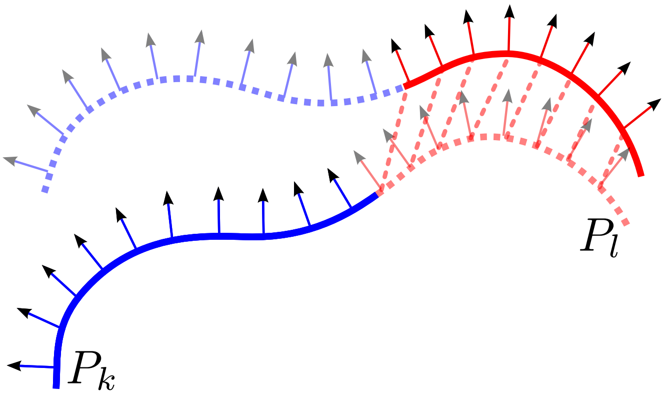


Figure C.4: **Virtual source points**.

2D. Nearest neighbor search is hence the very basic routine in 3D geometry that yields many speed-up algorithms to improve the efficiency.

3D registration aims to align two sets of indexed points: source and target. The association step in ICP strives to match the index between them so that one can construct a least-square data term for the subsequent optimization step. In [93, 162], Li *et al.* and Zollhöfer *et al.* present a data term where target points are unknown variables to be estimated during optimization. This strategy is feasible for them because their 3D data is generated from organized 2D depth maps, the so called 2.5D data. The unknown target points are still parameterized as the 2D coordinates discussed above. To the best of our knowledge, so far no one considers the full 3D coordinates as unknown variables and solves them in optimization. That said, we still observe some cases where people implicitly introduce auxiliary virtual correspondences to improve 3D registrations.

Virtual source points. First we provide an example on virtual source points. Fig C.4 recalls the patch-based deformation framework in Fig 2.3. Each vertex has several locations predicted by its neighboring patches (dashed line). These points do not physically exist but they are involved in the nearest neighbor search and are associated to the target point, which is a good example that virtual correspondences are already used in 3D registrations.

Virtual target points. We also give an example of virtual target points. One of the biggest traits in EM-ICP is the soft-associations. That is, there are no deterministic 0-1 correspondences but only probabilistic assignments, which typically corresponds to a weighted least square problem. As noted in the Eq. C.16 below, due to the property of least square, the minimizer of multiple weighted squared error is actually the weighted (averaged) target point, meaning that the source point is associated a synthetic point that may not exist in the observed target point set.

$$\min_{\mathbf{x}} \sum_i w_i \left\|\mathbf{x} - \mathbf{y}_i\right\|^2 \equiv \min_{\mathbf{x}} \|\mathbf{x} - \sum_i w_i \mathbf{y}_i\|^2, \text{ where } \sum_i w_i = 1. \tag{C.16}$$

In [35], Cagniart *et al.* demonstrate that introducing the predicted dashed points as candidates (virtual source) helps getting out of local minima, and that EM-ICP (virtual target) usually yields higher accuracy than traditional ICP, confirming that the superiority of subsidiary correspondences still holds in 3D.

C.3.2 Sampson error

Last but not least, we also derive the Sampson approximation of Gold Standard error in 3D. Let us consider a typical 2-norm data term:

$$\text{algebraic error:} \min_{\mathbf{T}} \left\|\mathbf{T}\mathbf{x} - \mathbf{x}'\right\|^2, \tag{C.17}$$

which minimizes the distance between a transformed source point and the associated target point. Meanwhile, the reprojection error can be likewise constructed as it is in the case of homography:

$$\text{reprojection error:} \min_{\mathbf{T},\hat{\mathbf{x}}} \left\|\mathbf{x} - \hat{\mathbf{x}}\right\|^2 + \left\|\mathbf{x}' - \hat{\mathbf{x}}'\right\|^2, \text{ s.t. } \hat{\mathbf{x}}' = \mathbf{T}\hat{\mathbf{x}}. \tag{C.18}$$

Following Eq. C.14, one has to define the residual vector ϵ and the Jacobian matrix $\mathbf{J}$. The transformation in patch-based deformation framework is:

$$\mathbf{T}\mathbf{x} = \mathbf{R}(\mathbf{x} - \mathbf{c}^0) + \mathbf{c}. \tag{C.19}$$

Stacking $\mathbf{x}$ and $\mathbf{x}'$ into one 6-vector $\mathbf{X}$, we can derive ϵ and $\mathbf{J}$ as follows:

$$\epsilon_{3\times 1} = \mathbf{T}\mathbf{x} - \mathbf{x}' = \mathbf{R}(\mathbf{x} - \mathbf{c}^0) + \mathbf{c} - \mathbf{x}' \tag{C.20}$$

$$\mathbf{J}_{3\times 6} \triangleq \frac{\partial \epsilon_{3\times 1}}{\partial \mathbf{X}_{6\times 1}} = [\mathbf{R}| - \mathbf{I}]_{3\times 6}\,. \tag{C.21}$$

Next we see that:

$$\mathbf{J}\mathbf{J}^\top = \mathbf{R}\mathbf{R}^\top + \mathbf{I}\mathbf{I}^\top = 2\mathbf{I}, \tag{C.22}$$

$$\therefore \left\|\delta\right\|^2 = \epsilon^\top(\mathbf{J}\mathbf{J}^\top)^{-1}\epsilon = \frac{1}{2}\epsilon^\top\epsilon = \frac{1}{2}\left\|\mathbf{T}\mathbf{x} - \mathbf{x}'\right\|^2, \tag{C.23}$$

which means the *Sampson error actually equals to the algebraic error*. For this reason, most work in the literature already apply Sampson approximation without mentioning it explicitly.

C.4 Conclusion

In this chapter, we review several fundamental techniques and theories in 2D computer vision, with the aim to investigate their usage in 3D geometry processing and in turn to provide a unified view point that bridge the gap between 2D and 3D. By showing examples and deriving formula, we mark that, even though not explicitly mentioned or noted, two useful techniques, virtual correspondences and Sampson approximation, are already considered in the prior arts of 3D registrations and their efficacies have also been verified.

D Multi-view datasets

Sequence	Views	Frames	Error metric	Reference
Samba	8	175	A	Vlasic *et al.* [148]
Crane	8	175	A	
Jumping	8	150	A	
Bouncing	8	175	A	
Handstand	8	175	A	
Skirt	8	720	A	Gall *et al.* [62]
Dance	8	573	A	
Wheel	8	280	A	
Handstand2	8	400	A	
Free	8	200	A	Starck *et al.* [128]
Fighting	12	500	B	Liu *et al.* [98]
S4_walking	4	350	C & D	Sigal *et al.* [122]
UpJump	48	239	A	Allain *et al.* [6]
Ballet_Seq1	9	500	A	Allain *et al.* [5]
SideJump	48	150	A	
Ballet_Seq2	9	936	A & B	Huang *et al.* [69]
Thomas_Seq1	68	1500	A	
Thomas_Seq2	68	1400	A	
Cutting	9	81	A	Huang *et al.* [71]
WalkChair1	9	130	A	
WalkChair2	9	148	A & D	Huang *et al.* [73]
HammerTable	9	93	A & D	
SideSit	9	97	A & D	

Table D.1: **Sequences used for evaluation in this thesis**. Different subsets are used in previous chapters. Please refer to the summary in each chapter for more details. Four distinct error measures are considered. A: silhouette overlap error. B: distances in $\mathbb{R}^3$ to markers. C: 3D error on joint locations. D: 2D pixel error on joint positions. A and B are metrics for shapes while C and D for poses.

E
Sequences with outliers

To highlight the main contribution of this thesis, *i.e.* discriminative associations, against traditional ICP-based approaches, we record sequences that contains un-removed outliers, such as chairs and tables.

In Fig. E.1, we show one example frame of these new sequences. The occluding object, *i.e.* the chair, is kept after background subtraction and therefore remains in the subsequent reconstructed point cloud. The reference surfaces is the smoothed reconstructed visual hulls at $t = 0$. There is no need to register the surface to the point cloud with a rigid transformation to initialize the tracking.

We provide two different types of ground truths for evaluations. For shape evaluation, we manually remove the silhouettes of irrelevant objects if they are not connected to the subjects, as shown in Fig. E.1. The associated metric is the standard *silhouette overlap error* which measures the discrepancies between the contours of the projected surface and the observed silhouettes. To evaluate the estimated poses, we annotate the positions of joints in five cameras, and see how close to them the estimated joints are (*2D joint error*). These sequences are available at the *4D Repository*[1].

[1] http://4drepository.inrialpes.fr/

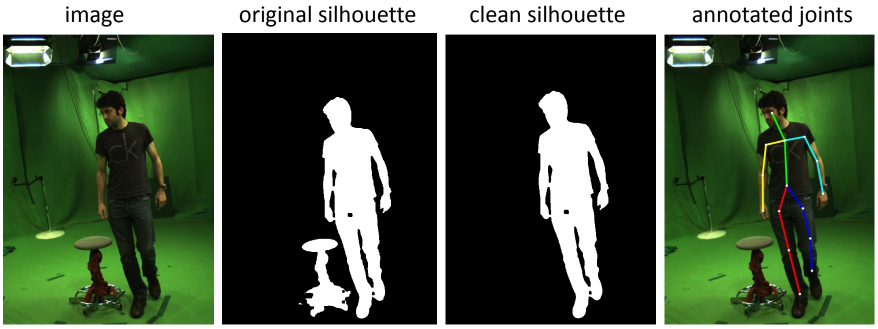


Figure E.1: **Example frame of newly recorded sequences.** From left to right: images, generated clean silhouettes, and annotated joint positions of *WalkChair*.

F Authored and Co-authored Publications

Authored:

1. Huang, C.H., Boyer, E., Allain, B., Franco, J.S., Tombari, F., Navab, N., Ilic, S.: Tracking-by-detection of 3d human shapes: from surfaces to volumes. Transactions on Pattern Analysis and Machine Intelligence (2017) (**Submitted**)

2. Huang, C.H., Allain, B., Franco, J.S., Navab, N., Ilic, S., Boyer, E.: Volumetric 3d tracking by detection. In: CVPR 2016-IEEE Conference on Computer Vision and Pattern Recognition (2016) (**Spotlight oral presentation**)

3. Huang, C.H., Cagniart, C., Boyer, E., Ilic, S.: A bayesian approach to multiview 4d modeling. International Journal of Computer Vision **116**(2), 115–135 (2016)

4. Huang, C.H., Tombari, F., Navab, N.: Repeatable local coordinate frames for 3d human motion tracking: From rigid to non-rigid. In: 3D Vision (3DV), 2015 International Conference on, pp. 371–379. IEEE (2015)

5. Huang, C.H., Boyer, E., do Canto Angonese, B., Navab, N., Ilic, S.: Toward user-specific tracking by detection of human shapes in multi-cameras. In: Computer Vision and Pattern Recognition (CVPR), pp. 4027–4035. IEEE (2015)

6. Huang, C.H., Boyer, E., Navab, N., Ilic, S.: Human shape and pose tracking using keyframes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3446–3453. IEEE (2014)

7. Huang, C.H., Boyer, E., Ilic, S.: Robust human body shape and pose tracking. In: 3DV. IEEE (2013) (**Oral presentation, best paper award runner-up**)

Co-authored:

1. Wang, X., Habert, S., Ma, M., Huang, C.H., Fallavollita, P., Navab, N.: Precise 3d/2d calibration between a rgb-d sensor and a c-arm fluoroscope. International journal of computer assisted radiology and surgery pp. 1–11 (2016)

2. Wang, X., Habert, S., Ma, M., Huang, C.H., Fallavollita, P., Navab, N.: [poster] rgb-d/c-arm calibration and application in medical augmented reality. In: Mixed and Augmented Reality (ISMAR), 2015 IEEE International Symposium on, pp. 100–103. IEEE (2015)

Bibliography

[1] Agarwal, S., Mierle, K., Others: Ceres solver. `http://ceres-solver.org`

[2] Aggarwal, J.K., Cai, Q.: Human motion analysis: A review. In: Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE, pp. 90–102. IEEE (1997)

[3] Akenine-Möller, T.: Fast 3d triangle-box overlap testing. In: SIGGRAPH 2005 Courses. ACM (2005)

[4] Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 157–164. ACM Press/Addison-Wesley Publishing Co. (2000)

[5] Allain, B., Franco, J.S., Boyer, E.: An efficient volumetric framework for shape tracking. In: Computer Vision and Pattern Recognition (CVPR). IEEE (2015). URL `https://hal.inria.fr/hal-01141207`

[6] Allain, B., Franco, J.S., Boyer, E., Tung, T.: On mean pose and variability of 3d deformable models. In: European Conference on Computer Vision (ECCV). Springer (2014). URL `https://hal.inria.fr/hal-01016981`

[7] Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. In: ACM Transactions on Graphics (TOG), vol. 24, pp. 408–416. ACM (2005)

[8] Ankerst, M., Kastenmüller, G., Kriegel, H.P., Seidl, T.: 3d shape histograms for similarity search and classification in spatial databases. In: Advances in Spatial Databases (1999)

[9] Aubry, M., Schlickewei, U., Cremers, D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In: ICCV Workshops. IEEE (2011)

[10] Baak, A., Muller, M., Bharaj, G., Seidel, H.P., Theobalt, C.: A data-driven approach for real-time full body pose reconstruction from a depth camera. In: International Conference on Computer Vision (ICCV). IEEE (2011)

[11] Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. In: TOG (2007)

[12] Baran, I., Vlasic, D., Grinspun, E., Popović, J.: Semantic deformation transfer. In: ACM Transactions on Graphics (TOG), vol. 28, p. 36. ACM (2009)

[13] Belagiannis, V., Amin, S., Andriluka, M., Schiele, B., Navab, N., Ilic, S.: 3d pictorial structures for multiple human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1669–1676 (2014)

[14] Belagiannis, V., Schubert, F., Navab, N., Ilic, S.: Segmentation based particle filtering for real-time 2d object tracking. In: European Conference on Computer Vision (ECCV), pp. 842–855. Springer (2012)

[15] Bengio, Y.: Learning deep architectures for ai. Foundations and trends® in Machine Learning **2**(1), 1–127 (2009)

[16] Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18**(9), 509–517 (1975)

[17] Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI) (1992)

[18] Bishop, C.M., Nasrabadi, N.M.: Pattern recognition and machine learning, vol. 1. Springer (2006)

[19] Bogo, F., Romero, J., Loper, M., Black, M.: FAUST: Dataset and evaluation for 3D mesh registration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3794–3801 (2014)

[20] Botsch, M., Bommes, D., Kobbelt, L.: Efficient linear system solvers for mesh processing. In: IMA Conference on the Mathematics of Surfaces (2005)

[21] Botsch, M., Pauly, M., Wicke, M., Gross, M.: Adaptive space deformations based on rigid cells. In: Computer Graphics Forum, vol. 26, pp. 339–347. Wiley Online Library (2007)

[22] Botsch, M., Pauly, M., Wicke, M., Gross, M.H.: Adaptive space deformations based on rigid cells. Comput. Graph. Forum (2007)

[23] Botsch, M., Sorkine, O.: On linear variational surface deformation methods. Visualization and Computer Graphics, IEEE Transactions on **14**(1), 213–230 (2008)

[24] Bouwmans, T., Porikli, F., Höferlin, B., Vacavant, A.: Background Modeling and Foreground Detection for Video Surveillance. CRC Press (2014)

[25] Boyer, E., Bronstein, A.M., Bronstein, M.M., Bustos, B., Darom, T., Horaud, R., Hotz, I., Keller, Y., Keustermans, J., Kovnatsky, A., et al.: Shrec 2011: robust feature detection and description benchmark. In: Eurographics 3DOR Workshop, pp. 71–78. Eurographics Association (2011)

[26] Bradie, B.: A friendly introduction to numerical analysis. Prentice-Hall (2006)

[27] Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

[28] Bronstein, A., Bronstein, M., Castellani, U., Dubrovina, A., Guibas, L., Horaud, R., Kimmel, R., Knossow, D., Von Lavante, E., Mateus, D., et al.: Shrec 2010: robust correspondence benchmark. In: Eurographics Workshop on 3D Object Retrieval (2010)

[29] Bronstein, M.M., Kokkinos, I.: Scale-invariant heat kernel signatures for non-rigid shape recognition. In: Computer Vision and Pattern Recognition (CVPR), pp. 1704–1711. IEEE (2010)

[30] Brooks, S.P., Morgan, B.J.: Optimization using simulated annealing. The Statistician pp. 241–257 (1995)

[31] Budd, C., Hilton, A.: Skeleton driven Laplacian volumetric deformation. In: CVMP (2009)

[32] Budd, C., Huang, P., Klaudiny, M., Hilton, A.: Global non-rigid alignment of surface sequences. International Journal of Computer Vision (IJCV) (2013)

[33] Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery **2**(2), 121–167 (1998)

[34] Cagniart, C., Boyer, E., Ilic, S.: Free-form mesh tracking: a patch-based approach. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 1339–1346. IEEE (2010)

[35] Cagniart, C., Boyer, E., Ilic, S.: Probabilistic deformable surface tracking from multiple videos. In: European Conference on Computer Vision (ECCV), pp. 326–339. Springer (2010)

[36] Chen, Q., Koltun, V.: Robust nonrigid registration by convex optimization. In: International Conference on Computer Vision (ICCV), pp. 2039–2047 (2015)

[37] Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: International Conference on Robotics and Automation. IEEE (1991)

[38] Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and vision computing **10**(3), 145–155 (1992)

[39] Cheng, Y.: Mean shift, mode seeking, and clustering. Pattern Analysis and Machine Intelligence, IEEE Transactions on **17**(8), 790–799 (1995)

[40] Chong, E.K., Zak, S.H.: An introduction to optimization, vol. 76. John Wiley & Sons (2013)

[41] Chua, C.S., Jarvis, R.: Point signatures: A new representation for 3d object recognition. International Journal of Computer Vision **25**(1), 63–85 (1997)

[42] Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(5), 603–619 (2002)

[43] Corazza, S., Mündermann, L., Gambaretto, E., Ferrigno, G., Andriacchi, T.P.: Markerless motion capture through visual hull, articulated icp and subject specific model generation. International journal of computer vision (IJCV) (2010)

[44] Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)

[45] Criminisi, A., Shotton, J.: Decision forests for computer vision and medical image analysis. Springer (2013)

[46] Criminisi, A., Shotton, J., Robertson, D., Konukoglu, E.: Regression forests for efficient anatomy detection and localization in ct studies. In: Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging. Springer (2011)

[47] De Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.P., Thrun, S.: Performance capture from sparse multi-view video. In: ACM SIGGRAPH 2008 (2008)

[48] De Aguiar, E., Theobalt, C., Stoll, C., Seidel, H.P.: Marker-less deformable mesh tracking for human shape and motion capture. In: Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)

[49] Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S.R., Kowdle, A., Escolano, S.O., Rhemann, C., Kim, D., Taylor, J., et al.: Fusion4d: Real-time performance capture of challenging scenes. ACM Transactions on Graphics (TOG) **35**(4), 114 (2016)

[50] Du, Q., Faber, V., Gunzburger, M.: Centroidal Voronoi tessellations: Applications and algorithms. SIAM review **41**, 637–676 (1999)

[51] Duveau, E., Courtemanche, S., Reveret, L., Boyer, E.: Cage-based motion recovery using manifold learning. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on, pp. 206–213 (2012)

[52] Fanelli, G., Dantone, M., Gall, J., Fossati, A., Van Gool, L.: Random forests for real time 3d face analysis. International Journal of Computer Vision (IJCV) **101**(3), 437–458 (2013)

[53] Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. International Journal of Computer Vision (IJCV) **70**(1), 41–54 (2006)

[54] Franco, J.S., Boyer, E.: Exact polyhedral visual hulls. In: British Machine Vision Conference (BMVC) (2003). URL `http://hal.inria.fr/inria-00349075`

[55] Franco, J.S., Boyer, E.: Efficient polyhedral modeling from silhouettes. IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI) **31**(3) (2009). doi: 10.1109/TPAMI.2008.104. URL `https://hal.inria.fr/inria-00349103`

[56] Franco, J.S., Boyer, E.: Learning temporally consistent rigidities. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 1241–1248. IEEE (2011)

[57] Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning, vol. 1. Springer series in statistics Springer, Berlin (2001)

[58] Fujiwara, K., Nishino, K., Takamatsu, J., Zheng, B., Ikeuchi, K.: Locally rigid globally non-rigid surface registration. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 1527–1534. IEEE (2011)

[59] Furukawa, Y., Ponce, J.: Dense 3d motion capture from synchronized video streams. In: Computer Vision and Pattern Recognition (CVPR) (2008)

[60] Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE transactions on pattern analysis and machine intelligence **32**(8), 1362–1376 (2010)

[61] Gall, J., Rosenhahn, B., Brox, T., Seidel, H.P.: Optimization and filtering for human motion capture. International Journal of Computer Vision (IJCV) **87**(1-2), 75–92 (2010)

[62] Gall, J., Stoll, C., De Aguiar, E., Theobalt, C., Rosenhahn, B., Seidel, H.P.: Motion capture using joint skeleton tracking and surface estimation. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 1746–1753. IEEE (2009)

[63] Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**(11), 2188–2202 (2011)

[64] Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 415–422. IEEE (2011)

[65] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., Kwok, N.M.: A comprehensive performance evaluation of 3d local feature descriptors. International Journal of Computer Vision **116**(1), 66–89 (2016)

[66] Gupta, S., Sengupta, K., Kassim, A.A.: Compression of dynamic 3d geometry data using iterative closest point algorithm. Computer Vision and Image Understanding **87**(1), 116–130 (2002)

[67] Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)

[68] Ho, T.K.: The random subspace method for constructing decision forests. Pattern Analysis and Machine Intelligence, IEEE Transactions on **20**(8), 832–844 (1998)

[69] Huang, C.H., Allain, B., Franco, J.S., Navab, N., Ilic, S., Boyer, E.: Volumetric 3d tracking by detection. In: CVPR 2016-IEEE Conference on Computer Vision and Pattern Recognition (2016)

[70] Huang, C.H., Boyer, E., Allain, B., Franco, J.S., Tombari, F., Navab, N., Ilic, S.: Tracking-by-detection of 3d human shapes: from surfaces to volumes. Transactions on Pattern Analysis and Machine Intelligence (2017)

[71] Huang, C.H., Boyer, E., do Canto Angonese, B., Navab, N., Ilic, S.: Toward user-specific tracking by detection of human shapes in multi-cameras. In: Computer Vision and Pattern Recognition (CVPR), pp. 4027–4035. IEEE (2015)

[72] Huang, C.H., Boyer, E., Ilic, S.: Robust human body shape and pose tracking. In: 3DV. IEEE (2013)

[73] Huang, C.H., Boyer, E., Navab, N., Ilic, S.: Human shape and pose tracking using keyframes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3446–3453. IEEE (2014)

[74] Huang, C.H., Cagniart, C., Boyer, E., Ilic, S.: A bayesian approach to multiview 4d modeling. International Journal of Computer Vision **116**(2), 115–135 (2016)

[75] Huang, C.H., Tombari, F., Navab, N.: Repeatable local coordinate frames for 3d human motion tracking: From rigid to non-rigid. In: 3D Vision (3DV), 2015 International Conference on, pp. 371–379. IEEE (2015)

[76] Huang, P., Hilton, A., Starck, J.: Shape similarity for 3d video sequences of people. International Journal of Computer Vision **89**(2-3), 362–381 (2010)

[77] Ibarria, L., Rossignac, J.: Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 126–135. Eurographics Association (2003)

[78] Jain, A., Thormählen, T., Seidel, H.P., Theobalt, C.: Moviereshape: Tracking and reshaping of humans in videos. ACM Transactions on Graphics (TOG) **29**(6), 148 (2010)

[79] Jolliffe, I.: Principal component analysis. Wiley Online Library (2002)

[80] Joo, H., Park, H., Sheikh, Y.: Map visibility estimation for large-scale dynamic 3d reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1122–1129 (2014)

[81] Kanaujia, A., Kittens, N., Ramanathan, N.: Part segmentation of visual hull for 3d human pose estimation. In: CVPR Workshop. IEEE (2013)

[82] Kavan, L., Collins, S., Žára, J., O'Sullivan, C.: Geometric skinning with approximate dual quaternion blending. ACM Transactions on Graphics (TOG) **27**(4), 105 (2008)

[83] Kehl, W., Navab, N., Ilic, S.: Coloured signed distance fields for full 3d object reconstruction. In: British Machine Vision Conference (BMVC) (2014)

[84] Klaudiny, M., Budd, C., Hilton, A.: Towards optimal non-rigid surface tracking. In: European Conference on Computer Vision (ECCV), pp. 743–756. Springer (2012)

[85] Ladikos, A., Benhimane, S., Navab, N.: Real-time 3d reconstruction for collision avoidance in interventional environments. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008, pp. 526–534. Springer (2008)

[86] Ladikos, A., Cagniart, C., Ghotbi, R., Reiser, M., Navab, N.: Estimating radiation exposure in interventional environments. In: MICCAI, pp. 237–244. Springer (2010)

[87] Lagrange, J.L.: Mécanique analytique, vol. 1. Mallet-Bachelier (1853)

[88] Lähner, Z., Rodolà, E., Bronstein, M.M., Cremers, D., Burghard, O., Cosmo, L., Dieckmann, A., Klein, R., Sahillioglu, Y.: Shrecąę16: Matching of deformable shapes with topological noise. In: Proc. of Eurographics Workshop on 3D Object Retrieval (3DOR) (2016)

[89] Lay, D.C.: Linear algebra and its applications, 2003. Addison Wesley, New York Google Scholar

[90] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4), 541–551 (1989)

[91] Lee, W., Woo, W., Boyer, E.: Silhouette segmentation in multiple views. Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**(7), 1429–1441 (2011)

[92] Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM (2000)

[93] Li, H., Sumner, R.W., Pauly, M.: Global correspondence optimization for non-rigid registration of depth scans. In: Computer graphics forum, vol. 27, pp. 1421–1430. Wiley Online Library (2008)

[94] Li, S.Z.: Markov random field modeling in image analysis. Springer Science & Business Media (2009)

[95] Liao, M., Zhang, Q., Wang, H., Yang, R., Gong, M.: Modeling deformable objects from a single depth camera. In: Computer Vision, 2009 IEEE 12th International Conference on, pp. 167–174. IEEE (2009)

[96] Lindholm, E., Kilgard, M.J., Moreton, H.: A user-programmable vertex engine. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 149–158. ACM (2001)

[97] Liu, Y., Gall, J., Stoll, C., Dai, Q., Seidel, H.P., Theobalt, C.: Markerless motion capture of multiple characters using multiview image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(11), 2720–2735 (2013)

[98] Liu, Y., Stoll, C., Gall, J., Seidel, H.P., Theobalt, C.: Markerless motion capture of interacting characters using multi-view image segmentation. In: Computer Vision and Pattern Recognition (CVPR). IEEE (2011). doi: 10.1109/CVPR.2011.5995424

[99] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: Smpl: a skinned multi-person linear model. ACM Transactions on Graphics (TOG) **34**(6), 248 (2015)

[100] Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: ACM siggraph computer graphics, vol. 21, pp. 163–169. ACM (1987)

[101] Mian, A., Bennamoun, M., Owens, R.: On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. International Journal of Computer Vision **89**(2-3), 348–361 (2010)

[102] Moeslund, T.B., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. Computer vision and image understanding **104**(2), 90–126 (2006)

[103] Mustafa, A., Kim, H., Guillemaut, J.Y., Hilton, A.: Temporally coherent 4d reconstruction of complex dynamic scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4660–4669 (2016)

[104] Newcombe, R.A., Fox, D., Seitz, S.M.: DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In: Computer Vision and Pattern Recognition (CVPR), pp. 343–352 (2015)

[105] Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: ISMAR. IEEE (2011)

[106] Noh, J.y., Neumann, U.: Expression cloning. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 277–288. ACM (2001)

[107] Novatnack, J., Nishino, K.: Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In: European Conference on Computer Vision (ECCV). Springer (2008)

[108] Nowozin, S., Lampert, C.H.: Structured learning and prediction in computer vision. Foundations and Trends in Computer Graphics and Vision **6**(3-4) (2011)

[109] Petrelli, A., Di Stefano, L.: On the repeatability of the local reference frame for partial shape matching. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 2244–2251. IEEE (2011)

[110] Petrelli, A., Di Stefano, L.: A repeatable and efficient canonical reference for surface matching. In: 3DimPVT. IEEE (2012)

[111] Pons-Moll, G., Romero, J., Mahmood, N., Black, M.J.: Dyna: A model of dynamic human shape in motion. ACM Transactions on Graphics, (Proc. SIGGRAPH) **34**(4), 120:1–120:14 (2015)

[112] Pons-Moll, G., Taylor, J., Shotton, J., Hertzmann, A., Fitzgibbon, A.: Metric regression forests for human pose estimation. In: British Machine Vision Conference (BMVC) (2013)

[113] Poppe, R.: Vision-based human motion analysis: An overview. Computer vision and image understanding **108**(1), 4–18 (2007)

[114] Ren, S., Cao, X., Wei, Y., Sun, J.: Global refinement of random forest. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 723–730 (2015)

[115] Robertini, N., Casas, D., De Aguiar, E., Theobalt, C.: Multi-view performance capture of surface details. International Journal of Computer Vision pp. 1–18 (2017)

[116] Robertini, N., De Aguiar, E., Helten, T., Theobalt, C.: Efficient multi-view performance capture of fine-scale surface detail. In: 3D Vision (3DV), 2014 2nd International Conference on, vol. 1, pp. 5–12. IEEE (2014)

[117] Rodola, E., Bulo, S.R., Windheuser, T., Vestner, M., Cremers, D.: Dense non-rigid shape correspondence using random forests. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4177–4184. IEEE (2014)

[118] Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pp. 3212–3217. IEEE (2009)

[119] Sampson, P.D.: Fitting conic sections to ?very scattered? data: An iterative refinement of the bookstein algorithm. Computer graphics and image processing **18**(1), 97–108 (1982)

[120] Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, vol. 1, pp. 519–528. IEEE (2006)

[121] Shotton, J., Fitzgibbon, A., Cook, M., Blake, A.: Real-time human pose recognition in parts from single depth images. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 1297–1304. IEEE (2011)

[122] Sigal, L., Balan, A.O., Black, M.J.: Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. International Journal of Computer Vision (IJCV) (2010)

[123] Sigal, L., Isard, M., Haussecker, H., Black, M.J.: Loose-limbed people: Estimating 3d human pose and motion using non-parametric belief propagation. International Journal of Computer Vision (IJCV) **98**(1), 15–48 (2012)

[124] Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of Eurographics/ACM SIGGRAPH symposium on Geometry processing, vol. 4 (2007)

[125] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 175–184. ACM (2004)

[126] Sridhar, S., Rhodin, H., Seidel, H.P., Oulasvirta, A., Theobalt, C.: Real-time hand tracking using a sum of anisotropic gaussians model. In: 3D Vision (3DV), 2014 2nd International Conference on, vol. 1, pp. 319–326. IEEE (2014)

[127] Starck, J., Hilton, A.: Correspondence labelling for wide-timeframe free-form surface matching. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1–8. IEEE (2007)

[128] Starck, J., Hilton, A.: Surface capture for performance-based animation. In: Computer Graphics and Applications (2007)

[129] Stoll, C., Gall, J., De Aguiar, E., Thrun, S., Theobalt, C.: Video-based reconstruction of animatable human characters. In: TOG. ACM (2010)

[130] Stoll, C., Hasler, N., Gall, J., Seidel, H.P., Theobalt, C.: Fast articulated motion tracking using a sums of gaussians body model. In: International Conference on Computer Vision (ICCV), pp. 951–958. IEEE (2011)

[131] Straka, M., Hauswiesner, S., Ruether, M., Bischof, H.: Skeletal graph based human pose estimation in real-time. In: British Machine Vision Conference (BMVC) (2011)

[132] Straka, M., Hauswiesner, S., Rüther, M., Bischof, H.: Simultaneous shape and pose adaption of articulated models using linear optimization. In: European Conference on Computer Vision (ECCV). Springer (2012)

[133] Su, H., Wang, F., Yi, L., Guibas, L.: 3d-assisted image feature synthesis for novel views of an object. In: International Conference on Computer Vision (ICCV) (2015)

[134] Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. In: ACM Transactions on Graphics (TOG). ACM (2004)

[135] Sumner, R.W., Schmid, J., Pauly, M.: Embedded deformation for shape manipulation. In: ACM SIGGRAPH 2007 (2007)

[136] Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. CGF **28**(5), 1383–1392 (2009)

[137] Tan, D.J., Ilic, S.: Multi-forest tracker: A chameleon in tracking. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pp. 1202–1209. IEEE (2014)

[138] Tan, D.J., Tombari, F., Ilic, S., Navab, N.: A versatile learning-based 3d temporal tracker: Scalable, robust, online. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 693–701 (2015)

[139] Taubin, G.: A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM (1995)

[140] Taylor, J., Shotton, J., Sharp, T., Fitzgibbon, A.: The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In: Computer Vision and Pattern Recognition (CVPR), pp. 103–110. IEEE (2012)

[141] The CGAL Project: CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`

[142] Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: European Conference on Computer Vision (ECCV), pp. 356–369. Springer (2010)

[143] Torr, P., Zisserman, A.: Robust computation and parametrization of multiple view relations. In: Computer Vision, 1998. Sixth International Conference on, pp. 727–732. IEEE (1998)

[144] Tung, T., Nobuhara, S., Matsuyama, T.: Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo. In: Computer Vision, 2009 IEEE 12th International Conference on, pp. 1709–1716. IEEE (2009)

[145] Urtasun, R., Darrell, T.: Sparse probabilistic regression for activity-independent human pose inference. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1–8 (2008)

[146] Vacchetti, L., Lepetit, V., Fua, P.: Fusing online and offline information for stable 3d tracking in real-time. In: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, vol. 2, pp. II–241. IEEE (2003)

[147] Viola, P., Jones, M.J.: Robust real-time face detection. International journal of computer vision (IJCV) **57**(2), 137–154 (2004)

[148] Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. ACM Transactions on Graphics (TOG) **27**(3), 97 (2008)

[149] Vlasic, D., Peers, P., Baran, I., Debevec, P., Popović, J., Rusinkiewicz, S., Matusik, W.: Dynamic shape capture using multi-view photometric stereo. In: ACM Transactions on Graphics (TOG), vol. 28, p. 174. ACM (2009)

[150] Wang, L., Hétroy-Wheeler, F., Boyer, E.: A hierarchical approach for regular centroidal Voronoi tessellations. In: CGF (2015)

[151] Wang, L., Hu, W., Tan, T.: Recent developments in human motion analysis. Pattern recognition **36**(3), 585–601 (2003)

[152] Wang, X., Habert, S., Ma, M., Huang, C.H., Fallavollita, P., Navab, N.: [poster] rgb-d/c-arm calibration and application in medical augmented reality. In: Mixed and Augmented Reality (ISMAR), 2015 IEEE International Symposium on, pp. 100–103. IEEE (2015)

[153] Wang, X., Habert, S., Ma, M., Huang, C.H., Fallavollita, P., Navab, N.: Precise 3d/2d calibration between a rgb-d sensor and a c-arm fluoroscope. International journal of computer assisted radiology and surgery pp. 1–11 (2016)

[154] Wei, L., Huang, Q., Ceylan, D., Vouga, E., Li, H.: Dense human body correspondences using convolutional networks. In: Computer Vision and Pattern Recognition (CVPR) (2016)

[155] Wei, X., Zhang, P., Chai, J.: Accurate realtime full-body motion capture using a single depth camera. ACM Transactions on Graphics (TOG) **31**(6), 188 (2012)

[156] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920 (2015)

[157] Zaharescu, A., Boyer, E., Horaud, R.: Keypoints and local descriptors of scalar functions on 2d manifolds. International Journal of Computer Vision (IJCV) **100**(1), 78–98 (2012)

[158] Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. International journal of computer vision (IJCV) **13**(2), 119–152 (1994)

[159] Zhang, Z.: Determining the epipolar geometry and its uncertainty: A review. International journal of computer vision **27**(2), 161–195 (1998)

[160] Zhou, F., De la Torre, F., Hodgins, J.K.: Hierarchical aligned cluster analysis for temporal clustering of human motion. IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI) **35**(3), 582–596 (2013)

[161] Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph Laplacian. In: SIGGRAPH. ACM (2005)

[162] Zollhöfer, M., Nießner, M., Izadi, S., Rhemann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., Stamminger, M.: Real-time non-rigid reconstruction using an rgb-d camera. ACM Transactions on Graphics (TOG) **33**(4) (2014)

Index