UNIVERSITY OF CALIFORNIA, SAN DIEGO

RECONSTRUCTION OF DYNAMIC ARTICULATED 3D MODELS FROM RANGE SCANS

A dissertation submitted in partial satisfaction of the

requirements for the degree

Doctor of Philosophy

in

Computer Science

by

William Young Chang

Committee in charge:

Professor Matthias Zwicker, Chair
Professor Serge J. Belongie
Professor Samuel R. Buss
Professor Henrik Wann Jensen
Professor Falko Kuester

2009

DEDICATION

To Mom and Dad

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

"Trust in the LORD with all your heart and lean not on your own understanding; in all your ways acknowledge him, and he will make your paths straight." (Proverbs 3:5-6 NIV)

First and foremost, I thank the Lord for guiding me through the Ph.D. program at UCSD. Through the easy and tough, the good and the bad, He has always been with me and has strengthened me by my side. In addition, I owe much gratitude and thanks for the many, many people who provided support and help during the formative years of my graduate study here at UCSD.

I would like to thank my advisor, Matthias Zwicker, for his support and guidance during my study. Early into the program, when I was struggling to define a research topic, Matthias helped me to get involved with a valuable collaborative research experience. During my last year, he graciously held skype meetings every week and supported me to keep pursuing my research topic. I also feel very fortunate to have visited Bern, Switzerland during the last summer to finish my thesis research. This dissertation would certainly not have been possible without his continued help and support over the past several years.

I thank the members of my committee, Serge Belongie, Sam Buss, Henrik Wann Jensen, and Falko Kuester, for their advice and their support. I will always remember their kindness and their willingness to help when I came knocking on the door for help.

The graphics and vision lab at UCSD was a stimulating and supportive environment. The countless hours of discussions and support from many friends helped to de-stress the grad school experience, especially during the hectic paper deadlines. During the beginning of my study, I was fortunate to have met many senior students, Sameer Agarwal, Kristin Branson, Piotr Dollár, Craig Donner, Satya Mallick, and Josh Wills, who paved the way and showed how it's done. I thank Neil Alldrin, Manmohan Chandraker, Wojciech Jarosz, Neel Joshi, and Vincent Rabaud, who were excellent lab mates and were always willing to entertain a question or two. I also thank Boris Babenko, Steve Branson, Krystle de Mesa, Toshiya Hachisuka, Arash Keshmirian, Wan-Yen Lo, Iman Mostafavi, Marios Papas, and Iman Sadeghi for many engaging discussions and helpful feedback. It was especially fun to have a graphics reading group meeting and discuss a new research paper each week.

In the CSE department, I was fortunate to have met many wonderful teachers, including Geoff Voelker, Russell Impagliazzo, T.C. Hu, Dean Tullsen, and Sanjoy Dasgupta. David Kriegman and Serge Belongie taught excellent classes and got me interested in computer vision, and Henrik Jensen's rendering algorithms class was a fun challenge and also a great software development experience. I also had a wonderful experience as a teaching assistant, and I will cherish the many fond memories interacting with students, debugging code, and solving math problems.

Personally, I would like to thank Hyun-Min Kang, Buhm Han, Won-Gyu Ju, and Seung-Nam Jung for their prayers, advice, friendship, and support during my study and stay at UCSD. During my undergraduate years at Harvey Mudd College, my thesis advisor Michael Orrison supported me with a valuable thesis experience that prepared me for writing this dissertation. I also thank Ran Libeskind-Hadas and Weiqing Gu for giving me an opportunity to perform research during those years. This experience gave me a feel of what research was like early on, even before I started grad school.

Through conferences and meetings, I met many researchers in my field who provided me support and encouragement. I would like to thank Niloy Mitra, Qi-Xing Huang, Hao Li, Michael Wand, Martin Bokeloh, and Alexander Berner, for their friendship, support, encouragement, and especially experimental comparisons and feedback. Even people who I have not met provided support by making their data and software available for other researchers. I would like to thank Brett Allen, Ilya Baran, Philip Fong, Craig Gotsman, Yuri Pekelny, Oliver Schall, Johnathan Starck, Bob Sumner, Thibaut Weise, and Li Zhang for providing much needed datasets for experiments. I also thank Gilles Debunne (libQGLViewer), David M. Mount and Sunil Arya (ANN library), Yuri Boykov, Olga Veksler, Vladimir Kolmogorov, and Ramin Zabih (Graph Cuts), Sivan Toledo (TAUCS), Tomas Akenine-Möller (Fast Triangle-Box Intersection Code), and Mutsuo Saito and Makoto Matsumoto (SFMT) for providing implementations of their software.

It has been thirteen years, through three schools and two universities, since I first set foot alone in the US. During the first several years, my aunt and uncle gave invaluable help by hosting me and supporting my studies. Through it all, my parents provided continued prayers, support, and guidance in academics and in life. Mom always encouraged me to stay on track, and Dad always gave valuable nuggets of advice as he recounted the days of his own graduate study. This thesis is dedicated to them.

Portions of this dissertation are based on papers which I have co-authored with others. My contributions to each of these papers are listed below.

- Chapter 4 is based on material published in the article:

  > Will Chang and Matthias Zwicker, "Automatic Registration for Articulated Shapes," Computer Graphics Forum (Proceedings of SGP), Vol. 27, No. 5, July 2008.

  I was the primary investigator and author of this paper.
- Chapter 5 is based on material published in the article:

  > Will Chang and Matthias Zwicker, "Range Scan Registration Using Reduced Deformable Models," Computer Graphics Forum (Proceedings of Eurographics), Vol. 28, No. 2, April 2009.

  I was the primary investigator and author of this paper.
- Chapter 6 is based on material that is in preparation for submission:

  > Will Chang and Matthias Zwicker, "Global Registration of Dynamic Range Scans for Articulated Model Reconstruction."

  I was the primary investigator and author of this paper.

VITA

2004	Bachelor of Science, Harvey Mudd College
2005–2009	Teaching & Research Assistant, Department of Computer Science and Engineering, University of California, San Diego
2006	Master of Science, University of California, San Diego
2009	Visiting Research Assistant, Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland
2009	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

Will Chang and Matthias Zwicker. "Range Scan Registration Using Reduced Deformable Models." *Computer Graphics Forum (Proceedings of Eurographics)*, Vol. 28, No. 2, April 2009.

Will Chang and Matthias Zwicker. "Automatic Registration for Articulated Shapes." *Computer Graphics Forum (Proceedings of SGP)*, Vol. 27, No. 5, July 2008.

Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. "Hair Photobooth: Geometric and Photometric Acquisition of Real Hairstyles." *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, Vol. 27, No. 3, 2008.

E. Miller, R. Libeskind-Hadas, D. Barnard, W. Chang, K. Dresner, W. M. Turner, and J. R. Hartline. "On the Complexity of Multicasting in WDM Networks with Tap-and-Continue and Multicast Capable Switches." *IEEE Journal on Selected Areas in Communications (JSAC-OCN)*, Vol. 22, No. 9, Pages 1601–1612, November 2004.

ABSTRACT OF THE DISSERTATION

Reconstruction of Dynamic Articulated 3D Models from Range Scans

by

William Young Chang

Doctor of Philosophy in Computer Science

University of California San Diego, 2009

Professor Matthias Zwicker, Chair

Our vision is to enable efficient acquisition and synthesis of highly detailed 3D surface models that are also easy to animate in a plausible and realistic way. The state-of-the-art surface acquisition technology is range scanning, which can measure surface geometry with a high degree of accuracy and speed. However, the output is only a partial view of the surface that has much missing data, and there is no tracking of the surface motion in the case of a moving subject. To reconstruct a complete model of the subject, we must align multiple range scans taken from different times and viewpoints to fill in the missing data and track the motion of the surface. At the same time, we would like to fit a reduced deformable model that expresses the surface motion in terms of a few intuitive parameters.

In this dissertation, we develop algorithms to process and align multiple range scans of a moving articulated subject. Our algorithms can automatically align multiple scans to a common pose, thus reconstructing the full geometry of an articulated subject along with a model of its motion. Our methods perform this alignment in a completely unsupervised way: without markers, a template, or a user-defined segmentation of the surface. A key contribution is the use of discrete optimization techniques to automatically estimate the articulated structure of the surface based on its motion.

First, we describe a method to align a pair of 3D surfaces that is robust to large motions and much missing data. This algorithm samples rigid transformations between the surfaces and performs an alignment by optimizing an assignment of the transformations to the surface. Its robustness to large motions makes it useful for initializing a registration.

Next, we present a technique to automatically fit an articulated surface motion model to a pair of range scans. We efficiently solve for the transformations and weights of this model by repeatedly estimating them in alternating fashion. The key benefit of this approach is that the solved model parameters can be used to easily and intuitively edit the pose of the scanned geometry.

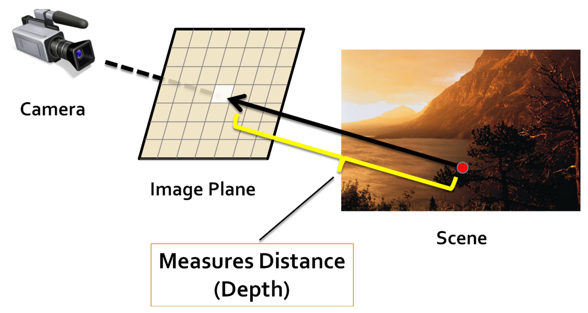
Finally, we improve and combine these two approaches to automatically reconstruct an articulated 3D model from multiple range scans. We reduce alignment error by simultaneously solving for the alignment of all input scans. We demonstrate that this method can reconstruct a variety of poseable, articulated 3D models from partial surface data acquired by a range scanner.

1 Introduction

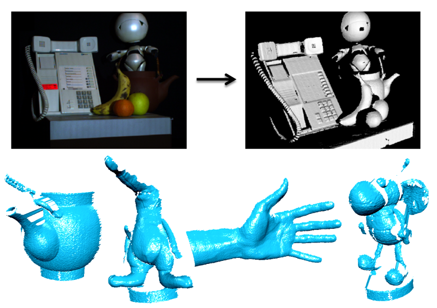
IN computer graphics, digital representations of 3D curves and surfaces (i.e. 3D models) are needed to generate virtual worlds and characters, which are then used to create realistic images that look like actual photographs. In addition, realistic motions and movement of these 3D models are needed to create natural animations and movies. Traditionally, these 3D models and animations are produced manually by the hand of an expert artist, with the help of highly specialized software tools. However, because of the great difficulty of this task and the high cost in production, researchers have sought to automate this task using two main approaches: physical simulation and capture.

Physical simulation can produce very accurate models and physical movement, but it is generally computationally expensive and difficult to control by an artist or director. In contrast, capture based methods provide a simpler approach that has been very popular in recent years. A classic example of this is motion capture, where a set of highly reflective markers are placed on a actor wearing a skin-tight suit. As the actor moves, the locations of the markers are tracked by multiple cameras placed around the person. While motion capture can acquire very natural stick-figure animations, it cannot capture fine surface movement such as the details of the skin and clothes of the actor. Instead, these details are synthesized manually or by physical simulation.

The vision of our work is to address this gap in the capturing of 3D models and



(a)



(b)

Figure 1.1: Range scanning technology measures the three-dimensional geometric content of a scene. (a) Range scanners capture high-resolution, per pixel depth that can be transformed into a detailed 3D point cloud of the scene. (b) Examples of surfaces captured using range scanners. These surfaces only offer a partial view of the object; notice that parts of the surface are missing. From Weise et al. [2007].

animations. The holy grail, so to speak, is to capture and generate highly detailed 3D surface models that are easy to animate in a plausible and realistic way. By adjusting only a few easy-to-understand parameters, an animator should be able to synthesize a realistic animation of a 3D model, incorporating effects such as the folding of skin, bulging of muscles, or wrinkling of cloth. These models then could be placed in movies and games to obtain a high quality result.

We believe that the key technology to enable this vision to capture and analyze an animated surface using range scanning. This technology allows us to acquire high-resolution, per-pixel depth images, which can be directly transformed into a detailed 3D point cloud of the scene (Figure 1.1a). While this was limited to the acquisition of rigid (i.e. static) objects in the past, recently the technology has developed to the point where high-resolution, real-time range scanners are available [Zhang et al., 2004; Zhang and Huang, 2006; Weise et al., 2007]. Unlike static range scans, the *dynamic range scans* (or range video, depth video) captured by these cameras contain both information about the surface and also how it moves. This provides an exciting opportunity to automatically acquire both a detailed surface and a model of its motion.

Although the capture technology has improved in leaps and bounds, many challenges still remain in processing the captured data to reconstruct the shape and motion of an entire object. First, since we typically cannot observe the entire surface of an object from a single viewpoint, range scans can only acquire partial surfaces with occlusion and missing data (Figure 1.1b). Second, even if we could observe the entire surface, range scanners do not track the motion of the scanned points over time. Range scans just tell you the location of a set of points on the surface, and they do not describe how each point on the surface moves over time.

These two primary challenges can be formulated as solving the *registration* task: the task of aligning multiple range scans based on the shape of the acquired geometry. If we can perform registration correctly, we can easily solve both issues. For the first issue, we can take multiple range scans from different viewpoints (or use multiple cameras), and align all scans to fill in the holes with surface data from different viewpoints. For the second issue, aligning the surface gives correspondences between the range scans, which allows us to track individual point locations and thus figure out the movement of each point. While registration offers a solution to these challenges, solving for an accurate registration is challenging in itself, especially with moving surfaces that have much occlusion and missing data.

Many ideas have been proposed to solve this problem. One idea is to place markers on the surface (much like motion capture) to track a few locations over time [Allen et al., 2003; Anguelov et al., 2005; Pauly et al., 2005]. These markers then guide the registration into the correct result. If we cannot physically place markers on the surface (because it affects the motion or it obstructs the capture process), these locations must be selected manually by clicking on corresponding points on the surface, which involves some manual work and may be difficult at times. Another idea is to use a template, which is a highly detailed, complete 3D scan of the object in a fixed, specific pose [Allen et al., 2002, 2003; Sand et al., 2003; Anguelov et al., 2005]. If this is available, then we can just align the template to every range scan to obtain an animation of the template that fits the data. However, with this type of approach, most of the surface detail comes from the template and not from the captured data.

In this dissertation, we develop algorithms to reconstruct the shape and motion of an object from range scans without using markers or a template. Instead, we focus on the specific class of *articulated* motion, which means that the surface consists of multiple rigidly moving parts. Many surfaces that we are interested in capturing have articulated motion (e.g.

people or animals), but it is less applicable to other common surfaces, such as cloth or liquid. This idea has been used for surface and motion capture given a user-provided segmentation of the surface (the division of the surface into parts) [Pekelny and Gotsman, 2008]. In our work, we do not assume that we have a segmentation of the surface provided by the user. Instead, we automatically estimate this segmentation automatically.

1.1 Summary of Original Contributions

To summarize our problem statement, we are given as input a set of range scans, where the scans are taken from a moving subject. This means that the pose of the shape is different in each range scan, so there is a spatially varying movement of the surface. Furthermore, we expect that there is a significant amount of missing data in each scan. We develop algorithms to align these input scans to a common pose, so that all of the surface data can be gathered in one place to obtain the complete surface of the subject. At the same time, our methods use an articulated motion model to describe the motion of the object, which means that we solve for (1) transformations aligning the surface and (2) the weights that associate these transformations to each point on the surface. Solving for this articulated model automatically gives us a reduced representation of the surface motion, which allows us to target the reconstructed model into new poses and create animations. Now, we outline our specific contributions below.

- **Automatic Registration for Articulated Shapes:** We present an unsupervised algorithm for aligning a pair of shapes in the presence of significant articulated motion and missing data. We explicitly sample the motion, which gives a priori the set of possible rigid transformations between parts of the shapes. This transforms the problem into a discrete labeling problem, where the goal is to find the assignment of transformations that produces the best alignment of the shapes. We apply graph cuts to solve this optimization problem. We demonstrate a good alignment can be obtained even when there is a large motion, and when there is much missing data. This technique can be used as a preprocessing step that gives a good initialization of the segmentation and registration between a pair of range scans.

- **Range Scan Registration Using Reduced Deformable Models:** We present a method for aligning a pair of articulated range scans. The key idea is to model the motion of the underlying object using a reduced deformable model (RDM) suitable for modeling articulated motion. We formulate the registration problem as estimating the parameters of this model, and we describe how to solve this efficiently using an EM-type algorithm. We demonstrate that this method can accurately register pairs of range scans with a significant amount of occlusion. One draw back is that this method is limited to aligning only a pair of scans, but we develop it further in the next contribution to handle multiple scans.

- **Global Registration for Articulated Model Reconstruction:** We extend and combine the algorithms of the previous chapters to develop an method capable of automatically reconstructing an articulated 3D model from multiple range scans. We use the first contribution as a pre-processing step to initialize the registration of the range scan sequence. We extend the second contribution to refine this initialization and solve for a consistent model of articulated motion that aligns all of the scans. Here, we develop a framework to simultaneously align multiple scans to reduce alignment errors. We demonstrate that we can reconstruct a complete 3D articulated model from a sequence of dynamic range scans. In addition, the estimated articulated structure allows us to target the reconstructed shape in new poses and create animations.

The main advantage of our algorithms is that they do not require user specified markers, a template, nor manual segmentation of the surface geometry into rigid parts.

1.2 Organization of the Dissertation

This dissertation is divided into seven chapters. In Chapter 2, we provide an overview of previous work that is related to the algorithms presented in this dissertation. In Chapter 3, we discuss in detail several closely related algorithms that we use and extend in the remaining chapters of the thesis. Chapter 4 presents our algorithm that uses a transformation sampling and assignment strategy to robustly align articulated shapes, and Chapter 5 discusses our method to solve for reduced motion parameters to align range scans. In Chapter 6, we extend these algorithms to reconstruct a full articulated model from multiple scans, and we conclude in Chapter 7 with a summary of our work and describe potential avenues for future work.

2

Related Work

WE divide the related work into five categories: (1) rigid registration, which focuses on aligning surfaces of rigid objects, (2) non-rigid registration, which handles moving objects, (3) reconstruction from silhouettes, which captures surface motion from from multi-view video, (4) motion modeling from mesh animations, and (5) non-rigid structure from motion.

2.1 Rigid Registration

Iterative Closest Points (ICP) Algorithm. The surface registration problem is a classic, well-studied problem in computer graphics and vision. A popular and well-developed method for aligning rigid surfaces is the Iterative Closest Point (ICP) algorithm, developed independently by Chen and Medioni [1991] and Besl and McKay [1992]. This approach aligns a pair of surfaces by iteratively pulling the surfaces together using closest point matches. This method works well for producing accurate registrations of surfaces, but it does require a good initial alignment of the surfaces. Without an initial alignment, the ICP algorithm may produce an incorrect registration. In our work, we extend the ICP method to handle articulated surfaces. The difference is that instead of having just one transformation that describes the movement, we have multiple rigid transformations (one for each part), and neighboring parts are constrained to meet together at joints. We will discuss the ICP algorithm in more detail in Chapter 3.

There have been many extensions to improve the robustness of the ICP algorithm and to handle the registration of multiple surfaces. A good survey of these techniques is given by Rusinkiewicz and Levoy [2001] and Rusinkiewicz [2001]. There are improvements in the selection of points on the surface to include more salient features such as lines and corners [Rusinkiewicz and Levoy, 2001], the matching of points for faster and reliable performance (for example, projecting points from one surface to another [Blais and Levine, 1995; Bergevin et al., 1996] or incorporating color, intensity, or surface normal information [Godin et al., 1994; Johnson, 1997; Weik, 1997; Pulli, 1999]), and weighing/rejecting points based on boundary, normals, distances, scanner angle, and occlusion [Turk and Levoy, 1994; Bergevin et al., 1996; Weik, 1997; Dorai et al., 1998; Neugebauer, 1997; Pulli, 1999]. In addition, as an alternative to the least-squares error minimization in the original ICP algorithm, a robust M-estimator can be used instead to reduce the effect of outlier correspondences that bias the minimization in favor or large residuals [Nishino and Ikeuchi, 2002]. We also incorporate many of these improvements in our algorithm, and the specific improvements we use are discussed along with the ICP algorithm in the next chapter.

Handling Multiple Range Scans. Although the ICP algorithm only applies to a pair of surfaces, we can extend it to align multiple surface fragments to reconstruct a complete 3D surface from multiple range scans. A basic strategy is to align each surface to all previously registered surfaces [Chen and Medioni, 1991]. While this strategy works often for smaller datasets, it causes an accumulation of alignment error for larger datasets. Researchers have explored some alternative strategies, for example, repeatedly picking one surface and aligning it to all other surfaces [Bergevin et al., 1996; Benjemaa and Schmitt, 1998; Williams and Bennamoun, 2000], or optimizing the alignment of all scans simultaneously [Stoddart and Hilton, 1996; Neugebauer, 1997]. In our work, we apply the simultaneous registration idea to reduce alignment error, because it offers faster convergence compared to the methods that align only one surface at a time.

One drawback of simultaneous registration is the memory requirement needed to keep large sets of range scans in memory. Our method shares the same drawback, but the simultaneous registration approach makes it straightforward to solve for a single articulated structure that reflects the movement of all frames. An approach to reduce the memory requirements is discussed by Pulli [1997], Pulli [1999], and Brown and Rusinkiewicz [2007]. In this line of work, correspondences between pairs of scans are precomputed in advance, and the registration of all scans is performed using this smaller set of correspondences. It may be possible to precompute pairwise correspondences in our work, but we have not explored this idea and leave it for future work.

As mentioned before, a limitation of ICP is that it is only effective if we have a good initial registration of the surfaces. Although a user could interactively select correspondences to provide an initial registration, this quickly becomes cumbersome for large datasets. A popular technique to automate the initial registration is to directly match feature descriptors derived from the surface geometry. This gives "informed correspondences" which we can use to roughly estimate the initial registration of the surfaces. A variety of shape descriptors have been proposed, for example, spin images [Johnson, 1997], 3D shape contexts [Frome et al., 2004], 3D tensors [Mian et al., 2006], HMM descriptors [Castellani et al., 2008], scale dependent/invariant features [Novatnack and Nishino, 2008], and many more. In addition, geometric constraints between points can be used to identify matches as well [Aiger et al., 2008]. In Chapter 4, we develop an algorithm that uses spin image descriptors to find a set of correspondences between a pair of articulated surfaces. In contrast to other methods, we go one step further by deriving transformations for each correspondence, clustering these transformations, and optimizing for an assignment of the transformations to surface that produces the best alignment.

Alternatives to ICP. Finally, we mention some alternatives to the ICP method that are different from our work. The geometric hashing approach by Lamdan and Wolfson [1988] hashes a collection of objects, or different poses of the same object. Then, given an object (or pose of an object) that we want to identify, we can query the hash table to retrieve the object identity or pose parameters. This can be a fast approach, but it does not produce as accurate results as

ICP and requires a significant amount of additional storage. Pottmann et al. [2004] develop a technique to register a pair of surfaces by using precomputing approximations the squared distance function to a surface, storing this approximation in a hierarchical structure called the d2tree. This work has inspired researchers to combine the point-to-point and point-to-plane error metrics in ICP, which we use in our articulated ICP registration method. Finally, Li and Hartley [2007] describe a branch-and-bound approach on the space of rotations to find the globally optimal alignment between two surfaces. This work is limited to solving for a single rotation that aligns a pair of point clouds, and it is much slower than the ICP method because it investigates the alignment error on the space of all rotations. In our work, we do not adopt this approach because hundreds of rotations need to be estimated to align a collection of articulated range scans. In this case, the search space is much larger, so the registration would be computationally expensive using a branch-and-bound type method.

2.2 Non-Rigid Registration

In recent years, there has been a considerable amount of research interest in the acquisition and processing of non-rigid surfaces from range scans. In this section, we discuss techniques that address problems in non-rigid registration.

Template and Marker Based Methods. Acquiring shapes from range scans is a challenge because they contain holes and occlusion, even when the scans are taken from multiple viewpoints. To alleviate this problem, researchers have used a complete template shape and fit it to the data, which results in a clean surface without any holes and artifacts. To solve the problem of fitting the template shape to range scans, a number of algorithms rely on tracking physical markers placed on specific locations on the surface. The markers are identified in each scan, resulting in initial correspondences to guide the registration to the correct result. Our work is different from these approaches because we do not use a template or markers to reconstruct the shape of surface from range scans. However, we briefly survey some methods that are related to our work.

One similar approach is the correlated correspondence algorithm by Anguelov et al.

[2004b]. Given a template shape, they find a good correspondence assignment by formulating a Markov Random Field (MRF) optimization that best fits the observed data and preserves the shape of the template. They are able to match range scans to the template shapes well, in spite of significant changes in object pose. This technique has also been applied to analyze human body shapes [Anguelov et al., 2005] by fitting a template mesh to the scans with the additional help of a few markers. Our algorithm in Chapter 4 is similar to the correlated correspondence algorithm, because our algorithm also formulates an MRF optimization which is solved using graph cuts. However, our method differs because we find an optimal assignment of *transformations* rather than an explicit assignment of corresponding points. With our method it is possible to assign transformations even when no corresponding point is available, so we do not require a template shape as additional input.

Allen et al. [2002] use range scans with marker positions to construct a deformable human upper body model. The markers are used to determine the pose of the body, and the fine details of the surface are reproduced by solving for a dense field of displacements on a simplified cylindrical model. They also extend this technique for full body scans [Allen et al., 2003] by fitting a full-body template model to the range scans. Sagawa et al. [2007] rely on matching both color texture and shape features to register a sequence of deforming range scans, using a similar optimization technique as Allen et al. [2003]. Also, the example-based 3D scan completion work by Pauly et al. [2005] uses markers to optimize a per-point displacement aligning complete examples to partial range scans. Other than the use of markers and a template, these methods are different from our work because they focus on solving for per-vertex displacements or transformations. Our work focuses on solving for a small number of rigid transformations that are associated with many vertices, not just a single vertex. As a result, we solve for a coarser, articulated alignment of the surface, whereas these methods focus on more detailed, high-quality per-vertex alignments that capture more non-rigid surface detail.

Finding Symmetries and Correspondences. Mitra et al. [2006] estimate symmetries within and among shapes by clustering transformations sampled from a rough set of correspondences.

In a subsequent paper they further extend this to make a shape more symmetric, and also to perform registration on an articulated shape [Mitra et al., 2007b]. In Chapter 4, we apply the symmetry detection technique by Mitra et al. [2006] to estimate the partial movement of the surface. However, if there are multiple parts that are shaped similarly (e.g. arms or legs of a person), this technique alone cannot distinguish between these parts and results in multiple possible matches. We address this limitation by optimizing an assignment of transformations that penalizes incorrect matches causing excessive stretch on the surface.

Another class of methods focuses on finding a set of correspondences between a pair of shapes. Notably, Zhang et al. [2008] extract a small set of correspondences by selecting the set that minimizes the distortion between the surfaces. While this method tries to establish only a few correspondences between different shapes of the same class (such as a dog and a wolf, possibly in different poses), our work attempts to find *dense correspondences* among a set of range scans acquired from the same object.

Starck and Hilton [2007] introduce a method similar to the approach by Anguelov et al. [2004b] that formulates a MRF optimization of correspondences between a pair of free-form surfaces. This method is applicable for complete, closed shapes, where geodesic distances between points can be computed. As we have mentioned before, our work is different because we optimize for an assignment of transformations on the surface, rather than optimizing for correspondences.

Tevs et al. [2009] randomly sample a set of geodesically consistent correspondences using an importance sampling based approach. The samples are drawn based on a probability distribution that reflects (1) how well the shape descriptors match between corresponding points, and (2) how well each correspondence preserves the geodesic distances to the set of correspondences that have already been picked. While this method is purely a random sampling approach to finding correspondences, our work additionally derives rigid transformations from the correspondences and optimizes their assignment to the surface.

Parameterization Based Methods. A large body of work on mesh morphing solves the correspondence problem by finding a common base parameterization across multiple meshes of a common topological type [Kaul and Rossignac, 1991; Kent et al., 1992; Kanai et al., 1997; Gregory et al., 1998; Lee et al., 1998, 1999; Kanai et al., 2000; Ohbuchi et al., 2001; Praun et al., 2001; Takahashi et al., 2001; Kraevoy and Sheffer, 2004]. Most of these parameterization-based methods use a set of common user-placed markers between the shapes. Recently Lipman and Funkhouser [2009] developed a markerless approach based on flattening a pair of closed surfaces to the complex plane. They observe that isometric deformations are related by a Möbius transformation, and they develop an algorithm that aligns the flattened surfaces and votes on mutually closest points to extract a reasonable set of correspondences between the pair of shapes.

These parameterization-based approaches are usually not applicable for aligning range scans, because finding a parameterization requires a closed surface with no holes or boundaries. It is possible to overcome this limitation with a rough template and a few markers providing correspondence [Kraevoy and Sheffer, 2005]. This method was used by Bradley et al. [2008] for aligning and reconstructing deforming garments from 3D point clouds obtained via multiview stereo. However, our work does not use markers or a template to reconstruct the shape, but does this directly with range scans using no additional input.

Pairwise Non-Rigid ICP Methods. A well-studied class of algorithms align surfaces by generalizing the ICP algorithm to handle non-rigid deformation. These methods resemble the ICP algorithm because they use the closest points between the surfaces as correspondences to optimize for a non-rigid alignment. These algorithms can be roughly categorized by how they model the motion of the deforming surface. Methods that model using thin-plate splines [Chui and Rangarajan, 2003; Brown and Rusinkiewicz, 2007] are able to solve for a globally smooth deformation that aligns the surfaces, but are unable to effectively express large or piecewise rigid deformations. Many techniques model the motion of each individual point using an affine transformation [Allen et al., 2003; Amberg et al., 2007] or just a per-point displacement vector [Shelton, 2000; Hähnel et al., 2003; Pauly et al., 2005], but these techniques also constrain the overall deformation to be smooth. In contrast to these methods, our work solves for a segmentation of the surface into rigid parts, which allows us to easily express piecewise rigid motions.

In a related approach, Li et al. [2008] solve for a reduced set of transformations to model the deformation by prescribing weights on the surface that associate the transformations to the surface points. Unlike traditional ICP algorithms, they do not maintain an explicit set of correspondences, but instead rely on a 2D parameterization of range scan to automatically detect overlapping regions and minimize the distance between the surfaces. Our work does not require a parameterization of the range scans, and also we do not prescribe weights but solve for the weights (i.e. segmentation) that associate the transformations to the surface.

Huang et al. [2008] describe a method to improve the traditional closest-point ICP strategy by extracting a set of geodesically consistent correspondences using the spectral matching technique [Leordeanu and Hebert, 2005]. This allows the technique to handle much larger motions, but it requires constructing a graph on the surface that reflects the geodesic distance between points on the surface. Unfortunately, constructing such a graph is not always possible for range scans, because they may have too much missing data. In these cases, the spectral matching does not produce good correspondences. In our work, we construct a graph as well, but the graph does not need to approximate geodesic distances.

After determining consistent correspondences, Huang et al. [2008] efficiently solve for transformations at many sampled locations on the surface, and interpolate the rest of the surface using prescribed influence weights. Here, they accelerate the registration by clustering the surface into rigid parts during the optimization. Although this idea sounds similar to our work, it is different because it just merges samples that have similar transformations. In contrast, we actually optimize for the assignment of transformations to the samples that produces the closest alignment.

Multiple Frame Methods. So far, most of the techniques we have discussed involve aligning a pair of frames. To solve the registration problem for multiple frames, a class of methods models the deforming surface as a single surface in 4D space-time. For example, Mitra et al. [2007a] model a sequence of range scans as a space-time surface and analyzes local kinematic properties of this 4D surface for aligning an entire sequence of range scans. In addition, Süßmuth et al. [2008] propagate a template mesh in space and time to reconstruct a 4D space-time function. These techniques rely on dense temporal and spatial sampling to reconstruct the 4D surface, and are less robust to larger motions or a coarser sampling of the surface. In contrast, our method does not require a dense temporal sampling and can handle the registration of reasonably large motions.

Sharf et al. [2008] apply a robust 4D surface reconstruction technique to find the 4D space time volume that fits a sequence of range scans. This technique is applicable when most of the surface is visible in the point cloud, and it focuses on filling holes in the surface. Also, this method reconstructs a 4D volume and not a mesh animation; therefore, the reconstructed 3D meshes at each time are not in correspondence. In contrast, we produce a single 3D surface, with associated transformations and weights that fit the surface to the data observed in each frame.

The statistical optimization technique by Wand et al. [2007, 2009] offer an alternative that is closer to the ICP-based methods. The recent method by Wand et al. [2009] presents a hierarchical registration approach, where they align every two frames, merge, then align every two pairs of frames, merge, and so on until they process the entire sequence. In addition, they decouple the surface and deformation representations, and use an adaptive deformation field to make the optimization more efficient. Their method is also robust to topological changes as well. However, because they use a closest point strategy to align the frames, their method is less robust to large motions caused by low frame rates or occlusions. Also, in contrast to our method which focuses on articulated motion, the adaptive deformation field is defined spatially and is more appropriate for smooth and non-rigid deformations.

Segmentation-Based Methods. A method that is closely related to this thesis is the method by Pekelny and Gotsman [2008], who reconstruct a full 3D model and articulated skeleton with the help of a user-provided segmentation. They perform ICP for each rigid part and accumulate the scanned geometry from each frame. In Chapter 5 and 6 we also develop an ICP based method to optimize the registration, but we estimate the segmentation automatically based on the motion of the input data. Also, we apply the articulated registration algorithm developed in Chapter 4 to make our method robust to larger motions.

2.3 Silhouette Based Shape Capture

The algorithms in our thesis reconstruct articulated models from detailed 3D point clouds obtained by a range scanner. However, it is also possible to capture surfaces and their motion using just video data. The main idea behind these approaches is to capture the silhouette of the shape using multiple synchronized and calibrated video cameras placed around the subject. To do this effectively, the subject usually needs to be in a special studio with a green or white background so that the silhouette is clearly visible. Then, a rough estimate of the 3D shape called the *visual hull* can be computed from the silhouettes, and this can be used to estimate the pose of the subject and/or to fit a more detailed template shape.

For example, the method by Cheung et al. [2003] refines the visual hull extracted from silhouettes in a multi-view video sequence to reconstruct the surface of the shape. Furthermore, they estimate joints in the articulated model by analyzing sequences of videos where one joint moves at a time. The resulting articulated model can then be used to track the motion in new video sequences. The method by de Aguiar et al. [2004] represents the visual hull using 3D voxels, to which ellipsoids are fit in order to track the pose of the subject. These methods are outside the scope of our work, because the algorithms apply to fundamentally different input data. While range scans offer high-quality, detailed surface capture, silhouettes only capture the shape at very sparse locations and can only provide a rough estimate of the overall shape.

To further improve the quality of the reconstruction, researchers have often used a template model (which is typically a high-quality complete 3D scan of the subject) and fit this model to the silhouettes. For example, Carranza et al. [2003] deform an articulated template model to match multi-view silhouettes in a video sequence. Here, they try to maximize the the overlap of the deformed template with the silhouettes by using an XOR operation implemented on graphics hardware. Sand et al. [2003] use both markers and silhouette data to deform a template to match a video sequence. The markers determine the initial pose of the template, which is further refined to reproduce the surface shape by solving for a dense set of displacements on the surface. Here, the actor needs to wear a skin-tight suit, so the deformations captured by this method are limited. In addition, de Aguiar et al. [2007] use silhouettes and optical flow to guide the deformation of a template, de Aguiar et al. [2008a] use silhouettes, stereo reconstruction, and image features tracked on the shape, and Vlasic et al. [2008] and Gall et al. [2009] use the visual hull to estimate the skeletal pose and deform the template to fit the silhouettes exactly.

These template-based approaches are nice because they produce a single 3D mesh that can be animated to reproduce the video sequence exactly. However, The drawback of this work is that most of the fine surface detail in the video is lost in the reconstructed shape. Instead, much of the surface detail comes from the surface of the template model itself. It is possible also to “add back” the surface detail using the image data, but this has to be detected and simulated in a separate step [Popa et al., 2009]. In contrast, range scans can directly capture the fine surface detail and can produce higher quality results.

2.4 Modeling Motion from Mesh Animations

Methods that learn a surface motion model from a mesh animation is also related to our work. A mesh animation is an animated, complete 3D surface, where the position of each vertex moves to a known 3D coordinate at each point in time. Therefore, all of the animation frames are in complete correspondence, so the surface and correspondence information is already known. The focus of this area is not to estimate the shape of the object, but instead to find correlated patterns of the surface motion.

The motivation is that specifying the location of every vertex to animate a shape is difficult and very unintuitive for a user. In addition, since the position of each vertex is specified separately, lot of storage space is required to store the animation. The main observation to improve this situation is that the motion of each vertex is not independent; instead it is highly correlated to the motion of other nearby vertices. Therefore, it should be possible to find patterns of the surface motion, and further use this to extract a set of few parameters that completely describe the motion of the surface. This can be thought as fitting a reduced model of surface motion to a mesh animation. Once this model is fit, the mesh animation can be stored more efficiently, or manipulated more easily in new poses to create new animations.

An example of this is to find an articulated skeleton that fits the shape, and associate the vertices of the surface with the bones of the skeleton using "vertex weights" [Anguelov et al., 2004a; Schaefer and Yuksel, 2007; de Aguiar et al., 2008b]. Then, by moving the bones, the surface also moves accordingly. This process is also known as "rigging" or "skinning" in the computer animation literature, but here the skeleton and vertex weights are computed automatically from a mesh animation. James and Twigg [2005] take a more abstract approach and do not construct an explicit skeleton. Instead, they directly approximate the surface motion as a set of affine transformations that describe the motion of specific regions of the surface. These regions are specified by the vertex weights, which also smoothly blend boundaries between regions to faithfully reproduce the input animation.

Another example area is to analyze shape variation across a class of similar shapes. The goal is to find a set of few parameters that describe the complete shape variation. For example, a collection of human body examples can be analyzed to find a correlation between body shape and age, sex, weight, or height, or to correlate between body shape deformation and specific body poses or body types (e.g. muscle bulging when flexing arms, or skin folding differently depending on body weight) [Allen et al., 2003; Anguelov et al., 2005; Allen et al., 2006; Hasler et al., 2009].

In contrast to these methods, the algorithms developed in this thesis compute reduced motion models from incomplete range scans without a template or any predefined correspondences. Our problem is more general, because we do not have predefined correspondences nor a complete shape of the object. However, the idea that we would like to fit a reduced motion model is similar in our work.

2.5 Non-Rigid Structure From Motion

A final area that is related to our work is the non-rigid structure from motion (NRSFM) literature from computer vision. Given a non-rigidly moving shape, specified by several 2D coordinates tracked over time using a single camera, the goal is to reconstruct the 3D coordinates of these points in every frame. In addition, some of the 2D features may be missing in some frames due to occlusion, so an additional goal is to fill the 3D locations of these missing features. Typically, only a small number of 2D locations (50-100) are used, and the correspondence over time is already given as the locations are individually tracked. Successful methods have modeled the shape movement as a linear combination of a few basis shapes [Torresani et al., 2008a] or as a non-linear manifold [Rabaud and Belongie, 2008]. The problem statement is different in our work: our method assumes that the 3D points are already given, but no correspondences are given between frames. The goal in our work is to find correspondences and gather the surface data from all frames to reconstruct the full shape of the object. At the same time, we model the surface motion using an articulated deformation model.

3

Technical Background

TO reconstruct an articulated model from range scans, we build on a variety of work in computer graphics and vision. In this chapter, we explain in detail the building blocks that we use to develop our algorithm.

Our problem statement is to align a set of range scans to a common pose, so that all of the surface data can be gathered in one place to obtain the complete surface of the subject. Here, the subject is moving to a different pose in each scan, and we assume that the motion is articulated. Thus, to align the scanned surface, (1) we divide the surface into multiple rigid parts and (2) align the surface of each part separately, with (3) methods to nicely blend the transformations at boundaries between parts.

To align the surface of each part, we employ the ICP algorithm. This method is used extensively in the next chapters, so we discuss the ICP algorithm in detail in the first part of this chapter. A drawback of the ICP algorithm is that it requires a good initial alignment of the parts. Therefore, we estimate an initial alignment by computing and matching shape descriptors derived from many locations on the surface. In our work, we use spin images and principal curvatures, and matching these descriptors gives “informed correspondences” which we use to generate reasonable guesses of the initial transformations. This is used to generate candidate transformations in Chapter 4, and also to generate initial correspondences in Chapter 5. In addition, we employ the mean-shift clustering algorithm to identify groups of similar transformations that affect significant portions of the surface. This is used in Chapter 4. We discuss the spin images, principal curvatures, and the mean-shift clustering algorithm in the second part of this chapter.

The other major component of our algorithm is to estimate the division of the surface into multiple rigid parts. Our approach is to formulate this as a discrete labeling problem, where we assign "part labels" to the vertices of the surface to produce the best alignment between the input scans. Note that each part label is associated with a rigid transformation, so we use the terms "assigning part labels" and "assigning transformations" interchangeably. To solve the discrete labeling efficiently, we use the graph cuts algorithm. We use this technique for Chapters 4, 5, and 6, and we discuss the algorithm in detail in the third part of this chapter.

Finally, we conclude this chapter with a discussion of articulated deformation models. A discrete assignment of the part labels often causes artifacts at boundaries between different parts. These deformation models describe how to smoothly blend the transformations at the boundaries between rigid parts to remove artifacts and produce a higher quality result. We use this technique in Chapter 5 and 6.

3.1 Registration Using the ICP Algorithm

The first building block we will discuss is the iterative closest point (ICP) algorithm that is used for aligning the rigid parts of the surface Besl and McKay [1992]. Given two surfaces, the ICP algorithm finds a rotation and translation that minimizes the distance between the two surfaces, producing an alignment or registration. We use this algorithm because it gives a fast and accurate registration, compared to alternatives that require more storage, time, or precomputation.

In the following sections, we will discuss the basic ICP algorithm, followed by a discussion of relevant extensions. For this discussion, we will denote the source shape as $\mathcal{P}$, the target shape as $\mathcal{U}$, and we will assume that each shape is just a set of 3D points. Note that this algorithm is applicable beyond just 3D point sets; it applies to the registration of line segment sets, implicit curves/surfaces, parametric curves/surfaces, triangle meshes, etc., as long as there is a routine to compute the closest points between the source and target shapes.

3.1.1 Basic Algorithm

To minimize the distance between the source and target shapes, we first need a good description of the distance between the shapes. In the ICP algorithm, we define a distance between a source point $\mathbf{x} \in \mathcal{P}$ and the target shape $\mathcal{U}$ as

$$d(\mathbf{x}, \mathcal{U}) = \min_{\mathbf{y} \in \mathcal{U}} \|\mathbf{x} - \mathbf{y}\|^2, \tag{3.1}$$

that is, the distance between the $\mathbf{x}$ and the closest point $\mathbf{y} \in \mathcal{U}$. By summing the total distance over all points in $\mathcal{P}$, we obtain a measure of distance between the two shapes:

$$d(\mathcal{P}, \mathcal{U}) = \sum_{\mathbf{x} \in \mathcal{P}} d(\mathbf{x}, \mathcal{U}). \tag{3.2}$$

Now, the goal is to apply a rotation and translation to $\mathcal{P}$ so that the above distance is minimized. We can express this optimization problem in the following objective function:

$$\underset{R, \vec{\mathbf{t}}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathcal{P}} \min_{\mathbf{y} \in \mathcal{U}} \left\| R\mathbf{x} + \vec{\mathbf{t}} - \mathbf{y} \right\|^2. \tag{3.3}$$

Thus, we want to find the optimal rigid transformation $(R, \vec{\mathbf{t}})$, such that when we apply the transformation to the source points as $R\mathbf{x} + \vec{\mathbf{t}}$, the total distance to the closest points in $\mathcal{U}$ is minimized. Note there are two minimization problems in this objective: (1) finding the closest points in $\mathcal{U}$, and (2) finding the best $(R, \vec{\mathbf{t}})$ minimizing the total distance.

Notice that for each value of $(R, \vec{\mathbf{t}})$, the closest point $\mathbf{y} \in \mathcal{U}$ can change to a different location on the shape. It would be nice if we could find an equation for the closest point $\mathbf{y}$ in closed-form, but this does not seem easy to do in general. To solve this problem, the basic strategy of the ICP algorithm is to perform an alternating optimization. Each step of the optimization minimizes one component of the objective function, keeping all other components fixed. Specifically, in the first step we first keep the $(R, \vec{\mathbf{t}})$ fixed, transform the

Algorithm 3.1: ICP($\mathcal{P}, \mathcal{U}, \tau$)

```
Data: 3D points of the source shape $\mathcal{P}$ and target shape $\mathcal{U}$, convergence
      tolerance $\tau$
Result: Optimal aligning rigid transformation $(R_k, \vec{\mathbf{t}}_k)$
1  begin
2      Let the iteration number $k \leftarrow 0$;
3      Let the initial rotation and translation $R_0 \leftarrow I, \vec{\mathbf{t}}_0 \leftarrow \vec{\mathbf{0}}$;
4      Let the initial alignment error $d_0 \leftarrow \infty$;
5      while Not converged do
6          Increment iteration count $k \leftarrow k+1$;
7          Compute the closest points $\mathcal{U}_k$ for this iteration:
               $\mathcal{U}_k = \left\{ \mathbf{y}_k \,\middle|\, \text{for each } \mathbf{x} \in \mathcal{P},\ \mathbf{y}_k = \text{argmin}_{\mathbf{y} \in \mathcal{U}} \left\| R_{k-1}\mathbf{x} + \vec{\mathbf{t}}_{k-1} - \mathbf{y} \right\|^2 \right\}$;
8          Find the best transformation $(R_k, \vec{\mathbf{t}}_k)$ minimizing
               $d_k = \sum_{\mathbf{x} \in \mathcal{P}, \mathbf{y}_k \in \mathcal{U}_k} \left\| R_k\mathbf{x} + \vec{\mathbf{t}}_k - \mathbf{y}_k \right\|^2$;
9          Check for convergence by examining if $|d_k - d_{k-1}| < \tau$;
10     return Final transformation $(R_k, \vec{\mathbf{t}}_k)$;
11 end
```

source positions $\mathbf{x}$ according to $(R, \vec{\mathbf{t}})$, and find the closest points $\mathbf{y}$ to each source position. Then, in the second step we keep the closest points $\mathbf{y}$ fixed, and we estimate the optimal transformation $(R, \vec{\mathbf{t}})$ aligning the source positions to the target positions. This alternation continues until it converges; i.e. the closest points do not change, or equivalently, the estimated $(R, \vec{\mathbf{t}})$ does not change any longer. This strategy is summarized in Algorithm 3.1.

The two main components of the algorithm are (1) determining the closest point on the target shape $\mathcal{U}$ and (2) estimating the optimal $(R, \vec{\mathbf{t}})$ given the correspondences $\mathbf{x}$ and $\mathbf{y}$. The efficiency of these two key steps determines the efficiency of the overall algorithm. We will discuss these steps next. Note that the ICP is inherently a local search method, and it does not guarantee to find the global minimum of the objective (3.3). Thus, having a good initial alignment is important for the success of this method.

Efficient Closest-Point Queries. The simplest way to search for the closest point on the target shape is to compare the distance between the query point and all points on the target, and choose the point that produces the minimum distance. However, this approach is costly at $O(N)$ time per query (where N is the number of points on the target), especially when the query is repeated for all points in the source.

We can reduce this cost by precomputing a kd-tree of the points in the target shape [Bentley, 1975]. This tree allows nearest neighbor queries to be answered much faster, in expected-case $O(\log N)$ time [Maneewongvatana and Mount, 1999]. Although the precomputation takes $O(N \log N)$ time [Maneewongvatana and Mount, 1999], this only needs to be computed once for the target shape [1].

If the target shape is a range scan, then an alternative to nearest-neighbor search is the projection method, also known as inverse (or reverse) calibration [Blais and Levine, 1995; Neugebauer, 1997]. A range scan is a digital image where each pixel value is a depth, measuring the distance from the camera to the corresponding point in the scene. By using the calibration information of the camera, we can map each pixel in the image to a 3D ray in the scene. Then, combining this with the depth values, we can convert each pixel in the range scan to a 3D location in the scene. This process converts the range scan into a 3D point cloud, where the points represent sampled locations on the surface of the captured scene.

The projection method uses the calibration information to project any position in the scene to a pixel location on the camera image. Thus, given a source position $\mathbf{x}$, we can use the point in the target range scan corresponding to the projected pixel coordinate of $\mathbf{x}$. This corresponding point is not necessarily the closest point on the target range scan. However, it is a fast approximation that works well in practice [Blais and Levine, 1995; Neugebauer, 1997; Rusinkiewicz and Levoy, 2001].

Optimizing for Rigid Transformations for Two Point Sets. The second key step in the ICP algorithm is to optimize for the optimal rigid transformation to align the source positions $\mathbf{x}$ to the corresponding closest points $\mathbf{y}_k$. Fortunately, there exist closed-form solutions in the case of minimizing the squared Euclidean distance between corresponding points. The most well-known methods are the method of quaternions by Horn [1987] and the SVD method by Arun et al. [1987]. There are also other methods; however a comparison study by Eggert et al. [1997] shows that there is little practical difference between them.

[1] There are also many good implementations of kd-tree based nearest neighbor search. A popular implementation that is publicly available is the Approximate Nearest Neighbor (ANN) library [Mount and Arya, 2006].

Algorithm 3.2: LEAST-SQUARES REGISTRATION($\mathbf{x}, \mathbf{y}$)

```
  Data: A set of $N$ corresponding points $\mathbf{x}, \mathbf{y}$
  Result: The optimal rotation and translation minimizing $\sum_{\forall \mathbf{x},\mathbf{y}} \left\| R\mathbf{x} + \vec{\mathbf{t}} - \mathbf{y} \right\|^2$
1 begin
2     Compute centroid of the two point sets, $\mu_\mathbf{x} = \frac{1}{N}\sum \mathbf{x}, \mu_\mathbf{y} = \frac{1}{N}\sum \mathbf{y}$;
3     Compute the cross-covariance matrix $\Sigma$:
          $\Sigma = \frac{1}{N} \sum_{\forall \mathbf{x},\mathbf{y}} (\mathbf{x} - \mu_\mathbf{x})(\mathbf{y} - \mu_\mathbf{y})^\top = \frac{1}{N} \sum_{\forall \mathbf{x},\mathbf{y}} \mathbf{x}\mathbf{y}^\top - \mu_\mathbf{x}\mu_\mathbf{y}^\top$;
4     Form the matrix $Q$ from $\Sigma$:
          $Q(\Sigma) = \begin{bmatrix} \mathrm{tr}(\Sigma) & \Delta^\top \\ \Delta & \Sigma + \Sigma^\top - \mathrm{tr}(\Sigma) I \end{bmatrix}$,
      where $\Delta = [A_{23}, A_{13}, A_{12}]^\top, A_{ij} = \left(\Sigma - \Sigma^\top\right)_{ij}$;
5     Find the eigenvector $q = [q_0, q_1, q_2, q_3]^\top$ of $Q(\Sigma)$ corresponding to the
      maximum eigenvalue;
6     Convert the quaternion into a rotation matrix
          $R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}$;
7     Compute the optimal translation $\vec{\mathbf{t}} = \mu_\mathbf{y} - R(q)\mu_\mathbf{x}$;
8     return $R(q), \vec{\mathbf{t}}$;
9 end
```

We summarize two well-known methods of finding the optimal rigid transformation. The approach by Horn [1987] transforms the problem into finding the optimal quaternion representation of the rotation. This method is summarized in Algorithm 3.2. The SVD method [Arun et al., 1987] is even simpler: after computing the cross-covariance matrix Σ, we compute the SVD $\Sigma = U\Lambda V^\top$, and the matrix $R = VU^\top$ is our desired rotation. Here, it may be the case that $\det(R) = -1$, then we have a reflection. We correct this by checking if one of the singular values (e.g. λ_3) is zero. If this is the case, we flip the sign of the corresponding column (e.g. the third column) of V. Otherwise, the point correspondences form a degenerate case for the algorithm.

3.1.2 Improving the ICP algorithm

The basic algorithm discussed above can be improved in a number of different ways. First, the basic algorithm is applicable only when the source is a subset of the target shape; i.e. it does not handle the case where there is no corresponding point in the target shape. Second,

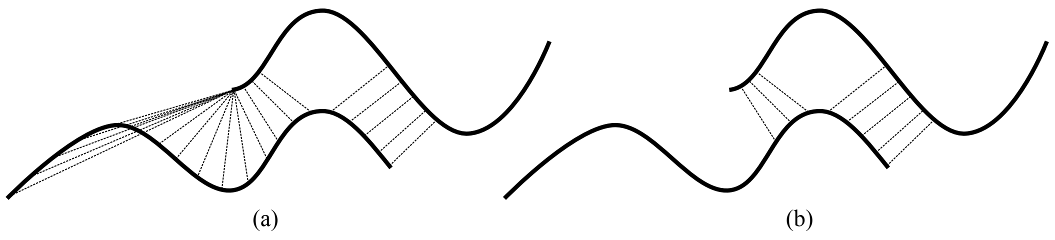


Figure 3.1: Rejecting closest points mapping to the boundary. Here we would like to move the bottom curve to match the top curve. (a) Many points on the source curve map to the boundary of the target curve, heavily biasing the distance between the curves. (b) By removing the correspondences that map to the boundary of the target, we minimize the distance between the shapes only where they overlap.

for shapes that overlap well but need to "slide" into the correct alignment, the algorithm may converge slowly to the correct result. We would like to improve the convergence speed in this case. Third, we would like to extend the algorithm to handle the alignment of multiple scans. In the following discussion, we describe improvements that have been made to the algorithm, extending its applicability to these other cases.

Handling Incomplete Data. To motivate the problem with incomplete data in the target shape, consider the 2D alignment example shown in Figure 3.1. Many points on the source curve (bottom) map to the boundary of the target curve (top). This biases the distance between the two shapes, and minimizing this distance moves the shape to the right. This is an example where the source shape is not a subset of the target, which is a condition where the ICP algorithm may fail to give a good result.

There are several strategies to handle this case [Rusinkiewicz and Levoy, 2001]. One of the most effective strategies is to remove the correspondences that map to the boundary of the target shape. This is illustrated in the right side of Figure 3.1. Notice that this allows us to measure the distance between the shapes only in the region where they overlap.

Another strategy to handle this case is based on the observation that the overlapping surface regions should be relatively close together, and the surface normals in the overlapping region should match closely. Therefore, we can remove correspondences where the distance between the points or the angle between their surface normals exceeds a threshold. This threshold is a user-specified parameter, and changing this parameter can result in different alignments of the surfaces.

A final heuristic we want to mention is to use mutually closest point correspondences. In this strategy, given a correspondence we check to see if the target point is closest (among all target points) to the source point, and also if the source point is closest (among all source points) to the target point. In practice this criteria may be too strict, and we can relax it by checking if the distance between the source point and the closest point (on the source shape) to the target point exceeds a small threshold. This strategy produces a similar result when applied to the example of Figure 3.1, but it is more costly to evaluate (since it involves two closest point queries), and its success could depend on an appropriate choice of the threshold.

Improving Convergence via Point-To-Plane Metric. Chen and Medioni [1991] describe an alternative formulation of the ICP algorithm. The difference in their approach is the choice of distance function between the two shapes. Instead of minimizing the Euclidean distance between the corresponding points

$$d_{\mathrm{pt}}\left(\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right),\mathbf{y}_k\right)=\sum_{\mathbf{x}\in\mathcal{P},\mathbf{y}_k\in\mathcal{U}_k}\left\|\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right)-\mathbf{y}_k\right\|^2, \tag{3.4}$$

their formulation measures the distance between the source position $R_k\mathbf{x}+\vec{\mathbf{t}}_k$ to the plane passing through $\mathbf{y}_k$ with normal equal to the surface normal $\vec{\mathbf{n}}_k$ at $\mathbf{y}_k$:

$$d_{\mathrm{pl}}\left(\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right),\mathbf{y}_k,\vec{\mathbf{n}}_k\right)=\sum_{\mathbf{x}\in\mathcal{P},\mathbf{y}_k\in\mathcal{U}_k}\left[\left(\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right)-\mathbf{y}_k\right)\cdot\vec{\mathbf{n}}_k\right]^2. \tag{3.5}$$

In contrast to the point-to-point metric, the point-to-plane metric gives a small residual as long as the source point $\mathbf{x}$ is close to the plane at $\mathbf{y}_k$. The difference between the two approaches is illustrated in Figure 3.2. While the point-to-point metric produces "sticky" constraints, the point-to-plane metric allows the surface to "slide" on the planes passing through the target points $\mathbf{y}_k$. This provides faster convergence in many cases [Rusinkiewicz and Levoy, 2001]. Perhaps one drawback of this method is that, unlike the point-to-point metric, we cannot solve the optimal rotation and translation in closed form. Instead, we use the Gauss-Newton algorithm to optimize this metric, which we describe shortly in Section 3.1.3.

An important observation here is that the point-to-point and point-to-plane dis-

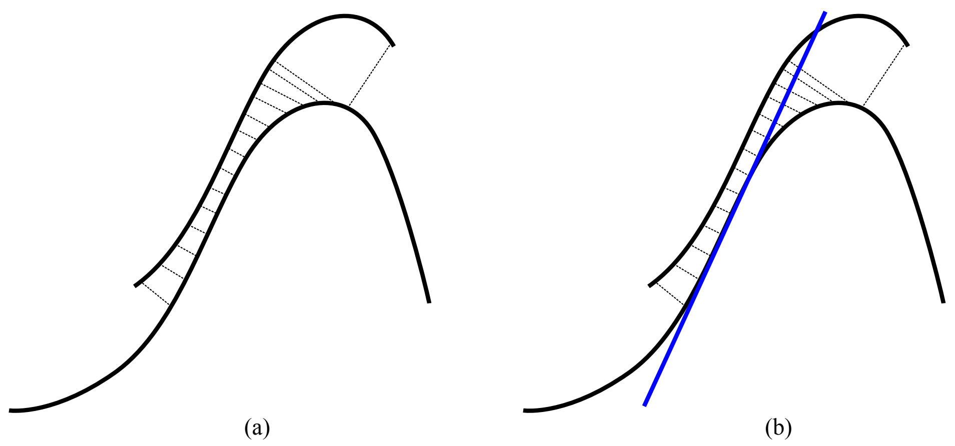


Figure 3.2: Point-to-point vs. point-to-plane ICP. Here we would like to move the top curve to match the bottom curve. (a) The point-to-point metric produces "sticky" constraints, specifying that the source positions match the exact location of the closest target points. (b) In contrast, the point-to-plane metric gives "slippery" constraints by minimizing the distance between the source points and the plane at each target point. This can allow the surface to slide easily in the flat planar region, where the surface normal is mostly perpendicular to the blue line.

tances using the closest point are approximations of the squared distance function between the source and target shapes. This was noticed by Mitra et al. [2004], performing a detailed convergence analysis of their behavior with experimental results. They demonstrate that using *quadratic* approximations of the squared distance function of the surface result in more stable convergence properties than either metric alone.

Motivated by this analysis, Huang et al. [2008] propose a simpler method by combining the point-to-point and point-to-plane metrics using a weighted average to get a hybrid distance metric:

$$d_{\text{hybrid}}\left(\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right),\mathbf{y}_k,\vec{\mathbf{n}}_k\right)=\sum_{\mathbf{x}\in\mathcal{P},\mathbf{y}_k\in\mathcal{U}_k}\eta_{\text{pt}}\,\left\|\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right)-\mathbf{y}_k\right\|^2+\eta_{\text{pl}}\,\left[\left(\left(R_k\mathbf{x}+\vec{\mathbf{t}}_k\right)-\mathbf{y}_k\right)\cdot\vec{\mathbf{n}}_k\right]^2. \tag{3.6}$$

Huang et al. [2008] use the weights $\eta_{\text{pt}} = 0.6$ and $\eta_{\text{pl}} = 0.4$, but the user could adjust these weights to give a different convergence behavior.

Simultaneous ICP for Multiple Scans. So far, we have focused on strategies to align a pair of shapes. However, when we want to build complete 3D models, we often need to align multiple

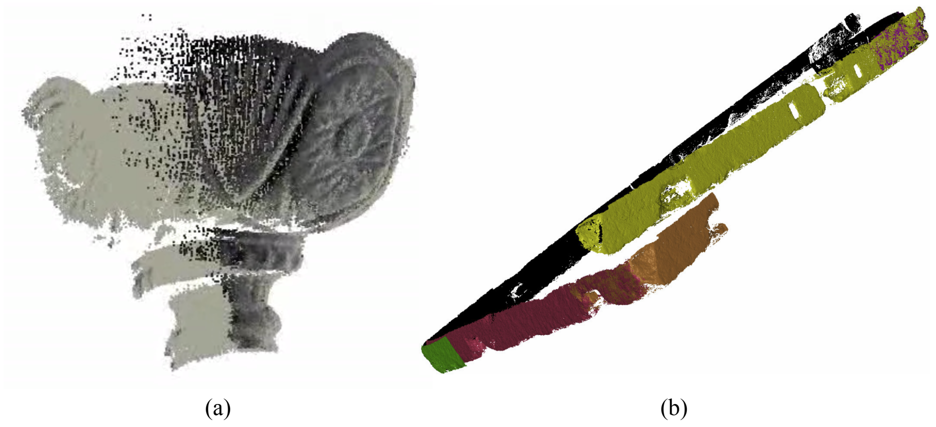


Figure 3.3: Examples of accumulation error using a sequential alignment strategy. In both examples, the range scans wrap around the object and should form a closed ring. (a) We see that the alignment error accumulates and the vase "twists" upwards. (b) Here the first and the last scans do not align well as a result of accumulated error. From Rusinkiewicz et al. [2002]; Brown and Rusinkiewicz [2007].

range scans taken from multiple viewpoints. For this case, a straightforward way to extend the pairwise ICP registration is to sequentially align each scan to the previous scan, or to all previous scans [Chen and Medioni, 1991]. However, small errors in each alignment can propagate to the alignment of future scans, eventually creating a large accumulated error [Bernardini and Rushmeier, 2002; Rusinkiewicz et al., 2002]. This behavior occurs especially for a large number of scans which wrap around the surface of an object [Brown and Rusinkiewicz, 2007]. Some examples of accumulated alignment error are shown in Figure 3.3.

An alternative to sequential registration is to solve for the optimal alignment (from each frame to all other frames) simultaneously over all frames. Neugebauer [1997] describes such a simultaneous registration for multiple scans, using the point to plane metric and the projection method for establishing correspondences. In contrast to a sequential approach, this type of simultaneous registration method is good at distributing the alignment error over all scans, rather than creating a gradual accumulation of error.

The basic idea of simultaneous registration is to solve for an aligning transformation T_i from each range scan $\mathcal{S}_i$ to a hypothetical world coordinate system. Here, T_i denotes both the rotational and translational components; i.e. $T_i = (R_i, \vec{\mathbf{t}}_i)$, and the subscript represents the transformation for frame i rather than the iteration number in the ICP algorithm. This situation is illustrated in Figure 3.4. Just like the pairwise case, the alignment is measured

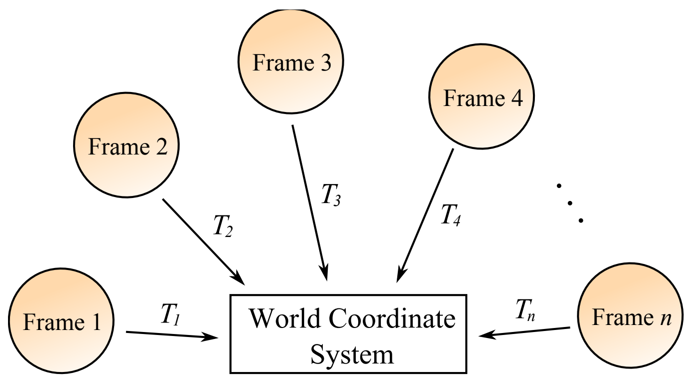


Figure 3.4: Organization of the transformations for simultaneous registration. We solve for the transformations that align all frames to a common world coordinate system.

quantitatively by establishing correspondences between each pair of scans $\mathcal{S}_i, \mathcal{S}_j$ (in their pairwise region of overlap). These correspondences are generated by projecting each position $\mathbf{x}_k^{(i)} \in \mathcal{S}_i$ into all other scans $\mathcal{S}_j$ (see Section 3.1.1), obtaining a corresponding point $\mathbf{y}_{ik}^{(j)} \in \mathcal{S}_j$ with surface normal $\vec{\mathbf{n}}_{ik}^{(j)}$. Once we have these correspondences, a good alignment should transform each correspondence to the same location in the world coordinate system. Thus, we can express the alignment distance by measuring how close the *transformed* positions are in the world coordinate system. Using the point-to-plane metric in Equation (3.5) we get

$$\operatorname*{argmin}_{T_1,T_2,\ldots,T_n} \sum_{i=1}^{n} \sum_{\mathbf{x}_k^{(i)} \in \mathcal{S}_i} \sum_{\mathbf{y}_{ik}^{(j)} \in \mathcal{S}_j} d_{\text{pl}}\left(T_i\left(\mathbf{x}_k^{(i)}\right), T_j\left(\mathbf{y}_{ik}^{(j)}\right), T_j\left(\vec{\mathbf{n}}_{ik}^{(j)}\right)\right). \tag{3.7}$$

Here, $T_i\left(\mathbf{x}_k^{(i)}\right)$ is the location $\mathbf{x}_k^{(i)}$ transformed to the world coordinate system, $T_j\left(\mathbf{y}_{ik}^{(j)}\right)$ is the location $\mathbf{y}_{ik}^{(j)}$ transformed to the world coordinate system, and the same for $T_j\left(\vec{\mathbf{n}}_{ik}^{(j)}\right)$. This is an optimization problem, and we can solve for the optimal T_i using the Gauss-Newton algorithm (described in the next two sections). This is similar to the pairwise case, except that we have a larger optimization solving for many transformations at once.

Note that simultaneous registration is still inherently a local search method that requires a good initial alignment of the scans. Neugebauer [1997] obtains this initialization via an interactive user interface, where the user needs to click at least three corresponding points from each scan to all previously integrated scans. Perhaps another drawback of the method is that it requires that all range scans are in memory, since we need to project the points to each scan to find correspondences during the registration. Depending on the size of the dataset, this requirement may be prohibitively expensive. Alternative strategies for automating the initial registration and for lowering the memory footprint are discussed in Section 2.1.

3.1.3 Gauss-Newton Algorithm

In this thesis, we often want to optimize for the best rigid transformation $g = (R, \vec{\mathbf{t}})$ that minimizes a residual $r(g)$ in the least-squares sense (e.g. Chapters 5 and 6). The main difficulty in such a minimization is that R is strictly constrained to be a rotation in $SO(3)$, i.e. $R^\top R = I$. An alternative to constrained optimization is to explicitly parameterize the rotation matrix, and such parameterizations result in a non-linear least-squares objective. We take this latter approach and use the Gauss-Newton algorithm to solve the non-linear objective.

The Gauss-Newton algorithm is a generic algorithm for optimizing non-linear least-squares problems. It is based on the observation that it is often much easier to solve a linearized version of the non-linear problem, rather than trying to solve the non-linear problem directly. At each step of the algorithm, the non-linear residual function is approximated by a linear residual function, and the optimal parameters minimizing the linear residual are computed. By repeating this linearization and optimization, the hope is to arrive (or converge) to a locally optimal set of parameters. The following details are based on Heath [2002].

Suppose we want to find the optimal parameters x which minimize a non-linear least-squares residual function $r(x)$ in the least squares sense; for example

$$\operatorname*{argmin}_{x\in\mathbb{R}^m} \; r(x) = f(x) - y, \qquad r : \mathbb{R}^m \to \mathbb{R}^n.$$

The first-order Taylor expansion about a point $a \in \mathbb{R}^m$ is

$$r(x) \approx r(a) + J_r(a)(x - a),$$

where J_r is the Jacobian matrix of r with respect to variables x. The Gauss-Newton algorithm says that, given an initial guess $x = x_0$ for minimizing $r(x)$, we can iteratively update x to a location that has a smaller residual, based on a linearization of the residual (as above). At each step k of the iteration with current location x_k, we use the Taylor expansion to approximate the residual function at this location:

$$r(x) \approx r(x_k) + J_r(x_k)(x - x_k). \tag{3.8}$$

The last term here $(x - x_k) = s_k$ is considered as the "step" that we should take to improve our minimization of $r(x)$. Thus we find the s_k that minimizes the above expression in the least-squares sense:

$$r(x_k) + J_r(x_k)s_k \approx 0 \tag{3.9}$$

$$J_r(x_k)s_k \approx -r(x_k). \tag{3.10}$$

This minimization can be solved using a variety of methods, such as the method of normal equations, using an orthogonal factorization of the left-hand side, or using the SVD. Once we have obtained a solution for s_k, we then update the current location using the formula

$$x_{k+1} = s_k + x_k. \tag{3.11}$$

To detect if the optimization has converged, we can monitor the change of the objective function $F_k = r(x_k)$, its gradient, and the magnitude of the step s_k [Gill et al., 1989]:

$$\begin{aligned} |F_k - F_{k+1}| &< \epsilon\,(1 + F_k) \\ \|F_k\|_\infty &< \sqrt[3]{\epsilon}\,(1 + F_k) \\ \|s_k\|_\infty &< \sqrt{\epsilon}\,(1 + \|s_k\|_\infty)\,, \end{aligned} \tag{3.12}$$

for some small ϵ. The Gauss-Newton algorithm has excellent quadratic convergence when the initial parameters x_0 are started close to the optimal solution. However, the algorithm may fail to converge unless it is started close enough to the solution. A line search improves the robustness of the algorithm, but it is not necessarily the case that the step s_k taken here is a descent direction. Also, in cases with a large residual, the linear approximation used here may be inaccurate.

The problem with these cases is that the Gauss-Newton step direction may be a direction where the overall residual is *increasing*, rather than decreasing. For a better guarantee of convergence, one can also employ an extension to the Gauss-Newton method known as the Levenberg-Marquardt method. This method has an additional parameter μ that blends between a Gauss-Newton step and a gradient step, the latter of which is guaranteed to be a descent direction. Even better, one can maintain an explicit "trust region", which indicates how far you can travel from the current solution and still produce a step that reduces the overall error. Powell's Dog Leg Method is an example of such an approach [Madsen et al., 2004]. It selects between a Gauss-Newton step and a gradient step within the trust region to always reduce the residual error, and it automatically adjusts the size of the trust region from feedback of how much the residual has actually decreased in the previous iteration. In our work, we found that the Gauss-Newton method was sufficient for our experiments, but these advanced methods could be used to produce a faster convergence to the solution.

3.1.4 Optimizing for Rigid Transformations using Gauss-Newton

In this section, we adapt the Gauss-Newton algorithm to specifically address the problem of optimizing a residual function of a rigid transformation. The main component in the Gauss-Newton algorithm is the linearization of the residual function about a current estimate of the x_k solution. Once we have this linearization, then we can solve the system for a step and iterate to minimize the residual. For rigid transformations, our mathematical tool to derive this linearization is the *axis-angle* formulation of a rotation, or *twist coordinate* formulation for a rigid transformation $g = (R, \vec{\mathbf{t}})$. We first demonstrate that these formulations locally parameterize the space of rotations and the space of rigid transformations about the identity. We then use these tools to derive a linearization of a residual function $r(g)$ [2].

We begin with the hat $\hat{\cdot}$ operator, which converts a cross product in $\mathbb{R}^3$ to a matrix

[2]A similar discussion is in Taylor and Kriegman [1994] and Krishnan et al. [2005].

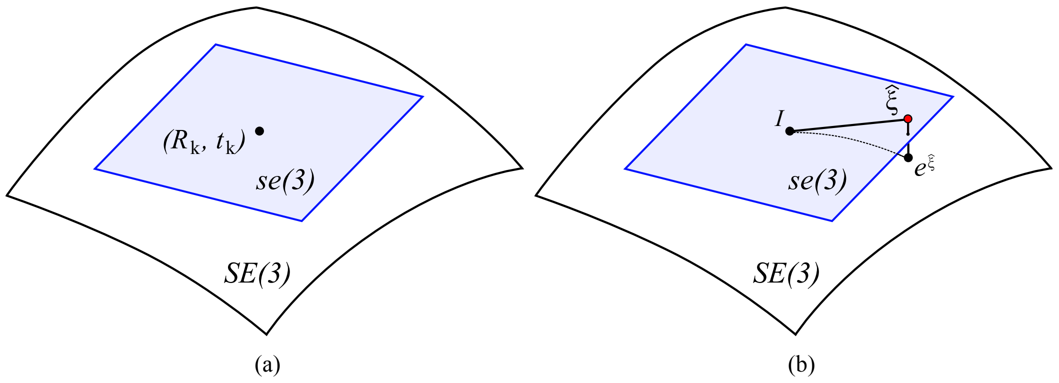


Figure 3.5: Optimizing in $SE(3)$, the space of rigid transformations. (a) At each iteration, we linearize the space of rigid transformations about the current estimate of the solution. This is depicted as a tangent plane $se(3)$ on the manifold of rigid transformations, $SE(3)$. (b) Once we minimize the residual in the tangent plane, we project the solution $\hat{\xi}$ back to the manifold via the exponential map $e^{\hat{\xi}}$. This allows us to take a step in the optimization algorithm.

that can be multiplied with a vector. Given a vector $\omega \in \mathbb{R}^3$, the operation of taking the cross product $\omega \times \mathbf{x}$ between the vector ω and some $\mathbf{x} \in \mathbb{R}^3$ can be described as the product $\widehat{\omega}\mathbf{x}$ where

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{3.13}$$

Here, $\widehat{\omega}$ is a skew-symmetric matrix, which means that taking the transpose is the same as multiplying by -1, or $\widehat{\omega}^\top = -\widehat{\omega}$. It is also the case that any skew symmetric matrix can be expressed as $\widehat{\omega}$ for some $\omega \in \mathbb{R}^3$. Interestingly, the space of skew-symmetric matrices forms a Lie algebra $so(3)$ [Ma et al., 2003]. In addition to this property, some manipulation reveals that

$$\widehat{\omega}^2 = \omega\omega^\top - \|\omega\|^2 I, \qquad \widehat{\omega}^3 = -\|\omega\|^2\widehat{\omega}. \tag{3.14}$$

It turns out that the space of rotations $R \in SO(3)$ around the identity I can be described using the set of skew-symmetric matrices [Ma et al., 2003] (The following discussion is taken from Ma et al. [2003]). This can be shown by taking a derivative of a trajectory $R(t) : \mathbb{R} \to SO(3)$ that describes a continuous rotational motion. The rotation must satisfy the constraint $R(t)^\top R(t) =$

I, or equivalently

$$R(t)R(t)^\top = I. \tag{3.15}$$

Taking the derivative of this with respect to t, we obtain that

$$\begin{aligned}\dot{R}(t)R(t)^\top + R(t)\dot{R}(t)^\top &= 0\\ \dot{R}(t)R(t)^\top &= -R(t)\dot{R}(t)^\top\\ \dot{R}(t)R(t)^\top &= -\left(\dot{R}(t)R(t)^\top\right)^\top.\end{aligned} \tag{3.16}$$

This last equation shows that the matrix $\dot{R}(t)R(t)^\top$ is a skew-symmetric matrix, and therefore it must be the case that

$$\begin{aligned}\dot{R}(t)R(t)^\top &= \widehat{\omega}(t),\\ \dot{R}(t) &= \widehat{\omega}(t)R(t) \qquad \text{for some } \widehat{\omega}(t).\end{aligned} \tag{3.17}$$

Here, we see that if $t_0 = I$, then $\dot{R}(t_0) = \widehat{\omega}(t_0)$. Thus, around the identity element, we can approximate the rotations to the first order using the expression

$$\dot{R}(t_0 + dt) \approx I + \widehat{\omega}(t_0)dt. \tag{3.18}$$

This means that the rotations locally around I depend only on the parameter $\omega \in \mathbb{R}^3$. In fact, the Lie algebra $so(3)$ forms a *tangent space* of $SO(3)$ at the identity element I. Moreover, we can apply this formula to a residual function of a rotation to obtain a linearization of the function about the identity.

This is almost what we need, except for three issues. First, this linearization is only about the identity I; we would like to linearize around an arbitrary rotation. Second, once we solve for the optimal $\omega \in so(3)$ that locally minimizes the residual, we need a way to project this solution (which is in $so(3)$) back to the original group $SO(3)$. This is necessary because we must apply this step to the current rotation to complete an iteration in the Gauss-Newton algorithm. Finally, we would like to handle the case of a rigid transformation $g = (R, t)$ where a translation t is also involved.

For the first issue, suppose that we would like to linearize the local space of rotations about a current estimate R_k. We can describe this space as the set of rotations produced by the expression $R^\delta R_k$, where R^δ is a small rotation. We can verify this by observing that when $R^\delta = I$ the result is R_k, which is exactly the current location. Also, a non-identity R^δ produces a rotation on top of the current location R_k; thus $R^\delta R_k$ describes rotations relative to R_k. Taking the tangent space of R^δ around the identity yields $R^\delta \approx I + \widehat{\omega}$ for $\omega \in \mathbb{R}^3$ (from Equation (3.18)), and applying this to our expression, we obtain $(I + \widehat{\omega})R_k = R_k + \widehat{\omega}R_k$ as the local description of the rotations about R_k.

Second, the translation between $R \in SO(3)$ and $\omega \in so(3)$ is done via the exponential and logarithm maps. A simple expression of the exponential map $\exp : so(3) \rightarrow SO(3),\ \ R = e^{\widehat{\omega}}$ is given by *Rodrigues' formula*:

$$e^{\widehat{\omega}} = I + \widehat{\omega} + \frac{1}{2!}\widehat{\omega}^2 + \frac{1}{3!}\widehat{\omega}^3 + \cdots + \frac{1}{n!}\widehat{\omega}^n + \cdots \tag{3.19}$$

$$= I + \frac{\widehat{\omega}}{\|\omega\|}\sin(\|\omega\|) + \frac{\widehat{\omega}^2}{\|\omega\|^2}(1 - \cos(\|\omega\|)), \tag{3.20}$$

which can be derived using the properties of the skew-symmetric matrix. Once we have solved for an ω^* that minimizes the residual locally around a current estimate R_k, we can map this to a rotation matrix using the exponential map and apply it to the current estimate. Using the exponential map for the linearized rotation $(I + \widehat{\omega^*})$ we obtain

$$\begin{aligned} e^{(I+\widehat{\omega^*})} &= e^I e^{\widehat{\omega^*}} \qquad (\text{since } I\widehat{\omega^*} = \widehat{\omega^*} I) \\ &= Ie^{\widehat{\omega^*}} = e^{\widehat{\omega^*}}. \end{aligned}$$

Applying this to the current rotation R_k results in the expression $e^{\widehat{\omega^*}}R_k$ to update the current estimate.

Finally, we address the issue concerning the rigid transformations $g = (R, \vec{\mathbf{t}})$. These transformations, like the space of rotations, also form a special group denoted $SE(3)$. In addition, they can be parameterized locally around the identity $(I, 0)$ using a *twist*, which is of the form

$$\widehat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix} \in se(3).$$

As we see above, the twist is described by the parameters $\omega, v \in \mathbb{R}^3$, and has a total of 6 degrees of freedom. ξ itself is called the *twist coordinate*, and consists of the parameters $\xi = [v, \omega]$ to describe the twist matrix. Using a calculation similar to what we had done with the rotations [Ma et al., 2003], we can show that for a continuous rigid motion $g(t) : \mathbb{R} \to SE(3)$,

$$\begin{aligned} \dot{g}(t)g(t)^{-1} &= \widehat{\xi}(t) \\ \dot{g}(t) &= \widehat{\xi}(t)g(t) \qquad \text{for some } \widehat{\xi}(t). \end{aligned} \tag{3.21}$$

If $g(t_0) = I$ we obtain the first order approximation $g(t_0 + dt) \approx I + \widehat{\xi}(t_0)dt$. Thus the rigid transformations about a *particular* g_k is described by $\left(I + \widehat{\xi}\right) g_k$, or equivalently $g_k + \widehat{\xi} g_k = g_k + \widehat{\omega} g_k + v$.

Similar to the rotations, the rigid transformations also have the exponential and logarithm maps to translate between $SE(3)$ and $se(3)$. In particular, we are interested in the exponential map for twists, which is the expression

$$e^{\widehat{\xi}} = \begin{cases} \begin{bmatrix} e^{\widehat{\omega}} & \frac{(I - e^{\widehat{\omega}})\widehat{\omega} v + \omega\omega^\top v}{\|\omega\|^2} \\ 0 & 1 \end{bmatrix} & \text{if } \omega \neq 0 \\ \begin{bmatrix} I & v \\ 0 & 1 \end{bmatrix} & \text{if } \omega = 0. \end{cases} \tag{3.22}$$

This gives us the final component needed for a Newton iteration: once we have solved for a ξ^* that minimizes the residual locally around a current estimate g_k, we use the exponential map to update the current estimate using $e^{\widehat{\xi^*}} g_k$.

Example Derivation of Gauss-Newton Using a Linearized Rigid Transformation

We give an example of linearizing the residual $r(g) = \mathbf{y} - g\mathbf{x}$ in the context of the Gauss-Newton algorithm to illustrate the tools introduced in this section. If we want to linearize this residual about a location g_k, we substitute the linearized expression for the rigid transformations around g_k yielding

$$r(g) \approx \mathbf{y} - (I + \hat{\xi}) g_k \mathbf{x} \tag{3.23}$$

$$= \mathbf{y} - g_k \mathbf{x} - \hat{\xi} g_k \mathbf{x}. \tag{3.24}$$

Note that we need to use the homogeneous coordinates for $\mathbf{x}$ and $\mathbf{y}$ for the dimensions of the matrices to match. Substituting the twist matrix for $\hat{\xi}$,

$$r(g) \approx \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} - \begin{bmatrix} R_k & \vec{\mathbf{t}}_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} - \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R_k & \vec{\mathbf{t}}_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{3.25}$$

$$\approx \begin{bmatrix} \mathbf{y} - (R_k \mathbf{x} + \vec{\mathbf{t}}_k) \\ 0 \end{bmatrix} - \begin{bmatrix} (\omega \times (R_k \mathbf{x} + \vec{\mathbf{t}}_k)) + v \\ 0 \end{bmatrix}. \tag{3.26}$$

Now, for any $\mathbf{x} \in \mathbb{R}^3$, $\hat{\omega}\mathbf{x} = \omega \times \mathbf{x} = -\mathbf{x} \times \omega = -\hat{\mathbf{x}}\omega$. Using this identity and removing the trivial bottom row, we can rearrange this above in terms of the parameters ω, v as

$$r(g) \approx \mathbf{y} - (R_k \mathbf{x} + \vec{\mathbf{t}}_k) - \begin{bmatrix} -\widehat{(R_k \mathbf{x} + \vec{\mathbf{t}}_k)} & I \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix}. \tag{3.27}$$

We can then minimize this formula by finding the ω, v that takes $r(g)$ as close as possible to $\mathbf{0}$ in the least-squares sense:

$$\begin{bmatrix} -\widehat{(R_k \mathbf{x} + \vec{\mathbf{t}}_k)} & I \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix} \approx \mathbf{y} - (R_k \mathbf{x} + \vec{\mathbf{t}}_k). \tag{3.28}$$

This is a linear least-squares problem that can be solved efficiently using the method of normal equations, via a Cholesky factorization and back-substitution steps [Heath, 2002]. The resulting solution ω^*, v^* is a step that minimizes this linearized residual. We then use the exponential map to apply the step to the current solution

$$g_{k+1} = e^{\widehat{\xi^*}} g_k, \tag{3.29}$$

where $\xi^* = [v^*, \omega^*]$. This illustrates a complete iteration in the Gauss-Newton algorithm, where the residual is a function of a rigid transformation.

3.2 Shape Descriptors for Local Surface Matching

A challenge of using range scans is to establish correspondences between multiple range scans, because the scans themselves do not provide any correspondence information. Given two range scans and a point in each scan, the objective is to determine if the points belong to the same location on the surface. If this is true, then we say that the two points "match" or "correspond" to each other. Shape descriptors are a useful tool for solving this problem. These shape descriptors compactly encode the shape of small, localized patches on the surface. By computing and matching the shape descriptors between the two points, we can determine if the shape of the surface around the two points are similar or different. This is useful because, at the very least, we can quickly determine incorrect matches when the descriptors are very different.

In this section we will discuss two classic examples of shape descriptors: principal curvatures and spin images. These two components are used to guess correspondences in the motion sampling step of Chapter 4 and the initialization step of Chapter 5. The original work on motion sampling used principal curvatures, and this is the reason why we use them in our work. Since spin images are simple to implement and are more discriminative than principal curvatures, we extend the original motion sampling work and use the spin images in conjunction with the principal curvatures to improve the quality of the sampling. Using other shape descriptors could make a difference, but spin images were robust enough in our experiments, and our focus was not to investigate which descriptor performs the best.

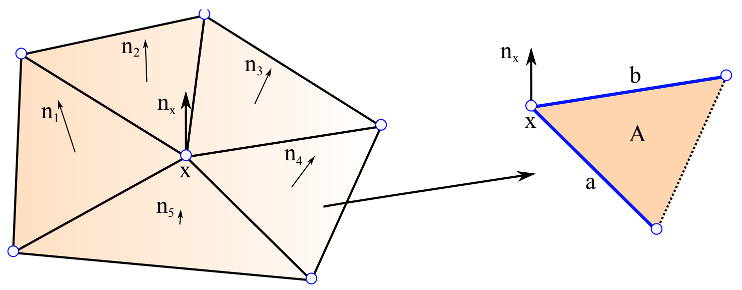


Figure 3.6: Estimating per-vertex normals on a triangle mesh. To estimate the normal at x, we take a weighted average of the normals at the incident triangles of x, where the weight of each triangle is $A/(ab)^2$.

3.2.1 Estimating Normals and Curvatures

Some of the most basic descriptions of curves and surfaces in space come from the study of differential geometry. The curvature of the surface, which measure how much the surface bends at a point, is perhaps one of the most basic descriptions of a surface. Although they may be less descriptive than spin images or other shape descriptors, they are still useful for classifying regions of the surface based on how much the surface is flat or curved. In addition, the principal curvatures are useful for defining a canonical frame for points on the surface. In this section, we review these concepts and discuss a few approaches for estimating them in the case of a triangle mesh.

Computing Normals. The surface normal of a surface S at a point x is the normal vector of the plane tangent to S at x. It provides a first-order linear approximation of the surface at x, and also it provides an orientation for the surface as well. In the case of a triangle mesh, the surface normals are well-defined within each triangle (they are just the normal vectors of the plane of the triangle), but are not naturally defined on the edges and vertices of the mesh. If we think of the surface as a piecewise linear approximation of an underlying smoother surface, then we can ask if it is possible to estimate the surface normal of the underlying surface at each vertex of the triangle mesh. This is the definition of the per-vertex normal, which defines a normal vector at each vertex that is different from the normals of the incident triangles.

Max [1999] shows that it is indeed possible to estimate an accurate per-vertex normal. This approach generates per-vertex normals by taking a weighted average of the incident

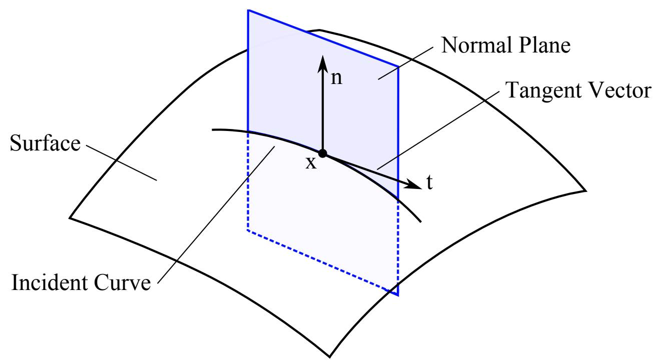


Figure 3.7: Illustrating the concept of the normal curvature of a surface. On a point x with normal n on the surface, the plane along the normal intersects the surface at the incident curve. The curvature of this curve is the *normal curvature* at x in direction t. Note that both n and t lie on the normal plane.

triangle normals (Figure 3.6). Using the illustration on the right of Figure 3.6, the weight of this incident triangle is the triangle area A divided by $(ab)^2$, the squared product of the lengths of the two edges of the triangle incident to x. This technique is fast to compute, and even provides an exact normal vector in the case where the vertices of the triangle mesh lie on a sphere.

Computing Curvatures. The surface curvature measures how much the surface bends in different directions about a point. The basic idea, illustrated in Figure 3.7, is to take the plane through the surface normal at $\mathbf{x}$ (which we call the *normal plane*) and examine the incident curve on the surface. The curvature of this curve, which is the inverse of the radius of the circle that best approximates the curve at $\mathbf{x}$, is called the *normal curvature* of $\mathbf{x}$ in direction t. Rotating the normal plane about the normal traces out different curves on the surface, resulting in different curvatures. Then, the minimum and maximum curvatures (with corresponding directions) are called the *principal curvatures* (and principal directions). The principal directions are always orthogonal to each other and to the normal, so the three vectors define a unique orthonormal frame at $\mathbf{x}$.

A concise way of representing the varying curvatures at $\mathbf{x}$ is via the *shape operator* (also known as the Weingarten Matrix or Second Fundamental Tensor). It is a 2×2 symmetric matrix that expresses the linear relationship between a tangent direction on the surface and the change of the normal in that direction. Given a basis $\vec{\mathbf{u}}, \vec{\mathbf{v}}$ of the tangent plane at $\mathbf{x}$, the shape operator S is defined as

$$S = \begin{bmatrix} D_{\vec{\mathbf{u}}}\vec{\mathbf{n}} \cdot \vec{\mathbf{u}} & D_{\vec{\mathbf{v}}}\vec{\mathbf{n}} \cdot \vec{\mathbf{u}} \\ D_{\vec{\mathbf{u}}}\vec{\mathbf{n}} \cdot \vec{\mathbf{v}} & D_{\vec{\mathbf{v}}}\vec{\mathbf{n}} \cdot \vec{\mathbf{v}} \end{bmatrix}, \tag{3.30}$$

where $D_{\vec{\mathbf{u}}}\vec{\mathbf{n}}$ and $D_{\vec{\mathbf{v}}}\vec{\mathbf{n}}$ are directional derivatives of the surface normal $\vec{\mathbf{n}}$ along the direction of $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ on the tangent plane. The shape operator is a concise representation of all the normal curvature information at $\mathbf{x}$. For example, the principal curvatures are the eigenvalues of this matrix, and the corresponding eigenvectors are the principal directions on the tangent plane.

In the literature, there are many methods developed to estimate the curvature on triangle meshes, for example by fitting a quadratic surface or by estimating normal curvatures. Perhaps one of the simplest and most effective approaches is to estimate the shape operator S directly [Rusinkiewicz, 2004; Kalogerakis et al., 2007]. The key idea is that the per-vertex normals provide direct constraints on the shape operator, so that we can estimate S that best fits these constraints.

Given a direction vector $\vec{\mathbf{d}}$ on the tangent plane, multiplying S with $\vec{\mathbf{d}}$ gives the derivative of the normal along direction $\vec{\mathbf{d}}$. Applying this idea for the case of a triangle mesh, given two points $\mathbf{p}, \mathbf{q}$ in the vicinity of $\mathbf{x}$ with normals $\vec{\mathbf{n}}_p, \vec{\mathbf{n}}_q$, and a basis $\vec{\mathbf{u}}, \vec{\mathbf{v}}$ for the tangent plane at $\mathbf{x}$, the shape operator S is constrained by

$$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} (\mathbf{p}-\mathbf{q}) \cdot \vec{\mathbf{u}} \\ (\mathbf{p}-\mathbf{q}) \cdot \vec{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \left(\vec{\mathbf{n}}_p - \vec{\mathbf{n}}_q\right) \cdot \vec{\mathbf{u}} \\ \left(\vec{\mathbf{n}}_p - \vec{\mathbf{n}}_q\right) \cdot \vec{\mathbf{v}} \end{bmatrix}. \tag{3.31}$$

Gathering these constraints for multiple pairs $\mathbf{p}, \mathbf{q}$, we can estimate the entries of S that best fit the constraints in the least-squares sense.

Rusinkiewicz [2004] describes a technique to estimate per-vertex curvature by taking a weighted average of incident per-triangle curvatures, similar in spirit to the technique of Max [1999]. Given the per-vertex normals, we first estimate S at each triangle (with its own basis u, v) using the edges of the triangle (and the per-vertex normals at the vertices) as constraints. Note that this does not make sense if we use the triangle normal as constraints; then the curvature would be zero. Next, at each vertex **x**, we transform S of each incident triangle to express it in terms of the basis $\vec{\mathbf{u}}, \vec{\mathbf{v}}$ defined at **x**, and take a weighted average, where the weight equals the area of the part of the triangle that is closest to **x**. This an efficient algorithm and gives accurate results, even exact results in the case of a sphere with exact vertex normals.

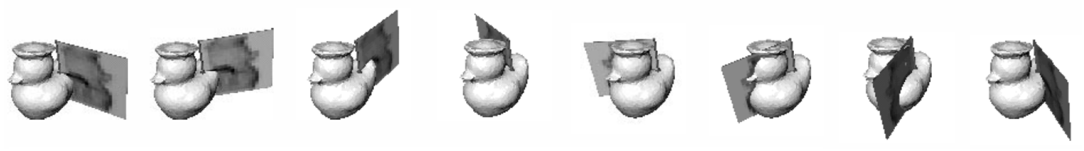
In our work, we use a more robust alternative developed by Kalogerakis et al. [2007], because it does not require a triangle mesh, handles noisy data, and supports curvature estimation at multiple scales. Unlike the method discussed above, this method directly estimates S at each point **x** using constraints between every pair of points in a neighborhood around **x**. Because there may be outliers among the constraints, the algorithm uses M-estimation to fit S instead of using least-squares. This tends to give smoother and higher quality results especially with noisy data, but the cost of the method is a little more expensive. Note also that taking larger neighborhoods give smoother estimates of S, so we can also compute the curvatures on multiple scales.

3.2.2 Spin Images

Spin images, introduced by Johnson [1997], is a 3D surface descriptor that converts local patches of geometry into digital images that can be compared directly. These images are a histogram of the points of the surface, generated by spinning a plane about the normal at a point. We first describe how to construct the spin images and then how to match spin images between two shapes.

Spin Image Construction. The input to the spin image generation algorithm is a triangle mesh, where the resolution of the mesh (defined as the median of all the edge lengths) is known. At each point **p** in the mesh, we must estimate the surface normal $\vec{\mathbf{n}}$, because the spin image generation depends on this normal vector.

Now, imagine a plane (with finite extent) that is attached to the normal. As we rotate the plane about the normal, other points on the surface “hit” the plane at specific locations on the plane. This is illustrated in the top of Figure 3.8, and the resulting projection of the points are shown in the bottom. We keep track of these hit locations concisely by partitioning the



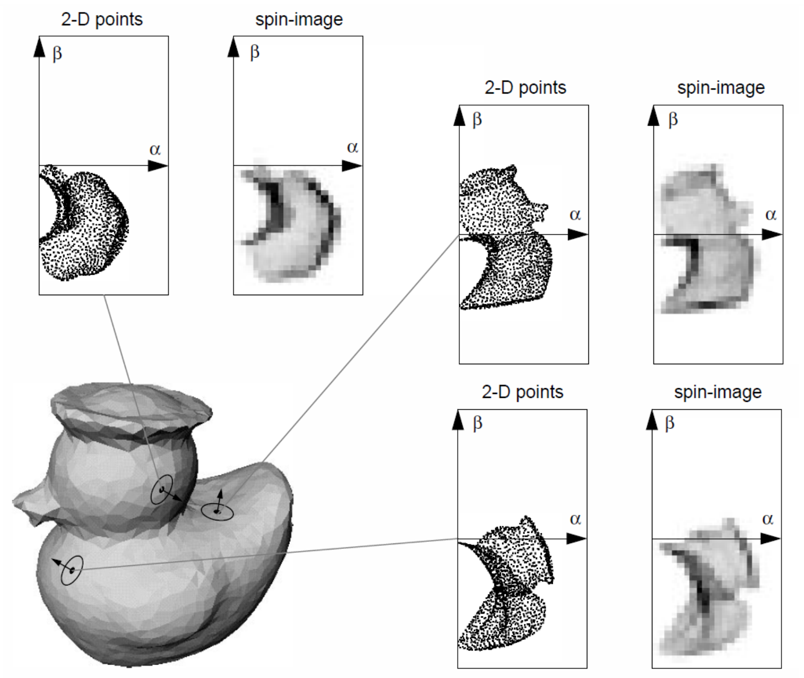


Figure 3.8: Generating spin images. (Top) Spin images are histograms of points on the surface created by rotating a plane about the surface normal at locations on the surface. (Bottom) Examples of spin images created at different points of the 3D model. From Johnson [1997].

plane in to a regular grid, and counting the number of hits per grid square. If we visualize these bins in 3D, then we see that the bins of the spin image histogram are concentric rings stacked along the normal direction (Figure 3.9). We can express the location of the hit point of **x** on the plane (passing through **p** and $\vec{\mathbf{n}}$) concisely using the formula:

$$S(\mathbf{x},\vec{\mathbf{n}},\mathbf{p}) = \left[\sqrt{\|\mathbf{x}-\mathbf{p}\|^2 - \left(\vec{\mathbf{n}}\cdot(\mathbf{x}-\mathbf{p})\right)^2} \quad \vec{\mathbf{n}}\cdot(\mathbf{x}-\mathbf{p})\right]^\top. \tag{3.32}$$

To obtain a higher quality histogram, we use bilinear interpolation that smoothly spreads the contribution of each point to the neighboring bins, resulting in a "soft" floating-point count of the number of points at each bin. The final result is a 2D image, which we call the spin image at point **p**.

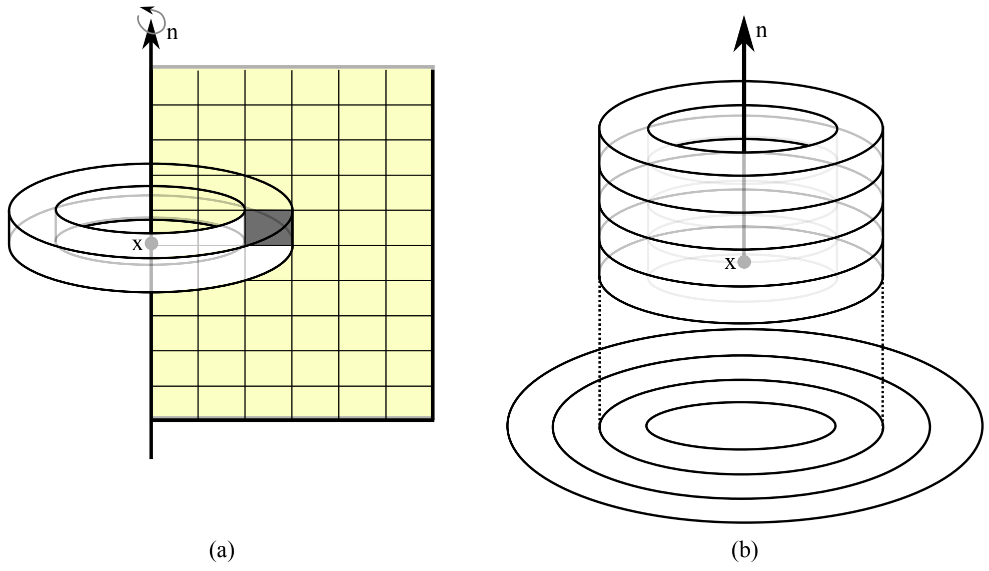


Figure 3.9: A spin image is a histogram, where the bins are concentric rings stacked along the normal direction. (a) Shows the analogy between a spinning normal plane and the histogram bin. (b) Several stacked bins.

There are a few parameters here that control the generation of spin images: the size of each grid bin, the total width of the image, and the support angle. The bin size determines the resolution of the spin image: the smaller the bin size, the higher the resolution. Since the image is sampled discretely, a very high resolution leads to many bins that contain no points. A good value for the bin size is the mesh resolution (the distance between the points), which gives a balance between the interpolation and the image resolution. The image width determines how large of a geometric patch we want to encode in the image. This setting can depend on how large of a patch we would like to compare. Finally, the support angle controls which points actually contribute to the spin image. This is done by comparing the surface normal $\vec{\mathbf{n}}_{\mathbf{p}}$ at $\mathbf{p}$ with the surface normal $\vec{\mathbf{n}}_{\mathbf{x}}$ at $\mathbf{x}$. If the angle between the normals exceeds the support angle threshold, we discard the point and do not accumulate it on the spin image. Since the rotating plane may hit other sides of the surface, the support angle is useful for limiting the information to similarly oriented points.

Spin Image Comparison & Matching. Once we have the spin images computed at each point, we can compare the images directly to see if the surface geometry around these points match. However, even a slightly different sampling of points on the same object may produce different spin images. Therefore, we cannot compare the images if the match exactly, but instead we need a measure of how similar (or different) two images are. For this purpose, Johnson [1997] applies the linear correlation coefficient to compare if corresponding bins in two images have similar value across the entire image. Given two spin images P, Q, the correlation $R(P, Q)$ is defined using the formula:

$$R(P,Q) = \frac{N\sum p_i q_i - \sum p_i \sum q_i}{\sqrt{\left(N\sum p_i^2 - \left(\sum p_i\right)^2\right)\left(N\sum q_i^2 - \left(\sum q_i\right)^2\right)}}, \tag{3.33}$$

where p_i and q_i is the value in corresponding bins of P and Q, and N is the total number of bins. The value of R ranges between 1 and -1, where a value of 1 means that the P and Q are highly correlated, 0 means they are not correlated, and -1 means they are inversely (or anti) correlated.

We may also consider the case where there is missing data or additional clutter of other surfaces. Addressing the problem of missing data is especially important for range scans, where most of the surface is occluded due to a limited viewpoint or the scanning technology. Here, it is often the case that in spin image P there is a bin with a positive value, but in the other spin image Q the bin has a zero value (because the geometry was missing). To reduce the effect on the correlation coefficient on these cases, Johnson [1997] proposes to only include bins in the computation of $R(P, Q)$ that have a non-zero value in both P and Q; i.e. only use the bins i for which both $p_i > 0$ and $q_i > 0$, or where the spin images “overlap.” This prevents us from comparing bins where there is data in one bin but missing in the other. The value of N then is the total number of bins where both p_i and q_i are positive. Here, note that the number of bins N will also affect the final value of R. Taking this into account, Johnson [1997] propose a similarity formula $C(P, Q)$ of the form

$$C(P,Q) = \operatorname{atanh}^2\left(R(P,Q)\right) - \frac{\lambda}{N-3}, \tag{3.34}$$

where N is the number of overlapping pixels, and λ is the expected overlap between two images. A higher value of C indicates that P and Q are similar, and a lower value of C indicates that they are different. Intuitively this formula balances the similarity value when there is too much (or too little) overlap relative to the expected number of overlapping bins λ.

Now, given a point $\mathbf{x}$ on shape $\mathcal{P}$, we can find the most similar points $\mathbf{y}$ on the another shape $\mathcal{U}$. We simply compute the similarity C between the spin image at $\mathbf{x}$ and all spin images for $\mathcal{U}$. The points on $\mathcal{U}$ which produces the largest value of C are taken to be the matching points. This allows us to find "informed" correspondences where the matching points have similar geometric structure. However, we cannot expect the matches to be perfect. For example, there could be repeating surface structures on the surface, leading to multiple matches with very high scores. Also, it may be difficult to produce meaningful matches in flat, cylindrical, or spherical regions where the surface is smooth and there are not many distinct geometric features. Nevertheless, spin images provides a useful tool to match similar geometric patches between multiple surfaces.

3.3 Clustering Using the Mean-Shift Algorithm

A key idea that we use in Chapter 4 is to identify points on the surface that move together as a group, using the correspondences obtained by descriptor matching. This is done by estimating the rigid transformation of each correspondence and clustering the resulting set of transformations. This allows us to find rigid parts of the surface that move together, and also to discard correspondences that are not consistent with others.

To perform the clustering, we use the mean-shift clustering method [Cheng, 1995; Comaniciu and Meer, 2002], which is a robust technique to identify clusters in distributions of points. Compared to traditional clustering techniques such as k-means, this method does not need to know the number of clusters in advance, it works well with noisy data, and it is suitable for finding clusters with arbitrary shapes. It has been successfully used previously to identify rigidly moving parts of a deforming surface [James and Twigg, 2005; Tuzel et al., 2005; Mitra et al., 2006], and we also use it in our work.

The mean-shift idea comes from estimating the density of a distribution of points using kernel density estimation. Suppose we have N data points $\mathbf{x}_i \in \mathbb{R}^d$. Then, the *kernel* *density estimate* at some point $\mathbf{x} \in \mathbb{R}^d$ is given by

$$xf(\mathbf{x}) = \frac{c_{k,d}}{Nh^d} \sum_{i=1}^{N} k\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h^2}\right), \tag{3.35}$$

where k is the *profile* of a radially symmetric kernel, h is the *bandwidth* of the kernel, and $c_{k,d}$ is a normalization constant for the kernel so that $K(\mathbf{x}) = c_{k,d}k\left(\|\mathbf{x}\|^2\right)$ satisfies the condition $\int_{\mathbb{R}^d} K(\mathbf{x})d\mathbf{x} = 1$. Some popular examples of the profile k include the profile of the Epanechnikov kernel

$$k_E(x) = \begin{cases} 1-x & 0 \le x \le 1 \\ \\ 0 & x > 1 \end{cases} \tag{3.36}$$

and the normal kernel

$$k_N(x) = \exp\left(-\frac{1}{2}x\right), \qquad x \ge 0. \tag{3.37}$$

The main connection between clustering and density estimation is that clusters of densely distributed points will show up as peaks of the density estimate. The key idea behind mean-shift is to provide an efficient way to find these peaks of the density, by estimating the gradient of the density function and following the direction of the gradient until it converges to a peak. Without going into a detailed derivation, it turns out that the gradient of the estimated density $f(\mathbf{x})$ is proportional to the mean shift vector

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{N} \mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h^2}\right)}{\sum_{i=1}^{N} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h^2}\right)} - \mathbf{x}, \tag{3.38}$$

where $g(x) = -\dot{k}(x)$ is the negative derivative of the profile $k(x)$. Intuitively this mean-shift vector is the difference between the current position $\mathbf{x}$ and the normalized weighted mean of the points $\mathbf{x}_i$ around $\mathbf{x}$. Since it is proportional to the gradient of the density, it always points in the direction of the maximum increase in the density. Thus, we can travel to the nearest peak of the density by iteratively updating the current position with the mean-shift

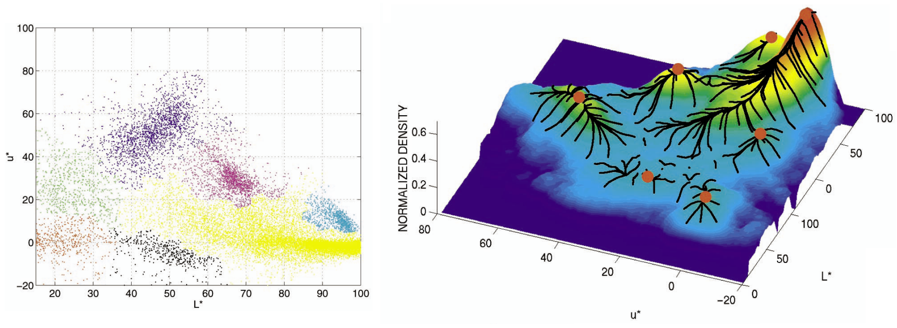


Figure 3.10: Example of clustering using the mean-shift algorithm. From Comaniciu and Meer [2002].

vector as $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{m}(\mathbf{x})$, until we converge to a specific location where $\mathbf{m}(\mathbf{x}) = 0$. This method is guaranteed to converge, as proved by Comaniciu and Meer [2002].

We can identify all the peaks of the density function by performing this gradient ascent starting at every data point $\mathbf{x}_i$, or at regularly spaced locations near the data points. As we expect the number of peaks to be much smaller than the number of points, the set of data points which converge to the same peak forms the *basin of attraction* of this peak. This provides a natural way of clustering the set of data points, where locations of the peaks are the cluster centers, and the basin of attraction of each peak are its cluster members. The one parameter needed to be specified by this method is the bandwidth parameter h. This can be estimated empirically by the user, or alternatively automatic bandwidth selection methods can be used, which was proposed by Comaniciu et al. [2001].

Figure 3.10 shows an example of the clusters found using this procedure. The left image shows the distribution of the data points, which are taken by plotting the colors of an image (the L* and u* components in L*u*v* color space). The points are colored according to the clusters classified by the mean-shift algorithm. The right image shows the estimated density, with the dots representing peaks and the lines representing the trajectory of the mean-shift algorithm. Although this dataset is noisy, we see that the procedure converges nicely to the peaks, and the identified clusters form arbitrary shapes in the color space.

3.4 Discrete Optimization With Graph Cuts

A common problem in computer vision is to estimate some spatially varying quantity from the given data. This can be formulated as assigning a label f_p in a finite set of labels $\mathcal{L}$ for each pixel p in an image $\mathcal{P}$. Then, the goal is to find a labeling of the pixels that is smooth and consistent with the data. For example, the labels may be disparity values that are assigned to pixels in a stereo image pair. In this case, the assigned disparity values would match the movement of the pixels in the pair of stereo images.

This type of formulation is also applicable to the articulated shape registration problem in our work. In this context, the rigid transformations are the "labels" and the points of the surface are the "pixels." A labeling of the surface points produces a division of the surface into multiple parts, where each part moves according to its assigned rigid transformation. The objective is to find the labeling that minimizes the registration error.

We can express this kind of objective as finding a minimum of an energy function. Commonly this energy function has two terms: (1) a data term $D_p(f_p)$ which measures how well label f_p is suited for pixel p, and (2) a smoothness term $V_{p,q}(f_p, f_q)$ which can encourage a smooth labeling by specifying that neighboring pixels p, q should have the same label $f_p = f_q$. Summing up these terms over all pixels $\mathcal{P}$ and all neighbors $\mathcal{N}$ we obtain the energy function $E(f)$ of a labeling f as

$$E(f) = \sum_{p\in\mathcal{P}} D_p(f_p) + \sum_{\{p,q\}\in\mathcal{N}} V_{p,q}(f_p, f_q). \tag{3.39}$$

A useful type of smoothness term V is a *discontinuity preserving* smoothness term. This is useful because often in vision applications (e.g. stereo matching) the labels can suddenly change at object boundaries. This is also true in our case, where we want to produce a labeling of the shape into articulated parts. The Potts interaction model is a good example of a discontinuity preserving smoothness term, where $V_{p,q}(f_p, f_q) = u_{\{p,q\}} T(f_p \neq f_q)$ where $u_{\{p,q\}}$ is a constant for each neighbor pair $\{p, q\}$ and $T(\cdot)$ is 1 if the argument is true, 0 otherwise. Under this model, any discontinuity is penalized equally, allowing for large jumps in the assigned label values equally as well as small jumps.

It turns out that finding the global minimum of this energy function for the Potts energy is an NP-hard problem [Boykov et al., 2001], because it can be reduced to finding a minimum cost multiway cut (which is NP-hard). However, Boykov et al. [2001] develop a good approximation algorithm that can produce a strong local minimum of the energy, via transformation to the binary max-flow/min-cut problem. We refer to this algorithm as "graph cuts" in our work, and we use it extensively to optimize for an articulated matching of deforming surfaces. The graph cuts algorithm is especially useful because it is fast and can optimize a variety of objective functions, even non-linear ones as well. In the following, we discuss graph cuts in detail, along with an intuitive explanation of the associated optimality guarantees.

3.4.1 Alpha-Beta-Swap and Alpha-Expansion Algorithms

The basic idea of the optimization is to transform the multi-labeling problem into a sequence of binary labeling problems. Each solution of the binary problem lowers the cost of the overall labeling, until the algorithm converges to a local minimum. The idea used in this transformation is the concept of the α-β-swap move and the α-expansion move.

Suppose that we have an initial labeling f of all pixels $p \in \mathcal{P}$. Given any pair of labels $\alpha, \beta \in \mathcal{L}$, the swap move swaps the label between α and β in a subset of all pixels labeled α or β in the current labeling f. For the expansion move, given any single label α, this move expands the region of pixels labeled α by changing a non-α pixel to have the label α.

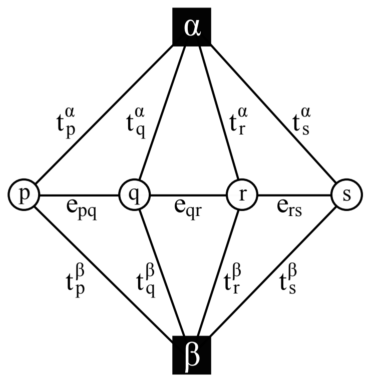
Since each labeling produces an energy, the key question is: given the current labeling, is it possible to lower the total cost of the labeling by performing a single swap or expansion move? Therefore, performing each swap or expansion becomes a binary decision problem. For swap, we need to make a decision on each pixel (with current label α or β) whether to keep the current label or to swap the label, in order to lower the total energy. Similarly, for expansion, we must decide whether to keep the current label or to change it to α, for every pixel in the entire image.

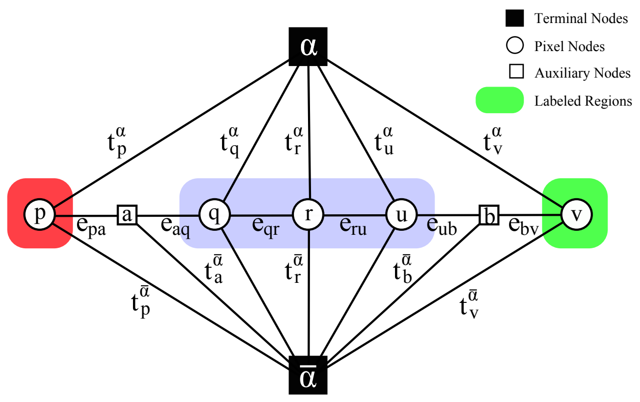
Algorithm 3.3: α-β-SWAP($\mathcal{L}, \mathcal{P}, \mathcal{N}, E(f), f$)

```
Data: A set of labels $\mathcal{L}$, set of sites $\mathcal{P}$, set of neighbors between sites $\mathcal{N}$, energy
      function $E(f)$, and initial labeling $f$
Result: A final labeling $f$ that is a local minimum of $E(f)$ with respect to one
        swap move
1  begin
2      Given initial labeling $f$;
3      Define success ← true;
4      while success == true do
5          success ← false;
6          foreach Pair of labels $\alpha, \beta \in \mathcal{L}$ (Label $\alpha \in \mathcal{L}$) do
7              Find the minimum energy labeling $f'$ minimizing $E(f')$ within one
               $\alpha$-$\beta$-swap move ($\alpha$-expansion move) from $f$;
8              if $E(f') < E(f)$ then
9                  $f \leftarrow f'$;
10                 success ← true;
11     return Final labeling $f$;
12 end
```

As Boykov et al. [2001] show, we can construct a graph to efficiently solve these binary decision problems via reduction to max-flow min-cut. This motivates an iterative algorithm to efficiently minimize the energy $E(f)$, summarized in Algorithm 3.3 (the α-expansion case is similar, and it is indicated by the text in parentheses). In this algorithm, we repeatedly try to minimize the energy $E(f)$ by performing swap (expansion) moves for all pairs of labels $\alpha, \beta \in \mathcal{L}$ (all labels $\alpha \in \mathcal{L}$). We continue this until we cannot perform a swap move for any pair α, β (expansion move for any α) to reduce the energy. In this sense, the algorithm converges to a local minimum of the energy with respect to a single swap of any pair α, β (single expansion of any α).

In order to solve the binary swap/expansion problem, Boykov et al. [2001] devise a graph $G = (V, E)$ where solving for the minimum cut in the graph naturally corresponds to finding the swap/expansion move that minimizes $E(f)$. Recall that in the max-flow min-cut problem, there are two terminal nodes $s, t \in V$ denoting the source and the sink, and each edge in the graph has a non-zero weight. Then, a *cut* is the partition of V into two sets, S, T, where $s \in S$ and $t \in T$. The cost of this cut is the sum of the weights of all *cut edges*: edges





Edge	Weight	For
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
e_{pq}	$V(\alpha, \beta)$	$\{p, q\} \in \mathcal{N}$ $p, q \in \mathcal{P}_{\alpha\beta}$

Edge	Weight	For
$t_p^{\bar\alpha}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar\alpha}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^α	$D_p(\alpha)$	$p \in \mathcal{P}$
e_{pa}	$V(f_p, \alpha)$	$\{p, q\} \in \mathcal{N}, f_p \neq f_q$
e_{aq}	$V(\alpha, f_q)$	$\{p, q\} \in \mathcal{N}, f_p \neq f_q$
$t_a^{\bar\alpha}$	$V(f_p, f_q)$	$\{p, q\} \in \mathcal{N}, f_p \neq f_q$
e_{pq}	$V(f_p, \alpha)$	$\{p, q\} \in \mathcal{N}, f_p = f_q$

Figure 3.11: Constructed graphs and edge weights for the swap (left) and expansion (right) algorithms. In the graph for the expansion algorithm, additional auxiliary nodes are created between pixel nodes whose current labels are different. From Boykov et al. [2001].

going between a vertex in the set S and a vertex in the set T. The main idea is that finding the minimum cost cut of the graph is equivalent to finding the maximum flow from s to t according to the edge weights defined on the edges. This problem can be solved using a number of different algorithms, for example the Ford-Fulkerson algorithm.

The specific construction of the graph is shown in Figure 3.11. Given a labeling on the pixels f, suppose that $\mathcal{P}_\alpha$ is the set of pixels p such that its label $f_p = \alpha$. Also, let us denote $\mathcal{P}_{\alpha\beta} = \mathcal{P}_\alpha \cup \mathcal{P}_\beta$. For the swap algorithm, the vertices of the graph are the two terminal nodes α, β and a pixel node for every $p \in \mathcal{P}_{\alpha\beta}$. There are three types of edges: terminal edges from a pixel node p to α with weight t_p^α, terminal edges from a pixel node p to β with weight t_p^β, and neighbor edges between each neighbor $\{p, q\} \in \mathcal{N}$ with weight e_{pq}. The values of the weights are summarized in the above figure. The α-expansion case is similar, except that (1) we create a pixel node for all pixels $p \in \mathcal{P}$, and (2) for each edge $\{p, q\} \in \mathcal{N}$ where $f_p \neq f_q$, we create an additional auxiliary node a and create three edges $e_{pa}, e_{aq}, e_a^{\overline{\alpha}}$.

Note that a cut on these graphs corresponds to a labeling on each pixel node. Let us denote the set of all cut edges for a given cut (S, T) as C. Note that for each node, exactly one of its terminal edges will be in C; otherwise both s, t will be in S (or T). Therefore, for each pixel p, we assign the label of the terminal corresponding to the cut edge for p in C. For example, in the swap case we assign the label α if $t_p^\alpha \in C$ or β if $t_p^\beta \in C$, and for the expansion case, we assign the label α if $t_p^\alpha \in C$ or the label f_p if $t_p^{\overline{\alpha}} \in C$.

Using this labeling, we can show that the cost of a cut is exactly the cost of the corresponding labeling (maybe plus some constant) for both the swap and the expansion cases. The proof only works when the smoothness term V is a semi-metric (for the swap algorithm) or a metric (for the expansion algorithm). This means that $V(\alpha, \beta) = 0$ if and only if $\alpha = \beta$, $V(\alpha, \beta) = V(\beta, \alpha)$ (to be a semi-metric), and additionally $V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)$ (to be a metric). We will not reproduce the proof here, but provide a brief explanation. Conceptually, the weights on the edges are engineered precisely so that the cost of a minimum cut on the graph will produce exactly the energy of the corresponding labeling. The α-expansion case is a little more complicated due to the auxiliary node, but this node is created because there are three cases for the smoothness term when the labels on neighbors p, q are different: (1) $V(f_p, \alpha)$, (2) $V(\alpha, f_q)$, and (3) $V(f_p, f_q)$. The triangle inequality requirement (i.e. requirement that V is a metric) is required here for the minimum cut to include exactly one of the three edges connected to each auxiliary node.

Boykov et al. [2001] go on to show that the energy of the $E(f)$ local minimum obtained by the α-expansion is at most a factor of $2c$ (as small as a factor of 2) of the cost of the global minimum solution f^*. Here c is the maximum ratio of any two smoothness costs of a neighbor pair p, q. Along with this guarantee, the algorithm itself has a practical running time for a variety of problems and smoothness energies. This has made the algorithm very popular in recent years, and such graph cut methods have been successfully applied for stereo matching, multi-camera scene reconstruction, surface reconstruction, shape matching, image resizing, and many other applications.

3.4.2 Further Reading

Many further developments have been made to the basic algorithms discussed above. Kolmogorov and Zabih [2004] have characterized the class of energy functions (called “regular” or “submodular” functions) that can be solved via reduction to the min-cut problem, both for smoothness terms involving pairs of sites or triplets of sites. They also discuss a simpler construction of the graph for the α-expansion case. In addition, Boykov and Kolmogorov [2004] discuss fast max-flow/min-cut algorithms that can be applied for graph cut algorithms.

Energy functions of the form that we have discussed in Equation (3.39) also arise in the context of labeling Markov Random Fields. There are other algorithms that apply in this context, such as loopy belief propagation (LBP). Szeliski et al. [2006] give a comparative survey of these techniques in the context of stereo matching, image segmentation, and other applications. More recently researchers have also investigated the application of graph cuts to non-submodular functions [Kolmogorov and Rother, 2007], computing feature correspondences for non-rigid motion in images [Torresani et al., 2008b], and strategies to speed up graph cut computation by combining multiple solutions and reducing the number of iterations [Lempitsky et al., 2009].

3.5 Deformation Models

The graph cuts algorithm is useful for obtaining a discrete segmentation of the surface into rigid parts. However, the discrete assignment often causes an undesirable surface deformation at boundaries between rigid parts. In our work, we use *deformation models* that describe how to smoothly and continuously blend multiple transformations to create a smooth transition at boundaries between parts. We use this technique as a post-processing step in Chapters 5 and 6 to remove artifacts and produce a higher quality surface deformation.

The idea is to smoothly blend the boundary between parts by specifying *weights* at each point on the surface. These weights specify exactly how much each point is influenced by the transformations, and they are chosen so that there is a smooth transition at the boundary between neighboring parts. Finally, the deformation model specifies how to blend the transformations according to these weights. The method of choice for our work is the simplest alternative, which is to linearly blend the transformation matrices (i.e. linear blend skinning, or LBS). In this case, it is easy to optimize the weights, because the resulting position is determined by a linear function of the weights. We also discuss higher-quality alternatives, such as linearly blending the dual quaternion representations of the transformations.

3.5.1 Linear Blend Skinning (LBS) and Dual Quaternion Linear Blending (DLB)

There are two basic components to the deformation models: the weights and the transformations. The transformations describe the configuration (or movement) of the bones of the skeleton, and the weights define the association of each surface point to the bones. Concretely, if we assume that there are a total of B transformations, each transformation $T_j = \left(R_j, \vec{\mathbf{t}}_j\right)$ is a rigid transformation describing the position and orientation of each bone. In addition, each weight is a B-dimensional vector $\mathbf{w}_\mathbf{x}$ assigned to every point $\mathbf{x}$ on the surface, where the jth component of the vector $w_{\mathbf{x}j}$ describes the weight or influence of transformation j to $\mathbf{x}$. To describe a movement of the surface, we first specify a weight on each point of the surface. Then, the transformations move each point according to its weights via a weighted sum of the transformed points. This is summarized by the function $D(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^3$, where

$$D(\mathbf{x}) = \sum_{j=1}^{B} w_{\mathbf{x}j} T_j(\mathbf{x}) = \sum_{j=1}^{B} w_{\mathbf{x}j} \left(R_j \mathbf{x} + \vec{\mathbf{t}}_j\right). \tag{3.40}$$

Here, the weights are assumed to be non-negative and sum to 1 for all $\mathbf{x}$ on the surface S:

$$\sum_{j=1}^{B} w_{\mathbf{x}j} = 1 \quad \text{and} \quad w_{\mathbf{x}j} \geq 0 \quad \forall\, j \in \{1..B\} \text{ and } \mathbf{x} \in S.$$

This model can be also generalized to include affine transformations which include stretching and shearing movements of the surfaces. This representation of the surface movement is a popular and widely adopted strategy to describe movement, especially for articulated characters. However, a key limitation of the model is that the linearly combined transformations does not produce a valid rigid transformation. This results in well-known artifacts such as the candy wrapper effect, where a twist causes the surface to shrink to a point, or extreme bending causes an excessive loss (or increase) of volume in the surface.

An alternative that overcomes this limitation is the dual-quaternion linear blending model introduced by Kavan et al. [2008]. It provides a natural way to blend rigid transformations so that the resulting transformation is also rigid. The idea is to convert the rigid transformations $T_j = \left(R_j, \vec{\mathbf{t}}_j\right)$ into their dual-quaternion representation $\hat{\mathbf{q}}_j$, and compute the normalized weighted sum of the dual quaternions:

$$DLB\left(\mathbf{w_x}, \hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_B\right) = \frac{\sum_{j=1}^{B} w_{\mathbf{x}j} \hat{\mathbf{q}}_j}{\left\| \sum_{j=1}^{B} w_{\mathbf{x}j} \hat{\mathbf{q}}_j \right\|}. \tag{3.41}$$

The resulting unit dual-quaternion corresponds to a rigid transformation, which is then applied to transform the point $\mathbf{x}$. This dual-quaternion blending is an approximation of exact spherical averages of rotations [Buss and Fillmore, 2001] and removes the artifacts in LBS. However, the model is non-linear, so the weights are not as easy to estimate as LBS. For our work, we primarily use the LBS model, but we use DLB in one specific case to blend a set of transformations (see Section 6.3.3).

3.5.2 Other Improved Models

In this section, we discuss a variety of other models that have been developed to overcome the limitations of the basic LBS model. Although we mainly use LBS in our work, we give a brief survey of these alternatives to inspire future work to model the surface motion more accurately.

The main idea of these approaches is to learn a corrective model or fit a more general deformation model from a set of example poses. Lewis et al. [2000] correct LBS by adding surface displacements to the rest pose. The idea is to learn a mapping from the domain of the skeleton parameters to the value of the displacements on the surface. This mapping is learned from a set of example poses, and to generate poses beyond the example set, the displacements are interpolated using Radial Basis Function (RBF) interpolation. Building on this approach, Kry et al. [2002] reduce the space requirements by performing PCA compression on the displacements. Wang and Phillips [2002] generalize LBS in a different way by adding weights per matrix component in T_j instead of having a single weight per T_j. These per-component weights are learned from example poses as well. Mohr and Gleicher [2003] improve LBS by proposing to add extra joints to the skeleton, also based on examples. Finally, Wang et al. [2007] takes the displacement idea one step further and adds deformation gradient corrections [Sumner and Popović, 2004]. The general formulation involves solving a sparse linear system to reconstruct the mesh that best satisfies the deformation gradients. However, they develop a simplified formulation that generalizes LBS to add deformation gradients directly to the transformations. Interestingly these deformation gradients themselves are expressed as linear combinations of “key” deformation gradients.

4

Automatic Registration for Articulated Shapes

OUR goal is to register multiple range scans of a moving, articulated subject. While ICP produces a fast and accurate registration, it depends on having a good initial alignment. Therefore, we need a robust way to find a registration that does not depend on the initial alignment of the scans, but only on the shape of the scanned surface.

In this chapter, we present a method that robustly aligns a pair of surfaces. This technique works well with large motions, and it does not depend on the initial orientation of the surfaces. In addition, this method can disambiguate between different matchings of surface parts and produces a registration that is consistent and does not distort the surface. However, this method does not produce an articulated motion model. Later, in Chapter 6, we use this algorithm to initialize the registration of multiple scans and integrate the registration with a motion model.

4.1 Contributions

Our approach optimizes a novel cost function to evaluate the alignment between two surfaces. The optimization is divided into two steps: sampling the motion between the two surfaces, and applying graph cuts to assign the best spatially varying motion that aligns the shapes while preserving its structure. Our method is able to disambiguate between multiple possible assignments of similarly shaped parts given sufficient connectivity within the shape. These contributions are summarized in the list below:

- We formulate the registration problem as a label assignment problem, where the labels are rigid transformations that can be assigned to the surface.
- We develop a novel cost function for simultaneously solving a consistent assignment of forward transformations (from the source to the target) and backwards transformations (from the target to the source).
- We optimize this cost function using graph cuts, resulting in an assignment of transformations that aligns the shapes. In addition, contiguous regions with the same transformation give a segmentation of the shape into rigid parts.

We apply our approach to a number of real-world and synthetic datasets, and we demonstrate that we can align the shapes accurately. In addition, our algorithm is robust to missing data in both shapes, and the resulting alignment can fill in the missing parts of one shape with the other. Once this basic registration is completed, one can perform higher level tasks, such as constructing templates, interpolating shapes, automatically rigging skeletons, and building general shape models.

4.2 Registration Algorithm

In this section, we describe our approach for automatically registering pairs of articulated shapes with significant missing data. We interpret the registration problem as finding, for each point in one shape, a rotation and translation that moves it to the corresponding point in the other. A solution would be to formulate this as a continuous optimization problem and solve for a smoothly varying transformation field. However, this results in a difficult continuous non-linear optimization problem.

In our approach, we take advantage of the observation that articulated objects consist of a few rigidly moving parts. Our main idea is to first determine a finite set of significant

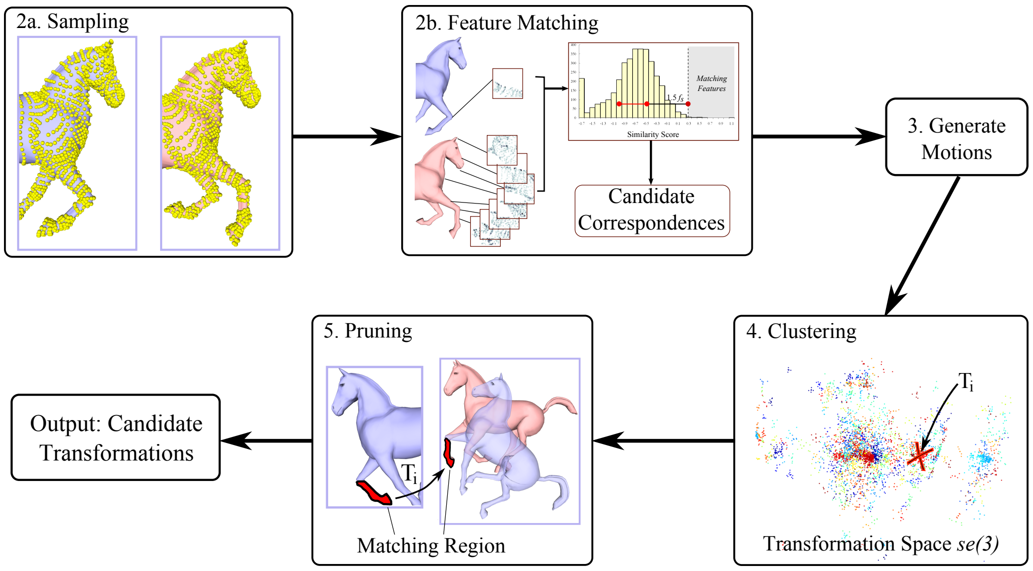


Figure 4.1: Motion sampling overview. After precomputing coordinate frames and feature descriptors, we sample the shapes and compare spin image features to find candidate correspondences between the shapes. Using the precomputed coordinate frames, we generate a rigid transformation for each correspondence. Then, we cluster the resulting set of candidate transformations to obtain the final set of transformations. Optionally, we prune unnecessary transformations based on matching regions.

motions between the mostly rigid parts of the pair of shapes. We show that it is possible to extract the significant motions even if large parts of the shapes are missing. We then find the registration by solving a label assignment problem, where each transformation corresponds to a label.

A main challenge of this approach is that many shapes contain several similar parts, as for example the four legs of a horse. Therefore, many significant transformations that lead to good alignments of parts will not lead to consistent global registration. We formulate a cost function for the labeling problem that prefers an assignment of transformations that keeps the shape intact and ensures that each part is mapped in a globally consistent manner. Adding this regularization enables us to apply graph cuts for solving the resulting optimization problem.

4.2.1 Motion Sampling

In this section, we describe an algorithm to find a finite set of rigid transformations $\mathcal{T} = \{T \mid T \in SE(3)\}$ that describes the movement of each part of the shape. We follow the work of Mitra et al. [2006] for estimating this set. The algorithm has the following steps, illustrated in Figure 4.1:

1. Precompute per-vertex coordinate frames and feature descriptors
2. Sample the shapes and match similar features
3. Generate the motion for each match
4. Cluster the transformations
5. Prune transformations based on matching regions

The purpose of the feature matching, clustering, and selection steps is to narrow down on a concise set of transformations that describes the movement of all rigid parts of the shape. Here we discuss the specific design choices made in the implementation of this method, and we refer the reader to Mitra et al. [2006] for additional details.

Coordinate Frames. As a preprocessing step, we estimate a coordinate frame on each vertex of the shape. The frame (R, t) contains the 3D location (the position of the vertex, t) and three orthonormal vectors (collected in matrix R) consisting of the normal vector and the two principal curvature directions. The principal curvature directions are estimated by least-squares fitting, as described in Section 3.2.1. These frames are used in Step 3 to find a rigid transformation between a pair of corresponding points.

Feature Descriptor. Also in the preprocessing step, we compute the spin image at each vertex. As we described in Section 3.2.2, a spin image is a histogram of the vertices where the bins are concentric rings stacked along the normal direction. This is visualized as a sweeping plane rotating about the normal direction, collecting vertices in a grid defined on the plane. Spin images have been successfully used for computing correspondences in range data and are robust to clutter and occlusion in static scenes. Since we assume that our object has articulated motion composed of several rigidly moving parts, the spin images work well as long as they are localized to small neighborhoods. We have also tried using multi-scale principal curvatures, but we found that spin images were more discriminative.

Sampling and Feature Matching. In these steps, we use the precomputed feature descriptors to find possible corresponding points between the source and target shapes. First, we randomly subsample the set of vertices in each shape, in order to reduce the number of comparisons during the matching step. Next, for a single point p in the source, we find corresponding points in the target by computing the similarity score (higher is better) from p's spin image to all of the spin images in the target (Figure 4.1). Collecting these scores into a histogram, we find the features that are significantly more similar than the rest—to such a degree that they are outliers in the histogram. We use a moderate threshold of $1.5f_s + m_u$ to determine outliers, where m_u is the median of the upper half of the measurements, m_l is the median of the lower half of the measurements, and $f_s = m_u - m_l$. We repeat this matching process for all vertices in $\mathcal{P}$ to obtain a large set of correspondence "candidates."

Generating Motions. Now, for each correspondence candidate (p, u) where $p \in \mathcal{P}$ and $u \in \mathcal{U}$, we use the precomputed coordinate frames $T_p = (R_p, t_p)$, $T_u = (R_u, t_u)$ to generate a rigid transformation $T = (R, t)$ from p to u, given by

$$R = R_u R_p^\top, \quad t = t_u - R\, t_p.$$

Here, R is the rotation and t is the translation from T_p to the T_u, and $SE(3)$ is the space of all rigid transformations. As a result, we have a set of rigid transformations, where each transformation is associated with a single correspondence candidate. Note that sometimes the coordinate frames have opposite handedness (i.e. left handed / right handed frame) due to an ambiguity in the sign of the eigenvectors. This results in R having negative determinant, and so we address this case by flipping the principal direction in R_u that yields the smaller rotation.

Clustering Motions. In this step, we use a variant of the non-linear mean-shift framework to cluster the set of transformations [Tuzel et al., 2005]. Mean-shift is a non-parametric clustering algorithm that finds peaks of the local density of a point set using gradient ascent. When mean-shift clustering is applied for rigid motions, the challenge is to define an appropriate distance measure for comparing transformations. The non-linear mean-shift approach defines the distance between any two transformations $X, Y \in SE(3)$ as

$$d(X, Y) = \| \log(X^{-1}Y) \|$$

where $\log(X)$ maps a transformation $X \in SE(3)$ to the corresponding element x in the Lie algebra $se(3)$, a 6D vector containing axis-angle rotation and linear velocity [Tuzel et al., 2005]. Considering the Lie group $SE(3)$ as a differentiable manifold, this distance $d(X, Y)$ corresponds to the length of the geodesic curve between $X^{-1}Y$ and the identity $e \in SE(3)$.

This distance measure is a natural definition that is based on the structure of a Lie group, but it is a non-linear function that is expensive to evaluate and cannot support a data structure for fast range queries. Therefore, we opt for an approximation

$$d'(X, Y) = \| -x + y \|$$

where $x, y \in se(3)$ are the axis-angle and linear velocity representations of $X, Y \in SE(3)$, respectively. The difference between $d(X, Y)$ and $d'(X, Y)$ can be expressed in terms of the Baker–Campbell–Hausdorff (BCH) formula:

$$\begin{aligned}\log(X^{-1}Y) &= \log(\exp(-x)\exp(y)) \\ &= \log(\exp(\mathrm{BCH}(-x, y))) \\ &= \mathrm{BCH}(-x, y) \\ &= -x + y + O([-x, y])\end{aligned}$$

where $[x, y] = xy - yx$ is the Lie bracket operator or commutator of the Lie algebra, and $BCH(x, y)$ is a series expansion based on nested iterated commutators. Blanes and Casas [2004] show that $x + y$ is a good approximation to $\log(e^x e^y)$ when the norm of the commutator $[x, y]$ is sufficiently small. Furthermore, since $d'(X, Y)$ is just Euclidean distance in the Lie algebra $se(3)$, it is faster to evaluate and supports fast approximate range queries using a k-d tree. This is particularly useful when we have a large number of rigid transformations to cluster.

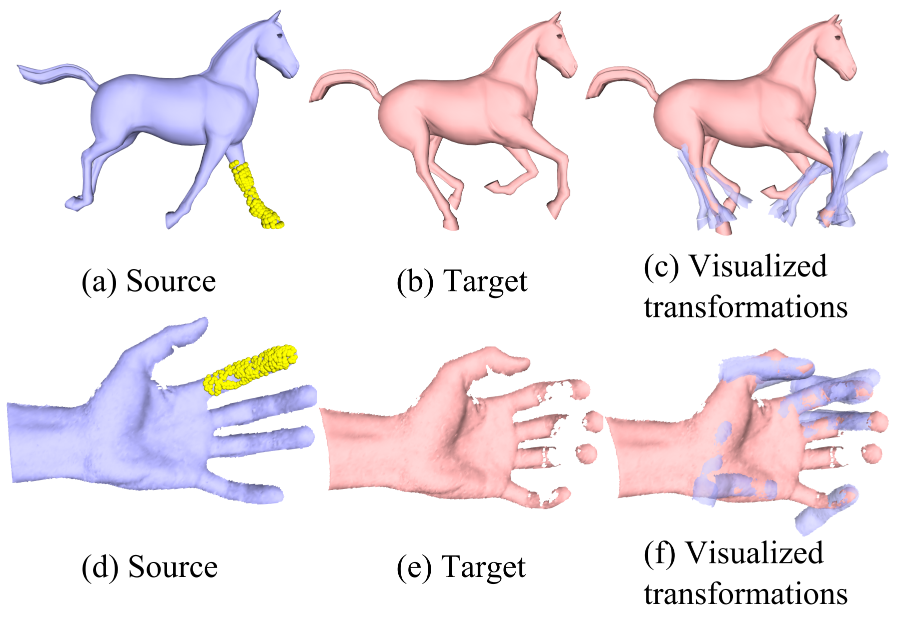

(a) Source
(b) Target
(c) Visualized transformations
(d) Source
(e) Target
(f) Visualized transformations

Figure 4.2: Visualizing the set of estimated transformations $\mathcal{T}$ from the source (a,d) to the target (b,e). We manually select a region in the source and plot a transformed copy of the region for each associated transformation (c,f). We see that there are many transformations between different but similarly shaped parts. The graph cuts optimization assigns the transformations to map each part correctly.

To utilize the distance $d'(X,Y)$, we first map each transformation $T=(R,t)\in SE(3)$ to its corresponding element in $se(3)$. The scale of the rotations R usually differs from the scale of the translations t, so we perform zero mean, unit standard deviation normalization on the translations. As a result, the range of norms for the rotational and translational components are in $\approx[0,\pi]$. Finally, we construct a k-d tree and apply mean-shift clustering using the Epanechnikov kernel. As a result, we obtain a set of clusters, where each cluster is a group of similar transformations. Finally, we compute the cluster modes and collect all of them to obtain the final set of estimated transformations $\mathcal{T}$. A visualization of these transformations is shown in Figure 4.2.

Pruning. This is an optional step to further refine the set of transformations $\mathcal{T}$. Even after clustering, there are many spurious transformations resulting from incorrect correspondences. Therefore, we want to retain the best transformations and get rid of unnecessary ones.

Most reliable transformations will have the largest matching regions (Figure 4.1). Therefore, we find the matching region for each transformation, and retain only the top k

transformations with the largest matching regions. However, since good transformations corresponding to very small regions may be potentially discarded, we found it acceptable to optionally skip this step if the feature matching is sufficiently accurate and discriminative.

In our implementation, to find the matching region, we start at a random cluster member (a pair of vertices on the source and target) and incrementally grow the region, only to the point where applying the transformation keeps the region within a small distance of the target. In addition, we refine the accuracy of the transformation during this time by performing ICP at regularly scheduled intervals. When there is missing data in the mesh, there will typically be boundary vertices, which are the vertices of edges that are adjacent to a single triangle. To help ICP to be more robust to missing data, we slightly modify the point selection step: in the case where the closest point u is on a boundary, we instead use the closest point on the tangent plane at u. This strategy helps to provide accurate transformations when there are large missing regions in the data.

Conclusion. The final output of the motion sampling is a finite set of rigid transformations $\mathcal{T} = \{T \mid T \in SE(3)\}$ of the best transformations determined by the clustering or selection step.

4.2.2 Graph Cuts Optimization

The next step in our method is to assign the transformations to the shapes. This is essentially a labeling problem, where the vertices of the shape are the nodes and the transformations are the labels assigned to each node. We first develop a cost function that measures the quality of assigning the transformations, and describe how we apply graph cuts to optimize this cost.

Cost Function. We expect that a good transformation assignment aligns the shapes well and results in a consistent deformation that preserves the connectivity of the shape. We express this preference as minimizing a cost function

$$\underset{\{T_p \in SE(3)\ \forall p \in \mathcal{P}\}}{\operatorname{argmin}} \sum_{p \in \mathcal{P}} D(p, T_p) + \sum_{(p,q) \in E_P} V(p, q, T_p, T_q) \tag{4.1}$$

where $D(p, T_p)$ represents a *data cost* measuring how well the application of rigid transformation T_p matches p to $\mathcal{U}$, E_P specifies the connectivity of $\mathcal{P}$, and $V(p, q, T_p, T_q)$ is a *smoothness cost* measuring the consistency of the deformation between a pair of points $(p, q) \in E_P$. This cost function is reminiscent of one developed by Allen et al. [2003], except that they consider T_p to be affine transformations. Stated purely in this form, minimizing this function results in a continuous, non-linear optimization problem that is generally difficult to minimize.

Our observation is that motion sampling gives *a priori* the set of possible transformations to assign to each vertex $p \in \mathcal{P}$. Instead of finding each T_p in the continuous space of rigid transformations, for each p we make a selection f_p of some transformation in a prescribed set $\mathcal{T}$. Thus, we transform the above continuous optimization problem to a discrete one:

$$\underset{\{f_p \in \mathcal{T} \,\forall\, p \in \mathcal{P}\}}{\operatorname{argmin}} \sum_{p \in \mathcal{P}} D(p, f_p) \; + \sum_{(p,q) \in E_P} V(p, q, f_p, f_q). \tag{4.2}$$

If we consider the transformations as labels to assign to each vertex, we see indeed that minimizing the above cost function corresponds to solving for an optimal labeling of the vertices. Therefore, we can use graph cuts for the optimization.

What remains is a definition of the data and smoothness terms in the objective function. For the data term, we provide a catalog of useful functions in Table 4.1. The point-to-point metric is the most basic matching function, illustrated in Figure 4.3b. Compared to the point-to-point metric, the point-to-plane error metric is more robust to missing data and the sampling of the surface. We prefer to use this metric when the sampling of the shape is very sparse. The hybrid metric is designed for missing data and measures a combination of the point-to-point and point-to-plane metrics, extrapolating the distance at the boundary of the shape. We choose this term when there is a substantial amount of missing data. Finally, the point-and-normal metric measures both the point-to-point distance and the difference of the normal vectors. Here, ν is an additional parameter controlling the influence of the normals. We use the point-and-normal term when precise normals are available.

For the smoothness term, the goal is to preserve the consistency of the shape. A

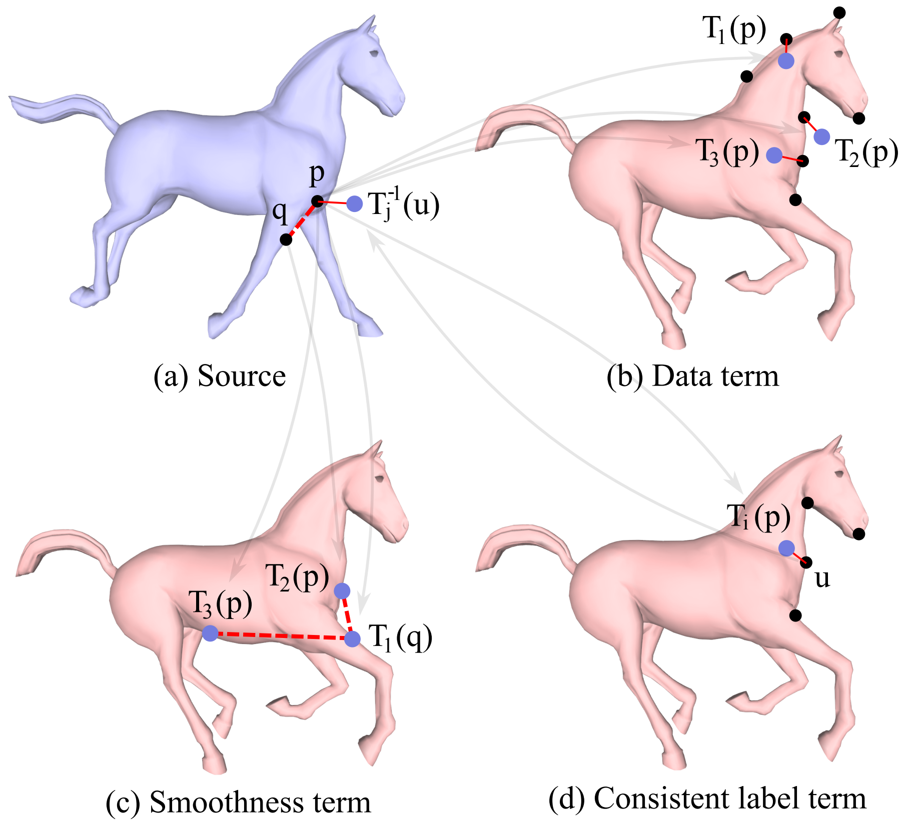


Figure 4.3: Illustration of the cost function from the source (a) to the target (b,c,d). The data term (b) measures the distance of the transformed point $T(p)$ to the closest point on the target. The smoothness term (c) measures the change of length between neighboring points p, q after applying the transformations. The consistent label term (d) ensures that the transformation T_i assigned to p is consistent with T_j assigned to the closest point u, i.e. $T_i = T_j$.

possibility is to directly compare the labels or to compare the transformations between neighboring vertices. However, we do not want to penalize differing transformations due to potential joints in the articulated shape. Therefore, we preserve the *edge lengths* of neighboring points

$$V(p, q, f_p, f_q) = \Big| \|p - q\| - \|f_p(p) - f_q(q)\| \Big|. \tag{4.3}$$

If $f_p = f_q$ then we see that $V(p, q, f_p, f_q) = 0$, as is desirable for a smoothness function. In addition, this expression does not penalize the assignment f_p and f_q as long as it preserves the edge length $\|p - q\|$. This means that we can assign completely different transformations at neighbors p and q as long as it does not break the shape apart. As an example, in Figure 4.3c, we see that assigning T_2 to p preserves the edge length, while T_3 is penalized because it

Table 4.1: A catalog of data cost functions. For a vertex p with normal n_p and a selected transformation f_p, $f_p(p)$ applies f_p to p (only rotation for normals), and $u \in \mathcal{U}$ is the closest point to $f_p(p)$ with associated surface normal n_u.

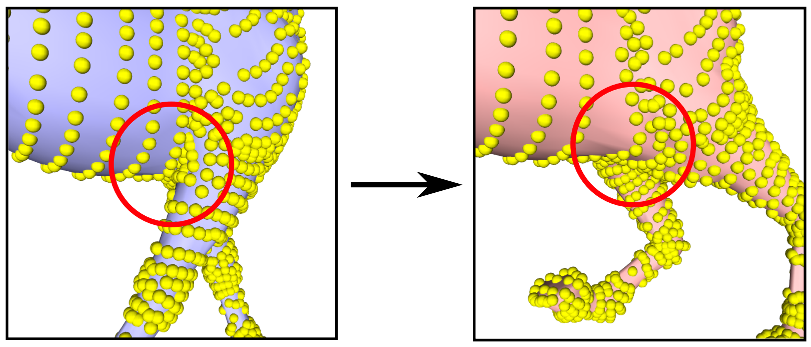
Data Cost Type	**Formula**
Point-to-point	$D_p(p, f_p) = \|f_p(p) - u\|$
Point-to-plane	$D_l(p, f_p) = \left\|(f_p(p) - u) \cdot n_u\right\|$
Hybrid	$D_h(p, f_p) = \begin{cases} D_l & \text{if } u \in \text{boundary} \\ D_p & \text{otherwise} \end{cases}$
Point-and-normal	$D_n(p, f_p) = \sqrt{D_h + \nu \|f_p(n_p) - n_u\|}$

stretches the edge.

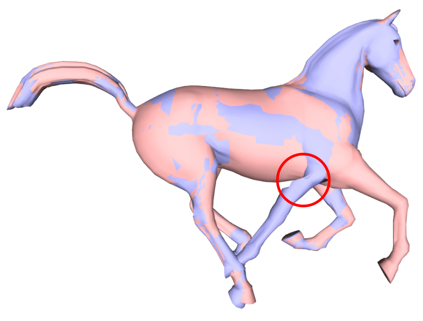
Graph Cuts. To apply graph cuts to minimize the objective in Equation 4.2, we construct an instance where the points $p \in \mathcal{P}$ are the sites, the edges $(p, q) \in E_P$ are neighbors, and the transformations $f \in \mathcal{T}$ are labels. We then solve the resulting multiple-label assignment problem using the α-expansion algorithm, as discussed previously in Section 3.4.

Symmetric Cost Function. In many cases swapping the inputs $\mathcal{P}$ and $\mathcal{U}$ gives different results, because in Equation 4.2 we are only optimizing for an assignment of the *forward* transformations from $\mathcal{P}$ to $\mathcal{U}$. To make maximum use of both shapes, we formulate a *symmetric* cost function, where we simultaneously solve for a consistent assignment of the forward transformations $\mathcal{T}$ to $\mathcal{P}$ and backward transformations $\mathcal{T}' = \{T^{-1} \mid T \in \mathcal{T}\}$ to $\mathcal{U}$. For convenience, we can assign forward transformation labels $f_u \in \mathcal{T}$ to points $u \in \mathcal{U}$, except that we apply the inverse transformation when evaluating the cost functions.

To construct a symmetric graph cuts instance, first we include the vertices and connectivity of both shapes $\mathcal{P}$ and $\mathcal{U}$ to simultaneously solve for the assignment in both directions. Next, we enforce corresponding points under a transformation assignment to have the same label. For example, in Figure 4.3d, we penalize the case where $T_i \neq T_j$. Formally, for each transformation $T \in \mathcal{T}$, we introduce a new edge $e_T = (p, u)$ for all $p \in \mathcal{P}$ and $u \in \mathcal{U}$ such that $u \in \mathcal{U}$ is the closest point to $T(p)$, or $p \in \mathcal{P}$ is the closest point to $T^{-1}(u)$. Let E_T be the set of all such edges. Then the *consistent label term* W is given by

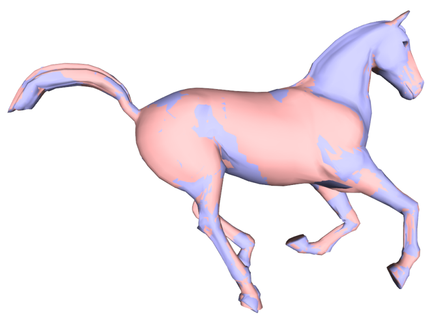


(a) Source vertices (closeup) (b) Target vertices (closeup)



Smooth/Data (7.58/422.85)
Total Cost: 430.43

(c) Non-symmetric solution



Smooth/Data (15.52/320.52)
Total Cost: 336.04

(d) Symmetric solution

Figure 4.4: Comparing non-symmetric and symmetric graph cuts. In this example, the skin joining the thigh and the torso is significantly stretched (a,b). The non-symmetric solution prefers to preserve the edge lengths in this region (c), resulting in a sub-optimal local minimum where the front right leg is aligned incorrectly. However, assigning consistent forwards and backwards transformations gives enough incentive to afford the stretch in the front right leg (d).

$$\underset{\substack{f_p, f_u \in \mathcal{T} \\ \forall\ p \in \mathcal{P}, u \in \mathcal{U}}}{\operatorname{argmin}} \sum_{(p,u) \in E_T} W(p, u, f_p, f_u). \tag{4.4}$$

This term serves to penalize assignments where the transformations disagree between corresponding points. Thus

$$W(p, u, f_p, f_u) = \begin{cases} c_W & \begin{array}{l} f_p \neq f_u \text{ and } u \text{ is closest to } f_p(p) \text{ or,} \\ f_p \neq f_u \text{ and } p \text{ is closest to } f_u(u), \end{array} \\ 0 & \text{otherwise,} \end{cases}$$

where c_W is a constant penalty of an inconsistent label assignment. An example of the benefit of this term is shown in Figure 4.4. Including the consistent label term and solving for labels on both the source and the target, we obtain the symmetric cost function:

$$\begin{aligned} \underset{f_p, f_u \in \mathcal{T}'}{\operatorname{argmin}} \quad & \sum_{p \in \mathcal{P}} D(p, f_p) + \sum_{(p,q) \in E_P} V(p, q, f_p, f_q) \\ & + \sum_{u \in \mathcal{U}} D(u, f_u) + \sum_{(u,v) \in E_U} V(u, v, f_u, f_v) \\ & + \sum_{\substack{(p,u) \in E_T, \\ p \in \mathcal{P} \text{ and } u \in \mathcal{U}}} W(p, u, f_p, f_u). \end{aligned} \tag{4.5}$$

One caveat in using graph cuts is that the smoothness term in Equation 4.3 is not a semimetric or a metric as required to obtain a strong local minimum. However, in practice we obtain good results with the α-expansion algorithm. This may be due to the relatively few number of smoothness constraints that actually violate the metric property. We have observed that typically 0.01% ~ 0.02% of the smoothness constraints (edges in the graph) violate this property in practice. In addition, lower cost solutions have consistently yielded qualitatively better results, which suggests that registration quality can be improved by using a stronger optimization technique.

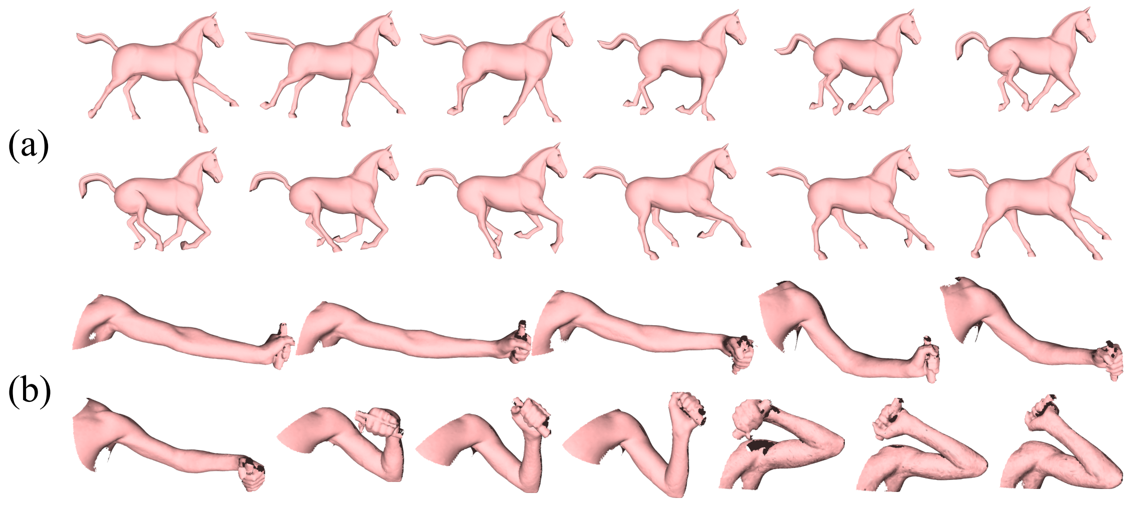


Figure 4.5: Examples used for testing our algorithm. Shown are the twelve poses in the horse dataset (a) and the arm dataset (b).

4.3 Results

4.3.1 Registration

In this section we present results of aligning one synthetic dataset of a galloping horse mesh animation, one real-world dataset of human arm scans, and another real-world dataset of human hand scans [Sumner and Popović, 2004; Allen et al., 2003; Weise et al., 2007]. Some examples used in our experiments are shown in Figure 4.5. To our knowledge, ours is the only algorithm which does not use any prior information about the shape or alignment, and is robust to both significant motion and missing data.

We were successfully able to automatically align most pairs of the 12 galloping horse examples (Figure 4.5a). To measure the quality of the alignment, we use the maximum symmetric Hausdorff distance, which is the maximum distance to the closest point for each vertex in both shapes. Out of a total 66 pairs, we obtained a good registration in 64 trials, where the distance was at most 5.6% of the bounding box. In only 2 examples, some legs were swapped, resulting in a maximum distance of 20% of the bounding box diagonal. This demonstrates that our algorithm can handle a wide range of motion and is particularly robust in this dataset.

Figure 4.6 shows an example of aligning two frames of this animation. Although

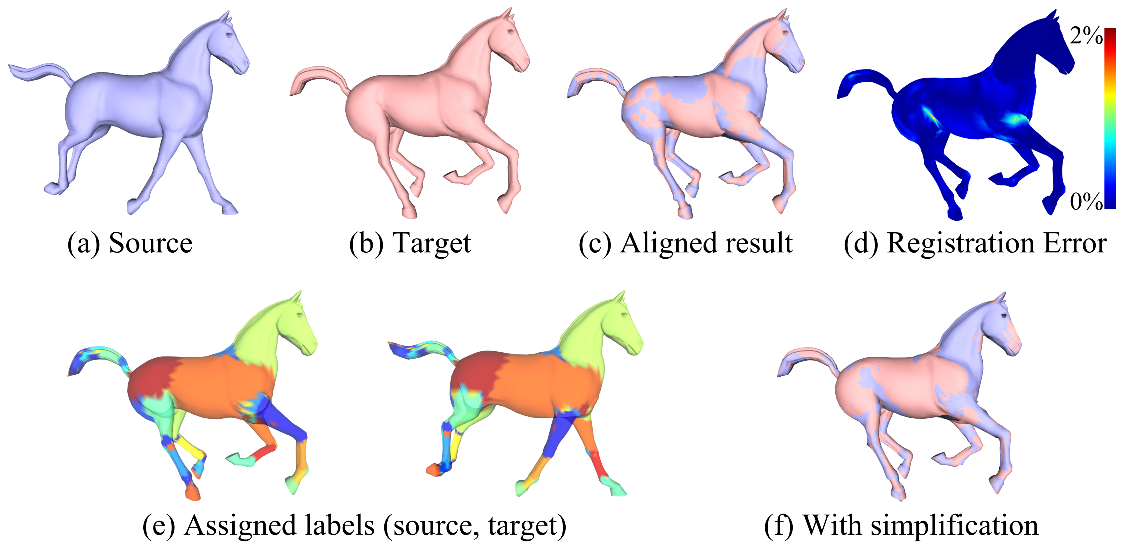
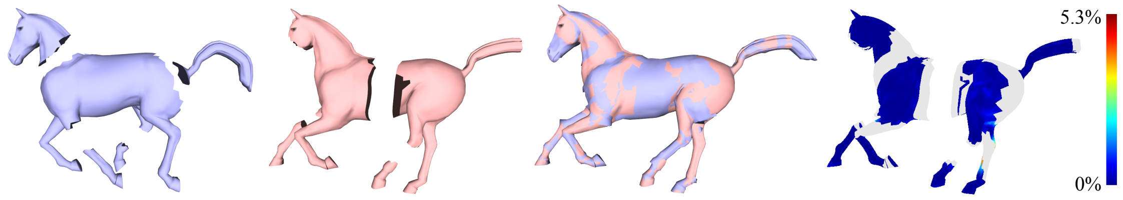


Figure 4.6: Registration results for the synthetic horse dataset. Given a pair of meshes (a) and (b), our algorithm results in a close alignment (c). Notice that all four legs are matched to the correct part. We also obtain a low registration error in (d), which shows the per-vertex distance to the target shape as a percentage of the bounding box diagonal. In addition, our method gives a meaningful segmentation of the model into rigid parts that is consistent in both shapes (e). The last image (g) shows the result of aligning a 50% simplified version of the same pair. This demonstrates that our method performs well independent of the sampling and parameterization of the shapes.

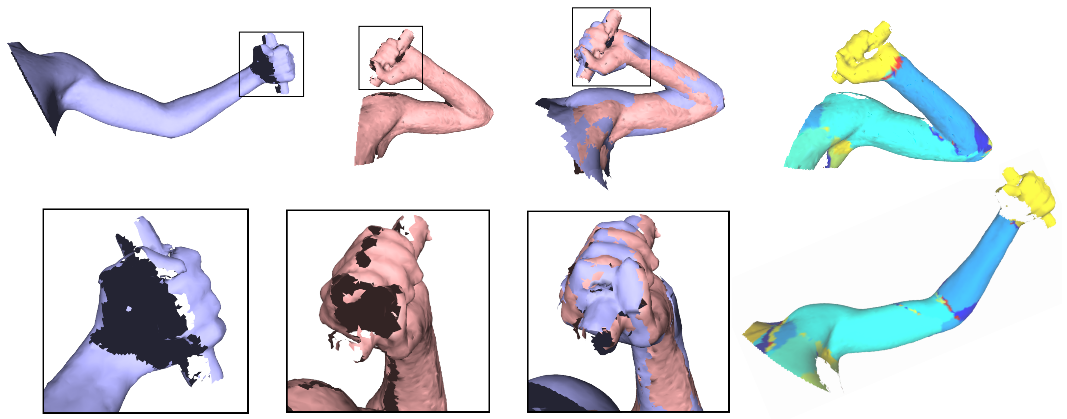
our method has no input other than just the shapes, we obtain an accurate alignment with a registration error within 2% of the entire scene size (length of the shape's bounding box diagonal). All four legs in the source mesh have aligned to the correct leg in the target mesh, demonstrating that our optimization can map similar parts correctly. We see that the assigned transformations naturally give a high quality segmention of the mesh into rigid components (Figure 4.6e), which can be used to automatically create a skeleton. In addition, the segmentation is consistent between both source and target shapes, as we can see in the figure where corresponding parts are colored with the same label. Finally, we obtain a good registration even when simplifing the meshes to 50%, demonstrating that our method is independent of the specific parameterization of the shape.

To test our algorithm on examples with significant missing data, we have manually removed parts of both the source and target meshes (Figure 4.7-abc). The holes were placed in different locations so that the alignment result would complete the entire shape. Even though the holes completely disconnect the meshes into several fragments, we obtain a good



(a) Source (b) Target (c) Alignment result (d) Registration Error

Figure 4.7: Registration with significant missing data. Even after manually removing parts in both source and target, our method is able to align the meshes well.



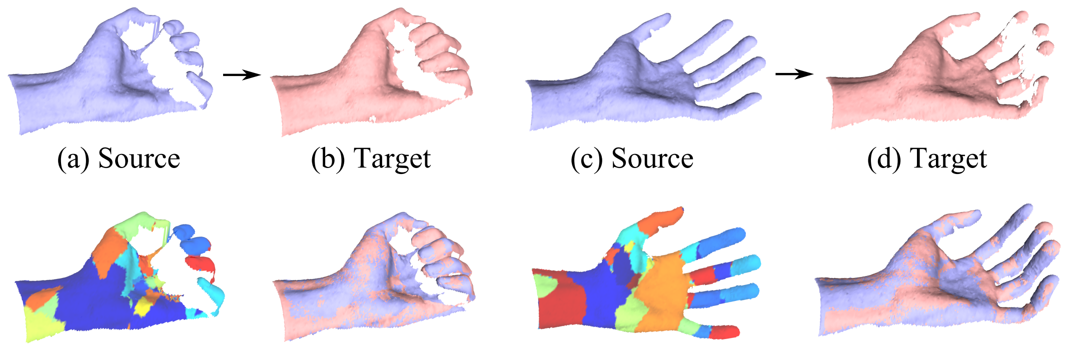
(a) Source (b) Target (c) Aligned result (d) Assigned labels

Figure 4.8: Registration for an arm dataset pair. The source mesh (a) is aligned to the target mesh (b). The hand region is missing a significant amount of data in both meshes, but after alignment the surface of the hand is completed nicely (c). The assigned labels are shown in (d) for the source (bottom) and the target (top), and corresponding parts have the same label assignment. Also, the segmentation naturally corresponds to the different parts of the arm.

alignment result. This shows that our algorithm is robust to missing data and that it can be used for automatic shape completion using a set of incomplete examples.

For the arm scan dataset, our algorithm successfully aligned all pairs of the 12 examples (Figure 4.5b). The registration performed well even when there was significant missing data in both the forearm and the hand. In the example shown in Figure 4.8, notice that the aligned result has nicely completed the thumb and index finger region of the hand area (c). Just like the horse example, the assigned transformations segment the shape into meaningful rigid parts (d), which can be used to automatically create a skeleton.

Finally our algorithm was able to align many examples for the hand grasping dataset. This dataset is particularly challenging because of the occlusion in the fingers. Figure 4.9 shows



(a) Source (b) Target (c) Source (d) Target

(e) Assigned labels (f) Aligned result (g) Assigned labels (h) Aligned result

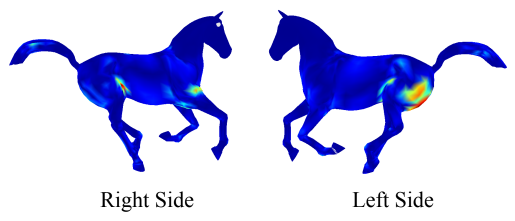
Figure 4.9: Registration results for hand dataset examples. Although the hand is missing a significant amount of data in the fingers (b,d), our method is able to successfully align all four fingers (f,h). Notice that the segmentation produced in (e,g) roughly corresponds to the actual segments in the fingers of a hand.

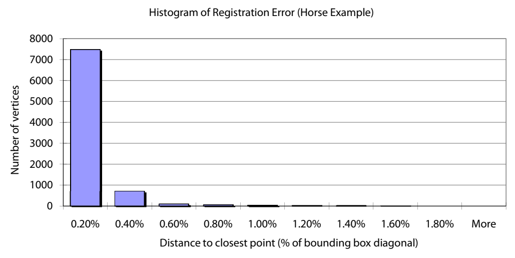
two examples of aligning a pair of these scans. Despite significant missing data and movement in the fingers, our method successfully aligns all four fingers to the correct position. As a result, the holes in the fingers are filled in using data from the source shape.

4.3.2 Registration Error Analysis

To numerically quantify the registration error, we measure the registration error at each vertex by computing the distance to the closest point on the other shape. Note that we do not measure the distance when the closest point is on the boundary of the shape. These vertices are indicated by a gray color in our visualizations. In Figure 4.10, we show a visualization of the registration error for the registration examples that we have shown, and next to this we plot a histogram of the distance for all vertices. Looking at the histogram, we see that most vertices have a very low error, typically below 0.5% of the bounding box diagonal length. Only a few vertices have higher error.

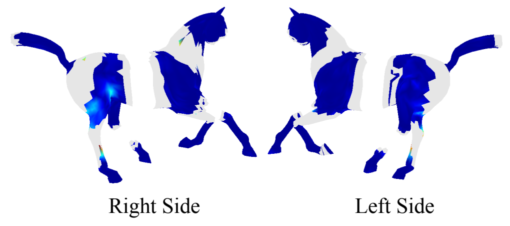
In Figure 4.11, we quantify the performance of our algorithm on an entire dataset. For each pair, we compute the maximum symmetric Hausdorff distance, which is the maximum distance to the closest point on the other shape taken over all vertices in the source and in the target. This is purely an upper bound on the distance between two shapes, and it is a very conservative estimate of the registration error. The figure shows a histogram of this distance

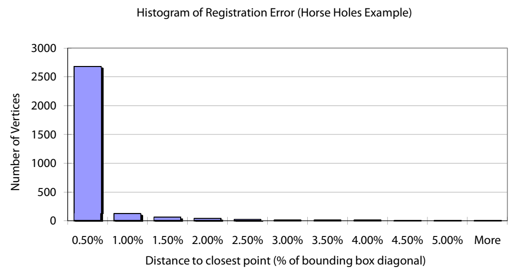




Horse Dataset Example

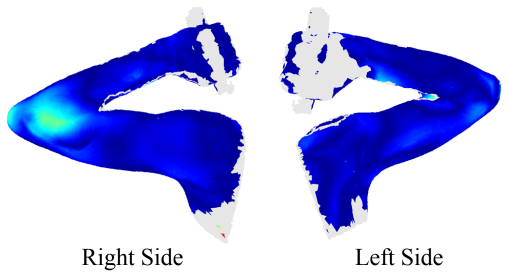
Histogram of Registration Error

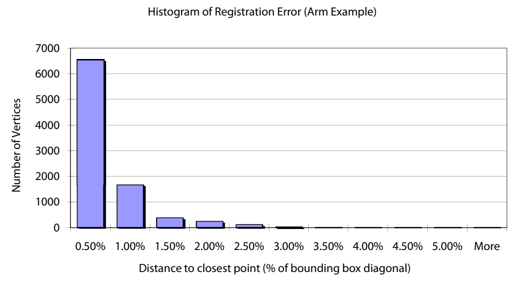




Horse Dataset Example with missing data

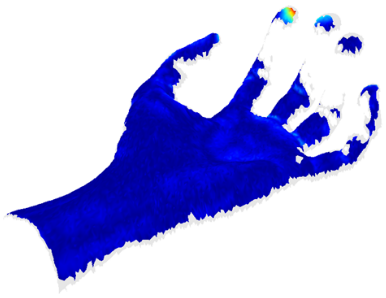
Histogram of Registration Error

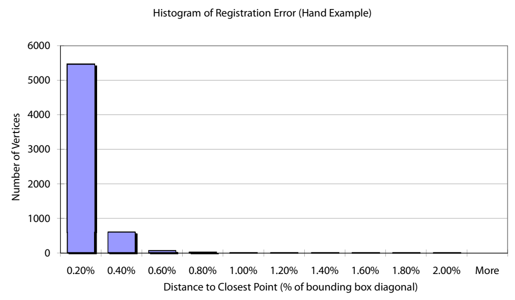




Arm Dataset Example

Histogram of Registration Error

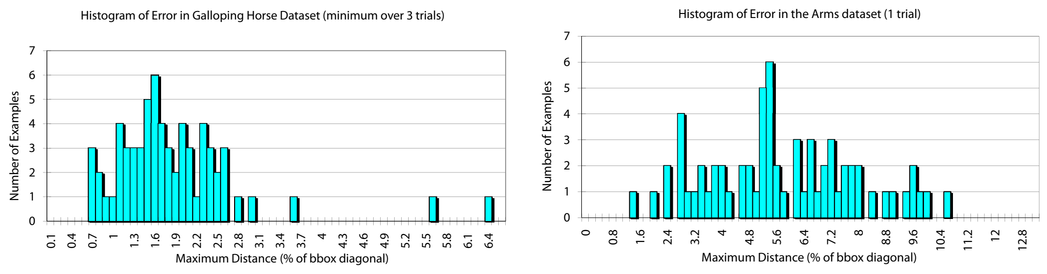



Histogram of Registration Error (Hand Example)
Number of Vertices
6000
5000
4000
3000
2000
1000
0
0.20%
0.40%
0.60%
0.80%
1.00%
1.20%
1.40%
1.60%
1.80%
2.00%
More
Distance to Closest Point (% of bounding box diagonal)

Hand Dataset Example

Histogram of Registration Error

Figure 4.10: Detailed error analysis of examples in each dataset. On the left, we show visualizations of the registration error. On the right, we plot the histogram of the registration error.



Horse Dataset Error Distribution Arm Dataset Error Distribution

Figure 4.11: Histogram of maximum error for all pairs in the horse and arm datasets.

over all pairs of the horse and arm datasets. For the horse dataset we performed 3 trials per pair and took the minimum distance, and for the arm dataset we performed 1 trial per pair. For the horse dataset shown on the left, we see that the registrations were very successful because only a few examples have maximum Hausdorff distance above 3% of the bounding box diagonal. In the arm dataset, the distance is higher because the examples have more non-rigid bulging and stretching deformations. These deformations are not modeled well by our discrete labeling of transformations. Nevertheless, the maximum distance is below 10% for almost all examples in this dataset.

4.3.3 Limitations

A key component of our algorithm is the motion sampling step. The discrete labeling alignment depends on this step to produce the transformations that correctly describe the movement of the shape. Therefore, we expect our algorithm to fail when the motions are not correctly sampled. This can happen when there are few distinct features on the shapes, or when the shapes exhibit extreme non-rigid deformation. To investigate the limits of our algorithm, we tested our algorithm with boxes with missing corners, featureless spheres, and a flapping flag.

The results of the alignment are shown in Figure 4.12. Even though are little or no features in the box and sphere examples, our algorithm is able to sample enough transformations and assign the one that matches the shapes exactly. One critical step here is the ICP verification that refines the sampled transformations. As an example, Figure 4.13 shows a

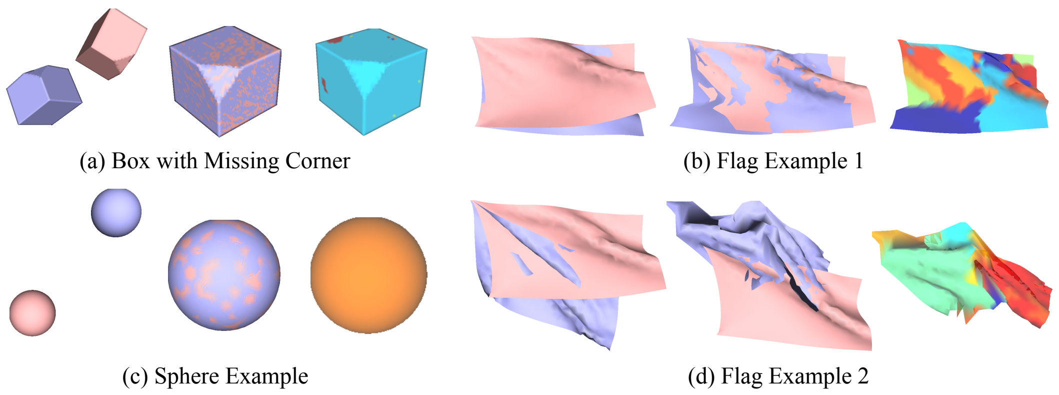


Figure 4.12: Special cases for our algorithm. For each example, we show the source and target on the left, the registration result in the middle, and the assigned labels on the right.

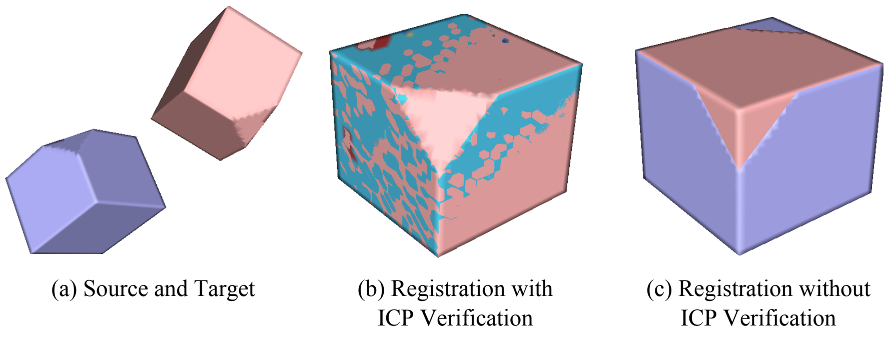

(a) Source and Target
(b) Registration with ICP Verification
(c) Registration without ICP Verification

Figure 4.13: The alignment of boxes with missing corners fails when the ICP verification is not performed.

case where the box example fails without the ICP step. Here, the “correct” transformation that aligns the missing corner was not refined using ICP to produce an exact alignment. Instead, an “incorrect” transformation that aligned the surfaces better (but not the corner) was chosen instead.

For the flapping flag example, we see that our algorithm works reasonably for a small deformation where the flag shape does not change much. However, for extreme deformations, we see that the alignment fails completely. This is because there are no features that are common in the two shapes, preventing our motion sampling step from finding the correct transformations to align the meshes.

4.3.4 Parameters & Performance

For the feature matching, the spin images were quite discriminative, so we limited the number of matching features for each vertex to 5-7 vertices. For the mean-shift clustering, since we take the cluster modes to be the estimated transformations, we set the mean-shift bandwidth to a small $h = 0.1$ in order to prevent over-smoothing of the transformations. We use a fraction of the bounding box diagonal as parameter values because it naturally relates to the data and smoothness costs which measure distances between points. We set the error of an inconsistent label assignment c_W to 100 times the bounding box diagonal. Also, we set the closest point distance threshold to 0.5% of the bounding box diagonal for finding matching regions in the pruning step and for determining corresponding points in Equation 4.4. Finally, the data cost that we use for each dataset is summarized in the rightmost column of Table 4.2. We set v to be 2% of the bounding box diagonal for the point-and-normal metric. To control the relative influence of the data and smoothness terms, we multiply each with a constant weight c_D and c_V, respectively. In our experiments, we used $c_D = 1$ for all datasets, $c_V = 5$ for the arm and hand datasets, and $c_V = 10$ for the horse dataset.

The statistics and running time of the experiments are summarized in Table 4.2. The most time-consuming portion of our algorithm is the graph cuts optimization, which depends on the number of sites as well as the number of labels. Since our graph cuts instance optimizes

Table 4.2: Averaged performance and timing statistics for a typical subset of our experiments. The running time statistics were gathered from testing our implementation on a single core of a dual core 2.4 GHz Intel processor.

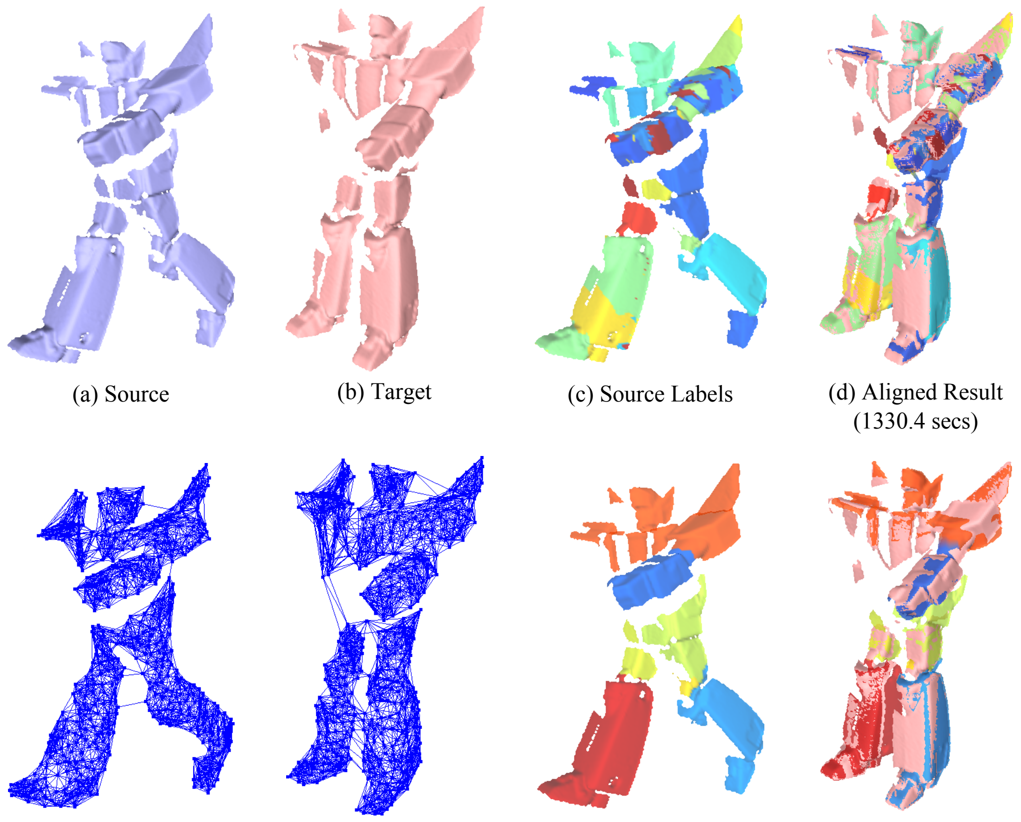
Dataset	**Vertices**	**Samples**	**Labels**	**Match**	**Cluster**	**Verify**	**Graph Cuts**	**Data Cost**
Horse	8431	4339	1500	2.1 min	3.0 sec	(skip) 1.6 sec	1.1 hr	D_n
Arm	11865	6094	1000	55.0 sec	0.9 sec	12.4 min	1.2 hr	D_p
Hand (Front)	8339	2945	1500	14.5 sec	0.7 sec	7.4 min	1.2 hr	D_l
Hand (Back)	6773	2669	1500	17.3 sec	0.9 sec	9.4 min	1.6 hr	D_l

the assignment of all vertices in both source and target meshes, we simplify the meshes using the quadric error metric technique [Garland and Heckbert, 1997] to reduce the running time.

4.4 Improving Performance by Subsampling

The basic algorithm discussed so far solved for an optimal assignment on all vertices of the source and target shape. With range scans that typically have several thousand, or even several tens of thousands of points, this method is too slow to process an entire range scan sequence with many frames. Therefore, we simplify the method by assigning the sampled transformations only to a small subset of the points in each frame.

Specifically, we sample a small number of points (e.g. 500–1000) in the source and target frames $\mathcal{P}$ and $\mathcal{U}$ using best-candidate sampling [Mitchell, 1991]. To replace the connectivity of the vertices given by the triangle mesh, we construct a k-nearest neighbor graph on the sampled points, where k is typically 15. The motion sampling step (Section 4.2.1) is performed exactly in the same way as before to obtain a set of transformations from $\mathcal{P}$ to $\mathcal{U}$. For the symmetric graph cut instance, two components need to be adapted to work with this reduced set of samples. First, to compute the data term $D(p, f_p)$ and $D(u, f_u)$ for each sample $p \in \mathcal{P}$ ($u \in \mathcal{U}$) and label f_p (f_u for samples in $u \in \mathcal{U}$), we transform the point p according to the transformation f_p and take the point-to-point distance to the closest point among *all points* of frame $\mathcal{U}$ (similarly for points $u \in \mathcal{U}$). Notice that we do not take the distance to the closest *sampled point* in the other shape. Second, for edges between the source and target frames that enforce a consistent symmetric labeling, for each sample point p and label f_p, we transform p according to f_p and create an edge from p to the closest *sample point* $u \in \mathcal{U}$ (vice versa for all



(a) Source (b) Target (c) Source Labels (d) Aligned Result (1330.4 secs)

(e) Source Graph (f) Target Graph (g) Source Labels (h) Aligned Result (87.3 secs)

Figure 4.14: Comparison between the original and simplified transformation assignment methods. The top row (a,b,c,d) shows the result with the original method, while the bottom row (e,f,g,h) shows the result using a graph of 500 nodes on each frame. With the same parameters, the simplified method produces a very similar alignment result in a fraction of the original time.

u and f_u). Then, the smoothness terms $V(p,q,f_p,f_q)$ and $W(p,u,f_p,f_u)$ is specified the same way as Equations (4.3) and (4.4). Finally, once we solve the labeling problem on the sampled points, we can propagate these labels to all remaining points of each frame, by finding the closest sample for that point and assigning its label. A comparison of the original method with the new method is given in Figure 4.14. We can see that we get a good alignment in both cases, while obtaining a significant speedup. Also, the use of the graph improves the connectivity between parts that may be disconnected in the original mesh. In our experiments with this example, we saw that this improved the robustness of the method by preventing the legs of the robot from being swapped.

4.5 Discussion and Future Work

In this section we discuss the limitations of our method and point out several avenues for future work.

Motion Sampling. The quality of our registration result depends on how accurately we sample the transformations. A good sampling should have all the necessary transformations to align each part, while accurately narrowing down on a concise subset of transformations. In some horse examples, small parts such as hooves or legs were misaligned, because the correct transformation was not in the set of estimated transformations $\mathcal{T}$. In these cases, there were too few samples to extract these transformations accurately. In our experiments, we found that when the feature matching is noisy, the clusters found by our algorithm are spread out more uniformly in the transformation space. In this case, the clustering acts somewhat like a sampling strategy rather than finding densely clustered rigid components. It would be interesting to investigate other sampling strategies for the motion sampling step, such as adaptive sampling. Furthermore, it may be beneficial to adaptively sample the motion *during* the optimization to automatically refine the registration result.

Another interesting avenue is to search for better ways to represent the range of motions in a moving object. Automatically learning the space of motions using dimensionality reduction techniques (e.g. PCA, MDS, LLE, ISOMAP, LSML [Dollar et al., 2007]) may lead to an improved representation and also a more efficient optimization technique. When there is non-rigid motion in the shape, there is a continuously varying range of motions that are needed for alignment. Since our clustering step attempts to extract only rigid components, our method has problems if the shapes are strongly non-rigid. A good extension, as also pointed out by Mitra et al. [2006], is to incorporate a method for extracting a continuous set of motions for non-rigid examples.

Another issue is that the assignment of transformations can disconnect or imperfectly match the boundaries between rigid components. In these cases, our method can be used as an initialization for an additional refinement step (such as non-rigid ICP) to obtain a smoother and more precise registration.

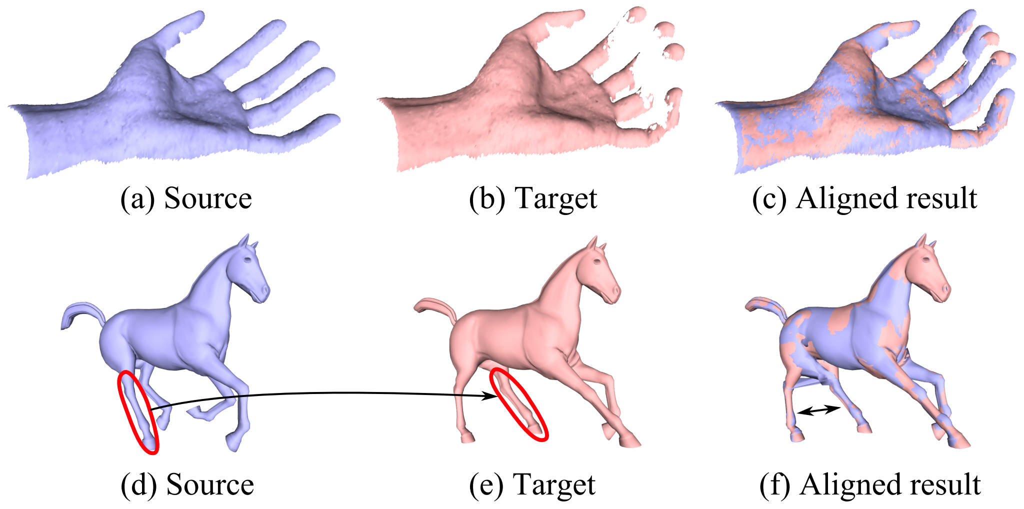


Figure 4.15: Typical errors in the registration. In the hand example (a,b,c), the missing data causes the fingers to become stretched, slightly disconnected, or misaligned. In the horse example (d,e,f), the rear legs are swapped.

Finally, an open problem is to characterize exactly how much data is required for an accurate alignment. For example, the alignment in Figure 4.15-abc is less accurate. The main difficulty with missing data in our method is that there is not enough data to estimate the correct motion or to guide the optimization. It would be interesting to investigate what are the fundamental limitations for resolving missing data.

Optimization. In general, the optimal labeling problem is known to be NP-hard [Boykov et al., 2001]. However, utilizing recent methods such as sequential tree-reweighted message passing (TRW-S) [Kolmogorov, 2006] or Log-cut [Lempitsky et al., 2007] may improve both the quality and efficiency of the registration. Also, applying a coarse-to-fine optimization technique may help as well.

In our experiments, we found that often there is a trade-off between minimizing the smoothness and the data costs. If the smoothness weight is too high, then the optimization can prefer a badly aligned solution in favor of preserving the edge lengths of the mesh (e.g. Figure 4.4c). On the other hand, if the data weight is too high, the optimization may choose to break the mesh and map parts incorrectly. For example, in Figure 4.15-def, the rear right leg of the source is most similar to the rear left leg of the target. In this case, the low data cost managed to compensate for the penalty of the smoothness term when the rear legs were swapped. As an area of future work, it would be useful to determine the weights automatically. Another interesting idea is to include a “dummy label” to explicitly label vertices with no correspondence due to missing data. In our preliminary experiments, it was difficult to define an appropriate penalty of assigning such a label. Investigating whether such a label could be incorporated well in the optimization would be an interesting avenue for future work.

Finally, since our method is currently limited to a pair of shapes, extending it to simultaneously align more than two shapes may help to resolve more missing data.

4.6 Conclusion

We have presented an automatic method for aligning a pair of articulated shapes in the presence of significant motion and missing data. We formulate the registration problem as assigning rigid transformations to each vertex of the shape. Our algorithm first determines a finite set of possible motions of the shape by sampling the motion. Then, it uses graph cuts to optimize a cost function, which measures the quality of an assignment for aligning the shapes and preserving the consistency of the transformed mesh. Furthermore, we develop a symmetric cost function for simultaneously obtaining a consistent assignment for both source to target and target to source. Our experimental results show that despite no prior knowledge of a template, user-placed markers, segmentation, or the skeletal structure, the algorithm is able to find good alignments between different poses of the shape. We also obtain a natural segmentation of the shape into rigid parts, given by contiguous regions with the same label.

4.7 Acknowledgments

We wish to thank Bob Sumner for the galloping horse animation, Brett Allen for the body scans, and Thibaut Weise for the hand scan dataset. Additional thanks to Gilles Debunne for providing the libQGLViewer library, David M. Mount and Sunil Arya for providing the ANN library, and Yuri Boykov, Olga Veksler, Ramin Zabih for providing an implementation of their graph cuts algorithm.

The material in this chapter is, in part, a reproduction of published material: Will Chang and Matthias Zwicker, “Automatic Registration for Articulated Shapes,” Computer Graphics Forum (Proceedings of SGP), Vol. 27, No. 5, July 2008. The dissertation author was the primary investigator and author of this paper.

5

Range Scan Registration Using Reduced Deformable Models

IN this chapter, we propose to align surfaces by modeling the motion of the surface using an reduced, articulated deformation model. The two main components of such models are: (1) the transformations, which describe the movement of each part, and (2) the influence weights, which indicate how strongly each transformation influences each point on the surface. In particular, we choose the linear blend skinning (LBS) model due to its simplicity and ease of optimization. In order to align range scans using this model, we will solve for its deformation parameters, i.e. the transformations for each part and influence weights for each point on the surface.

Compared to the previous chapter that focused only on registration, the algorithm in this chapter explicitly models the articulated motion, for example, modeling joints and smoothly blending the boundary between parts. This allows an animator to easily target the input range scans into new poses for creating an animation.

5.1 Contributions

We formulate the registration problem using an objective function that enforces close alignment of the 3D data. We develop a novel algorithm to solve for the deformation

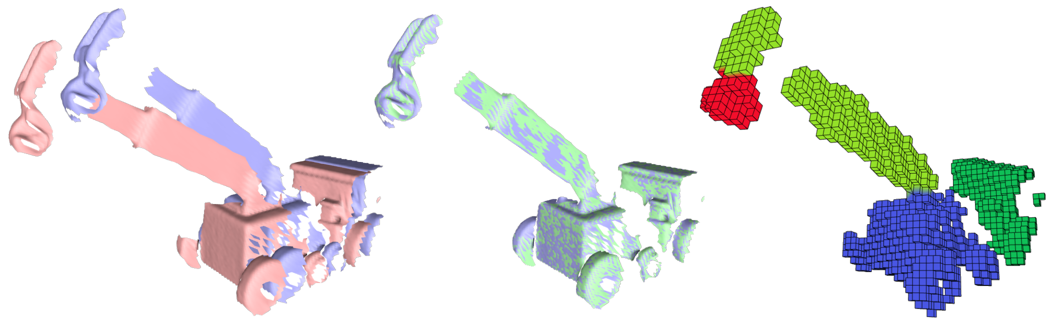


Figure 5.1: Given a pair of range scans, our technique automatically finds the parameters of a linear blend skinning (LBS) model to register the scans accurately.

parameters in alternating fashion. Since range scans usually have tens of thousands of points, determining the influence weights on every point would significantly slow down the optimization. In addition, range scans have many occlusions and holes, resulting in disconnected surface geometry. To address these issues, we propose a novel representation of the influence weights on a grid enclosing the range scan surface. Not only does this reduce the number of variables, but it also provides a regular structure to define the weight function that is decoupled from the incomplete surface representation. Finally, to obtain a coherent deformation and to prevent neighboring bones from moving apart, we formulate joint constraints directly on this grid and incorporate it into the objective. Also, we propose an efficient solution for solving the weights via a combination of discrete and continuous optimization. In summary, the contributions of our work are:

- A novel method to align scans by solving for articulated deformation parameters,
- Formulation of influence weights and joint constraints using a regular grid,
- Application of discrete optimization for efficiently solving the weights,
- Solving for continuous weights on range scans.

Using our algorithm, we demonstrate that we can efficiently and accurately register pairs of range scans despite significant motion and missing data. In contrast to other approaches, our method does not require user specified markers, a template, nor a manual segmentation of the surface geometry.

5.2 Problem Formulation

We assume that the scanned object is articulated, which means that we expect the movement of the surface to be piecewise rigid. Therefore, we use the linear blend skinning model (Section 3.5.1) to describe the motion of the observed surface. This means that we align the pair of scans by solving for a set of transformations and the set of weights (for each point on the scan) that associate the transformations with the surface. Because our input data consists of incomplete range scans without correspondence information, we propose representing the weight functions on a regular grid enclosing the scanned geometry. Let us denote a regular grid by specifying values on a set of grid points

$$G = \{(\mathbf{p}_c, \mathbf{v}_c)\}_{c=1}^{m},$$

where c is the index of the grid point, $\mathbf{p}_c$ is the position of grid point c, $\mathbf{v}_c$ is the vector of weights at c, and m is the number of grid points. Alternatively, we can think of the grid as a collection of grid *cells*, denoted in capital letters $C \in G$, where each grid cell contains eight grid points, and adjacent grid cells share four grid points. For the 2D case, this situation is illustrated in Figure 5.2.

To compute the value of the weight function $w_j(\mathbf{x})$ on the surface geometry, we find the grid cell C that contains $\mathbf{x}$ and perform trilinear interpolation of the values $\mathbf{v}_c$ defined at the cell's grid points $c \in C$. This allows us to specify the weight function everywhere within the grid, and leads to a compact representation of the vertex weight function that is well-defined even in the presence of missing data. In addition, we adapt the grid to the scanned geometry. This is accomplished by using only those grid cells that contain points from the range scan. This reduces the numbers of variables needed to solve, and localizes the deformation to the surface geometry. We can also control the smoothness of the weight function by adjusting the resolution of the grid.

Registration. Our goal is to find the deformation parameters that best align the input range scans. The parameters of our model are the set of transformations $\mathcal{T} = \{T_j\}$ and the weight

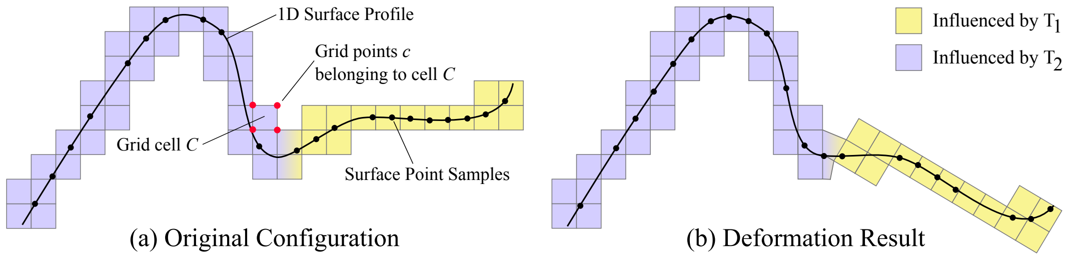


Figure 5.2: We represent the motion of a range scan using LBS, which consists of a fixed number of transformations and their associated influence weights. We decouple the weights from the incomplete surface representation by defining them on a grid enclosing the range scan data.

functions $\mathcal{W} = \{\mathbf{w_x}\}$. We formulate an objective function $E(\mathcal{T}, \mathcal{W})$ that describes the quality of a registration as a function of these unknown parameters:

$$\underset{\mathcal{T},\mathcal{W}}{\operatorname{argmin}} \;\; \alpha E_{\text{dist}}(\mathcal{T}, \mathcal{W}) + \beta E_{\text{joint}}(\mathcal{T}, \mathcal{W}) + \;\mu E_{\text{smooth}}(\mathcal{W}) + \nu E_{\text{norm}}(\mathcal{W}). \tag{5.1}$$

The first term E_{dist} measures the accuracy of the alignment by evaluating the distance between the registered surfaces, as discussed in the beginning of Section 5.3.1. The second term E_{joint} enforces a joint constraint that encourages transformations to act intuitively at joints. We discuss this term in more detail in Section 5.3.1. The final two terms E_{smooth} and E_{norm} are regularization constraints on the weight functions, which we discuss in Section 5.3.2. We solve this problem in an iterative fashion, where we update the model parameters $\mathcal{T}$ and $\mathcal{W}$ until convergence. At each iteration, we perform an alternating optimization similar to the EM algorithm [Dempster et al., 1977] where we

1. (T-step) fix the current weights $\mathcal{W}$ to be constant and solve for the rigid transformations $\mathcal{T}$, then
2. (W-step) using the rigid transformations $\mathcal{T}$ from the T-step, solve for the weight functions $\mathcal{W}$ on the grid.

The entire optimization procedure is illustrated in Figure 5.3 and Algorithm 5.1. In the following sections, we discuss the T-step and the W-step in further detail.

Algorithm 5.1: REGISTRATION OVERVIEW

```
1  begin
2      Generate grid & initial segmentation (Section 5.3.3);
3      Generate initial correspondences using spin image features and RANSAC
       (Section 5.3.3);
4      repeat
5          Solve for the transformations $\mathcal{T}$ (Equation 5.5);
6          Update correspondences;
7          Solve for discrete weights $\mathcal{W}$ (Equation 5.11);
8          Update correspondences;
9      until Stopping criteria are met ;
10     Solve for continuous weights $\mathcal{W}$ (Equation 5.8);
11 end
```

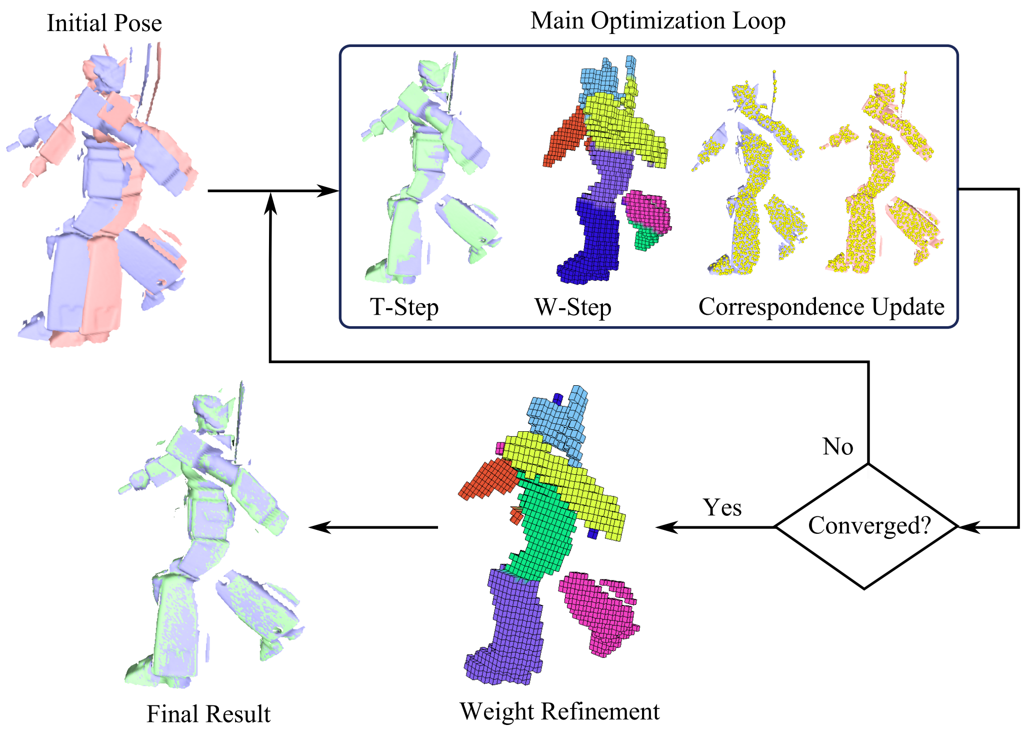
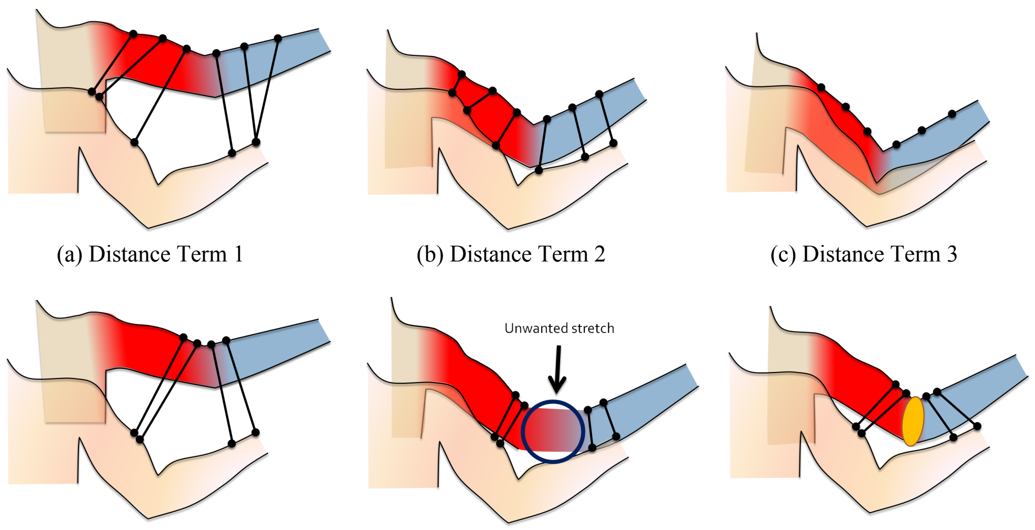


Figure 5.3: Overview of our optimization algorithm. The main optimization loop alternates between solving for the transformations (T-step), solving for the weights (W-step), and updating correspondences. We obtain the final registration result after a weight refinement step that smooths unnatural deformations.



(a) Distance Term 1 (b) Distance Term 2 (c) Distance Term 3

(d) Distance Constraints (e) Without Joint Constraint (f) With Joint Constraint

Figure 5.4: Overview of the distance and joint terms in the T-step optimization. The top row shows the role of the distance term, which is to align the shapes by minimizing the distance between the correspondences (shown as the black points and lines). The bottom row shows the role of the joint term, preserves the structure of the shape by constraining a joint region.

5.3 Optimization Algorithm

5.3.1 Solving the T-step

The goal of the T-step is to obtain the rigid transformations $\mathcal{T}$ that minimize the objective. In this step the weights $\mathcal{W}$ are fixed, therefore the optimization includes only the distance term E_{dist} and the joint constraint E_{joint} term. To give an overview of these two terms, we illustrate the each term with an example in Figure 5.4. The distance term tries to align the shapes as closely as possible with the help of correspondences between the shapes. Then, the joint term tries to preserve the structure of the shape by constraining neighboring transformations to move the joint region to the same location. We now discuss each term in more detail.

Distance Term. The distance term E_{dist} measures how close the deformation aligns the two shapes together. This term uses point correspondences between the source and target shapes. Suppose that we are registering a pair of range scans $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{P}$ is the source and $\mathcal{Q}$ is the target. First, we generate a uniform sampling of points $\mathbf{x}$ on $\mathcal{P}$. For each of these points, we find a corresponding point $\mathbf{y}$ on $\mathcal{Q}$. Denoting the set $\mathcal{U} = \{(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in \mathcal{P}, \mathbf{y} \in \mathcal{Q}\}$ as the entire set of correspondences from $\mathcal{P}$ to $\mathcal{Q}$, the distance term E_{dist} is given by

$$E_{\text{dist}}(\mathcal{T},\mathcal{W}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{U}} \left[\left(\sum_j w_{\mathbf{x}j} T_j(\mathbf{x}) - \mathbf{y} \right) \cdot \vec{\mathbf{n}}_{\mathbf{y}} \right]^2, \tag{5.2}$$

where $\vec{\mathbf{n}}_{\mathbf{y}}$ is the surface normal associated with $\mathbf{y} \in \mathcal{Q}$. This is exactly the point-to-plane distance discussed in Section 3.1.2, which we favor because it allows the surface to slide easily into the right location. Initially, the corresponding points $\mathbf{y} \in \mathcal{Q}$ are found using feature descriptors, which we describe in more detail in Section 5.3.3). However, during the optimization, we switch to the closest-point strategy, where we find the point $\mathbf{y} \in \mathcal{Q}$ that is closest to the currently deformed position $D(\mathbf{x})$.

To improve the robustness of this term, we use a modified distance measure to compute the closest point [Johnson, 1997]:

$$d(\mathbf{x},\mathbf{y},\vec{\mathbf{n}}_{\mathbf{x}},\vec{\mathbf{n}}_{\mathbf{y}}) = \sqrt{\|\mathbf{x}-\mathbf{y}\|^2 + \omega_{\vec{\mathbf{n}}} \|\vec{\mathbf{n}}_{\mathbf{x}} - \vec{\mathbf{n}}_{\mathbf{y}}\|^2}, \tag{5.3}$$

where $\vec{\mathbf{n}}_{\mathbf{x}}$ is the surface normal of $\mathbf{x}$, $\vec{\mathbf{n}}_{\mathbf{y}}$ is the surface normal of $\mathbf{y}$, and $\omega_{\vec{\mathbf{n}}}$ is a weighting constant to control the relative influence of the normal distance. In our experiments, we set $\omega_{\vec{\mathbf{n}}} = 2s$, where s is the range scan grid spacing. In addition, we consider $\mathbf{y}$ as invalid when the $\mathbf{y}$ is on the boundary of $\mathcal{Q}$, the distance between the points exceeds a threshold, or the angle between the normals exceeds a threshold [Pekelny and Gotsman, 2008]. We then remove these correspondence entirely from E_{dist}. The values of these thresholds are discussed further in Section 5.4.

Joint Constraints. A joint is a region that is influenced by more than one bone, and we naturally expect that the bones always meet at these joints. Therefore, to solve for natural bone transformations, we employ the joint constraint in our optimization to keep the shape intact and prevent bones from sliding away from each other (Figure 5.4 and 5.5).

Specifically, this constraint enforces that the joint region is transformed to the same location by all bones that influence it. In order to determine which points in space are part

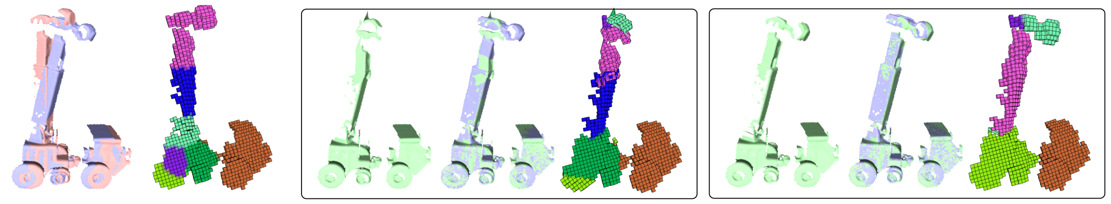

(a) Initial Pose and Labeling (b) Without Joint Constraint (c) With Joint Constraint

Figure 5.5: The joint constraint helps to overcome undesired local minima in the optimization by preserving the joint location between neighboring transformations. The initial alignment (a) with labeling (c) does not register correctly when the joint constraint is removed (b). By adding the joint constraint, however, we converge to the correct result (d).

of a joint, we use the product of the weight functions. For example, the joint region for a pair of bones T_i and T_j is given by the points $\mathbf{x} \in \mathbb{R}^3$ where $w_i(\mathbf{x})w_j(\mathbf{x}) > 0$. For every such point, we would like to enforce that $T_i(\mathbf{x}) = T_j(\mathbf{x})$. Thus, we measure the total misalignment for each potential joint between bone i and j as:

$$E_{\text{joint}}(\mathcal{T}) \;=\; \sum_{i,j} \sum_{C \in G} \tau_{C_{ij}} \int_{\mathbf{x} \in C} w_i(\mathbf{x})\, w_j(\mathbf{x})\, \| T_i(\mathbf{x}) - T_j(\mathbf{x}) \|^2 d\mathbf{x}, \tag{5.4}$$

where $\tau_{C_{ij}} = \left(\int_{\mathbf{x} \in C} w_i(\mathbf{x})\, w_j(\mathbf{x})\, d\mathbf{x}\right)^{-1}$ is a normalization constant for each grid cell $C \in G$ and each pair of transformations T_i, T_j. This term is similar to the deformation energies and regularization terms proposed elsewhere [Botsch et al., 2007; Sumner et al., 2007], which simply differ in the choice of weight functions. While other techniques prescribe the weight functions (e.g. [Li et al., 2008]), our formulation allows joint constraints for arbitrary weight functions. This is especially useful since we allow the weight function to change each time the weights are optimized in the W-step.

To compute the joint constraint term, we pull the unknown elements of the transformations out of the integral, since they are constants with respect to $\mathbf{x}$. We then analytically evaluate the integrals of the weight functions in each cell in closed form, and these form coefficients of a quadratic function of the unknowns T_i.

Optimization. Our final goal in the T-step is to minimize the objective consisting of the distance and the joint terms,

$$\underset{\mathcal{T}}{\operatorname{argmin}}\, \alpha\, E_{\text{dist}}(\mathcal{T}, \mathcal{W}) + \beta\, E_{\text{joint}}(\mathcal{T}, \mathcal{W}), \tag{5.5}$$

where α and β are coefficients to weight each term. Since we require the transformations $\mathcal{T}$ to be rigid, this becomes a non-linear optimization problem. We solve this problem iteratively using the Gauss-Newton method (see Chapter 3.1.4).

5.3.2 Solving the W-step

In the W-step, we keep the transformations $\mathcal{T}$ fixed and solve for the weight functions $\mathcal{W}$. Since the weight function is defined on a regular grid and interpolated in between the grid points, we just need to solve for the weight values at each grid point. To provide regularization, the W-step includes the smoothness term E_{smooth} and the normalization term E_{norm}. However, we leave out the joint constraint term E_{joint} for computational efficiency. In theory, this means that the overall error could increase after each W-step iteration, but we have not experienced any problems in practice.

The smoothness term penalizes variation between the weights of neighboring grid points. It is defined as

$$E_{\text{smooth}}(\mathcal{W}) = \sum_{(c,d)\in E_G} \|\mathbf{v}_c - \mathbf{v}_d\|^2, \tag{5.6}$$

where E_G is the set of all edges between neighboring grid points c and d. The normalization term encodes our preference that the weights sum to 1 at each grid point

$$E_{\text{norm}}(\mathcal{W}) = \sum_c \left(1 - \sum_j v_{cj}\right)^2, \tag{5.7}$$

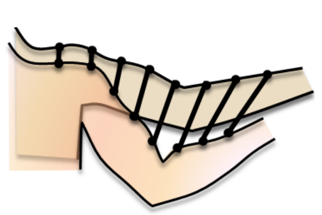
where v_{cj} is the j^{th} component of $\mathbf{v}_c$. Thus, in the W-step our goal is to optimize

$$\begin{aligned} &\underset{\mathcal{W}}{\operatorname{argmin}} \;\; \alpha E_{\text{dist}}(\mathcal{T},\mathcal{W}) + \mu E_{\text{smooth}}(\mathcal{W}) + \nu E_{\text{norm}}(\mathcal{W}) \\ &\text{subject to} \quad v_{cj} \geq 0 \quad \forall\, j,\; \forall\, \mathbf{v}_c \in G. \end{aligned} \tag{5.8}$$

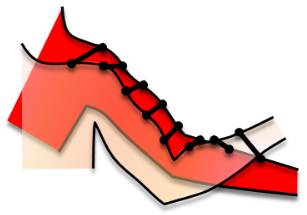
We have observed that optimizing this objective directly often converges to a local minimum that does not correspond to an intuitive solution. With continuous weights, the system is underconstrained because even a single transformation can be expressed as a weighted combination of one, two, three, or more transformations. Thus, instead of obtaining weight functions that are localized to each rigid part, the weights tend to be very smooth and influence large parts of the shape. At the same time, the transformations $\mathcal{T}$ are overfitted in the successive T-steps. Therefore, we use a two step procedure that starts out with a more aggressive regularization. Since articulated models are composed of nearly rigid parts, we expect that the weight function is mostly 0 or 1, except at joints where nearby transformations blend together. So instead of solving for a vector of continuous influence weights at each point, we first solve a *discrete labeling* problem, which assigns a single transformation at each point on the surface. Later in a weight refinement step, we adjust the weights using a continuous optimization step.

To illustrate this further, we show an example in Figure 5.6 of the difference between solving for continuous vs. discrete weights. Here, we are trying to align two arm shapes with three transformations. The movement of each transformation is illustrated in (a),(b), and (c). The first transformation aligns the torso region, the second transformation aligns the upper arm, and the third aligns the lower arm. Note at this point the transformations are not exact but approximate, because they are still being optimized in the algorithm.

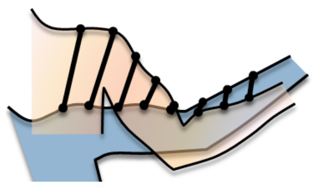
The goal of the weight optimization is to find the weights that satisfy the correspondence constraints (shown as black points and lines) as closely as possible. Since the transformations are only approximate, the optimizing for continuous weights in (d) leads to a smooth, overfitted result which blends all of the transformations (bones) so that the constraints are satisfied exactly. Instead, we prefer to use a discrete labeling like (e) which assigns a single



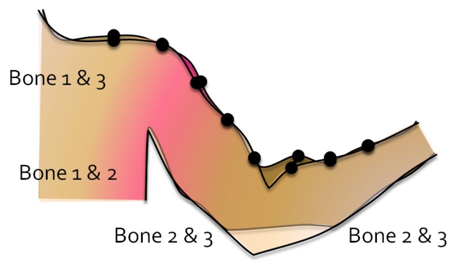
(a) Transformed Using Bone 1



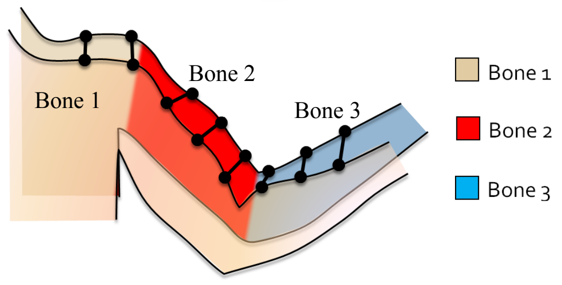
(b) Transformed Using Bone 2



(c) Transformed Using Bone 3



(d) Optimization with Smooth Weights



(e) Optimization with Discrete Weights

Figure 5.6: Solving for smooth weights leads to overfitting in the optimization. The top row shows each of the three transformations (bones) to align the shapes, and the bottom row compares a continuous weight solution and a discrete weight solution. The labels in (d) show which transformations have been blended in each region of the shape.

transformation to each point on the surface. This leads to weights that are not overfitted and cleanly separate each rigid part.

Labeling. To perform the discrete labeling on the grid, we solve for a single label at each cell that indicates which transformation is assigned to this cell (Figure 5.7). Note that we assign labels to each cell as opposed to each grid point. This is because it is more straightforward to evaluate the correspondence error for each grid cell, rather than each grid point. For the grid cell case, we can just find all correspondences contained within the cell, transform them according to the cell's label, and compute the resulting error. However, for the grid point case, the weight at the grid points are trilinearly interpolated in the interior of each grid cell. Therefore, multiple grid points (therefore potentially multiple labels) can affect the transformation applied to a correspondence, which makes the problem more complicated.

We replace the continuous smoothness term with a discrete version

$$\sum_{(C,D)\in A_G} V(f_C, f_D), \tag{5.9}$$

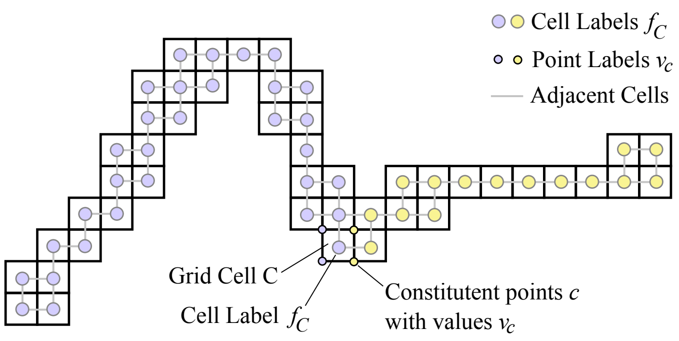


Figure 5.7: We solve for the influence weights by formulating it as a labeling problem on the grid cells.

where f_C is the bone index (or label) assigned to cell C, and A_G is the set of adjacent grid cell pairs $C, D \in G$. The discrete smoothness term assigns a constant penalty when neighboring cells have different labels:

$$V(f_C, f_D) = \begin{cases} 0 & \text{if } f_C = f_D \\ 1 & \text{otherwise.} \end{cases} \tag{5.10}$$

This term controls the degree to which the labels form contiguous regions over the grid. In addition, it also satisfies E_{norm} and the non-negativity constraint automatically. Thus, the overall optimization objective becomes

$$\underset{f_C \forall C}{\text{argmin}} \; \alpha \sum_{C \in G} W(C, f_C) \; + \; \mu \sum_{(C,D) \in A_G} V(f_C, f_D). \tag{5.11}$$

Here, $W(C, f_C)$ is the discrete version of E_{dist}, and it is essentially the same as in the continuous case, measuring the point-to-plane distance in each grid cell.

Compared to solving for continuous weights, this labeling approach dramatically reduces the complexity of the problem. We solve it efficiently using the graph cuts algorithm. First, we construct an instance of graph cuts where the grid cells $C \in G$ are the sites and adjacent cells $(C, D) \in A_G$ are neighbors. We then apply the α-expansion algorithm (Section 3.4) to find the cell labeling that minimizes this objective. Once the optimal labeling is obtained, we

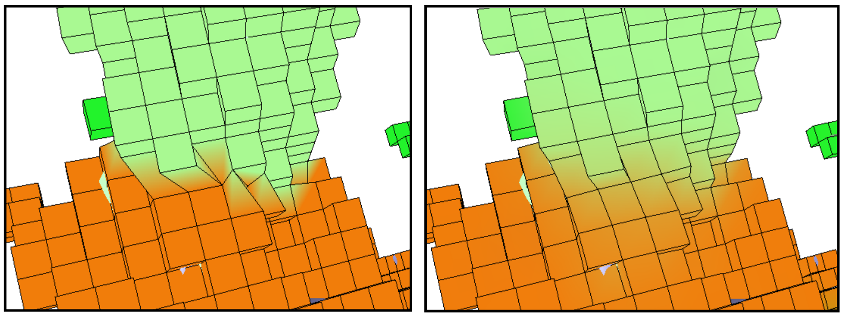

(a) Before weight refinement (b) After weight refinement

Figure 5.8: Refining the discrete weights with smooth weight values. (a) In many cases, discrete labeling causes unnatural deformations between neighboring transformations. (b) With the final weight refinement step, we solve for a smooth blend between neighboring transformations at the joint.

update the weights v_{cj} at each grid point c in each cell C by assigning a binary value

$$v^*_{cj} = \begin{cases} 1 & \text{if } j = f_C \\ 0 & \text{otherwise,} \end{cases} \tag{5.12}$$

where f_C is the label assigned to cell C.

Reusing Labels. If the smoothness weight is set too high, or if some transformations are very similar, the smoothness term will dominate the distance term. In these situations, two separate parts may be labeled with the same bone index. This often results in unused labels that are not assigned to any grid cell, with little chance that the label will be reintroduced. Therefore, we give a second chance for these unused labels by introducing them into regions with the highest registration error. The optimization then proceeds to the T-step, giving a chance to adapt the transformations before the next W-step.

In our implementation, we first find the regions with the highest registration error according to $W(C, f_C)$, and split each region randomly in half. The split is performed in the same fashion as we initialize the weights (Section 5.3.3). We continue this process until the registration error is below a threshold (typically 0.1 times the vertex sampling distance) or until there are no remaining unused labels.

Weight Refinement. The binary weight assignment according to Equation 5.12 has the limi-

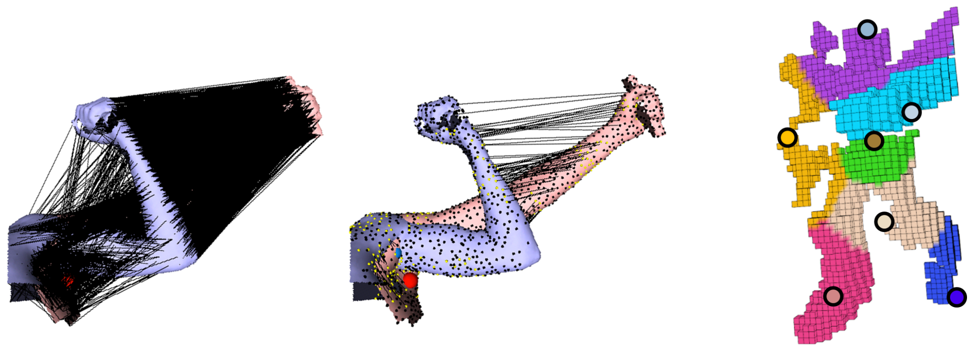

(a) Spin Image Matches (b) Filtered with RANSAC (c) Initialization of Weights

Figure 5.9: Initializing correspondences and weights. (a,b) Spin image matching alone produces many outlier correspondences, but filtering these correspondences with RANSAC selects the most consistent subset. (c) The weights are initialized by picking random bone centers and assigning the label of the closest center at each grid point.

tation that the surface may deform unnaturally at joints, as illustrated in Figure 5.8. In order to smooth these regions, we perform an optional weight refinement step by solving for the continuous weights according to the original optimization objective in Equation 5.8. Since we would like the weights to be as similar as possible to the discrete weights, we add an additional term to Equation 5.8 that pulls the weights to their discrete counterparts:

$$E_{\text{disc}} = \sum_{c} \left(v_{cj} - v_{cj}^{*} \right)^2, \tag{5.13}$$

where v_{cj}^{*} is the binary weights obtained from discrete labeling. We solve the resulting non-negative least squares (NNLS) problem using the approach by Schaefer and Yuksel [2007]. This solves the unconstrained least squares problem, removes variables with the smallest negative values (i.e. forcing their value to zero), and repeats until a non-negative solution is reached.

5.3.3 Initialization

To start our algorithm, we automatically initialize the weight functions and find initial correspondences using feature descriptors and RANSAC. Note that the user specifies the total number of bones used to estimate the motion of the range scans. This parameter does not need to be exact, since the graph cuts optimization tends to combine regions and throw away extra labels.

Initial Correspondences Using RANSAC. When the input scans are initially close together, we can use closest point correspondences. To provide better correspondences when the object has moved significantly, we use spin images as a feature descriptor and use Johnson's spin-image matching engine to determine reliable correspondences [Johnson, 1997]. We use these initial correspondences for only the first iteration of the T-step and use closest point correspondences for the rest of the optimization.

Since spin images are local descriptors, matching with them alone gives rise to many incorrect correspondences due to repeated shape features (Figure 5.9a). To deal robustly with such outliers, we treat each labeled region independently and perform RANSAC [Fischler and Bolles, 1981] to estimate a rigid transformation and remove spurious correspondences (Figure 5.9b). This step proved to be useful in our experiments: on average anywhere from 20% to 40% of correspondence candidates were removed as outliers. In some cases, even 60%~70% were outliers, most likely because the scan contained a large amount of symmetry or because regions in the initial segmentation overlapped multiple rigid parts. Occasionally this method confused symmetric parts, because it still relies on the comparison of local feature descriptors. However, it provided reasonably accurate starting guesses in almost all of our experiments.

Initial Segmentation. To initialize the weight function, we generate a random segmentation of the range scan, similar to the k-means clustering algorithm [Lloyd, 1982]. First, we randomly pick initial bone locations for each bone j on the source shape. In this step, we ensure that these locations are sufficiently spaced apart by using best-candidate sampling [Mitchell, 1991]. Then, we initialize binary weight values $\mathbf{v}_c$ at each grid point. To do this, we find the bone j whose location is closest to the grid point, and then we set the j^{th} weight component to be 1 and the rest to 0. This strategy is illustrated in Figure 5.9c. Even though this segmentation is often incorrect, the subsequent weight optimization is able to adjust the weights to express the motion accurately.

5.4 Experimental Results

5.4.1 Registration

We tested our algorithm with five datasets: one synthetic dataset of a walking figure generated by Pinocchio [Baran and Popović, 2007], two real datasets of a car and a robot [Pekelny and Gotsman, 2008], a dataset of a moving hand [Weise et al., 2007], and a dataset of human body scans [Allen et al., 2002]. Each dataset consists of a sequence of depth scans of a moving object. All tests were performed on a single core of an Intel Core 2 Duo 2.66GHz with 2GB of RAM.

Figure 5.10 shows some registered example pairs and the resulting influence weight function on the grid. In the color-coded visualization of the skinning weights, we can see that our algorithm determines intuitive weights for each rigid part. In addition, we were able to obtain an accurate registration with small registration error, visualized for some examples in Figure 5.11.

Our algorithm also worked well with a severe amount of occlusion in the examples. The car example shown in Figure 5.12 is completely missing the arm in one of the pairs. Also, one of the robot pairs is completely missing the torso region, and only a small part of the whole surface is observed in the example of the walking figure. Nevertheless, we are able to successfully align all three examples. We also tested our algorithm using a few examples from the human body scan dataset [Allen et al., 2002], shown in Figure 5.13. Our method produces accurate registrations, and the quality is comparable to that of previous work [Huang et al., 2008]. To test the robustness of our method, we degraded the data by manually cutting holes into the geometry, shown in the bottom of Figure 5.13. We were able to obtain a successful registration even in this case.

We tested our implementation with pairs of adjacent frames in the datasets. In order to evaluate the registration results, we verified the registration of each pair visually. For the car dataset, out of a total number of 89 registration pairs, 85 examples were registered correctly with only 4 pairs exhibiting an objectionable misalignment. The algorithm worked well with the more complex robot dataset as well, with 67 out of 89 pairs registered correctly. In addition,

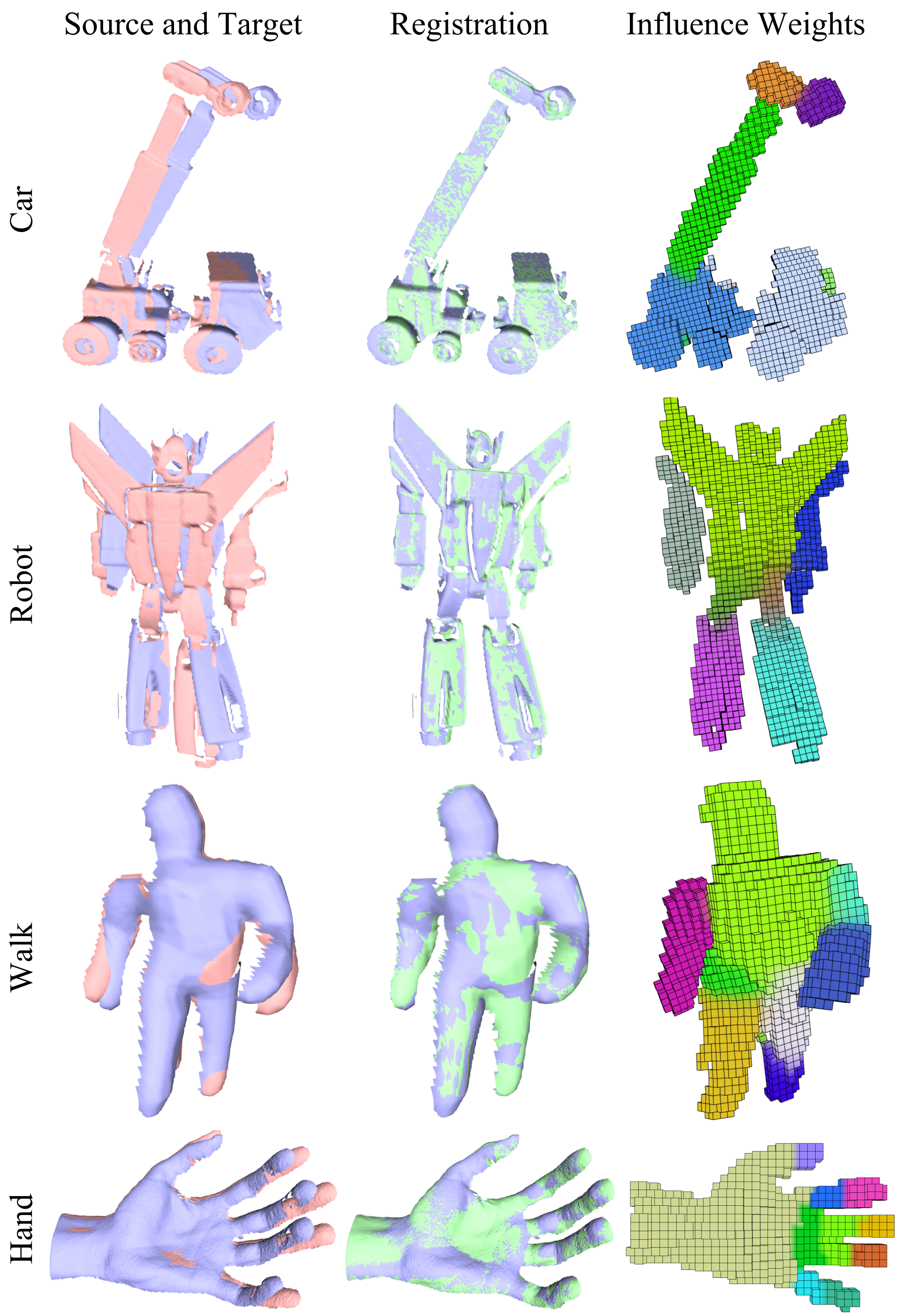


Figure 5.10: Some registration examples from the car, robot, walk, and hand datasets. The red shape is the source, the blue shape is the target, and the green shape is the source deformed to match the target.

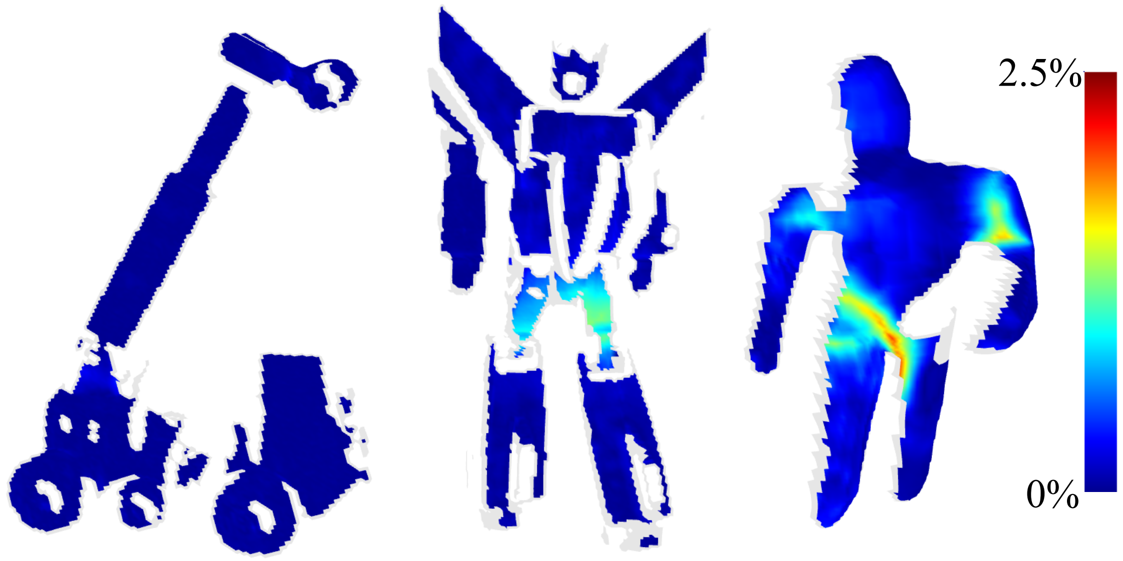


Figure 5.11: Registration Error. The color-coded visualization shows the distance to the closest point as a percentage of the bounding box diagonal.

167 of the 189 pairs were registered correctly in the synthetic walking figure dataset, and 31 out of 46 for the hand dataset. Some results that were less successful are shown in Figure 5.14. In these cases, the maximum correspondence distance was too low or high, causing local minima or wrong part mappings (Figure 5.14a,c), the grid resolution was too low or high, causing separate parts to be attached or separated (Figure 5.14b), or there was too much missing geometry, causing bad part mappings or misalignments (Figure 5.14d). After retrying with different parameter settings, we were able to reduce the misalignment rate to 0, 4, and 6 examples for the car, robot, and walking figure dataset, respectively. The remaining examples remained problematic because of too much missing data or bad part mappings.

5.4.2 Comparison with Huang et al. [2008]

We compare our technique with the recent method by Huang et al. [2008] in Figure 5.15. To test how both methods handle the alignment with missing data, we used the arm example from Figure 5.13 and a pair of frames from the robot dataset. We see that both techniques can align examples with missing data. Comparing the registration error, the result by Huang et al. [2008] aligns the arm examples better, while our method aligns the robot

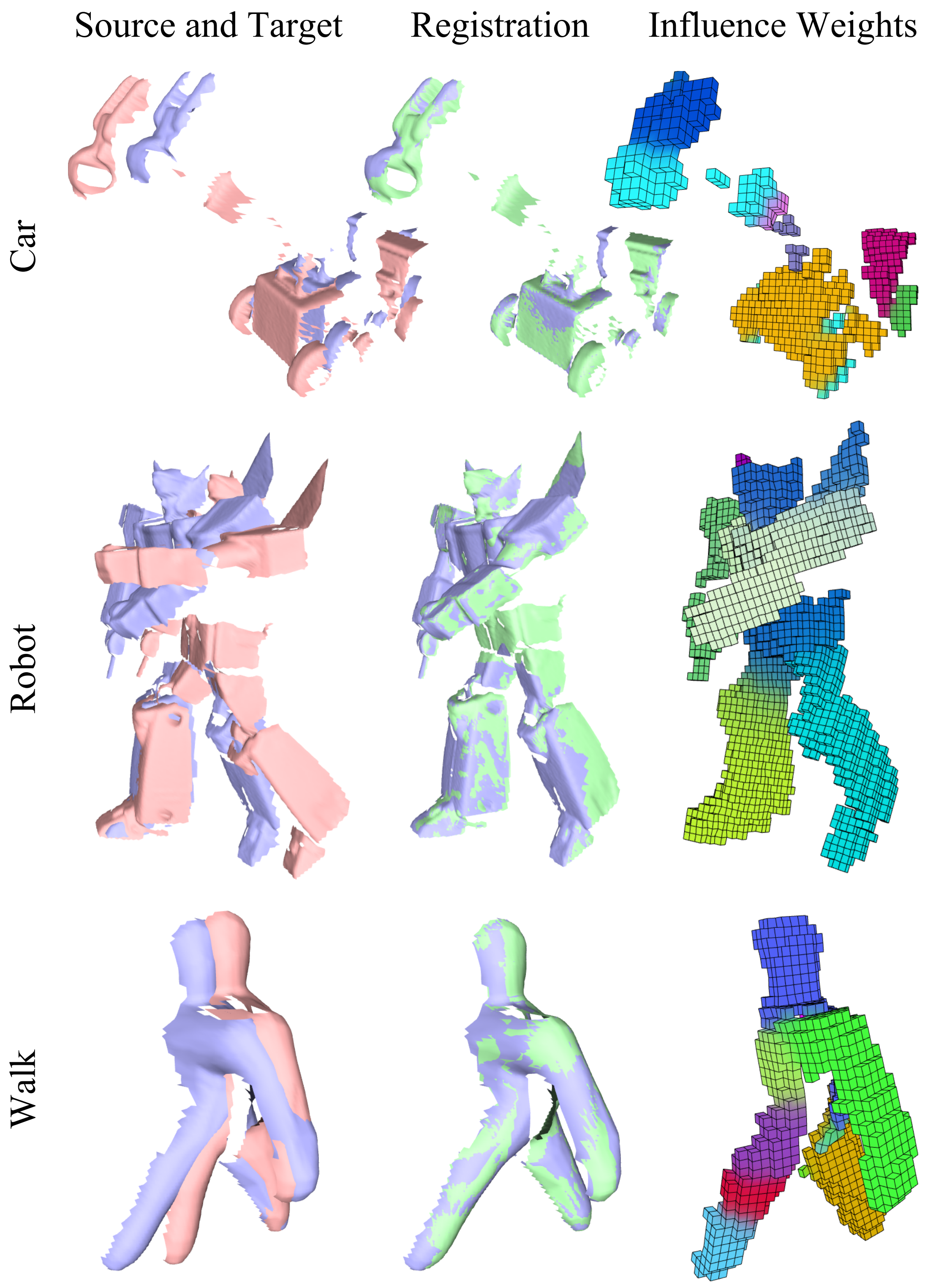


Figure 5.12: More difficult registration tests, each with a significant amount of motion and occlusion.

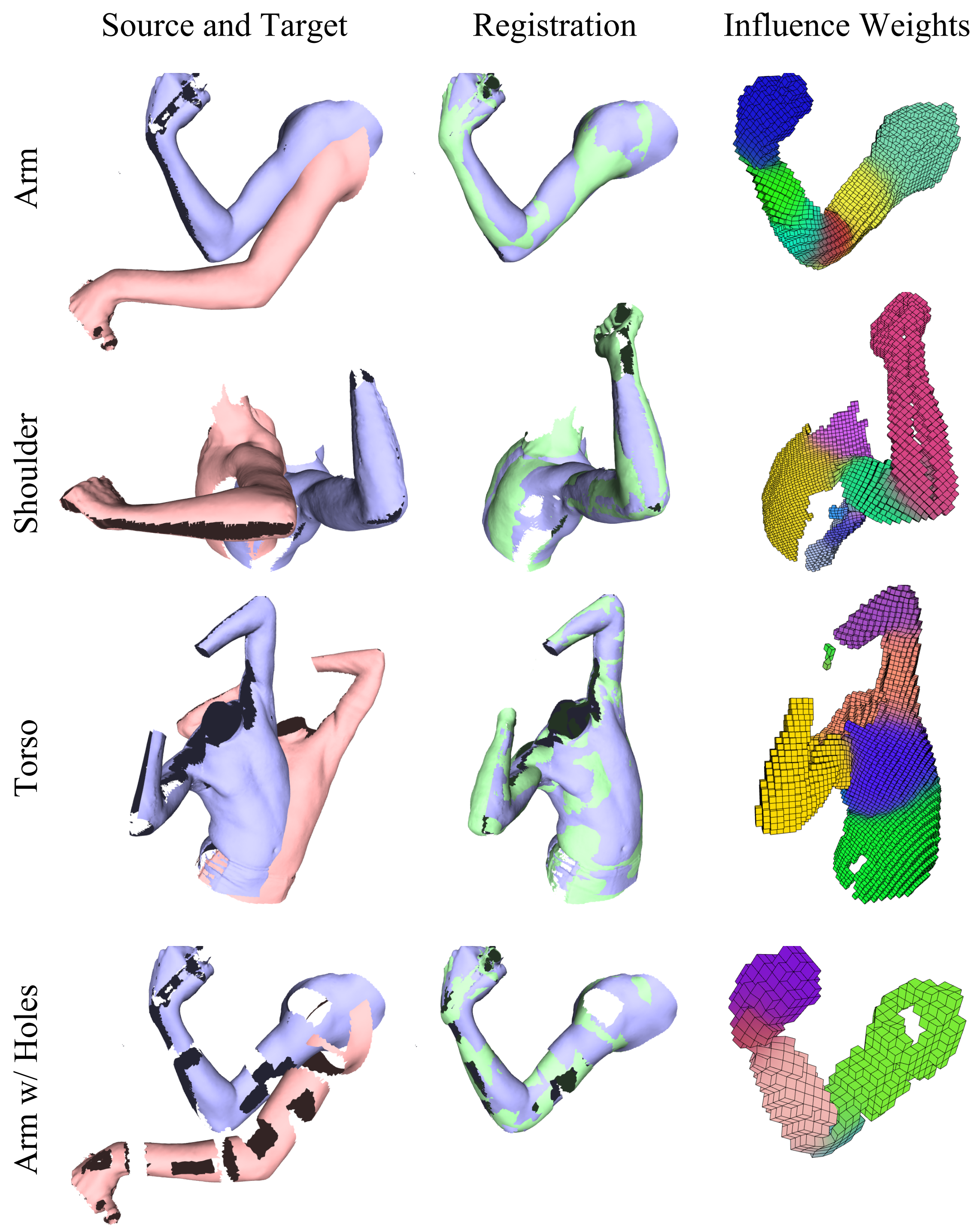


Figure 5.13: Registration examples from the human body dataset. The bottom example, a modified version of the top example, shows that our method is robust to missing data.

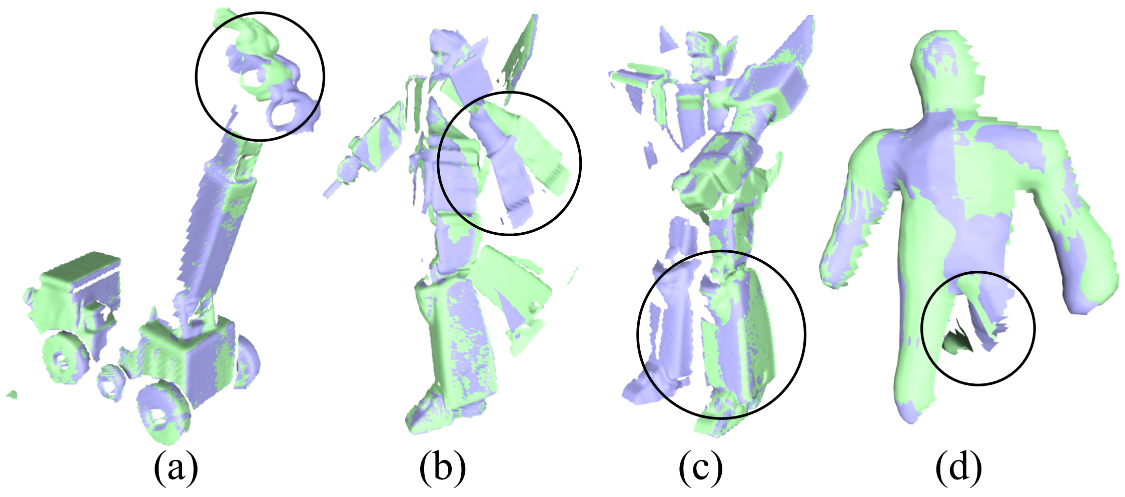
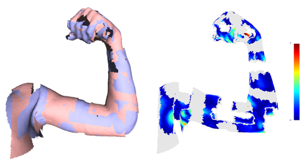
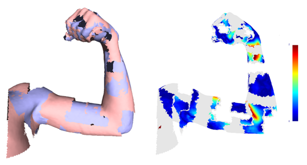


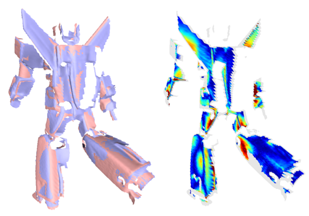
Figure 5.14: Less successful registration examples. (a) The registration converged to a suboptimal local minimum. (b) Shows an undesired connection between the left wing and left arm. (c) Both legs in the source map to the left leg in the target. (d) Insufficient data causes a misalignment.

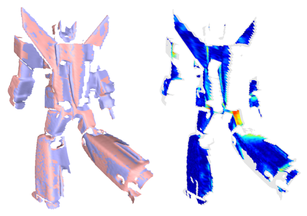

(a) Result by Huang et al. [2008]


(b) Our Result

Arm	Huang et al.	Our Result
Max	4.96	11.78
Average	0.296	0.378
Variance	0.118	0.314

(e) Registration Error for Arm


(c) Result by Huang et al. [2008]


(d) Our Result

Robot	Huang et al.	Our Result
Max	4.54	2.29
Average	0.325	0.135
Variance	0.0896	0.0208

(f) Registration Error for Robot

Figure 5.15: Comparison with Huang et al. [2008], using an arm example with holes and a pair of robot frames. The tables on the right show some statistics of the registration error. Here, the numbers indicate the maximum, average, and variance of the distance to the closest point (measured from all target vertices), as a multiple of the range scan grid spacing.

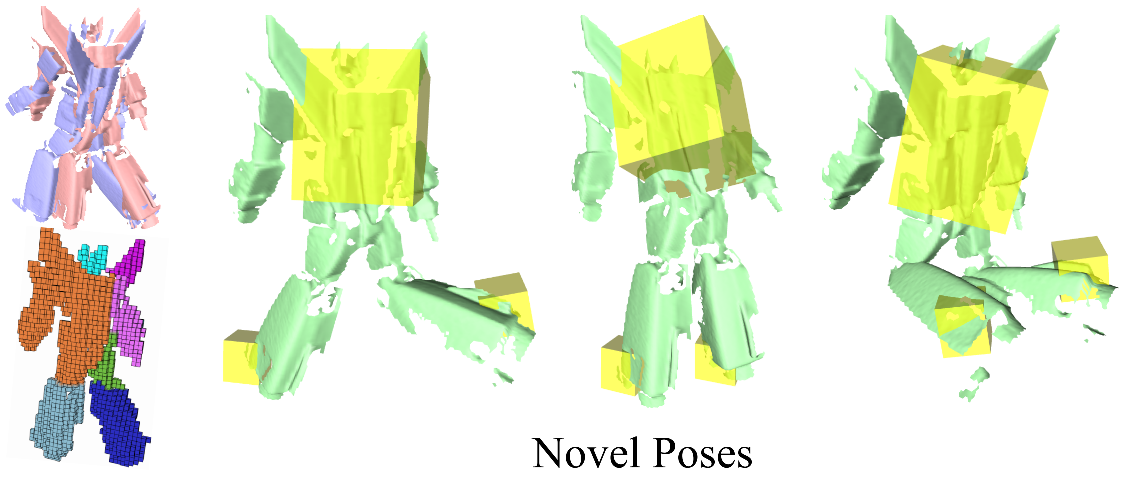


Figure 5.16: Interactive posing using our system. The weights (bottom left) were optimized by registering two scans (top left). The user can interactively select and drag constraints on the shape (shown as boxes) to generate a novel pose.

example better. This shows that our method works well when the examples are more rigid, while non-rigid deformations are aligned less precisely. An additional difference is that our method solves for an articulated motion model, which includes joints between neighboring parts. This allows us to target the range scan into novel poses, which is discussed below.

5.4.3 Creating Novel Poses

Since we solve for the skinning weights, our technique is especially useful for creating novel poses. Unlike previous work, we use the output of our algorithm directly to perform both forward and inverse kinematics, either by specifying or solving for the bone transformations. In fact, by substituting the correspondences with user-given constraints, we can use our T-step optimization directly to perform inverse kinematics. We created an interactive application that allows the user to define and manipulate end effectors to create novel poses. Figure 5.16 shows that interesting poses can be generated using our application.

5.4.4 Parameters and Performance

The number of bones, the number of correspondences, the grid resolution, and the maximum correspondence distance for each dataset is reported in Table 5.1. The grid resolution is specified as the number of divisions along the longest axis of the bounding box, and the maximum correspondence distance is specified as a multiple of the range scan sample spacing. Using a coarse grid resolution covers larger holes between parts. However, if the grid is too coarse, then it becomes harder to localize the weights to smaller parts. Also, a coarse grid may cause undesired parts to be connected, causing difficulties in the registration. For the maximum correspondence distance, a larger distance increases the running time for searching initial correspondences, because a greater number of spin images must be compared. This allows for larger movement, but also may cause similar parts (e.g. arms, legs) to be mapped incorrectly. In addition to the maximum distance, we use a dynamic threshold for the maximum normal angle (decreasing from 80 to 20 degrees) similar to [Pekelny and Gotsman, 2008].

For the weights of each error term, the weight of E_{dist} was $\alpha = 1.0$ for both the T-step and the W-step. For the joint constraint term E_{joint}, we decreased the weight from 1.0 to 0.05 in 5 iterations of the main optimization loop according to the function $e^{k \ln(0.05)/5}$, where k is the iteration number starting from 0. This allows the registration to be refined precisely during the latter part of the optimization. For the weight smoothness term E_{smooth}, a constant between 0.5 and 1.0 times the grid spacing size worked well. For E_{norm} we used $\nu = 1.0$, and for E_{disc} we chose a value between 0.1 and 0.5 depending on how close we wanted to be to the discrete weights. Finally, for generating the spin images, we set bin size equal to the grid spacing, the image size to 15, and the maximum support angle to 90 degrees.

We report averaged statistics and timings of our method in Table 5.1 using our implementation. The setup time is spent creating and initializing the data structures, the RANSAC step generates the initial correspondences using spin image matching, the align time is the total time the algorithm processes the main loop, and the weight time is spent performing the final NNLS weight refinement step.

Table 5.1: Average performance statistics for our tests. Timings (in seconds) represent the total time spent in each stage.

Statistic	**Car**	**Robot**	**Walk**	**Hand**
Bones	7	7	10	12
Corresp.	1200	1200	1000	1500
Vertices	5389	9377	4502	34342
Max Dist	20	40	20	30
Grid Res.	60	65	50	40
Grid Cells	1107	1295	1014	814
Grid Points	2918	3366	2553	1884
Setup	0.185s	0.234s	0.136s	0.078s
RANSAC	8.089s	20.001s	5.517s	N/A
Align	9.945s	19.644s	23.092s	49.918s
Weight	6.135s	10.713s	10.497s	3.689s
Total Time	24.355s	50.591s	39.242s	53.684s

5.5 Discussion and Future Work

Our algorithm is currently limited to registering pairs of shapes. In the next chapter, we extend this method to handle the alignment of multiple scans. Since a sequential alignment that accumulates the geometry in each frame causes registration error to accumulate, we formulate a simultaneous optimization to give a precise result.

We are currently using the LBS deformation model for its simplicity. In case LBS exhibits artifacts, such as the well known “candy wrapping” effect, our method could be adapted to optimize other RDMs, for example dual-quaternion blending [Kavan et al., 2007]. However, this would likely make the optimization procedure more computationally expensive.

Our method requires sufficient separation between different parts of the shape in the source scan. If two parts are joined together, this causes the grid cells to be connected between these parts, and the joint constraint will constrain the parts to move together. In the future, we plan to address these topological issues by using a spatially varying smoothness weight that allows the deformation to disregard problematic joints. Also it would be interesting to use a multiresolution hierarchical grid to represent the skinning weights.

We believe that improving the initial correspondences will make the algorithm more robust to problematic cases. Using a more discriminative feature detection, or using spectral clustering (as proposed by Huang et al. [2008]) could help our algorithm to avoid local minima and wrong part mappings in the registration. Our discrete labeling step also resembles clustering algorithms [Cohen-Steiner et al., 2004; Huang et al., 2008], which could be used as an alternative to the graph cut optimization.

5.6 Conclusions

We have presented a method to register deforming range scans by modeling the motion with a reduced deformable model (RDM). We have chosen linear blend skinning for its simplicity, but more sophisticated approaches could be used if necessary. A key idea of our approach is to represent the weight functions on a 3D grid surrounding the scanned geometry. This allows us to apply linear blend skinning (LBS) to range scans with occlusions and missing data. We solve for the parameters of the deformation model using an EM-type algorithm.

We have demonstrated that our approach is able to register articulated shapes robustly and with significant occlusions and missing data. The advantages of our technique are that it does not require any manual segmentation, user specified markers, nor a surface template. We believe that our work is a significant step forward in automatically reconstructing fully rigged 3D articulated models from range scans.

5.7 Acknowledgments

We wish to thank Y. Pekelny and C. Gotsman for sharing the car and robot datasets, T. Weise for the hand dataset, I. Baran for the walking figure dataset, and B. Allen for the body scan data. Additional thanks to G. Debunne for providing the libQGLViewer library, D. M. Mount and S. Arya for providing the ANN library, and Y. Boykov, O. Veksler, R. Zabih for providing an implementation of their graph cuts algorithm. Also we thank Q.-X. Huang for kindly providing us with comparisons.

The material in this chapter is, in part, a reproduction of published material: Will Chang and Matthias Zwicker, "Range Scan Registration Using Reduced Deformable Models," Computer Graphics Forum (Proceedings of Eurographics), Vol. 28, No. 2, April 2009. The dissertation author was the primary investigator and author of this paper.

6

Global Registration for Articulated Model Reconstruction

IN the last two chapters, we focused on the problem of aligning pairs of shapes in different poses. In this chapter, we build on these ideas to design a system that can automatically reconstruct a complete surface from a sequence of dynamic range scans. Consider Figure 6.1, which shows a simple example of an object that we would like to reconstruct. The toy robot is moving in each frame, and the relative position and orientation of each part is changing. At the same time, there is much missing data in each frame because the data is taken from a single range camera. Although this is the case, the motion of the object is relatively simple: it consists of the movement of a few underlying rigid parts.

The goal of this chapter is to demonstrate that it is possible to reconstruct the complete surface from this type of data without a template, a user-specified segmentation, or corresponding marker locations in each frame. Given the sequence of input data as range scans, we assume that the motion of the underlying surface can be effectively approximated by the movement of B rigid parts. As we saw in Chapter 5, prescribing this motion is equivalent to finding a set of B rigid transformations (one for each part) and a vector of weights for each scanned point (which serve as part assignments). A movement of the surface is produced by applying the transformations to each point according to its weights. So the problem for multiple scans is to find the best set of transformations (B transformations for each frame) and

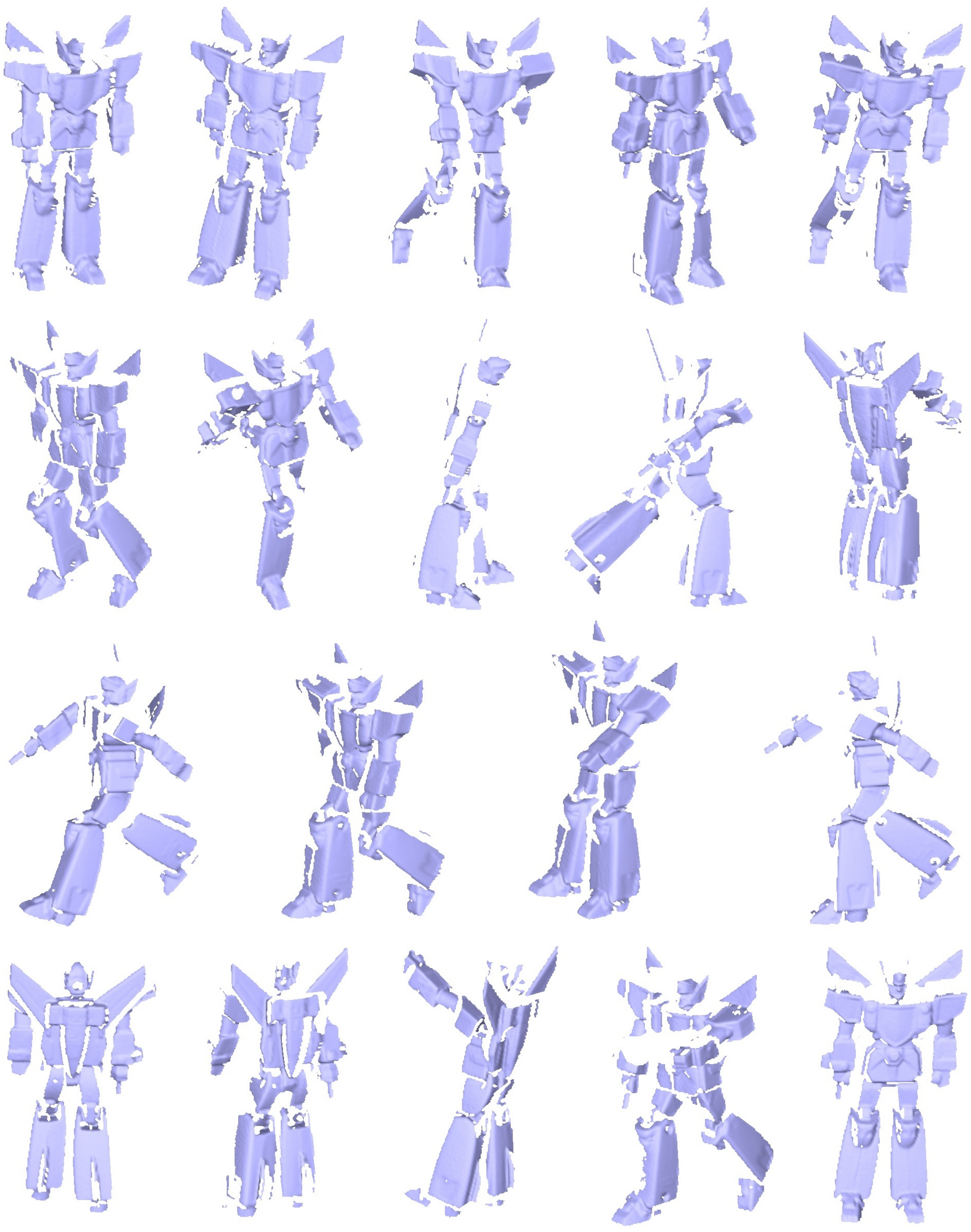


Figure 6.1: Selected range scans from the 90 frame robot sequence. Our goal is to automatically align all of these scans to a common location and a common pose. Data from Pekelny and Gotsman [2008].

influence weights at each scanned point (in all frames) that produces the closest alignment of *all* frames to a common pose. Solving for this articulated model is useful because the weights and transformations, along with joints between neighboring parts, can be used to easily and intuitively edit the pose of the reconstructed model.

6.1 Contributions

We develop an algorithm to solve this problem effectively for multiple range scans of an articulated object. The main contribution is to use the transformation assignment approach of Chapter 4 to obtain a rough initialization of the alignment between pairs of frames, and extend the algorithm of Chapter 5 to align multiple scans simultaneously.

For the initialization, we use the faster strategy discussed in Section 4.4 which solves the assignment problem on a reduced, subsampled representation of the input scan. This initialization is precomputed in advance and used as input to the simultaneous registration step. While the algorithm in Chapter 5 could be extended for multiple scans by sequentially aligning each input scan, it is well known in the rigid registration literature that a sequential alignment approach often causes accumulation of alignment error [Bernardini and Rushmeier, 2002; Rusinkiewicz et al., 2002]. The simultaneous registration prevents this accumulation of error. In addition, we also replace the regular grid used in Chapter 5 with a more flexible graph-based structure to improve performance.

6.2 Algorithm Overview

We first outline the basic structure of our method in Algorithm 6.1. The input to our algorithm is a sequence of range scans, denoted $F_0, \ldots, F_n$, which are expected to be in temporal order so that there is sufficient overlap between frames to align the scans. We assume that each range scan just is a set of 3D points. Although it is not mentioned in the pseudocode above, we estimate the normal at each point based on a triangle mesh constructed from these points (Section 3.2.1). If a mesh is not already available, we construct a simple mesh based on

Algorithm 6.1: ARTICULATED GLOBAL REGISTRATION($F_0, \ldots, F_n$)

```
Data: A sequence of range scans, denoted $F_0, \ldots, F_n$
Result: Sample set $S$ of the completed surface, part labels $\mathcal{W}$ for each sample
        $\mathbf{x} \in S$, rigid transformations $\mathcal{T}$ for each part
1  begin
2      Compute the initial registration between each pair of adjacent frames using
       transformation assignment (Section 4.4);
3      Subsample an initial set of points $S$ from $F_0$, and construct a Euclidean
       k-nearest neighbor graph on $S$;
4      $F_{\text{last}} \leftarrow F_0$;
5      while $F_{last} \neq F_n$ do
6          Let $F_{\text{new}}$ be the next frame after $F_{\text{last}}$;
7          Load the initial registration result for $F_{\text{last}} \to F_{\text{new}}$ (Section 6.3.3);
8          Check if any parts are occluded in $F_{\text{new}}$ (Section 6.3.6);
9          OPTIMIZE $\mathcal{T}, \mathcal{W}$ $(S, E, \mathcal{T}, \mathcal{W}, F_0, \ldots, F_{\text{new}})$ (Algorithm 6.6);
10         $(S, \mathcal{W}, E) \leftarrow$ RESAMPLE ASG$(S, \mathcal{W}, \mathcal{T}, F_0, \ldots, F_{\text{new}})$ (Algorithm 6.2);
11         $F_{\text{last}} \leftarrow F_{\text{new}}$;
12     return $S, \mathcal{W}, \mathcal{T}$;
13 end
```

the rectilinear structure of the scanned points.

The first step is to apply the algorithm from Chapter 4 independently to each pair of adjacent frames in the sequence (line 2). This method is used because it can align a pair of scans while being robust to missing data and large motions. We use the faster version of the algorithm developed in Section 4.4 to speed up the matching process. The output of this step is an initial registration between each adjacent frame.

The second step is to refine this initial registration using the technique of Chapter 5 and produce a global registration of all frames to a common pose. The basic idea is to optimize the transformations and weights simultaneously across all frames to align them to a common reference pose. The frames are introduced sequentially, one at a time, into the global registration (lines 5–11). For each frame, we load the initial registration (line 7), handle cases when parts are occluded (line 8), optimize the transformations $\mathcal{T}$ and weights $\mathcal{W}$ to simultaneously align the frames (line 9), and update the sample set used to optimize the registration (line 10).

An important component in the global registration is to optimize the alignment using a subset of positions S gathered from all frames. This set S is called the “sample set.”

As long as there are enough samples to unambiguously match each rigid part, this can speed up the algorithm while still producing a high-quality alignment of the frames. In addition, unlike the original technique, we define weights directly on each sample in S. This simplifies the algorithm by removing the overhead of translating between the locations of the surface samples and the grid cells. The regular grid connectivity, which was needed for specifying smoothness constraints between weights, is now replaced by a Euclidean k-nearest neighbor graph on the sample positions. This provides a more flexible structure to avoid sampling issues and topological problems.

As discussed previously in Section 5.3.2, solving for smooth weights during the registration leads to overfitting of both the transformations and the weights. Thus, like Chapter 5 we solve for a discrete labeling of the samples S, which corresponds to assigning a binary weight (where one component is exactly 1 and the rest are 0). In the following sections, we will use the terms "weight" or "binary weight" and "label" interchangeably.

During the global registration, some parts may entirely disappear (and reappear) in several frames. To handle these cases, we check if there are too few matching samples for each part. If this is the case, then the part is marked as occluded and is subsequently excluded from the optimization. Also, in case a part reappears (perhaps in a different location), we have a strategy to track the part again during the global registration. We will discuss this more in Section 6.3.6 and 6.3.7.

Now we describe each component of our algorithm in detail. After performing the registration on the entire sequence, we can resample the surface densely, reconstruct a mesh representing the completed surface, and fit smooth skinning weights to obtain a final polished reconstruction.

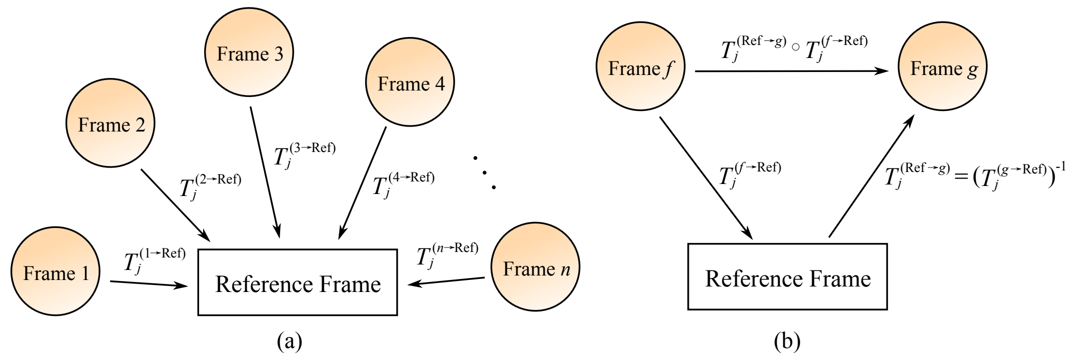


Figure 6.2: Organizing the transformations for simultaneous registration. (a) We solve for the set of transformations that align each input frame to the reference frame F_0. (b) We can transform between any pair of frames f and g by first transforming from f to the reference and applying the inverse transformation to g.

6.3 Global Registration

6.3.1 Organization of the Transformations

We need a way to represent the movement for each part in each frame. To organize this information concisely, we designate one of the frames as a *reference frame* or *reference pose* and define rigid transformations (each composed of a rotation and translation) relative to this reference[1]. This definition makes it easy to specify and solve for the alignment later in the global registration step (Section 6.3.4).

To specify the pose of the subject in each frame, we need to use multiple transformations, one for each rigid part. The user specifies a maximum number of transformations B to use in each frame. We use the notation $T_j^{(f\to\text{Ref})}$ to denote the jth transformation for frame F_f. Each transformation $T_j^{(f\to\text{Ref})}$ is composed of a rotation $R \in SO(3)$ and translation $\vec{\mathbf{t}} \in \mathbb{R}^3$. To apply the transformation to a point $\mathbf{x} \in \mathbb{R}^3$, we use the notation $T_j^{(f\to\text{Ref})}(\mathbf{x}) = R\mathbf{x} + \vec{\mathbf{t}}$. Note that the direction of transformation $T_j^{(f\to\text{Ref})}$ is from frame f to the reference frame. This is illustrated in Figure 6.2. We can also transform between any frames by transforming first to the reference frame and then applying the inverse transformation to the desired frame (Figure 6.2b). This is expressed using an inverse operator $(\cdot)^{-1}$ and a composition operator $\circ$,

[1]This is similar to the approach used by Neugebauer [1997] for registering scans of rigid objects.

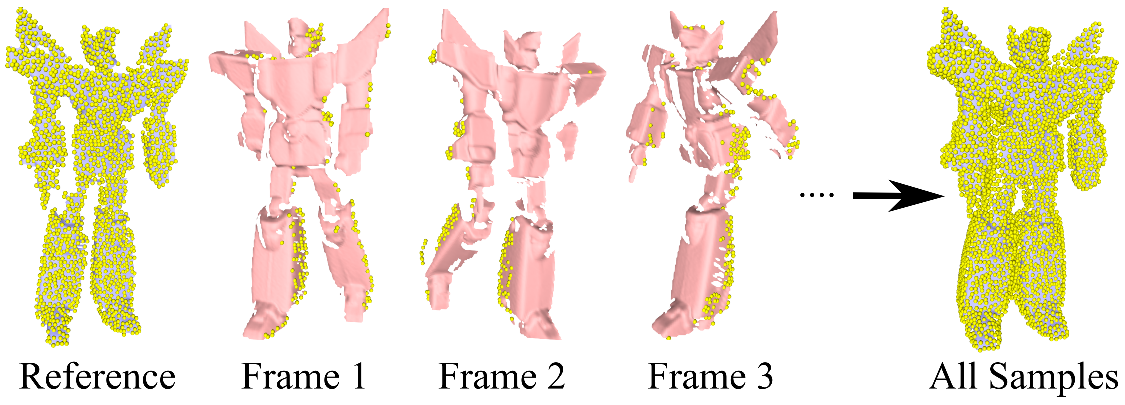


Figure 6.3: We use sample points on all input frames to measure the global alignment. For each frame, we only keep the samples that are from new geometry that has not been observed in any previous frames.

which translates to the formula

$$\begin{aligned}\left(T_j^{(g\to\mathrm{Ref})^{-1}} \circ T_j^{(f\to\mathrm{Ref})}\right)(\mathbf{x}) &= T_j^{(g\to\mathrm{Ref})^{-1}}\left(R_j^{(f\to\mathrm{Ref})}\mathbf{x} + \vec{\mathbf{t}}_j^{(f\to\mathrm{Ref})}\right)\\ &= R_j^{(g\to\mathrm{Ref})^{\top}}\left[\left(R_j^{(f\to\mathrm{Ref})}\mathbf{x} + \vec{\mathbf{t}}_j^{(f\to\mathrm{Ref})}\right) - \vec{\mathbf{t}}_j^{(g\to\mathrm{Ref})}\right].\end{aligned} \tag{6.1}$$

Thus, once we know the transformations relating each frame to the reference, we can transform from any source frame (where the weights are known) to the pose of any target frame.

6.3.2 Sample Set and All-Samples Graph (ASG)

The set of samples S plays a key role in optimizing for the precise alignment of the scans in the global registration step. Each member of S, which we call a *sample point*, is a scanned point $\mathbf{x} \in \mathbb{R}^3$ selected from an input frame F_f [2]. This set will serve as a coarse representation of the reconstructed 3D model. It is used to optimize the alignment between the frames (Figure 6.3). Associated with each sample is a vector of weights $\mathbf{w_x} = [w_{\mathbf{x}1}, w_{\mathbf{x}2}, \ldots, w_{\mathbf{x}B}]$, which each component $w_{\mathbf{x}j}$ indicates the strength or influence of transformation T_j on $\mathbf{x}$. During our optimization we will solve for binary weights (labels), thus all components of $\mathbf{w_x}$ are 0, and only one component $w_{\mathbf{x}j} = 1$.

Creating the Sample Set. Our purpose in using the sample set is to sparsely represent the

[2] In the subsequent text, we will implicitly assume that the sample point $\mathbf{x}$ is associated with the frame F_f.

entire observed surface, while minimizing the redundancy of geometric information. This will maximize the performance of the optimization, while still giving an accurate registration of the surfaces. Our approach is to select a well-distributed set of points in each registered frame and merge all of the sets together. During the merging step, we minimize redundancy in S by removing overlapping points. This sampling and merging process is performed every time a new frame is added to the global registration, so that any changes in the registration or newly observed parts of the surface are incorporated into S.

First, a well-distributed set of points are picked in each frame using the best-candidate technique [Mitchell, 1991]. We pick a fixed fraction of the points, where this fraction is a user-specified parameter. The use may adjust it lower to compute the registration faster, or higher to obtain a more precise alignment. We denote the resulting list of sampled points as U_f, and we store the points in this list as candidates for selection into the sample set S. Note that this step has to be performed only once per frame when we store the list of sampled points.

For the very first frame, the sample set S is just U_0. Every time a new frame is added to the global registration, S is resampled to produce a new set S', which incorporates changes in the registration and adds points from newly observed parts of the surface. The resampling simply merges all of the U_f by detecting and removing overlapping samples, and assigns a weight for each point in the newly resampled set.

The resampling (Algorithm 6.2) is performed using a sequential strategy. Starting from U_0, we iterate through each successive U_f and determine (1) which points are not overlapping with the currently selected set S' and (2) the weights of each $\mathbf{x} \in U_f$ extrapolated using the previous weights S and $\mathcal{W}$. Then, we add all the non-overlapping samples that have a valid weight in the new set $S', \mathcal{W}'$.

We adopt the strategies outlined by Pekelny and Gotsman [2008] to determine the non-overlapping samples and extrapolate the weights for the new samples in U_f. To determine the non-overlapping samples (Algorithm 6.3), for each sample we compute the distance d_{pt} to the closest existing point. We also the distance d_{OnPl} between the points projected on the plane passing through the surface normal at the closest point. If the surface normals differ

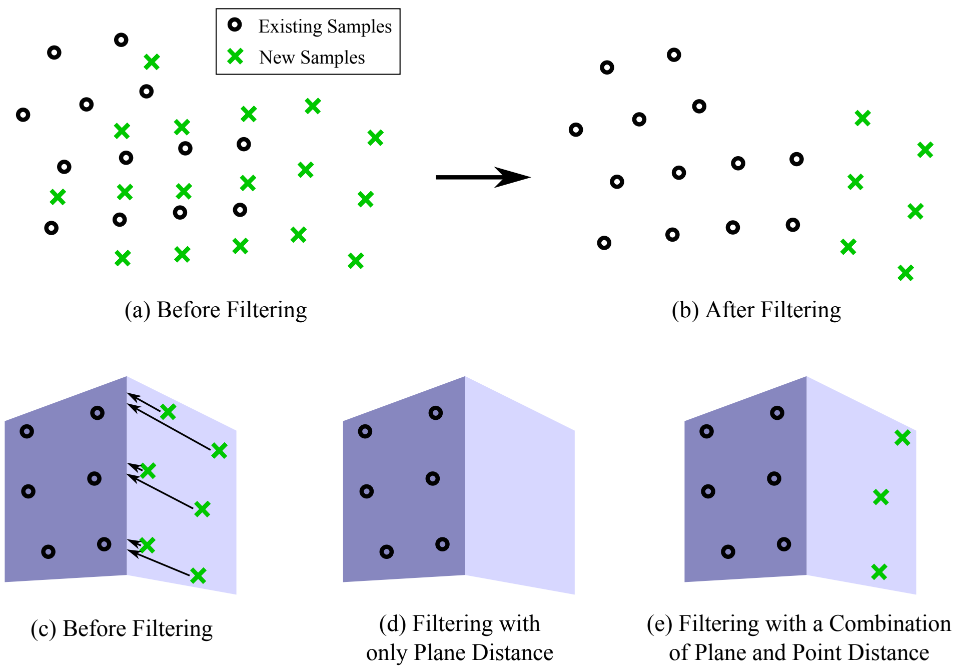
Algorithm 6.2: RESAMPLE ASG($S, \mathcal{W}, \mathcal{T}, F_0, \ldots, F_{\text{new}}$)

```
Data: Sample set $S$ with associated labels $\mathcal{W}$, transformation for all frames $\mathcal{T}$
Result: New sample set $S'$, assigned weights $\mathcal{W}'$, ASG graph $E'$
begin
    foreach Label $i \in [1..B]$ do
        Find all samples $\mathbf{x} \in S$ whose label is $i$;
        Transform these samples to the reference frame and store in $V_i$;
    $S' = \{\}, \mathcal{W}' = \{\}$;
    foreach Frame $F_f \in [F_0..F_{new}]$ do
        Let $U_f$ be the set of uniformly subsampled points in $F_f$;
        Transform $S'$ to frame $F_f$ and store in $V'$;
        Transform each $V_i$ to frame $F_f$ using $T_i^{(f \to \text{Ref})^{-1}}$ and store in $V_i'$;
        $U_f' \leftarrow$ NONOVERLAP($V', U_f, \tau_{\text{sample}}$) (Algorithm 6.3);
        $W' \leftarrow$ EXTRAPLABELS($V_i', U_f, \tau_{\text{sample}}$) (Algorithm 6.4);
        foreach $\mathbf{x} \in U_f'$ do
            if $\mathbf{w}_\mathbf{x}' \in W'$ is valid then
                Add $\mathbf{x}$ to $S'$ and its corresponding weight $\mathbf{w}_\mathbf{x}'$ to $\mathcal{W}'$;
    $E'$ = CONSTRUCT ASG($S', \mathcal{W}', \mathcal{T}, \tau_{\text{len}}, \tau_{\text{stretch}}$) (Algorithm 6.5);
    return $(S', \mathcal{W}', E')$
end
```

Algorithm 6.3: NONOVERLAP($V, U_f, \tau_{\text{sample}}$)

```
Data: Existing points and corresponding surface normals $V$, new points and
      normals $U_f$
Result: A subset $U_f'$ of $U_f$ that does not overlap with $V$
begin
    $U_f' = \{\}$;
    foreach $\mathbf{x} \in U_f$ do
        Find the point $\mathbf{y} \in V$ that is closest to $\mathbf{x}$;
        Let $\vec{\mathbf{n}}_\mathbf{x}, \vec{\mathbf{n}}_\mathbf{y}$ be the corresponding surface normals of $\mathbf{x}, \mathbf{y}$;
        $d_{\text{pt}} \leftarrow \|\mathbf{x} - \mathbf{y}\|$;
        $d_{\text{OnPl}} \leftarrow \sqrt{\|\mathbf{x}-\mathbf{y}\|^2 - \left((\mathbf{x}-\mathbf{y}) \cdot \vec{\mathbf{n}}_\mathbf{y}\right)^2}$;
        if $(\vec{\mathbf{n}}_\mathbf{x} \cdot \vec{\mathbf{n}}_\mathbf{y} < 0$ and $d_{pt} > \tau_{sample})$ or $(\vec{\mathbf{n}}_\mathbf{x} \cdot \vec{\mathbf{n}}_\mathbf{y} > 0$ and $d_{OnPl} > \tau_{sample})$ then
            Add $\mathbf{x}$ to $U_f'$;
        return $U_f'$;
end
```

by more than 90 degrees, then we apply a threshold on d_{pt}, otherwise we apply a threshold on d_{OnPl}. This is a modification of the original strategy, which only applied a threshold on



(a) Before Filtering (b) After Filtering

(c) Before Filtering (d) Filtering with only Plane Distance (e) Filtering with a Combination of Plane and Point Distance

Figure 6.4: Illustrating how we filter out overlapping points. (a,b) Each time there are new points to be added to the sample set S, we filter out samples that are too close to the existing points in S. (c,d,e) In this case, the existing samples lie on one plane and the new samples lie on another plane that is perpendicular to the first. When only d_{OnPl} is used, all of the new samples are discarded, because they are close to the existing samples when projected onto the first plane. We use a combination so we can keep samples that are further away from the existing set.

d_{OnPl}. We illustrate this with an example in Figure 6.4. The original strategy can potentially discard points that are far away from the existing set, because the projected on-plane distance happens to be small. Our modified strategy seeks to keep the points whose projected distance may be small but the point-to-point distance is large.

To extrapolate the labels from the original set S to the new samples U_f (Algorithm 6.4), we first gather all the samples for each label i into separate sets V_i, and transform these sets into frame F_f. To determine the label for a new sample point $\mathbf{x} \in U_f$, we first compute the distance from $\mathbf{x}$ to the closest point in each set V_i, using a weighted combination of point-to-point and point-to-plane distances [3]. These distances are then inverted and normalized to give a score for that label, and the label with the maximum score is taken as the label for the new point $\mathbf{x}$. However, if this label is marked as occluded for this frame, or the maximum

[3]The weights used here are discussed later, in Section 6.3.4.

Algorithm 6.4: ExtrapLabels(V_i for each part $i, U_f, \tau_{\text{sample}}$)

```
Data: Existing points and corresponding surface normals $V_i$ for each part, new
      sample points $U_f$
Result: A set of labels $W$ for each point in $U_f$
begin
    foreach $\mathbf{x} \in U_f$ do
        foreach Label $i \in [1..B]$ do
            Find the point $\mathbf{y} \in V_i$ that is closest to $\mathbf{x}$;
            Let $\vec{\mathbf{n}}_{\mathbf{y}}$ be the corresponding surface normal of $\mathbf{y}$;
            $d_i \leftarrow \eta_{\text{pt}} \|\mathbf{x}-\mathbf{y}\| + \eta_{\text{pl}} \left|(\mathbf{x}-\mathbf{y}) \cdot \vec{\mathbf{n}}_{\mathbf{y}}\right|$;
        $C_i \leftarrow \frac{1/d_i}{\sum_i 1/d_i}$;
        Sort the list of $C_i$, and find the median $m$ of the larger half of this list;
        $\tau_C \leftarrow 3m$;
        if $(\max_i C_i > \tau_C)$ and $(\operatorname{argmax}_i C_i$ is not occluded$)$ then
            Set $\operatorname{argmax}_i C_i$ as the label for $\mathbf{x}$;
        else
            Mark $\mathbf{x}$ as having no valid label;
    return The labels $W$ corresponding to each point $\mathbf{x} \in U_f$;
end
```

score is not significantly higher than the other scores, then we consider the label as invalid for this point. To determine if the maximum score is significantly higher, we use three times the upper quartile (median of the largest half of the scores) as a threshold.

Constructing the ASG. In addition to the sample set S, we construct a graph structure over S, which we call the *All-Samples Graph (ASG)*. The connectivity of this graph serve as smoothness constraints that help the optimization form large, contiguous parts. The graph is constructed by forming the k-nearest neighbor graph on the sample set S, where all samples are transformed to the reference frame. To prevent undesired smoothness constraints between separate (but spatially near) parts, we measure the length of each edge in all frames and discard edges that stretch too much. The procedure for constructing the graph is outlined in detail in Algorithm 6.5. We observed that pruning edges between connected parts may bias the discrete labeling optimization and create an unmovable boundary between parts. Therefore, we keep edges between parts that are connected by a joint (which are discussed in Section 6.3.4).

Algorithm 6.5: CONSTRUCT ASG($S, \mathcal{W}, \mathcal{T}, \tau_{\text{len}}, \tau_{\text{stretch}}$)

```
Data: Sample set $S$ with associated weights $\mathcal{W}$, transformations for all frames $\mathcal{T}$
Result: A list of edges $E$ of the constructed ASG
1  begin
2      Transform all samples $\mathbf{x} \in S$ to the reference pose;
3      Store the resulting samples in the set $V$;
4      Construct the k-nearest neighbor graph of $V$;
5      Store the resulting edges in $E$;
6      foreach Edge $e = (\mathbf{x}, \mathbf{y}) \in E$ do
7          if $\mathbf{w_x} \neq \mathbf{w_y}$, and their corresponding parts do not have a joint
           (Section 6.3.4) then
8              Transform the two endpoints $\mathbf{x}$ and $\mathbf{y}$ to each frame;
9              Find the maximum $l_{\text{max}}$ and minimum length $l_{\text{min}}$ of the edge;
10             if $l_{max} < \tau_{len}$ and $\frac{l_{max}}{l_{min}} > \tau_{stretch}$ then
11                 Mark the edge $e$;
12     Remove all marked edges from $E$;
13     return $E$;
14 end
```

6.3.3 Propagating the initial registration

The result of the initial registration between adjacent frames F_i and F_{i+1} (Section 4.4) is a set of transformations and their assignment to each point of F_i and F_{i+1}. Since the global registration works only with the weights defined on points of the sample set S, we need to find the transformations and labels of the initialization that pertain to S. There are two cases: (1) at the beginning of the registration, where we have a sample set S entirely from the first frame F_0 with no weights (labels) associated with the samples, and (2) during the registration, where we have both the sample set S and associated labels.

For the first case, we will translate both the labeling and the transformations from the initial registration to the sample set. Here, we would like to assign labels to each point in S and provide an initial transformation for each label. For each sample in S, we assign the same label that is assigned to that point in the initial registration. If the number of labels is greater than the user-suggested maximum B, then we find and keep only the largest B labels. For the samples that do not have a label in this top B, then we find the closest point that does have a top B label and assign this label to the sample. Finally, we just use the transformations corresponding to the labels assigned to the samples.

For the second case, since we already have a labeling on each point of S, our goal is just to find an initial transformation for each label. At a high level, for each label k that is assigned to S, we compute a weighted average of transformations assigned to F_i whose corresponding regions overlap with the region for label k.

First, we first transform all sample points to frame F_i. Since we do not expect each sample $\mathbf{x} \in S$ to have a matching point $\mathbf{y} \in F_i$ at exactly the same location as $\mathbf{x}$, we associate $\mathbf{x}$ with the closest point $\mathbf{y} \in F_i$. Then, the label assigned to sample point $\mathbf{x} \in S$ is considered to be overlapping with the label assigned to point $\mathbf{y} \in F_i$ (from the initial registration). Note that we take care to reject associations when (1) $\mathbf{y}$ is on the boundary of F_i, (2) the distance $\|\mathbf{x} - \mathbf{y}\|$ exceeds a threshold, and (3) the angle between the normals of $\mathbf{x}$ and $\mathbf{y}$ exceeds a threshold. The thresholds used here are the same values used for computing closest points in the global registration in Section 6.3.4.

Finally, to determine the initial transformation for each label k that is assigned to sample points S, we first gather a list of points $R \subseteq S$, where each $\mathbf{x} \in R$ has label k. Then, for each $\mathbf{x} \in R$ we look up the corresponding points $\mathbf{y} \in F_i$ and make a list of all labels L assigned to each $\mathbf{y}$, along with the number of points $\mathbf{y}$ with each label. The labels in this list are considered to be overlapping with region R, and the ratio of (# points for $l \in L$ / # total points) serves as a weight for label l. Finally, we just take a weighted average of the transformations corresponding to each l according to this weight. To compute the weighted average of the rigid transformations, we use the dual-quaternion linear blending (DLB) technique by Kavan et al. [2008].

6.3.4 Global Registration

Once a frame is initialized, it is introduced into the global registration step. This step optimizes for the best weights (denoted $\mathcal{W}$) and transformations (denoted $\mathcal{T}$) that simultaneously align all initialized frames. The optimization objective has three terms: (1) $\mathcal{E}_{\text{fit}}(\mathcal{T}, \mathcal{W})$, which measures the alignment distance of all frames to the reference, (2) $\mathcal{E}_{\text{joint}}(\mathcal{T})$, which constrains neighboring transformations to agree on a common joint location, and (3) $\mathcal{E}_{\text{weight}}(\mathcal{W})$, which constrains the weights to be smooth and to form contiguous regions. With weights α, β, γ for each term, we write the entire objective as

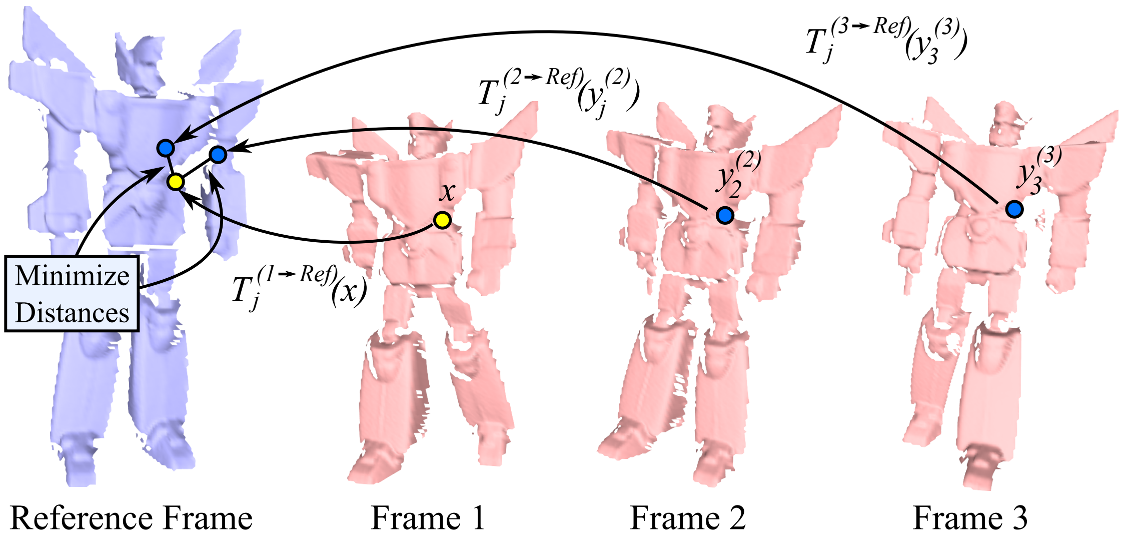
$$\underset{\mathcal{T},\mathcal{W}}{\operatorname{argmin}} \; \alpha \, \mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) + \beta \, \mathcal{E}_{\text{joint}}(\mathcal{T}) + \gamma \, \mathcal{E}_{\text{weight}}(\mathcal{W}). \tag{6.2}$$

Next, we describe each term and our optimization procedure in more detail. During the optimization, solving the weights in a continuous range leads to overfitting, as we mentioned before. To resolve this problem, we constrain the weights to be binary, where only one component can be 1 and the rest are 0. Thus the weights essentially become labels; we will use the two terms interchangeably (See Section 5.3.2).

Fitting Objective $\mathcal{E}_{\text{fit}}$

The key idea for this term is to measure the alignment distance between all frames using the sample points. For each sample point $\mathbf{x}$ on frame f, the weight $\mathbf{w}_{\mathbf{x}}$ assigned to $\mathbf{x}$ and the transformations $\mathcal{T} = \left\{ T_j^{(f \to \text{Ref})} \mid \forall F_f \text{ and } j \in [1..B] \right\}$ tell us the transformed location of $\mathbf{x}$ on all other frames. Therefore, in this term, we transform the sample location to all other frames and measure how close it is to the scanned data of these frames.

To measure the proximity of a transformed point $\mathbf{x}' = T_j^{(g \to \text{Ref})^{-1}} \circ T_j^{(f \to \text{Ref})}(\mathbf{x})$ to frame F_g, we take the distance to the closest point $\mathbf{y}_j^{(g)} \in F_g$. There are three important details to add to this basic strategy. (1) Notice that the closest point will change depending on which of the B transformations we use to transform $\mathbf{x}$ to frame g. This is the reason why we keep a separate closest point $\mathbf{y}_j^{(g)}$ for each j. (2) It may be the case that $\mathbf{x}'$ may not have a corresponding point in F_g due to missing data. To handle this case, for each sample point $\mathbf{x}'$ (i.e. transformed to frame g), we mark the corresponding target point $\mathbf{y}_j^{(g)}$ as *invalid* if (a) the distance between these points exceeds a threshold τ_d, (b) the angle between the normals exceeds a threshold τ_n, or (c) the target point lies on the boundary and the distance exceeds a smaller threshold τ_b [Pekelny and Gotsman, 2008]. (3) If the corresponding point for $\mathbf{x}'$ is indeed missing in frame F_g, then we do not expect $\mathbf{y}_j^{(g)}$ for any j to be valid. Therefore, if j^* is the current label



$T_j^{(3\rightarrow Ref)}(y_3^{(3)})$
$T_j^{(2\rightarrow Ref)}(y_j^{(2)})$
x
$y_2^{(2)}$
$y_3^{(3)}$
Minimize Distances
$T_j^{(1\rightarrow Ref)}(x)$
Reference Frame
Frame 1
Frame 2
Frame 3

Figure 6.5: To measure alignment, we compute distances between sample points $\mathbf{x}$ and target points $\mathbf{y}_j^{(g)}$ on the reference frame F_{ref}. We add up these distances to measure the alignment of all frames in the sequence. We optimize for the transformations and weights to minimize the total distance.

assigned to $\mathbf{x}$, we check to see if $\mathbf{y}_{j^*}^{(g)}$ is invalid. If this is the case, then we invalidate all target positions $\mathbf{y}_j^{(g)}$ for all j.

Given these corresponding points, we can precisely quantify the alignment distance between the sample points all all frames. Figure 6.5 illustrates this situation for a sample point $\mathbf{x}$ in frame 1 and corresponding points $\mathbf{y}_j^{(2)}, \mathbf{y}_j^{(3)}$ in frames 2 and 3, respectively. Mathematically, we measure the alignment distance using the following expression:

$$\mathcal{E}_{\text{fit}}(\mathcal{T}, \mathcal{W}) = \sum_{\mathbf{x}\in S} \sum_{\substack{\text{All } F_g \\ g\neq f}} \sum_{j=1}^{B} \sum_{\text{Valid } \mathbf{y}_j^{(g)}} w_{\mathbf{x}j}\, d\left(T_j^{(f\rightarrow\text{Ref})}(\mathbf{x}), T_j^{(g\rightarrow\text{Ref})}\left(\mathbf{y}_j^{(g)}\right)\right). \tag{6.3}$$

Here, we have computed the distance between $T_j^{(f\rightarrow\text{Ref})}(\mathbf{x})$ and $T_j^{(g\rightarrow\text{Ref})}\left(\mathbf{y}_j^{(g)}\right)$ instead of comparing the distance between $T_j^{(g\rightarrow\text{Ref})^{-1}} \circ T_j^{(f\rightarrow\text{Ref})}(\mathbf{x})$ and $\mathbf{y}_j^{(g)}$. These two alternatives are basically the same, except that the first computes the distance in the reference frame F_{ref}, whereas the second computes the distance in frame F_g. We chose the first option for $\mathcal{E}_{\text{fit}}$, because it does not involve a composition of the transformations seen in the second option. We want to avoid this composition of transformations because it is more difficult to approximate with a linear function in the optimization. Since we linearize each transformation separately, a composition of transformations would result in a multiplication of two linearized transformations, which is less accurate than having two separate linearized transformations.

In addition, notice that we have simplified the objective for the case of binary weights and pulled the weight term $w_{\mathbf{x}j}$ outside of the distance $d(\cdot,\cdot)$. Thus, we can think of the weight as "selecting" one of the distances according to which label is assigned to $\mathbf{x}$. Next, $d(\cdot,\cdot)$ measures the distance between the points, and the resulting distance is then summed up over all sample positions $\mathbf{x}$ to compute the total alignment distance. For $d(\cdot,\cdot)$ we use a weighted sum of the point-to-point and point-to-plane distance measures:

$$d(\mathbf{x},\mathbf{y}) = \eta_{\text{pt}} \, \|\mathbf{x}-\mathbf{y}\|^2 + \eta_{\text{pl}} \, \left((\mathbf{x}-\mathbf{y}) \cdot \vec{\mathbf{n}}_{\mathbf{y}}\right)^2 . \tag{6.4}$$

For the point-to-plane distance, we also need the normal vector $\vec{\mathbf{n}}_{\mathbf{y}}$ of $\mathbf{y}$. This vector is transformed to the reference frame F_{ref} as well. We typically use the weights $\eta_{\text{pt}} = 0.2$ and $\eta_{\text{pl}} = 0.8$ for our experiments.

Joint Objective $\mathcal{E}_{\text{joint}}$

The joint term constrains neighboring transformations to agree on a common joint location. This preserves a natural connection between different parts of the surface. Our method supports automatically detecting and constraining two types of joints: 3 DOF ball joints and 1 DOF hinge joints. Before we discuss how to detect these joints, we first describe the joint types and how we constrain them in the optimization.

A hinge joint specifies that two transformations always agree on a line in $\mathbb{R}^3$, which means that both transformations transform this line to exactly the same location. We call this line the *hinge axis*, which can be described using the parametric form $\mathbf{u} + t\vec{\mathbf{v}}$, where $t \in \mathbb{R}$. In contrast to the hinge joint, a ball joint says that the transformations agree only on a single point $\mathbf{u} \in \mathbb{R}^3$. We can also express a ball joint in the same form as the hinge joint, except that $\vec{\mathbf{v}} = \vec{\mathbf{0}}$. An example of hinge joints detected for the robot model is illustrated in Figure 6.6.

Once we know these joint locations and types, we can constrain the transformations to agree on the joint locations. Let us represent a joint between transformations for label i and

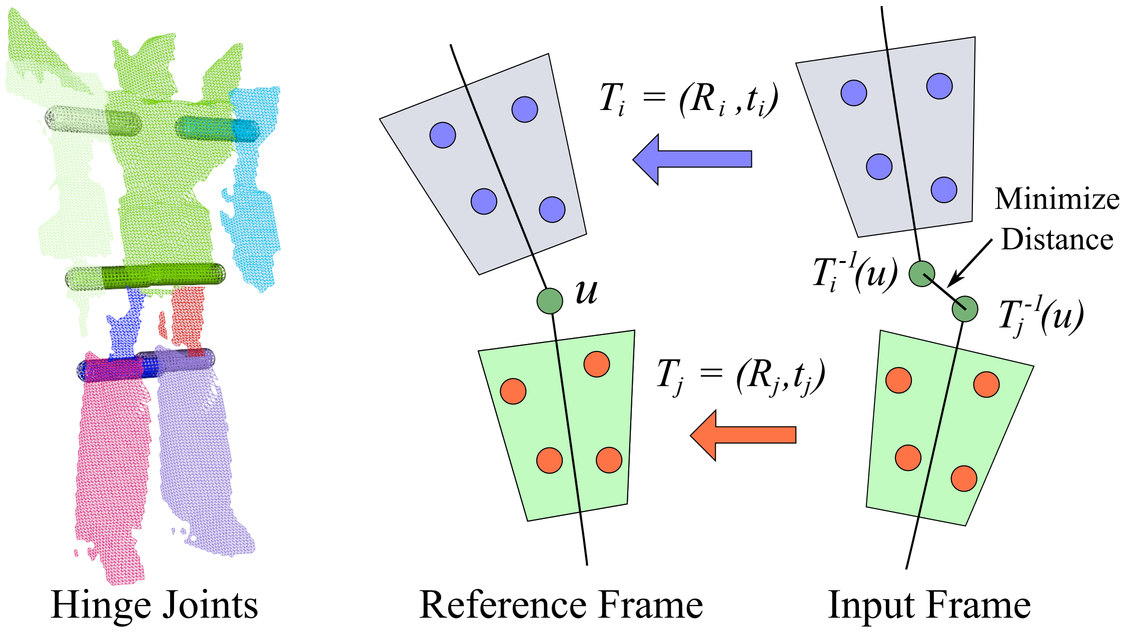


Figure 6.6: Estimating and constraining joints in our optimization. On the left, we show hinge joints that are automatically estimated. These joints are constrained in $\mathcal{E}_{\text{joint}}$ as shown on the right. This term constrains the transformed locations of $\mathbf{u}$ to agree on the same point by minimizing the distance between the transformed locations.

j using the tuple $(\mathbf{u}_{ij}, \vec{\mathbf{v}}_{ij})$. We additionally mark this tuple as valid or invalid depending on whether there actually is a joint between transformations i and j. Note that this also expresses ball joints by simply setting $\vec{\mathbf{v}} = \vec{\mathbf{0}}$. Now, we can constraint the joints using the term $\mathcal{E}_{\text{joint}}$:

$$\mathcal{E}_{\text{joint}}(\mathcal{T}) = \sum_{\text{All } F_f} \sum_{\substack{\text{Valid Joints} \\ (\mathbf{u}_{ij}, \vec{\mathbf{v}}_{ij})}} \sum_{t \in \mathbb{R}^3} \left\| T_i^{(f \to \text{Ref})^{-1}} (\mathbf{u}_{ij} + t\vec{\mathbf{v}}_{ij}) - T_j^{(f \to \text{Ref})^{-1}} (\mathbf{u}_{ij} + t\vec{\mathbf{v}}_{ij}) \right\|^2. \tag{6.5}$$

The inverse of the transformations are used here because the joint locations are defined on the reference frame. This situation is illustrated in Figure 6.6. Also, in practice we use a small number of values for t: typically 20 values of t in the range $[-10s..10s]$ where s is the mesh resolution (or grid sample spacing) [4]. This is the same approach that is used by Knoop et al. [2005]. In the case of a ball joint, we are constraining only one point $\mathbf{u}$ where transformations i and j agree, and for a hinge joint we are individually constraining a set of points along the hinge axis between transformations i and j.

[4]This parameter s is discussed later in the results, Section 6.5.2.

Detecting Joint Locations

To estimate the joints between the transformations, we first need to know which pairs of transformations are likely to share a joint in between. To determine this, we examine the ASG to see which pairs of labels are neighboring on this graph.

Consider the set of the edges E' in the ASG that have different labels assigned to the end points. If we have a large number of edges with labels i and j, this would indicate that transformations i and j are likely to share a joint. On the contrary, a small number of edges (or none) would indicate that the transformations are not related. To determine which label pairs are significant, for each pair i, j we count the number of edges $e \in E'$ whose labels are i, j. Also, for each label i, we count how many edges $e \in E'$ are incident to i (i.e. at least one endpoint is labeled i). The following ratios

$$\frac{\text{\# edges for } i, j}{\text{\# edges incident to } i} \quad \text{and} \quad \frac{\text{\# edges for } i, j}{\text{\# edges incident to } j} \tag{6.6}$$

give a measure of how dominant i, j are for labels i and j, respectively. If either of these ratios exceeds a threshold (set to 15%), then we take the pair i, j as a candidate for sharing a joint.

The edges also give a rough estimate of where we would expect the joint location. For each i, j candidate, we compute the average of all the endpoint locations (on the reference frame) of edges with label i, j. This position, which we denote as $\mathbf{u}_{\text{est}}$, gives us a guess of where the joint location is likely to be.

Once we have a set of candidate label pairs i, j and estimated joint locations $\mathbf{u}_{\text{est}}$, we solve for the true joint locations $\mathbf{u}$ on the reference frame using the transformations estimated so far at all frames. This is done by performing a least-squares minimization

$$\underset{\mathbf{u}\in\mathbb{R}^3}{\operatorname{argmin}} \sum_{\text{All Frames } F_f} \left\| T_i^{(f\to\text{Ref})^{-1}}(\mathbf{u}) - T_j^{(f\to\text{Ref})^{-1}}(\mathbf{u}) \right\|^2. \tag{6.7}$$

For hinge joints, the solution will be a set of points (on the reference frame) lying on the hinge axis. When we solve the above least-squares minimization using the SVD, we can detect hinges by examining if the ratio of the smallest singular value to the sum of the singular values is less than a threshold (set to 0.1 in our implementation). If this is the case, then we truncate the smallest singular value to zero and solve for the equation of the line $\mathbf{u}' + t\vec{\mathbf{v}}'$ satisfying the system. Finally, for the hinge joint parameters $(\mathbf{u}, \vec{\mathbf{v}})$, we take the point $\mathbf{u}$ on this line that is closest to $\mathbf{u}_{\text{est}}$ and normalize $\vec{\mathbf{v}} = \vec{\mathbf{v}}' / \|\vec{\mathbf{v}}'\|$.

If the joint is not a hinge, it will be a ball joint where we determine a single joint location $\mathbf{u}$. In this case, we add an additional regularization term in the optimization:

$$\underset{\mathbf{u}\in\mathbb{R}^3}{\operatorname{argmin}} \sum_{\text{All Frames } F_f} \left\| {T_i^{(f\to\text{Ref})}}^{-1}(\mathbf{u}) - {T_j^{(f\to\text{Ref})}}^{-1}(\mathbf{u}) \right\|^2 + \lambda \|\mathbf{u} - \mathbf{u}_{\text{est}}\|^2. \tag{6.8}$$

where λ is typically 0.1 [Pekelny and Gotsman, 2008]. This additional term helps to pull the location closer to $\mathbf{u}_{\text{est}}$ in case the joint is close to being a hinge (i.e. the first term admits multiple solutions).

Weight Objective $\mathcal{E}_{\textbf{weight}}$

The binary weights transform the problem into a discrete labeling problem, where we try to find an optimal assignment of transformations (interpreted as "labels") to the sample points $\mathbf{x} \in S$. The goal of the weight objective term is to ensure that neighboring samples have a similar label, ensuring that labels form large, contiguous regions on the ASG. Therefore, for $\mathcal{E}_{\text{weight}}$ we use a simple constant penalty when two neighboring samples in S are assigned different labels. We express this using the formula

$$\mathcal{E}_{\text{weight}}(\mathcal{W}) = \sum_{\substack{(\mathbf{x},\mathbf{y})\in E \\ \mathbf{w_x} \neq \mathbf{w_y}}} 1, \tag{6.9}$$

where E is the set of all edges in the ASG. This is the same as in Section 5.3.2, and it is a simple form of the Potts model, which is a discontinuity-preserving interaction penalty [Boykov et al., 2001].

6.3.5 Optimization

To solve the optimization, like we did in Chapter 5 we divide the solver into two phases and alternate between each phase until the solution converges (see Algorithm 6.6). In the first phase, we keep the weights fixed and solve for the transformations (lines 4-11), and in the second phase, we keep the transformations fixed and solve for the weights (lines 15-23). This strategy works well in practice and produces a good alignment within a few iterations (See Chapter 5).

In our experiments, we observed that the transformations for a frame does not change much after the first global registration pass when the frame is first introduced. Therefore, we provide an option in the global registration to solve for the transformations only on the newest k frames that have been introduced to the global registration. We can think of this as solving for the transformations on a sliding window of k frames. Lowering the value of k improves the speed of the registration, while raising this value may produce a more accurate registration at the cost of speed. Note that this only affects the step for optimizing the transformations. The weights are still optimized globally over all frames.

During the global optimization we try to detect if previously occluded parts have reappeared in the new frame (line 12). We discuss how we handle these cases in Section 6.3.6 and 6.3.7.

We observed that the registration was more precise if it terminates after solving for the transformations, rather than terminating after solving for the weights. This is because the optimization is usually able to refine the transformations further after the weights have changed. This is why we have placed the convergence check after we solve the transformations (line 13-14).

Optimizing the Transformations. For optimizing the first phase, we solve for the transformations minimizing the terms $\alpha\, \mathcal{E}_{\text{fit}}(\mathcal{T}, \mathcal{W}) + \beta\, \mathcal{E}_{\text{joint}}(\mathcal{T})$ from Equation 6.2. Since the target positions $\mathbf{y}_j^{(g)}$ corresponding to each sample $\mathbf{x} \in S$ changes depending on the transformations, we use an iterative approach in the spirit of ICP [Besl and McKay, 1992] and alternate between updating the transformations and the corresponding target positions until convergence. Since

Algorithm 6.6: OPTIMIZE $\mathcal{T}, \mathcal{W}$ $(S, E, \mathcal{T}, \mathcal{W}, F_0, \ldots, F_{\text{new}})$

```
Data: Sample set $S$ with associated labels $\mathcal{W}$, transformations for all frames $\mathcal{T}$, A
      list of edges $E$ of the constructed ASG, all initialized input frames
      $F_0, \ldots, F_{\text{new}}$
Result: Optimized transformations and labels $\mathcal{T}, \mathcal{W}$
1  begin
2      Select a subset of frames to optimize (e.g. a sliding window of 1–10 frames);
3      while Not converged do
4          begin (Phase 1: Solve for the transformations $\mathcal{T}$)
5              Re-estimate joint locations and types;
6              while Not converged do
7                  Update the closest points $\mathbf{y}_{j^*}^{(g)}$ for all $\mathbf{x} \in S$ and frames $F_g$;
8                  Construct the sparse matrices for $\mathcal{E}_{\text{fit}}$ and $\mathcal{E}_{\text{joint}}$;
9                  Solve linear system and update transformations;
10                 Check convergence criteria;
11         end
12         Handle reappearing parts in $F_{\text{new}}$ by aligning occluded label regions
           with unmatched surface points (Section 6.3.7);
13         Check convergence criteria;
14         if converged then break;
15         begin (Phase 2: Solve for the labels $\mathcal{W}$)
16             Update the closest points $\mathbf{y}_j^{(g)}$ for all $\mathbf{x} \in S$, frames $F_g$, and $j \in [1..B]$;
17             Precompute $\mathcal{E}_{\text{fit}}$ for each $\mathbf{x} \in S$ and $j \in [1..B]$;
18             Create a graph for $\mathcal{E}_{\text{weight}}$ using the edges $E$ of the ASG;
19             Solve discrete labeling on this graph using $\alpha$-expansion;
20             Discard labeled regions that are too small;
21             Reuse unassigned labels by splitting regions with highest $\mathcal{E}_{\text{fit}}$ error;
22             Update the labels for each $\mathbf{x} \in S$;
23         end
24 end
```

we keep the weights fixed at this step, only the non-zero components will contribute to the $\mathcal{E}_{\text{fit}}$ in this step. Therefore, we only need to update the target positions $\mathbf{y}_{j^*}^{(g)}$ for the currently assigned label j^* at each sample $\mathbf{x}$.

We perform the optimization using the Gauss-Newton algorithm (see Section 3.1.4 for a detailed explanation). We first describe the matrix equations resulting from the linearization of the objective function. Suppose that

$$\begin{aligned}
\mathbf{x}' &= R_j^{(f\to\text{Ref})}\mathbf{x} + \vec{\mathbf{t}}_j^{(f\to\text{Ref})} \\
\mathbf{y}_j^{(g)\prime} &= R_j^{(g\to\text{Ref})}\mathbf{y}_j^{(g)} + \vec{\mathbf{t}}_j^{(g\to\text{Ref})} \\
\vec{\mathbf{n}}'_{\mathbf{y}} &= R_j^{(g\to\text{Ref})}\vec{\mathbf{n}}_{\mathbf{y}}
\end{aligned}$$

that is, they are $\mathbf{x}$, $\mathbf{y}_j^{(g)}$, and the surface normal $\vec{\mathbf{n}}_{\mathbf{y}}$ corresponding to $\mathbf{y}_j^{(g)}$, transformed to the reference frame using the current jth transformation for frame f and g, respectively. Linearizing and rearranging the equation for $\mathcal{E}_{\text{fit}}$, we obtain the equation

$$\begin{bmatrix} -\widehat{(\mathbf{x}')} & I & \widehat{\left(\mathbf{y}_j^{(g)\prime}\right)} & -I \\ -\left(\vec{\mathbf{n}}'_{\mathbf{y}} \times \mathbf{x}'\right)^\top & \vec{\mathbf{n}}_{\mathbf{y}}^{\prime\top} & \left(\vec{\mathbf{n}}'_{\mathbf{y}} \times \mathbf{y}_j^{(g)\prime}\right)^\top & -\vec{\mathbf{n}}_{\mathbf{y}}^{\prime\top} \end{bmatrix} \begin{bmatrix} \omega_j^{(f)} \\ v_j^{(f)} \\ \omega_j^{(g)} \\ v_j^{(g)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}' - \mathbf{y}_j^{(g)\prime} \\ \vec{\mathbf{n}}_{\mathbf{y}}^{\prime\top}\left(\mathbf{x}' - \mathbf{y}_j^{(g)\prime}\right) \end{bmatrix}. \tag{6.10}$$

for both the point-to-point term and the point-to-plane term, respectively (see Equation (6.4)). In the point-to-plane term, we have simplified the objective by just using the pre-rotated normal $\vec{\mathbf{n}}'_{\mathbf{y}}$, instead of applying an additional local rotation $\omega_j^{(g)}$ to the normal. Also, notice that the top row in the above is actually three rows in the matrix (because each term is a 3×3 matrix) and the bottom row is a single row in the matrix. We also weight the top row (on both sides) with $\sqrt{\eta_{\text{pt}}}$ and the bottom row with $\sqrt{\eta_{\text{pl}}}$ to weigh the two terms appropriately. For the joint constraint term $\mathcal{E}_{\text{joint}}$, we obtain the equation

$$\begin{bmatrix} R_i^{(f\to\text{Ref})}\widehat{\mathbf{u}} & -R_i^{(f\to\text{Ref})} & -R_j^{(f\to\text{Ref})}\widehat{\mathbf{u}} & R_j^{(f\to\text{Ref})} \end{bmatrix} \begin{bmatrix} \omega_i^{(f)} \\ v_i^{(f)} \\ \omega_j^{(f)} \\ v_j^{(f)} \end{bmatrix} = \begin{bmatrix} \mathbf{u}'_i - \mathbf{u}'_j \end{bmatrix}. \tag{6.11}$$

where $\mathbf{u}'_i = R_i^{(f\to\text{Ref})}\mathbf{u} + \vec{\mathbf{t}}_i^{(f\to\text{Ref})}$ and $\mathbf{u}'_j = R_j^{(f\to\text{Ref})}\mathbf{u} + \vec{\mathbf{t}}_j^{(f\to\text{Ref})}$. Here, any variables ω, v appearing in the leftmost matrix were removed, because the linearization was evaluated at $[v, \omega] = \mathbf{0}$ (which corresponds to a linearization at the current estimated transformations). Gathering these equations in a big sparse matrix and solving the resulting linear least-squares problem, we obtain a step for the rigid transformations. To complete the iteration in the Gauss-Newton algorithm, we apply this step to the current rigid transformations using the exponential map (see Section 3.1.4). To solve for the transformations on a limited number of frames, we can simply remove the variables and constraints involving transformations from frames outside of the set of interest. This can significantly reduce the time needed to perform this step.

Optimizing the Weights. For the second phase, we solve the discrete labeling problem of $\alpha\,\mathcal{E}_{\text{fit}} + \gamma\,\mathcal{E}_{\text{weight}}$ using the α-expansion algorithm [Boykov et al., 2001; Boykov and Kolmogorov, 2004; Kolmogorov and Zabih, 2004]. We use the ASG directly to specify smoothness constraints between points. Unlike solving for the transformations, we always solve for the weights as a global optimization over all frames. This means that we need to compute the $\mathcal{E}_{\text{fit}}$ term over all frames and all labels.

Since the transformations are kept fixed in this stage, we can precompute $\mathcal{E}_{\text{fit}}$ to save computation time during the optimization. Here we simply compute the distance (6.4) between $\mathbf{x}$ and $\mathbf{y}_j^{(g)}$) over all samples $\mathbf{x}$ and all labels j. We then store the resulting in a hash table for a quick look-up during the optimization.

After the optimization, it may be the case that some labels are only assigned to a few sample points, or other labels may not be assigned at all. In the first case, we can discard these labels to obtain a solution that is simpler and not substantially different from the original solution. In our implementation, we discard a label if the percentage of samples assigned to that label is less than some small threshold (set to 1% in our implementation). In the second case, we reuse labels like we did for the pairwise case in Section 5.3.2.

Checking for Convergence. To detect if the optimization for the transformations has converged, we monitor the change of the objective function by examining the value of the minimized residual. We apply the criterion $|F_k - F_{k+1}| < \epsilon(1 + F_k)$ (where $\epsilon = 1.0 \times 10^{-6}$) and stop the iteration if this condition is met. We also have a maximum number of iterations, typically set to about 20–30 iterations, and stop if we exceed this maximum number. In our experiments, we observed that in most cases the optimization converges in about 10–15 iterations. However, the optimization may enter an oscillating mode, where the closest points in each iteration of the T-step switch back and forth indefinitely between a few points. Because of this, convergence is not guaranteed; but in practice we have not encountered any major problems.

6.3.6 Treating Occlusion

When a part of the surface is completely missing in a frame, the transformation for this part may have few or no valid correspondences constraining it in the optimization. In these cases, it may not be possible to solve for the rigid transformation of that part in that particular frame. In our algorithm, we automatically detect if this happens and exclude these transformations from the optimization. We perform this occlusion check right after we initialize a new frame (see Algorithm 6.1). If the initialization was unable to handle the case of a reappearing part (that was previously occluded), then we try to detect and align this part automatically during the global optimization that follows. We discuss how we handle the reappearing case in the next section, but first we discuss more details on how the occlusion is detected and handled.

To decide if a part is occluded, we update the closest points for this frame after the initialization. Then, we count the number of target positions $\mathbf{y}_j^{(g)}$ for each label. If this number falls below a small threshold (either below 5 points, or below 5% of the total number of samples for that label), then we consider the label as occluded for this frame.

We then exclude all occluded labels from the optimization. For optimizing the transformations, we simply remove the variables and constraints involving these occluded labels from the objective term. In this case, there will be no transformation that is estimated for the occluded label in the frame. However, it is still possible to estimate a reasonable transformation based on the joint constraints with neighboring transformations. We adopt the strategy of Pekelny and Gotsman [2008] and find an approximate transformation:

- If there is one neighbor, we copy this transformation for the occluded one,

- If there are two neighbors connected via ball joints, then we find the rigid transformation that fits both joints and minimizes the relative rotation between the joints,
- If there are two neighbors and at least one is a hinge, or there are more than two neighbors, we find the rigid transformation that best fits all of the joint constraints.

If there are no neighbors, then we just use the transformation of the last frame where it was not occluded. We estimate the transformation for occluded labels according to this strategy every time the transformations are optimized in the global registration.

For optimizing the weights, the situation is more complicated: since the weights are optimized globally over all frames, we cannot remove a label entirely from the optimization just because it was occluded some frames. Therefore, we must decide on an appropriate $\mathcal{E}_{\text{fit}}$ value when it involves a frame with an occluded label.

We cannot always rely on the approximate estimated transformation to give a reasonable value of $\mathcal{E}_{\text{fit}}$. A zero error or very low error does not work either, because then the occluded label may be assigned to strange locations where it produces a lower error than the actual "correct" label. On the other hand, assigning a high value may discourage from assigning this label to the "correct" locations, where the surface of the part is partially visible in the frame. In the end, we settle on a heuristic: we use the error value of the current label assigned to the sample, minus a small epsilon. If the current label is occluded as well, then we use the minimum error among all unoccluded labels at that sample. This heuristic worked well in our experiments, but there was still a handful of cases where the occluded label was assigned to a completely unrelated location.

6.3.7 Reappearing Parts

There is also the case where the occluded part may suddenly reappear in a new frame. This is not handled by our initialization step, because it can only align parts that appear in *both* the source and target. Basically there are two possibilities for reappearing parts. First, if the part reappears nearby its last approximated location, then the algorithm will be able to find a sufficient number of closest points to start tracking again. However, the other possibility is that the part reappears in a completely different location, and there are not enough closest point correspondences to start tracking again.

In this case, a large number of scanned points in the frame will not "overlap" with the sample set S. If the number of such "unmatched" points exceeds a threshold (10% of the total points in the frame), we attempt to align the occluded parts with these unmatched points. This is performed after each optimization of the transformations (line 12 of Algorithm 6.6). Here, we use the same procedure to optimize for the transformations (Section 6.3.5), but with some changes where

- we optimize only for the occluded transformations,
- we set the closest point threshold τ_d and normal angle threshold τ_n much higher (Section 6.3.4),
- and we increase the weight of the $\mathcal{E}_{\text{joint}}$ (β in Equation 6.2) to be very high.

After the match, we run the occlusion detection routine once more to check if we have obtained a sufficient number of target points to start tracking the part again.

6.4 Post-Processing

After the global registration, we have aligned all frames and we can reconstruct the surface of the entire model. To do this, we just resample the set S with a small sample distance τ_{samples}, where we use all of the points in each frame to resample S (instead of taking the uniform subsamples U_f as candidates). This results in a dense sample set S, which we can use to reconstruct a surface mesh using any favorite surface reconstruction algorithm. In our results, we use the streaming wavelet surface reconstruction algorithm by Manson et al. [2008]. Since our algorithm gives only binary weights at each point there may be artifacts at boundaries between parts. To reduce these artifacts, we can solve for smooth skinning weights at each point as a post-processing step. We do not apply this smoothing step in our results in Section 6.5, so that we can give a clearer picture of the quality using our binary weight pipeline.

6.5 Experimental Results

6.5.1 Reconstruction

We implemented our algorithm in C++ and tested it with several real-world and synthetic datasets exhibiting articulated motion. The car and robot datasets were acquired by Pekelny and Gotsman [2008] using a Vialux Z-Snapper depth camera. These sequences were created by animating the physical model, while capturing each frame from a different viewpoint. Each sequence has 90 frames, and consists of 5 and 7 parts, respectively. The reconstruction results using our algorithm are shown in Figures Figure 6.7 and Figure 6.8. The top row shows some of the input frames in the sequence. Notice that there is a significant amount of occlusion in some of the frames. The second row shows the labeled sparse sample set S used by our algorithm, and the third row shows the dense sample set obtained using a smaller τ_{sample} in the post-processing step. There are still some holes on the surface, which are locations that were occluded in all input frames, or locations where the algorithm could not extrapolate the label (Section 6.3.2). The fourth row shows the reconstructed mesh using the algorithm by Manson et al. [2008] (kindly provided by the authors), with labels on each vertex obtained by taking the label of the nearest point in the dense sample set. Since we meshed a single, closed surface in the pose of the reference frame, there are some stretching artifacts on the boundary between neighboring parts. This could be corrected by meshing each part separately, or by meshing the surface in a pose where the parts are further apart. Finally, the fifth row shows the estimated joint locations. Hinge joints are represented by a short stick, where ball joints are representing using a sphere. The reconstruction results are good, demonstrating that we can obtain an accurate registration without a segmentation given as input by the user. For the car dataset, our algorithm preferred a simpler configuration of 4 parts, instead of creating a separate part for the small rotating base in the middle. We think that this is a reasonable reconstruction of the car, because the surface for the rotating base is quite small.

To test our algorithm on a more deformable subject, we acquired sequences of a bendable, poseable pink panther toy. These sequences were acquired using a Konica Minolta

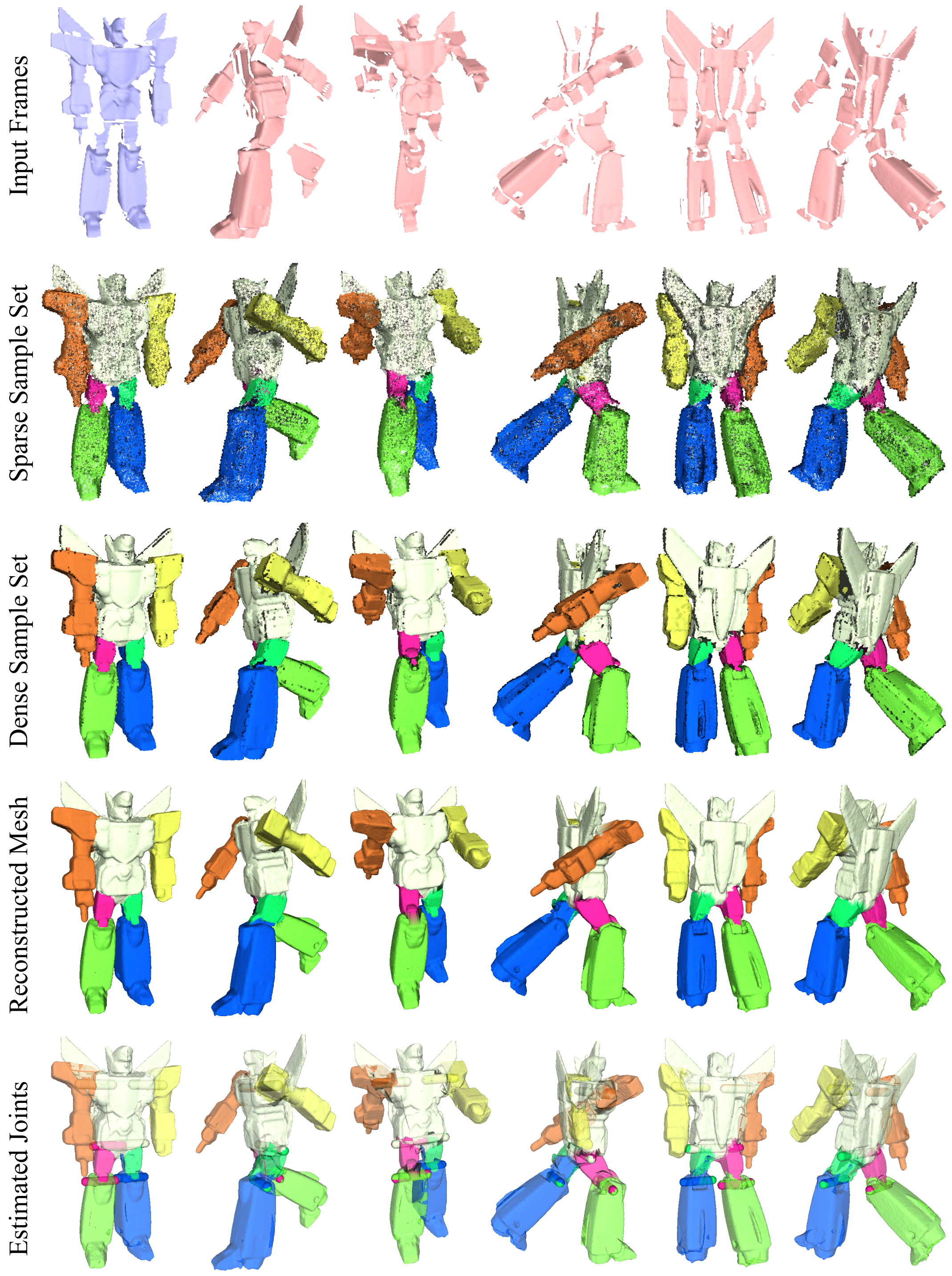


Figure 6.7: Reconstruction results for the robot dataset.

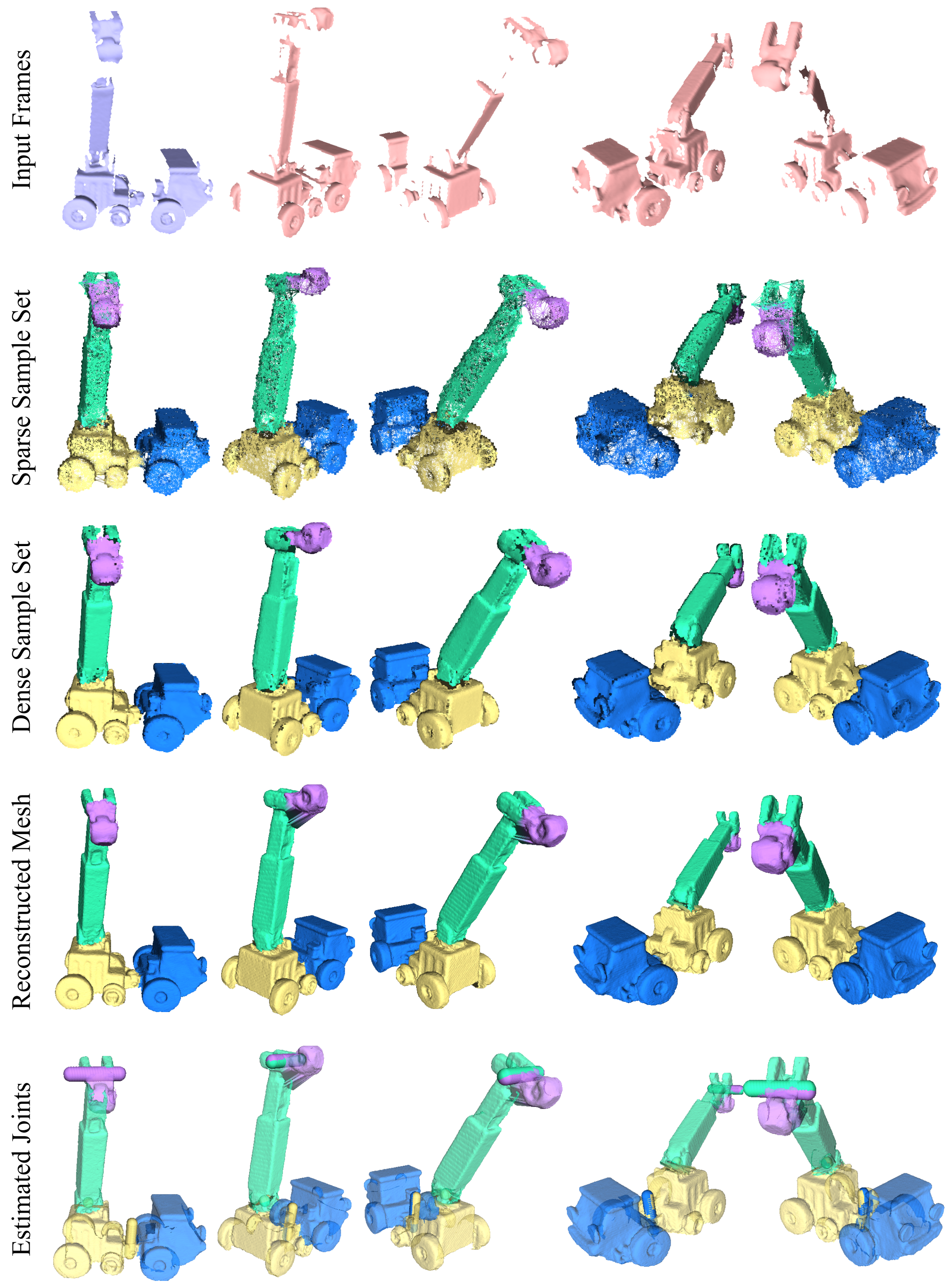


Figure 6.8: Reconstruction results for the car dataset.

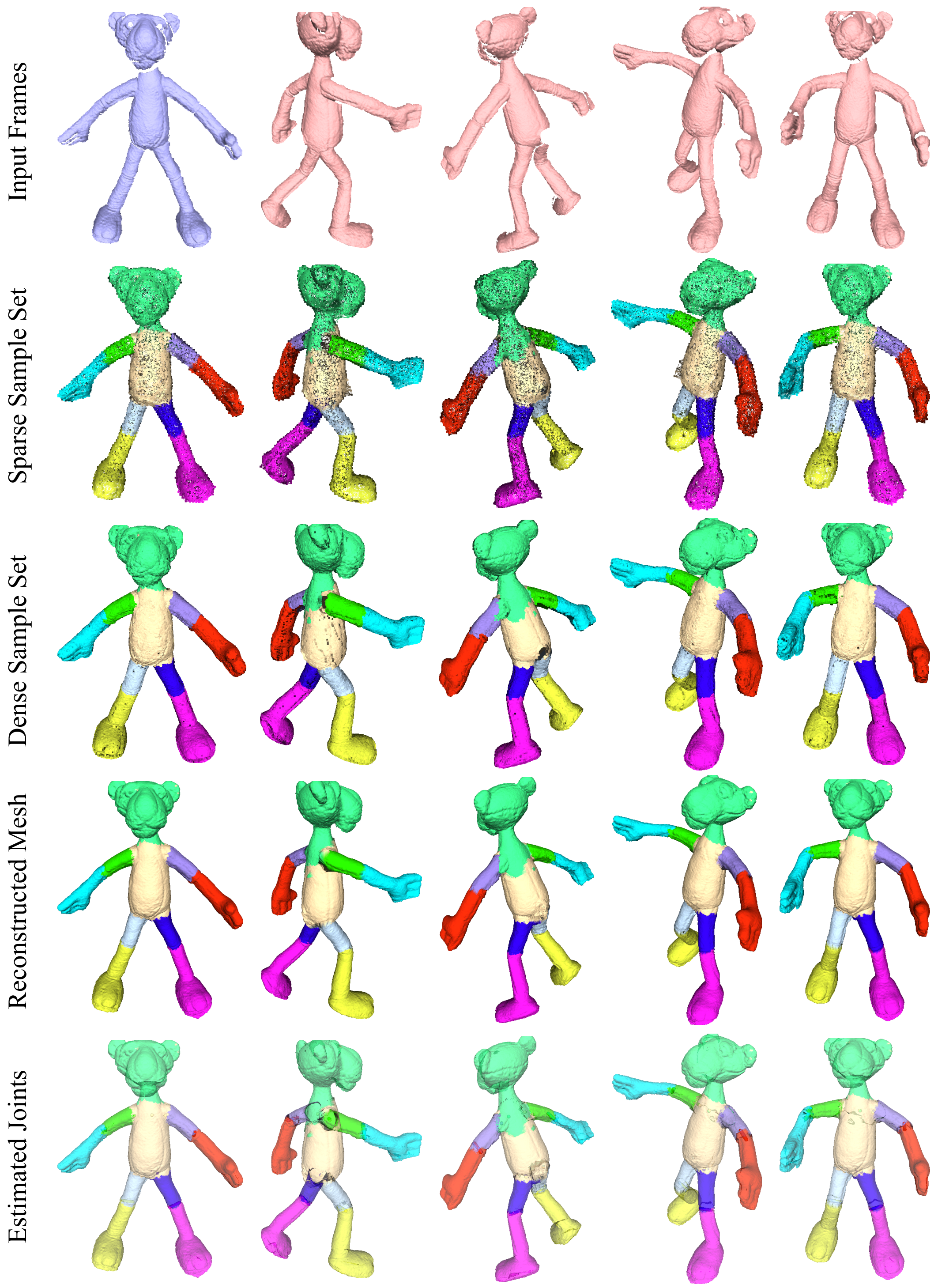


Figure 6.9: Reconstruction results for the first Pink Panther dataset.

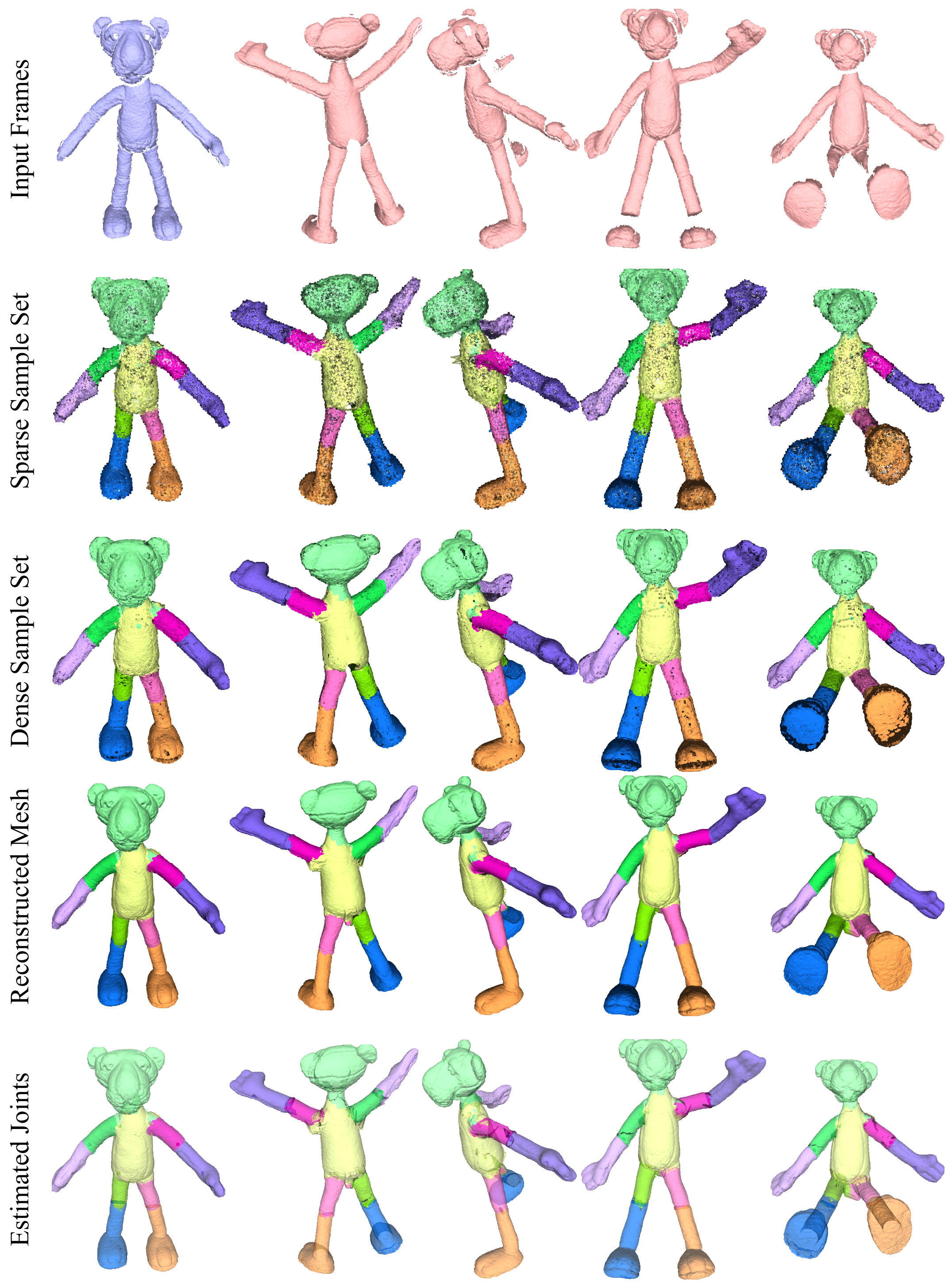


Figure 6.10: Reconstruction results for the second Pink Panther dataset.

VI-910 laser scanner. Each sequence has 40 frames consisting of 10 parts each. In the first sequence, we animated the toy with small motions while capturing each frame at a different viewpoint. In the second sequence we created larger motions of the toy while changing the viewpoint. In addition, the furry texture on the toy created a significant amount of noise on the scanned surface. The reconstruction results, shown in Figures 6.9 and 6.10, are very good for both the small motion and the large-motion case, except for some minor stretching artifacts on the reconstructed mesh, at the boundary between parts.

Finally, we generated synthetic depth sequences of a walking man, where the camera is rotating around the subject. These sequences were created by capturing the Z-buffer of an OpenGL rendering, and the modelview and projection matrices were inverted to convert the depth values into 3D coordinates. To test the effect of occlusion in our algorithm, we captured the first sequence using a single camera, and the second sequence using two cameras which were $90°$ apart. Since the frames were very close to each other, we did not use the transformation assignment initialization for these sequences, and we reduced the sliding window size from 5 frames (for the first $\sim$10 frames) down to 1 frame (for the rest of the sequence). The reconstruction results are shown in Figures 6.11 and 6.12. The first sequence was less successful due to the large amount of occlusion of the arms. In particular, both the left arm and right arm disappear from the front and reappear in the back, causing some alignment errors in the middle of the sequence. This resulted in a “larger” left hand, where the algorithm did not align the hand well and added extra points for this part. Also, in some of these frames, the arm and hand partially appeared but was not tracked, and this resulted in some “floating parts.” Nevertheless, the reconstructed mesh nicely approximates the entire shape. In the second sequence, the arm and hand do not disappear completely, and the algorithm is able to track all parts accurately for the entire sequence. This results in a very accurate reconstruction (especially for the hands). The stretching artifacts are more noticeable for these datasets, where the torso and arm connect together and also the hip region where the surface stretches significantly.

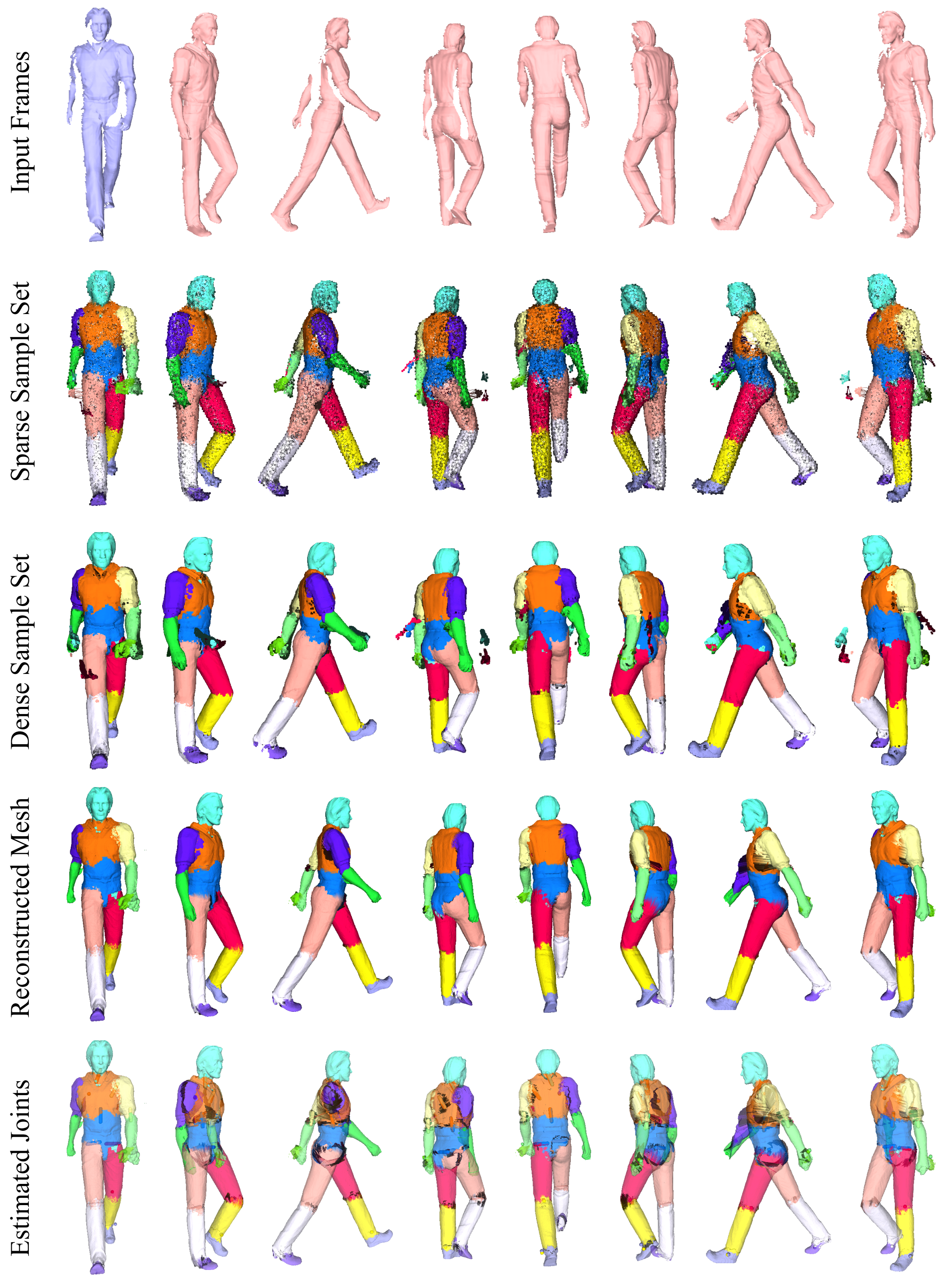


Figure 6.11: Reconstruction results for the synthetic Walking Man dataset taken using a single virtual camera.

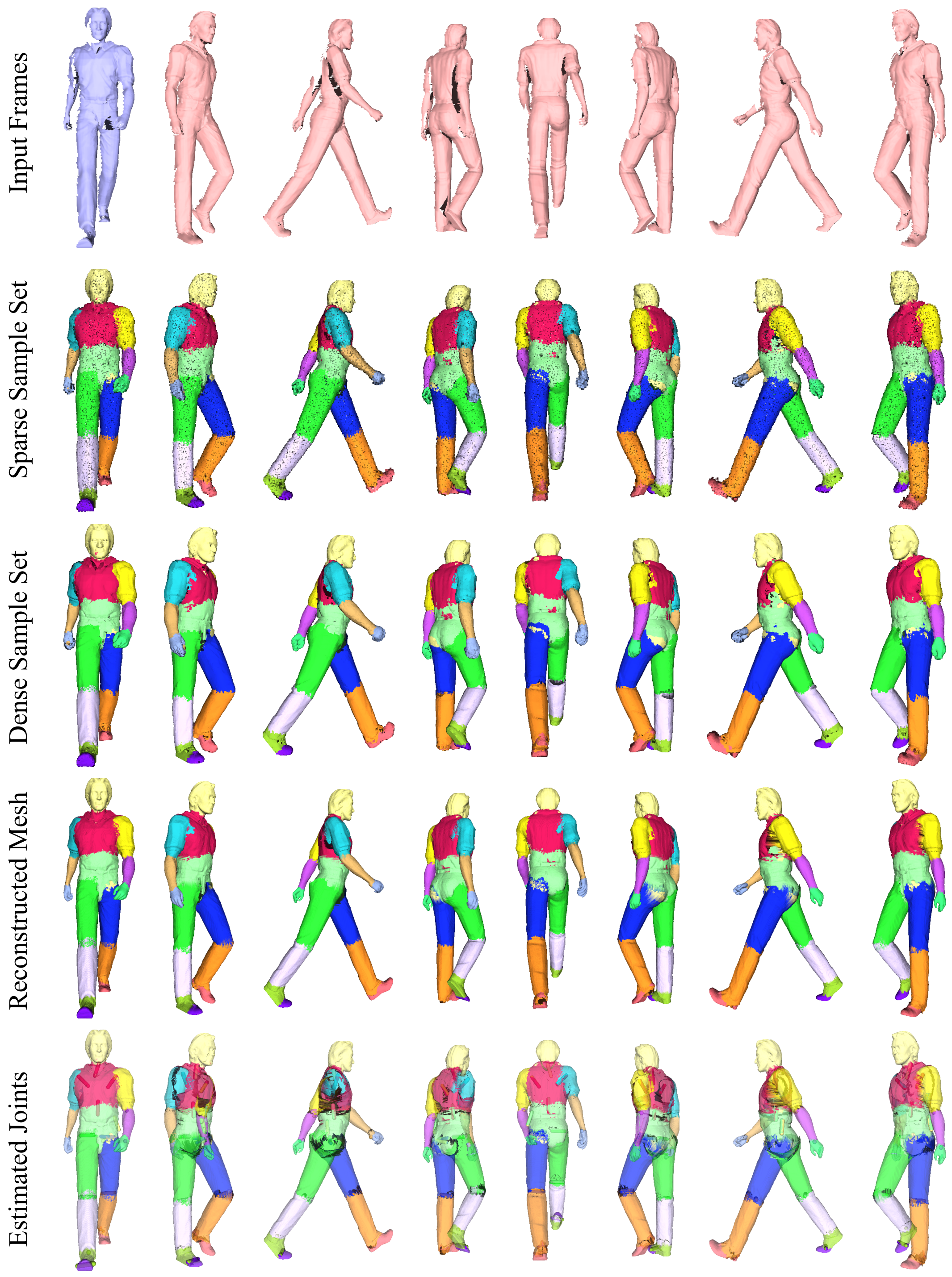


Figure 6.12: Reconstruction results for the synthetic Walking Man dataset taken using two virtual cameras.

6.5.2 Parameters

The main parameters of our algorithm are the the number of transformations B, weights for each term in the optimization and thresholds that control sampling and closest point computation. Although the user needs to specify the number of transformations to approximate the motion, the algorithm may settle on a smaller number of transformations if the registration error is small enough. An alternative strategy would be to have the user specify a maximum alignment error ϵ and make the algorithm add part labels until the alignment error is accurate within this ϵ. We did not explore this alternative, but this ϵ parameter would be similar to directly specifying the number of transformations.

We expressed many parameters relative to the grid sample spacing s, which is the average distance between samples in each frame. For the weights of each term in the objective function (6.2), we used $\alpha = 1$, β between 0.1 and 1.5, and γ either $0.5s$ or s. For the uniform subsampling U_f (Section 6.3.2), we specified a fraction of points to sample for the entire sequence, typically between 6% and 20% depending on the density of the scans. For the sample spacing parameter τ_{sample}, we used a value between $2s$ and $5s$ depending on how dense we wanted the sparse sample set to be. Finally, for determining if the closest point is valid (Section 6.3.4), we used $\tau_d = 10s, \tau_n = 45°$, and $\tau_b = s$. This changes when we match reappearing parts, for which we used τ_d between $50s$ and $100s$, τ_n between 45° and 80°, and $\beta = 100$. In our experiments, we experimented with a few different parameter settings but did not seriously optimize the parameters to give a better result.

6.5.3 Performance

The performance of our implementation using a single core of an Intel Xeon 2.5 GHz processor is reported in Table 6.1. In the robot and car datasets, the most time-consuming part was the initialization, but in the other cases it was the global registration. The global registration step can execute faster if a smaller sliding window is used, with the trade-off of having a less accurate registration. Like most ICP-based algorithms, the most time-consuming part is the closest point computation, which can typically take 30% of the total time. Note that

Table 6.1: Performance statistics for our experiments. The timings are expressed in seconds, and the bottom row reports the average execution time per frame in each sequence.

Statistic	**Robot**	**Car**	**PP1**	**PP2**	**Walking1**	**Walking2**
Max Bones	7	7	10	10	16	16
Used Bones	7	4	10	10	14	16
Frames	90	90	40	40	121	121
Sliding Window	5	5	5	5	$5 \rightarrow 1$	$5 \rightarrow 1$
Points/Frame	9,391.2	5,387.86	36,683.9	30,003.1	19,843.7	39,699.7
Total Points	845,208	484,907	1,227,356	1,200,125	2,401,082	4,803,662
Samples	4,970	2,672	4,077	4,203	8,305	8,539
Edges in ASG	37,678	20,707	30,758	31,841	61,711	63,043
Initialization	7,357.68	2,652.57	1,826.27	1,828.98	69.38	134.74
Global Reg	2,287.61	1,200.04	2,184.68	2,624.4	5,574.86	19,789.0
Resampling ASG	264.44	117.93	67.90	68.06	876.32	1,617.07
Total Time	9,909.73	3,970.54	4,079.85	4,521.44	6,520.56	21,540.81
Average Time	110.11	44.12	102.00	113.04	53.89	178.02

the times in the initialization step reported in Table 6.1 do not include the preprocessing time to compute spin images and estimate the principal curvature frame at each vertex.

6.5.4 Inverse-Kinematics Application

Solving for the weights and joints in the model is useful for re-posing and animating the reconstructed model. To demonstrate this, we implemented a tool to perform inverse kinematics on the reconstructed model. In this system, the user specifies point constraints, and we use our optimization of the transformations (Section 6.3.5) to satisfy these constraints. The result was an interactive tool for the user to intuitively re-pose and animate the reconstructed subject. Figure 6.13 shows examples of different poses of the robot created by our system.

6.5.5 Sequential Registration vs. Simultaneous Registration

To illustrate the benefit of performing simultaneous registration, we compare our algorithm with a sequential registration pipeline. In a sequential registration method, we optimize each frame of the sequence one-by-one, accumulate new samples directly on the reference frame, and discard the frame before moving on to the next. This strategy uses only

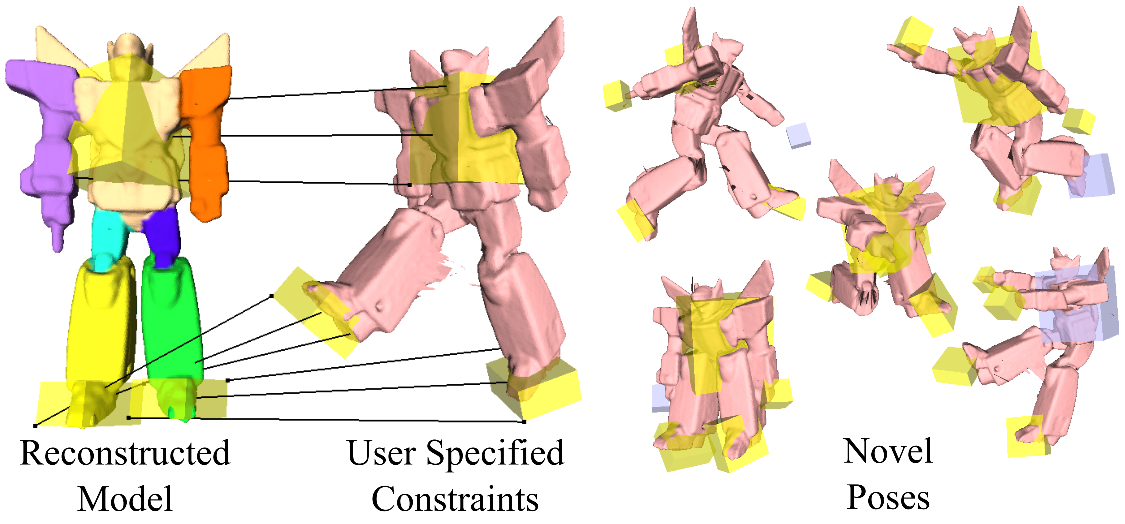
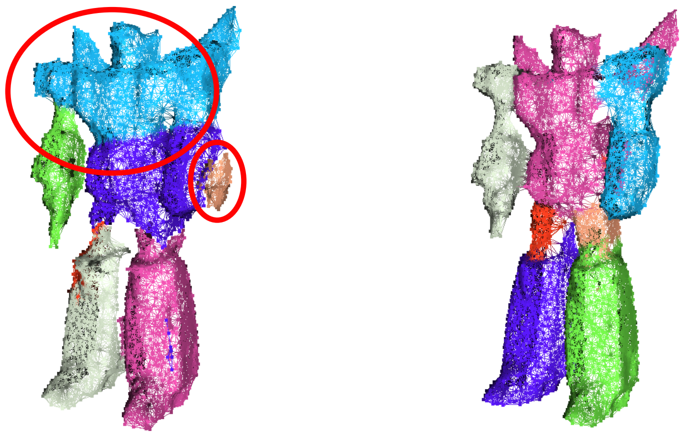


Figure 6.13: Reposing the reconstructed robot. By using the solved weights and the hinge joints, our optimization can satisfy point constraints given by the user.

correspondences between the accumulated samples and the current frame being optimized (for estimating both transformations and weights). Therefore, this strategy is essentially a pairwise registration that is applied repeatedly for each frame.

The main problem with the sequential registration approach is that it cannot reliably estimate the articulated structure (i.e. weights) based on the movement observed in just one frame. This complicates the situation further for occlusion detection and recovery, which rely on a reliable estimate of the articulated structure. A comparison between the sequential and simultaneous strategies is shown in Figure 6.14. Here, we have used the two strategies to align 40 robot frames, and we display the sparse ASG which roughly shows the estimated geometry. On the left, we can see that the sequential strategy did not produce a correct labeling. As a result, the registration was imprecise, and "extra" surfaces appear where the parts were not aligned properly (for example, on the left arm). On the right, we show a result obtained by simultaneous registration, where we kept the same parameters, used a 1-frame window for optimizing the transformations, and used the correspondences from all frames to optimize the weights. The result has a correct labeling that reflects the movement in all frames, and the registration and estimated geometry are precise.



(a) Sequential Registration (b) Simultaneous Registration

Figure 6.14: Comparing sequential and simultaneous registration. (a) As indicated by the large red circle on the upper body area, the sequential strategy gives an unreliable estimate of the articulated structure, because it only uses the movement observed in one frame. This leads to an imprecise registration, for example, in the left arm indicated by the smaller circle. (b) The simultaneous strategy can correctly estimate the structure that fits the movement observed in all frames. The registration is more precise, as well as the estimated surface geometry.

6.5.6 Grid-Based Weights vs. Graph-Based Weights

To compare the benefit of using a graph for defining the weight function vs. using a grid like we did in the last chapter, we implemented the simultaneous registration using a grid and compared the results. First, we found that the performance of the graph-based registration is much faster, because the grid-based method has an additional overhead of translating the weights from the grid to the samples. For registering the 90 frame robot sequence, the global registration took a total of 144.00 seconds per frame using the grid strategy, but it only took 28.36 seconds per frame for the graph based strategy (excluding the time for initialization).

Second, the graph-based representation dealt robustly with topology issues. An example of this is shown in Figure 6.15, where we display the grid and graph deformed according to the optimized weights and transformations. Unlike the graph based solution on the right, the grid based solution on the left shows many artifacts. This is because when the resolution of the grid is too coarse, a single grid cell overlaps multiple separate parts. In this example, there are several grid cells that overlap a little with both the right leg and the left leg of the robot. As a result, different weights are assigned to either side of the cell, so the cell “stretches” apart,

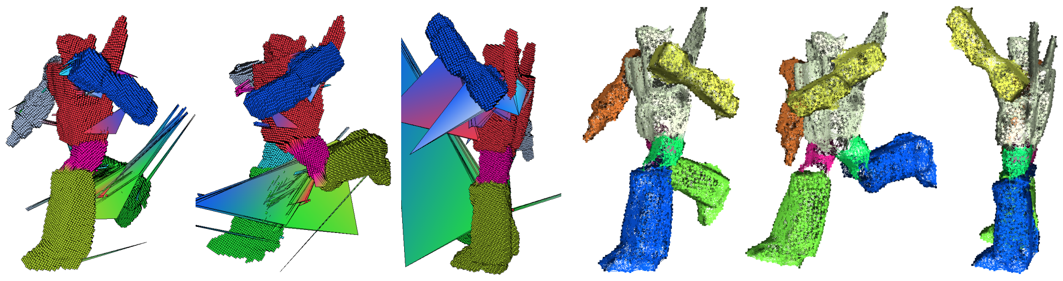

(a) Result Using a Grid-Based Representation (144.00 sec/frame)
(b) Result Using a Graph-Based Representation (28.36 sec/frame)

Figure 6.15: Comparing grid-based and graph-based weight representations. These images show the represented weight function, deformed into different poses according to the optimized transformations and weights. Notice the deformation artifacts with the grid-based representation, which is absent in the graph-based representation.

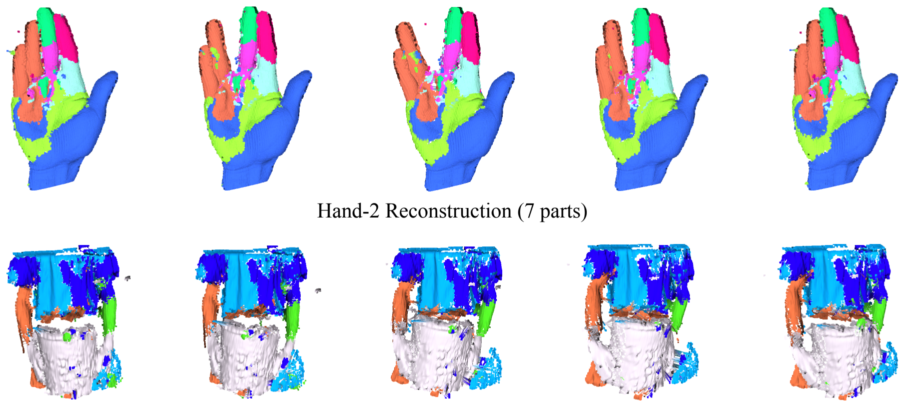

Hand-2 Reconstruction (7 parts)
Popcorn Tin Reconstruction (5 parts)

Figure 6.16: Articulated registration on the hand-2 and popcorn tin datasets used by Wand et al. [2009]. Our algorithm is able to produce coarse approximations of the non-rigid motion exhibited in these datasets.

causing the artifact that we see. This stretching behavior makes it difficult to look up weights for the scanned points inside this cell, and so we “lose” points in these situations. In contrast, for the graph-based strategy, since we define weights directly on each sample, it does not suffer from this issue. Furthermore, we can prune edges of the graph based on the optimized motion, so it handles these topology issues robustly.

6.5.7 Comparison with Wand et al. [2009]

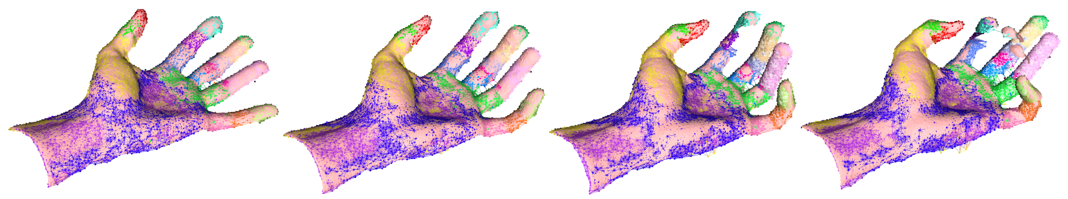
We compare our articulated reconstruction with the deformable reconstruction method by Wand et al. [2009]. For the car, robot, and pink panther datasets, their method was not able to fully reconstruct these sequences because there was too much motion between the frames. However, they were able to reconstruct some subsequences of these datasets. This is because they rely only on a local optimization using closest points, whereas our method uses a robust initialization that is able to automatically handle frames with large motion.

We also tested our algorithm on several examples from Wand et al. [2009]. Figure 6.16 shows reconstructions of the hand-2 and popcorn tin datasets, and Figure 6.17 shows a result for the grasping hand (hand-1) dataset. These sequences exhibit non-rigid motion, especially the popcorn tin dataset. Our algorithm can successfully capture the overall shape and produce a coarse articulated motion of the subject. However, we see that it does not reproduce fine details in the surface deformation.

6.6 Summary and Conclusion

We have presented a method to reconstruct an articulated shape from a set of range scans. From a sequence of range scans, we solve for the division of the surface into parts and the motion for each part to align all input scans. For this purpose, we first improved a transformation sampling and assignment strategy to obtain a robust initialization of the registration between pairs of adjacent frames in the sequence. Then, we formulated a simultaneous registration for all input frames to minimize registration error. This optimization included joint constraints that preserves the connectivity of each part, and automatically handles cases where parts are occluded or they reappear. We demonstrated that we can reconstruct a full 3D articulated model without relying on markers, an user-provided segmentation, or a template. Finally, we have demonstrated that the reconstructed model can be targeted in new poses for the purpose of creating an animation.

A limitation of our method is that there needs to be enough overlap between adjacent frames in the range scan sequence to obtain a good alignment. For example, if one frame captures the surface on the front of the object, and the next frame has the surface from the back of the object, then there will be not enough overlap to match these frames together in the registration. This means that the order of the range scans in the sequence should maintain



(a) Frame 1 (b) Frame 9 (c) Frame 15 (d) Frame 21

Figure 6.17: Registration for a hand sequence, where the hand starts from an open pose and gradually closes to a grasping pose. These pictures show the input data (displayed as a red color mesh) and the sparse ASG. Our algorithm tracks the hand well in the first part of the animation, where most of the surface is visible. In (c), the surface of the fingers start to gradually disappear, and the middle segment of the index finger starts to lose track and rotate backwards. In (d), the algorithm loses track of the middle and ring fingers, because most of these fingers are occluded (except for the fingertips).

a reasonable amount of overlap between every adjacent pair of frames. A temporal ordering of the scans, for example, would produce a sequence with a reasonable amount of overlap. Sometimes even this is not enough when there is severe occlusion. For example, our algorithm loses track of the fingers in the hand sequence because of too much missing data, as shown in Figure 6.17.

Another shortcoming of our ICP-based registration is the handling of “slippable” parts such as cylinders. For example, the fingers of a hand example shown in Figure 6.17 have cylindrical symmetry, and the ICP registration could converge into a state where the segments of the fingers are “twisted” or rotated about the axis of symmetry (Figure 6.17c). Although hinge joints could disambiguate cylindrical symmetries, we found that it was difficult to estimate accurate hinge joints in this case.

Our algorithm does not estimate scale, so it cannot handle the range scans where the scale of the object changes. While this was not a problem for any of our examples, automatically estimating scale changes could help capture regions that are stretching. Furthermore, it would be interesting to adapt our algorithm for aligning completely non-rigid examples. For this case, estimating “flexible” transformations would be appropriate (perhaps affine transformations), and it would be useful to find a way to optimize for smooth weights without causing overfitting. We believe that there should be a middle ground between specifying a separate transformation on every sample point [Sumner et al., 2007] and our method (solving for the weight at each sample point).

We would also like to reduce the parameters in our algorithm. An alternative to specifying various thresholds is to use robust error metric similar to the work of Nishino and Ikeuchi [2002]. In this case, the outliers would automatically be identified during the optimization, without a need to specify hard thresholds.

Finally, we would like to investigate ways of improving the performance of the algorithm. In particular, since our method estimates the weights and transformations for all frames simultaneously, we need to keep all of the input scans in memory. We would like to develop a streaming version of our algorithm that reduces the memory requirements. This would allow us to process longer sequences of range scans. In addition, once a reasonable segmentation is obtained, only the transformations need to be solved for each frame. We believe that this could be implemented in real-time to be used for various markerless motion capture applications.

6.7 Acknowledgments

We wish to thank Y. Pekelny and C. Gotsman for sharing the car and robot datasets, and we thank T. Weise for the hand dataset. Additional thanks to G. Debunne for providing the libQGLViewer library, D. M. Mount and S. Arya for providing the ANN library, and Y. Boykov, O. Veksler, R. Zabih for providing an implementation of their graph cuts algorithm, and J. Manson for providing surface reconstruction software.

The material in this chapter is, in part, a reproduction of material being prepared for submission for publication: Will Chang and Matthias Zwicker, “Global Registration of Dynamic Range Scans for Articulated Model Reconstruction.” The dissertation author was the primary investigator and author of this paper.

7

Conclusions and Future Work

THE vision of our work is to enable efficient acquisition and synthesis of highly detailed 3D surface models that are also easy to animate in a plausible and realistic way. This technology to capture and manipulate dynamic 3D models has many potential applications in human-computer interaction, manufacturing, medicine, virtual & augmented reality, and computer animation. While range scanners can acquire high-speed, high-resolution geometric data, each scan only offers a partial view of the surface, with no tracking of the motion. To reconstruct a complete model of the geometry and motion of the subject, we must align multiple range scans taken from different times and viewpoints to fill in the missing data and track the motion of the surface.

7.1 Contributions

In this dissertation, we focused on algorithms to process and align range scans of a moving articulated subject. We demonstrated that these methods can align range scans in a completely unsupervised way: without markers, a template, or a user-defined segmentation of the surface. A distinguishing feature of our research was the key observation that discrete optimization is useful for automatically estimating the division of the surface into parts, based on the observed motion of the surface.

Our first contribution was a method to robustly align a pair of shapes by optimizing an assignment of transformations on the surface. In this method, we first matched feature descriptors and clustered the resulting transformations to find a set of candidate surface movements. We then formulated a discrete labeling problem to solve for the best transformation assignment that minimizes the alignment error between the two shapes, while preserving the original structure of each shape. We found that this problem is efficiently solved by graph cuts, leading to an algorithm to align surfaces that is robust to large motions, partial surface data, and multiple similar parts.

Next, we presented a technique to solve for the parameters of a reduced deformable model (RDM) that produces the best alignment of a pair of surfaces. We showed that it is possible to align these surfaces by repeatedly estimating the transformations and weights in alternating fashion, similar to the EM algorithm. This resulted in an efficient alignment of articulated range scans, due to the few number of transformations and the automatic estimation of influence weights in the optimization.

Finally, we combined these techniques to automatically construct an articulated 3D model. We improved the performance of the algorithms by subsampling the surface points and solving for influence weights defined directly on the samples themselves. Then, we formulated an articulated global registration that solves for the transformations and the weights of an RDM simultaneously over all frames, while handling joint constraints and dealing with the occlusion and reappearing of parts. We demonstrated that this method can reconstruct a variety of moving 3D models based on a sequence of partial surface data acquired by a range scanner.

7.2 Future Research Directions

In the last few chapters, we focused on fitting a simple piecewise rigid model to express the surface motion. However, many articulated subjects have additional non-rigid surface motion, including bulging, stretching, bending, and flapping. An open research direction is to investigate how to represent and capture these additional effects. There are techniques that fit this type of motion using a template [Allen et al., 2003; Anguelov et al., 2005;

Weise et al., 2009; Li et al., 2009], or without an articulated model [Wand et al., 2009], and incorporating the ideas from these techniques could yield an effective algorithm capable of fitting non-rigid deformations on an articulated model.

Another idea is to apply our registration techniques to recover motion and correspondence in other types of data. For example, solving the non-rigid registration for images has important applications for object/pose recognition and medical imaging. This problem has been formulated as a graph matching problem by other researchers, similar to our framework in Chapter 4 [Berg et al., 2005; Torresani et al., 2008b]. While these methods solve explicitly for the correspondences between the two images, our approach would try to assign an optimal labeling of the motion between the two images. It would be interesting to apply this idea to see if it works well when features or correspondences are missing in the images.

A final avenue of work is to improve the performance of the range scan matching techniques to enable real-time surface tracking. This could be realized by having an initial scanning stage where the algorithm determines the articulated structure of the object, and a second stage where real-time tracking is performed. We believe that the development of a method to quickly and robustly track the motion of a surface would be a significant step in improving human-computer interaction, computer animation, medicine, and manufacturing applications. We hope that the algorithms developed in this dissertation will inspire new developments to improve quality of life through effective interaction between computers and the physical world.

Bibliography

Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH*, pages 1–10, 2008. 9

B. Allen, B. Curless, Z. Popović, and A Hertzmann. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *SCA*, 2006. 18

Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM SIGGRAPH*, 21(3):612–619, 2002. ISSN 0730-0301. 3, 11, 101

Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM SIGGRAPH*, pages 587–594, 2003. ISBN 1-58113-709-5. 3, 11, 13, 18, 67, 72, 155

Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *CVPR*, 2007. 13

Dragomir Anguelov, Daphne Koller, Hoi-Cheung Pang, Praveen Srinivasan, and Sebastian Thrun. Recovering articulated object models from 3d range data. In *UAI*, pages 18–26, 2004a. 18

Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *NIPS*, 2004b. 10, 12

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH*, pages 408–416, 2005. 3, 11, 18, 155

K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE TPAMI*, 9:698–700, 1987. 24, 25

Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. In *ACM SIGGRAPH*, page 72, 2007. 101

Raouf Benjemaa and Francis Schmitt. A solution for the registration of multiple 3d point sets using unit quaternions. In *ECCV*, pages 34–50, 1998. 8

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *CACM*, 18(9):509–517, 1975. 24

Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, pages 26–33, 2005. 156

Robert Bergevin, Marc Soucy, Hervé Gagnon, and Denis Laurendeau. Towards a general multi-view registration technique. *IEEE TPAMI*, 18(5):540–547, 1996. 8

Fausto Bernardini and Holly E. Rushmeier. The 3d model acquisition pipeline. *Computer Graphics Forum (Proceedings of Eurographics STAR)*, 21(2):149–172, 2002. 29, 114

P. J. Besl and H.D. McKay. A method for registration of 3-d shapes. *IEEE TPAMI*, 14(2):239–256, 1992. 7, 21, 131

Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE TPAMI*, 17:820–824, 1995. 8, 24

Sergio Blanes and Fernando Casas. On the convergence and optimization of the baker–campbell–hausdorff formula. *Linear Algebra and its Applications*, 378:135–158, 2004. 64

Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3):339–347, 2007. 93

Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI*, 26(9):1124–1137, September 2004. 55, 134

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, 2001. 51, 52, 53, 54, 83, 130, 134

Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM SIGGRAPH*, 27, 2008. 13

Benedict J. Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-d scans. In *ACM SIGGRAPH*, page 21, 2007. 9, 13, 29

Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph.*, 20(2):95–126, 2001. ISSN 0730-0301. 57

Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM SIGGRAPH*, 22(3):569–577, 2003. 16

U. Castellani, M. Cristani, S. Fantoni, and V. Murino. Sparse points matching by combining 3d mesh saliency with statistical descriptors. In *Computer Graphics Forum (Proceedings of Eurographics)*, 2008. 9

Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Proceedings of the IEEE International Conference on Robotics and Automation*, 3:2724–2729, 1991. 7, 8, 27, 29

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE TPAMI*, 17:790–799, 1995. 47

German K. M. Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *CVPR*, pages 77–84, 2003. 16

Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003. doi: 10.1016/S1077-3142(03)00009-2. 13

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *ACM SIGGRAPH*, pages 905–914, 2004. 110

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002. 47, 49

Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. The variable bandwidth mean shift and data-driven scale selection. In *ICCV*, 2001. 49

Edilson de Aguiar, Christian Theobalt, Marcus A. Magnor, Holger Theisel, and Hans-Peter Seidel. M^3: Marker-free model reconstruction and motion tracking from 3d voxel data. In *Pacific Conference on Computer Graphics and Applications*, pages 101–110, 2004. 16

Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *CVPR*, 2007. 17

Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM SIGGRAPH*, 2008a. 17

Edilson de Aguiar, Christian Theobalt, Sebastian Thrun, and Hans-Peter Seidel. Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum (Proceedings of Eurographics)*, 27(2):389–397, 2008b. 18

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL `http://www.jstor.org/stable/2984875`. 89

Piotr Dollar, Serge Belongie, and Vincent Rabaud. Learning to traverse image manifolds. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *NIPS*, pages 361–368, Cambridge, MA, 2007. MIT Press. 82

Chitra Dorai, Gang Wang, Anil K. Jain, and Carolyn R. Mercer. Registration and integration of multiple object views for 3d model construction. *IEEE TPAMI*, 20(1):83–89, 1998. 8

D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl.*, 9(5-6):272–290, 1997. 24

Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981. 100

Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bulow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *ECCV*, 2004. 9

Juergen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *CVPR*, 2009. 17

Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH*, pages 209–216, 1997. ISBN 0-89791-896-7. 80

P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, U.K., 1989. 32

Guy Godin, Marc Rioux, and Rejean Baribeau. Three-dimensional registration using range and intensity information. *SPIE*, 1994. 8

Arthur D. Gregory, Andrei State, Ming C. Lin, Dinesh Manocha, and Mark A. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *CA '98: Proceedings of the Computer Animation*, page 64, 1998. ISBN 0-8186-8541-7. 13

Dirk Hähnel, Sebastian Thrun, and Wolfram Burgard. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *IJCAI*, pages 915–920, 2003. 13

N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel. A statistical model of human pose and body shape. *Computer Graphics Forum (Proceedings of Eurographics)*, 28, 2009. 18

Michael T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, 2nd edition, 2002. 31, 38

B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4), 1987. 24, 25

Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. *Computer Graphics Forum (Proceedings of SGP)*, 27(5):1449–1457, 2008. ix, 14, 28, 101, 103, 106, 110

Doug L. James and Christopher D. Twigg. Skinning mesh animations. In *ACM SIGGRAPH*, pages 399–407, 2005. 18, 47

Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, August 1997. 8, 9, 43, 44, 46, 92, 100

Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai, and Karan Singh. Robust statistical estimation of curvature on discretized surfaces. In *SGP*, pages 13–22, 2007. ISBN 978-3-905673-46-3. 42, 43

Takashi Kanai, Hiromasa Suzuki, and Fumihiko Kimura. 3d geometric metamorphosis based on harmonic map. In *PG '97: Pacific Conference on Computer Graphics and Applications*, page 97, 1997. ISBN 0-8186-8028-8. 13

Takashi Kanai, Hiromasa Suzuki, and Fumihiko Kimura. Metamorphosis of arbitrary triangular meshes. *IEEE Comput. Graph. Appl.*, 20(2):62–75, 2000. ISSN 0272-1716. 13

A. Kaul and J. Rossignac. Solid-interpolating deformations: Construction and animation of pips. In *EUROGRAPHICS*, pages 493–505, 1991. 13

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *I3D*, pages 39–46, 2007. 109

Ladislav Kavan, Steven Collins, Jirí Zára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics*, 27(4), 2008. 57, 124

James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape transformation for polyhedral objects. *ACM SIGGRAPH*, 26(2), 1992. 13

S. Knoop, S. Vacek, and R. Dillmann. Modeling joint constraints for an articulated 3d human body model with artificial correspondences in icp. In *IEEE-RAS International Conference on Humanoid Robots*, 2005. doi: 10.1109/ICHR.2005.1573548. 128

Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE TPAMI*, 28(10):1568–1583, 2006. ISSN 0162-8828. 83

Vladimir Kolmogorov and Carsten Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE TPAMI*, 29(7):1274–1279, 2007. 55

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimizedvia graph cuts? *IEEE TPAMI*, 26(2):147–159, 2004. ISSN 0162-8828. 55, 134

Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM SIGGRAPH*, 23(3):861–869, 2004. 13

Vladislav Kraevoy and Alla Sheffer. Template-based mesh completion. In *Symposium on Geometry Processing*, pages 13–22, 2005. 13

Shankar Krishnan, Pei Yean Lee, John B. Moore, and Suresh Venkatasubramanian. Global registration of multiple 3d point sets via optimization-on-a-manifold. In *SGP*, 2005. 33

Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *SCA*, pages 153–159, 2002. 57

Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *ICCV*, 1988. 9

Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. Maps: multiresolution adaptive parameterization of surfaces. In *ACM SIGGRAPH*, pages 95–104, 1998. ISBN 0-89791-999-8. 13

Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *ACM SIGGRAPH*, pages 343–350, 1999. ISBN 0-201-48560-5. 13

V. Lempitsky, C. Rother, S. Roth, , and A. Blake. Fusion moves for markov random field optimization. *IEEE TPAMI, to appear*, 2009. 55

Victor Lempitsky, Carsten Rother, and Andrew Blake. Logcut – efficient graph cut optimization for markov random fields. In *ICCV*, 2007. 83

Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, pages 1482–1489, 2005. doi: 10.1109/ICCV.2005.20. 14

J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *ACM SIGGRAPH*, pages 165–172, 2000. 57

Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (Proceedings of SGP)*, 27(5):1421–1430, 2008. 14, 93

Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. Robust single view geometry and motion reconstruction. In *ACM SIGGRAPH ASIA, to appear*, 2009. 156

Hongdong Li and Richard Hartley. The 3d-3d registration problem revisited. *ICCV*, 2007. 10

Yaron Lipman and Thomas A. Funkhouser. Möbius voting for surface correspondence. *ACM SIGGRAPH*, 28(3), 2009. doi: 10.1145/1531326.1531378. 13

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982. 100

Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2003. ISBN 0387008934. 34, 37

K. Madsen, H. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. Technical report, Technical University of Denmark, 2004. 33

S. Maneewongvatana and D. M. Mount. It's okay to be skinny, if your friends are fat. In *4th Annual CGC Workshop on Comptutational Geometry*, 1999. 24

J. Manson, G. Petrova, and S. Schaefer. Streaming surface reconstruction using wavelets. In *SGP*, 2008. 137, 138

Nelson Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999. 40, 42

Ajmal Mian, M. Bennamoun, and R. Owens. A novel representation and feature matching algorithm for automatic pairwise registration of range images. *IJCV*, 66:19–40, 2006. 9

Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *ACM SIGGRAPH*, pages 157–164, 1991. 80, 100, 119

Niloy J. Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. Registration of point cloud data from a geometric optimization perspective. In *SGP*, 2004. 28

Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM SIGGRAPH*, 25(3):560–568, 2006. ISSN 0730-0301. 11, 12, 47, 61, 62, 82

Niloy J. Mitra, S. Flory, M. Ovsjanikov, N. Gelfand, Leonidas J. Guibas, and H. Pottmann. Dynamic geometry registration. In *SGP*, pages 173–182, 2007a. 14

Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Symmetrization. In *ACM SIGGRAPH*, page 63, 2007b. 12

Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM SIGGRAPH*, pages 562–568, 2003. 58

David M. Mount and Sunil Arya. Ann: A library for approximate nearest neighbor searching, version 1.1.1. http://www.cs.umd.edu/~mount/ANN, 2006. URL `http://www.cs.umd.edu/~mount/ANN`. 24

P. J. Neugebauer. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *International Journal of Shape Modeling*, 3(1/2): 71–90, 1997. 8, 24, 29, 30, 117

K. Nishino and K. Ikeuchi. Robust simultaneous registration of multiple range images. In *ACCV*, 2002. 8, 153

John Novatnack and Ko Nishino. Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In *ECCV*, 2008. 9

Ryutarou Ohbuchi, Yoshiyuki Kokojima, and Shigeo Takahashi. Blending shapes by using subdivision surfaces. *Computers & Graphics*, 25(1):41–58, 2001. doi: 10.1016/S0097-8493(00)00106-0. 13

Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus Gross, and Leonidas J. Guibas. Example-based 3d scan completion. In *SGP*, page 23, 2005. ISBN 3-905673-24-X. 3, 11, 13

Yuri Pekelny and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum (Proceedings of Eurographics)*, 27(2), 2008. 4, 15, 92, 101, 108, 113, 119, 125, 130, 135, 138

Tiberiu Popa, Quan Zhou, Derek Bradley, Vladislav Kraevoy, Hongbo Fu, Alla Sheffer, and Wolfgang Heidrich. Wrinkling captured garments using space-time data-driven deformation. *Computer Graphic Forum (Proceedings of Eurographics)*, 28(2):427–435, 2009. 17

H. Pottmann, S. Leopoldseder, and M. Hofer. Registration without icp. *CVIU*, 95:54–71, 2004. 10

Emil Praun, Wim Sweldens, and Peter Schröder. Consistent mesh parameterizations. In *ACM SIGGRAPH*, pages 179–184, 2001. ISBN 1-58113-374-X. 13

Kari Pulli. *Surface Reconstruction and Display from Range and Color Data*. PhD thesis, University of Washington, 1997. 9

Kari Pulli. Multiview registration for large data sets. In *3DIM*, pages 160–168, 1999. 8, 9

Vincent Rabaud and Serge Belongie. Re-thinking non-rigid structure from motion. In *CVPR*, 2008. 19

Szymon Rusinkiewicz. *Real-Time Acquisition and Rendering of Large 3D Models*. PhD thesis, Princeton University, 2001. 8

Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *3DPVT*, pages 486–493, 2004. ISBN 0-7695-2223-8. 42

Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *3DIM*, pages 145–152, 2001. 8, 24, 26, 27

Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. *ACM SIGGRAPH*, 21(3):438–446, 2002. 29, 114

Ryusuke Sagawa, Nanaho Osawa, and Yasushi Yagi. Deformable registration of textured range images by using texture and shape features. In *3DIM*, pages 65–72, 2007. ISBN 0-7695-2939-4. 11

Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. *ACM SIGGRAPH*, 22(3):578–586, 2003. 3, 16

S. Schaefer and C. Yuksel. Example-based skeleton extraction. In *SGP*, pages 153–162, 2007. 18, 99

Andrei Sharf, Dan A. Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. Space-time surface reconstruction using incompressible flow. *ACM SIGGRAPH ASIA*, 2008. 15

Christian R. Shelton. Morphable surface models. *Int. J. Comput. Vision*, 38(1):75–91, 2000. ISSN 0920-5691. 13

Jonathan Starck and Adrian Hilton. Correspondence labelling for wide-timeframe free-form surface matching. In *ICCV*, 2007. 12

A. J. Stoddart and A. Hilton. Registration of multiple point sets. *International Conference on Pattern Recognition*, 2:40, 1996. ISSN 1051-4651. doi: http://doi.ieeecomputersociety.org/10.1109/ICPR.1996.546720. 8

Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM SIGGRAPH*, pages 399–405, 2004. 58, 72

Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH*, page 80, 2007. 93, 152

Jochen Süßmuth, Marco Winter, and Günther Greiner. Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum (Proceedings of SGP)*, 27(5):1469–1476, 2008. 15

Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall F. Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV*, 2006. 55

Shigeo Takahashi, Yoshiyuki Kokojima, and Ryutarou Ohbuchi. Explicit control of topological transitions in morphing shapes of 3d meshes. In *PG '01: Pacific Conference on Computer Graphics and Applications*, page 70, 2001. ISBN 0-7695-1227-5. 13

Camillo J. Taylor and David J. Kriegman. Minimization on the lie group so(3) and related manifolds. Technical report, Yale University, 1994. 33

A. Tevs, M. Bokeloh, M.Wand, A. Schilling, and H.-P. Seidel. Isometric registration of ambiguous and partial data. In *CVPR*, 2009. 12

Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE TPAMI*, 30:878–892, 2008a. 19

Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008b. 55, 156

Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *ACM SIGGRAPH*, pages 311–318, 1994. 8

Oncel Tuzel, Raghav Subbarao, and Peter Meer. Simultaneous multiple 3d motion estimation via mode finding on lie groups. In *ICCV*, pages 18–25, 2005. 47, 63, 64

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM SIGGRAPH*, 2008. 17

Michael Wand, Philipp Jenke, Qixing Huang, Martin Bokeloh, Leonidas J. Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *SGP*, pages 49–58, 2007. 15

Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. Efficient reconstruction of non-rigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics*, 28, 2009. ix, 15, 150, 151, 156

Robert Y. Wang, Kari Pulli, and Jovan Popović. Real-time enveloping with rotational regression. In *ACM SIGGRAPH*, page 73, 2007. 58

Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA*, pages 129–138, 2002. 58

Sebastian Weik. Registration of 3-d partial surface models using luminance and depth information. In *3DIM*, pages 93–100, 1997. 8

Thibaut Weise, Bastian Leibe, and Luc J. Van Gool. Fast 3d scanning with automatic motion compensation. In *CVPR*, 2007. 2, 72, 101

Thibaut Weise, Hao Li, Luc Van Gool, and Mark Pauly. Face/off: Live facial puppetry. In *Eighth ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2009. 156

J. Williams and M. Bennamoun. A multiple view 3d registration algorithm with statistical error modeling. *IEICE Transactions on Information and Systems*, 83-D:1662–1670, 2000. 8

Hao Zhang, Alla Sheffer, Daniel Cohen-Or, Quan Zhou, Oliver van Kaick, and Andrea Tagliasacchi. Deformation-driven shape correspondence. *Computer Graphics Forum (Proceedings of SGP)*, 27(5):1431–1439, 2008. doi: 10.1111/j.1467-8659.2008.01283.x. 12

Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. In *ACM SIGGRAPH*, pages 548–558, 2004. 2

Song Zhang and Peisen Huang. High-resolution, real-time three-dimensional shape measurement. *Optical Engineering*, 45, 2006. 2