LEARNING MODELS OF SHAPE FROM 3D RANGE DATA

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dragomir Anguelov
December 2005

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daphne Koller
(Principal Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Andrew Ng

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marc Levoy

Approved for the University Committee on Graduate Studies:

To my family and Olya.

Abstract

Constructing shape models of complex articulated and deformable objects is a fundamental capability that enables a variety of applications in computer graphics, biomechanics, arts and entertainment. Current approaches require a significant amount of manual intervention in the model construction process.

In this thesis, we present algorithms for learning models of shape that reduce the need for human input. First, we describe an unsupervised algorithm for registering 3D surface scans of an object undergoing significant deformations. Our algorithm does not use markers, nor does it assume prior knowledge about object shape, the dynamics of its deformation, or scan alignment. It is based on a probabilistic model, which minimizes deformation and attempts to preserve geodesic distances and local mesh geometry. Second, we describe an algorithm whose input is a set of meshes corresponding to different configurations of an articulated object. The algorithm automatically recovers a decomposition of the object into approximately rigid parts, the location of the parts in the different object instances, and the articulated object skeleton linking the parts.

We also address the problem of learning the space of human body deformations from 3D scans. Unlike existing example-based approaches, our model spans variation in both subject shape and pose. We learn a model of surface deformation as a function of the joint angles of the articulated human skeleton. We also learn a separate model of the variation between different body shapes. We show how to combine these two models to produce realistic deformation for different people in different poses. Finally, we show how our framework can be used for shape completion – generating a complete surface mesh given a limited set of markers specifying the target shape. We use this capability to complete partial mesh geometry and to animate marker motion capture sequences.

Acknowledgements

I am profoundly grateful to my adviser Daphne Koller for many years of guidance and support. She has been the most influential person in my academic career, responsible for kindling my interest in research when I was a wide-eyed undergraduate, and for giving me the opportunity to develop this interest. From my years as her advisee, I gleaned insights into what kind of problems to tackle, how to go about addressing them and how to present my results in a compelling manner. She set an example for me with her brilliance, high standards and drive for excellence. In short, she has been a terrific adviser, and is the primary culprit for me getting to the point of writing these acknowledgements.

I feel privileged for having Sebastian Thrun be my guide to the exciting world of probabilistic robotics. I am forever indebted to his generosity — in practice he has been a second adviser to me, spending countless hours to help develop the ideas in this thesis and providing valuable help and advice. His never-ending energy, flair and quick wit have truly been an inspiration. It has been a joy to be around Sebastian; his research enthusiasm is truly contagious.

My first graduate adviser, Carlo Tomasi, helped me with my first steps in graduate life, and exposed me to the subject of computer vision. His kindness, generosity, and clarity of thought have been a great example for me.

I would like to acknowledge the kind help of Marc Levoy, who on many occasions shared his computer graphics expertise with this humble artificial intelligence student. He provided valuable advice, which helped me to shape my research ideas, and present them in a compelling manner to the computer graphics community. I would like to thank Andrew Ng and Ron Fedkiw for valuable discussions and research suggestions, as well as Tom Andriacchi for chairing my thesis defense committee and for generously providing me

access to the wonderful facilities of the Stanford biomechanics lab.

I am grateful to James Davis, a 3D sensor guru and multimedia wizard, who was instrumental in providing the sensor data, which is the input to the algorithms in this thesis. He is also responsible for taking the output of the algorithms and turning it into what I personally think is a work of art. The work on SCAPE would not have been possible if James Davis had not introduced me to two great Europeans in the Stanford biomechanics lab – Lars Mündermann and Stefano Corazza, who spent countless hours helping me with the data acquisition process. Their passion for markerless motion capture is contagious, which resulted in a great collaboration, which I hope will continue.

I feel privileged to be a member of our storied DAGS research group. I am greatly thankful to have had a chance to work, hang around, travel with, spend time in interesting discussions with, being motivated by the great work of: Pieter Abbeel, Alexis Battle, Luke Biewald, Rahul Biswas, Vassil Chatalbashev, Gal Chechik, Gal Elidan, Lise Getoor, Carlos Guestrin, Geremy Heitz, Uri Lerner, Uri Nodelman, Dirk Ormoneit, Jimmy Pang, Evan Parker, Ron Parr, Jim Rodgers, Suchi Saria, Eran Segal, Christian Shelton, Praveen Srinivasan, Simon Tong, David Vickrey, Haidong Wang and Ming Fai Wong.

I would like to say cheers mate! to my officemate Ben Taskar, friend in work and play, with whom we faced early morning deadlines and grizzly bears, shared research ideas, music, books and life experiences. I would like to say thank you! to Praveen Srinivasan, the czar of morphing, whose steadfastness, dedication and insights always kept us on track for the deadlines (meaning we would submit at the last moment after a mad scramble). I would like to thank Pieter Abbeel, a remarkable researcher, and a serious disciple of the game of tennis, for numerous interesting discussions that have enriched my understanding of many a concept. I would like to thank Uri Lerner, chess and Diplomacy master, for taking an inexperienced undergrad and showing him a thing or two in the ways of research. I would like to thank Jimmy Pang, and Jim Rodgers — great guys, whom I enjoyed working with. I would also like to thank Gal Elidan, an avid hiker and backpacker, a guy who knows how to do great research and enjoy life, for many interesting conversations and for useful advice. I thank Geremy Heitz for multiple discussions about various computer vision methods. I would like to thank Mark Paskin, Brian Gerkey and Michael Montemerlo from Sebastian's group, whom I enjoyed talking robots, and not simply robots, with.

I would like to thank many friends from my ten years at Stanford, who did not have direct impact on this thesis, but made my time here really memorable.

My parents, Irina and Dimitar, made everything possible for me to get a good education while I was growing up, and supported me in my years-long foray into America and then to grad school. Thank you mom and dad, I am truly grateful. I hope now that I am done I will be able to see you more often. And last, but not the least — Olya, thank you for lighting up these long months with your love and smile. Thank you for making me happy next to you.

Contents

List of Figures

Chapter 1

Introduction

Hamlet: Do you see yonder cloud that's almost in shape of a camel?
Polonius: By the mass, and 'tis like a camel, indeed.
Hamlet: Methinks it is like a weasel.
Polonius: It is backed like a weasel.
Hamlet: Or like a whale?
Polonius: Very like a whale.
William Shakespeare (1564 - 1616), "Hamlet", Act 3 scene 2

Computers are increasingly used for modeling and interaction with the physical world. Thanks to decades of research in computer graphics, computers can be utilized as animation and rendering tools to enable the creation of compelling virtual realities. A multi-billion industry specializes in conjuring up, creating and packaging these realities in the form of movies and computer games. As a result, billions of people have been transported to new worlds of fantasy, adventure and learning. Computers are also utilized for the creation of intelligent agents, which are becoming increasingly sophisticated in their ability to navigate the environment and to interact with people.

The tasks of modeling and interaction with the physical world are critically dependent on the ability of the computer to represent and reason about shape. When humans perform these tasks, they are able to transform the photon impulses hitting the eye's retina into a

symbolic representation of the world (”Ouch! The dog ate my chocolate!”). We tend to think of the world in terms of *objects* (dog, chocolate), which are entities with coherent properties (dogs like chocolate). *Shape* is one of the most fundamental properties, that is used to describe the 3D surface geometry of objects. The ability to acquire and reason about this geometry enables multiple practical applications in the entertainment, biomedical and robotics industries which include animation of scenes and characters, motion capture and analysis, and scene understanding.

Humans are able to perform sophisticated reasoning about shape on many levels. We are able to recognize various objects and *classes of objects* based on their shape, and many artists and sculptors possess uncanny ability to reproduce the shapes that they have previously seen. We can reason about object classes from nature such as giraffes, jellyfish and cobras, to name a few, and multiple human-made objects such as cars, chairs and text. Furthermore, we can deal rather effortlessly with the significant *shape variation* which may be present in a particular object class. Consider the familiar shape of the human body, which is remarkably diverse in spanning African pygmies and North American couch potatoes, as well as males and females. Another common object, the chair, comes in an incredibly rich set of shapes limited only by the designer’s imagination, while somehow remaining distinctly recognizable.

To make matters more difficult, the shape of each object can change (deform) over time. In order to animate humans, snakes and puffer-fish, we need to be able to represent these deformations. For many objects, the shape changes are often easier to understand if the object is viewed as a *collection of parts*. For example, we think of a human body in terms of head, torso, arms and legs, while a chair can have a seat, legs and a back. For many objects (giraffes, chairs), the configuration of the object parts relative to each other can change significantly and account for most of the shape deformation. The parts themselves can deform (when humans flex their muscles and cats stretch) and faithful models need to capture these deformations as well.

Most state-of-the-art algorithms for shape modeling, animation and tracking bypass the inherent complexity of the shape-modeling task by relying on significant amount of human input. For example, algorithms for finding the correspondence between two shapes often require that dozens of corresponding points on the shapes are specified by a human.

Algorithms for shape tracking often assume that the shape model and its decomposition into parts is provided, and rarely generalize to other instances in the object class. Shape-completion algorithms often assume that the object is placed in a particular pose.

In this thesis, we present a framework for learning complex shape models from example surfaces acquired with a 3D scanner. The framework consists of several algorithms, based on the theory of probabilistic graphical models, which allow us to learn complex shape models of different objects with minimal human intervention. First, we address the fundamental problem of non-rigid registration and describe an unsupervised algorithm for computing the correspondences between two drastically deforming surfaces. Then, we show how to automatically decompose a shape into its constituent parts, and find the joints between the parts. We also show how to combine the information from the registered surfaces and the part decomposition in order to learn the space of deformations for an entire object class. We demonstrate this approach by learning the space of human body deformations spanning different body physiques and different poses. Finally, we show applications of the learned shape models to popular tasks such as animation, shape completion and tracking.

1.1 3D Shape Models

We will start by giving a brief overview of the main shape-modeling paradigms.

1.1.1 Artist-designed Models

Most 3D character models, which are used in movies and games, are designed by graphics artists. The design process is known as *character modeling* and consists of several stages (Fig. 1.1(a)). First, specialized software is used to model the surface geometry of the character, using many possible approaches such as subdivision surfaces, nurbs or constructive solid geometry [46]. Other surface properties such as color, texture and reflection characteristics may also be defined at this time. In the next stage of the process called *rigging*, the character may be fitted with a skeleton, which allows easy editing of its poses and movement. The model can also be equipped with specific controls to make animation easier and

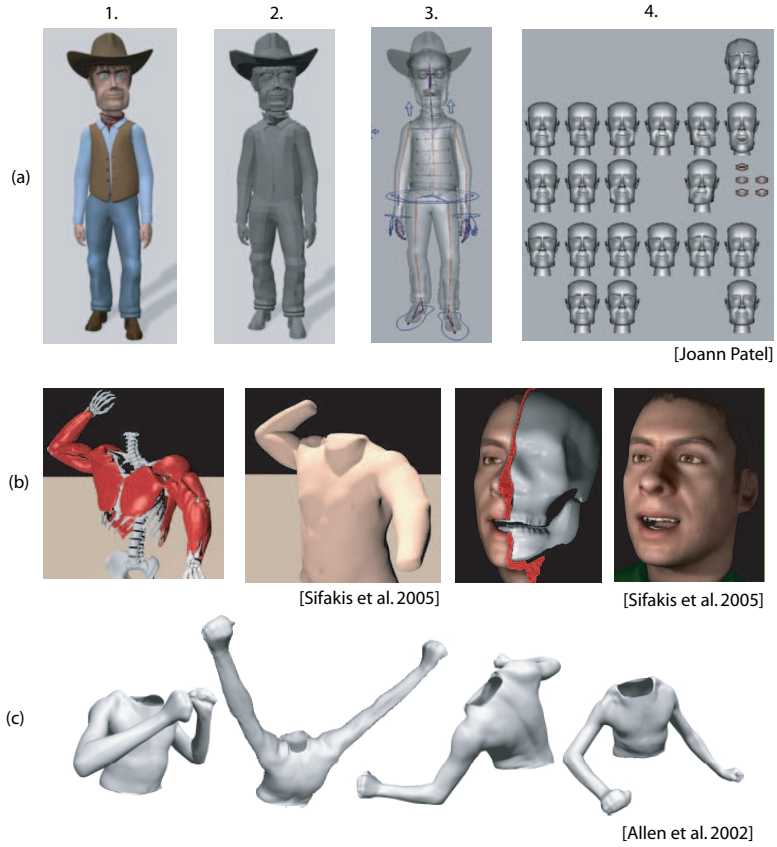


Figure 1.1: Different 3D modeling paradigms. (a) Artist-designed models are created by specifying the geometry and defining the skeleton and the facial expressions of the character. (b) Anatomical models simulate the deformation of the underlying muscle and tissue. (c) Example-based models are learned from scans of real-world objects.

more intuitive. For example, the character Woody from the movie Toy Story has about one hundred pre-defined facial expression controls and mouth shapes used for lip-synching.

Many memorable characters such as Yoda[1], Shrek and Buzz Lightyear and some less memorable ones such as Jar-Jar Binks have been generated in such a manner. The scope of different characters that can be created with the existing tools for 3D animation is truly astounding; the main limitation being the imagination of the designer. The other main limitation is time. The process is very labor-intensive, and may take months and many people to design a novel character. Specialized software helps to decrease design time for

[1] Yoda is computer-generated in Star Wars episodes I-III only.

commonly-used models such as humans [94], but the process still can take many days.

1.1.2 Anatomical Models

A different paradigm for obtaining accurate surface deformation is based on anatomical modeling of the major bones, muscles and other interior structures of the body (Fig. 1.1(b)). As the body moves, the deformation of these underlying structures induces a corresponding deformation of the skin that is wrapped over them. There is a large body of work on such physically-realistic models, including Wilhelms and Gelder [124], Scheepers *et al.* [100], and Aubel and Thalmann [9]. The primary strength of anatomical approaches is their ability to simulate dynamics and object interactions in a realistic way. However, such detailed physical models are difficult to construct, and computationally expensive to use. In each animation frame, one must perform a physical simulation of the entire body anatomy, while taking care to conserve muscle volumes, and stretch the skin appropriately.

1.1.3 Example-based Models

An increasingly popular approach is to learn shape models directly from examples, which can be acquired with a 3D scanner (Fig. 1.1(c)). This approach has been used to model face deformations [15, 120], human body deformations due to changes in pose [108, 123, 80] and human body deformations between different people [2, 102]. Example-based approaches produce realistic models that closely mimic the appearance of the scanned objects. Constructing them requires relatively little human involvement. Finally, they are considerably more efficient than anatomical approaches because they only model the object's surface but not its interior. These factors make example-based approaches the method of choice for objects which can be easily scanned. Example-based approaches for modeling shape will be the focus of this thesis. They merit a more detailed description, which is provided in the next section.

1.2 Constructing a Deformation Model from Examples

Most example-based modeling methods use the same basic data-processing pipeline. The process usually begins with a human-designed shape template of the object (Fig. 1.2(a)). In the case when pose deformations are modeled, this template includes the articulated object *skeleton* — a decomposition of the object surface into parts and the joints between these parts. Depending on the actual surface representation, additional controls for deforming the surface may be provided as well.

Then 3D scans of the object are acquired. In order to integrate the information present in the scan surfaces, they need to be brought into correspondence — a process also known as *registration*. Registration is usually performed between the object template and each scan [1, 2]. The output is a mapping between every point on the template surface and its corresponding point in the scan. Current registration algorithms need to be initialized with a subset of the point-to-point correspondences between the model and each scan. These correspondences can be obtained by placing markers on the scanned object (Fig. 1.2(b)) or by having a human click on corresponding points in a special software tool. The number of correspondences required is usually quite large — Allen *et al.* [2] needed more than 70 matching point pairs for the registration of two human body scans. Accurate placement of physical or virtual markers is usually time-consuming, and for full-body models can easily exceed half an hour per subject.

Now we have a set of registered scans, which specify the deformations of the template shape for a variety of object instances or poses. These deformations Y are associated with some intrinsic parametrization X of the object template, producing the tuples $(X_1, Y_1), \ldots, (X_n, Y_n)$. For example, if we are modeling pose deformations, the vectors X will contain the joint angles of the articulated skeleton. If we are modeling the space of human facial expressions, X may correspond to the parameters of some lower-dimensional subspace. Most commonly, these subspace parameters are obtained by performing *Principal Component Analysis (PCA)* on the deformation vectors Y.

We can predict the deformations of new object instances by interpolating from nearby examples (Fig. 1.2(c)). In particular, given a set of parameters X' defining a new shape instance, we can obtain the shape template deformations Y' by looking at the examples

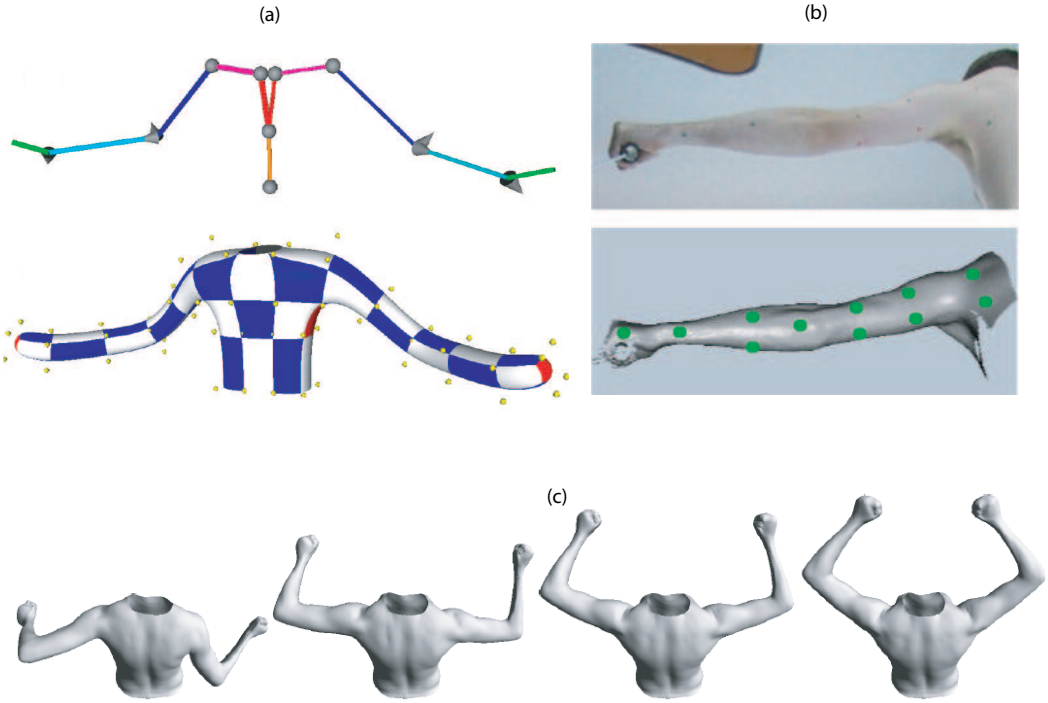


Figure 1.2: The pipeline for learning a shape model from examples (example courtesy of Allen *et al.* [1]). (a) A shape template for the object is defined, which includes an articulated skeleton and a deformable subdivision template. (b) Markers are placed on the object's surface and scans of the object are acquired. (c) After the shape template is registered with the scans, new examples can be generated by interpolation from existing ones.

whose parameters X are similar to X'. A variety of different methods exist, that differ only in the details of representing the deformation, and in the way the interpolation is done [70, 108, 123, 80, 102]. Many of these interpolation methods are very efficient, and can be used for real-time animation.

1.3 Contributions

This focus of this thesis is on algorithms which allow us to learn complex models of shape with little or no human supervision. We present novel algorithms for all stages of the modeling process.

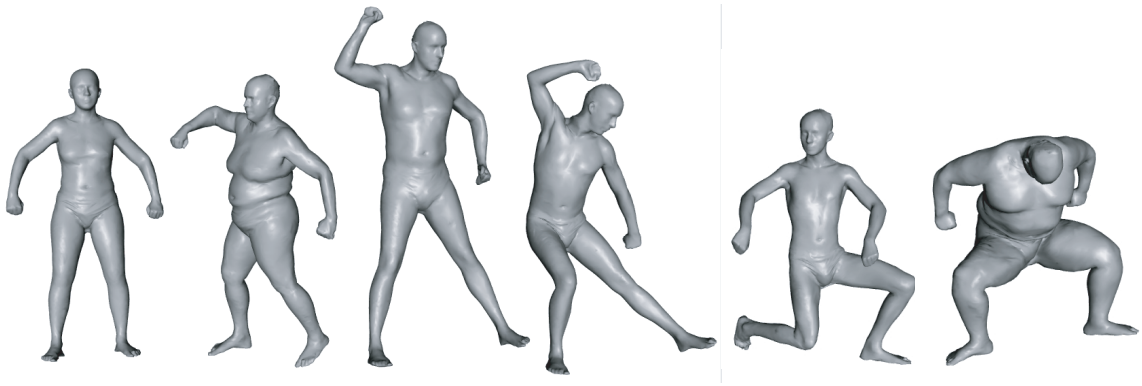


Figure 1.3: Shapes of different people in different poses, synthesized from our learned space of human body variations.

- **Registration of 3D surfaces**

  We propose an unsupervised algorithm for 3D surface registration. When the surfaces undergo significant deformations, previous approaches rely on the presence of markers on the scans, or on significant object-specific knowledge. In contrast, our algorithm does not need markers, nor does it assume prior knowledge about object shape, the dynamics of its deformation, or scan alignment. The algorithm registers two meshes by optimizing a joint probabilistic model over all point-to-point correspondences between them. This model enforces preservation of local mesh geometry, as well as more global constraints that capture the preservation of geodesic distance between corresponding point pairs. The algorithm applies even when one of the meshes is an incomplete range scan; thus, it can be used to automatically fill in the remaining surfaces for this partial scan, even if those surfaces were previously only seen in a different configuration. Our algorithm has certain limitations — it does not address the cases when there are significant changes in surface topology, nor does it offer a way of preserving the volume enclosed by the surface.

- **Articulated model recovery** We address the problem of learning a complex articulated object models from registered 3D scans. The algorithm automatically recovers a decomposition of the object into approximately rigid parts, the location of the parts in the different object instances, and the articulated object skeleton linking the

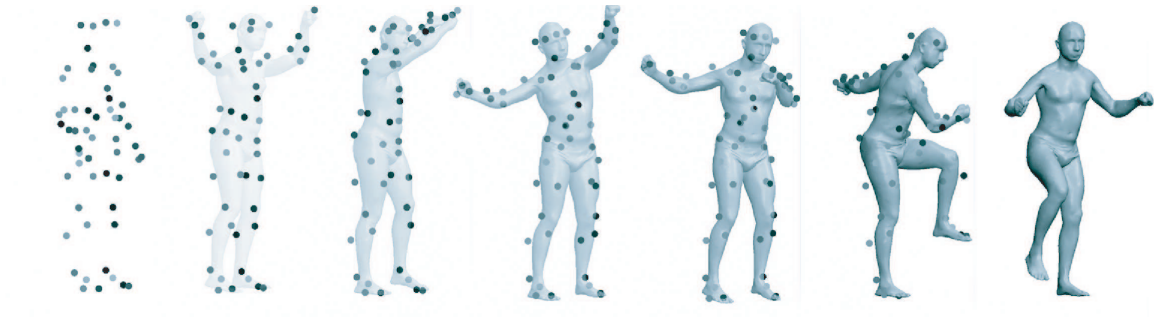


Figure 1.4: Animation of a motion capture sequence taken for a subject, of whom we have a single body scan. The muscle deformations are synthesized automatically from the space of pose and body shape deformations.

parts. The decomposition into parts is obtained by using the EM algorithm, using a graphical model that explicitly enforces the spatial contiguity of each part. Although the graphical model is densely connected, the object decomposition step can be performed optimally and efficiently, allowing us to identify a large number of object parts while avoiding local maxima. We demonstrate the algorithm on real world datasets, recovering complex models with up to 18 parts, even in the presence of non-trivial part deformations. To the best of our knowledge, this is the first algorithm to recover such complicated articulated models in a completely unsupervised manner.

- **Modeling the space of human body shapes** We introduce a data-driven method for building a human shape model that spans variation in both subject shape and pose. The method is based on a representation that incorporates both articulated and non-rigid deformations. We learn a *pose deformation model* that derives the non-rigid surface deformation as a function of the pose of the articulated skeleton. We also learn a separate model of variation based on body shape. Our two models can be combined to produce 3D surface models with realistic muscle deformation for different people in different poses, when neither appear in the original set of examples (see Fig. 1.3). We show how the model can be used for animation and *shape completion* — generating a complete surface model given a limited set of markers that specify the target shape (Fig. 1.4). We present applications of shape completion to partial view completion and motion capture animation.

1.4 Thesis Outline

Below is a summary of the rest of chapters in this thesis.

Chapter 2. Surfaces and transformations: We discuss and contrast the basic surface representations, including point clouds, meshes and sign distance maps. Then we review rigid surface transformations, and describe different representations of rotation such as twists and quaternions. We review how to compute the optimal alignment between two corresponding point sets.

Chapter 3. Probabilistic graphical models: We introduce and compare the Bayesian network and Markov network formalisms. Then we discuss inference algorithms for answering conditional probability and maximum a-posteriori (MAP) queries. We describe in detail the Belief Propagation algorithm and its performance on singly-connected and loopy graphs. We also present a linear programming algorithm for inference in Associative Markov networks. We describe maximum-likelihood parameter estimation for Bayesian networks when the data is fully observed. We also describe the Expectation-Maximization algorithm for parameter learning in the presence of hidden variables and missing data.

Chapter 4. Surface registration: We define the problem of surface registration, and describe the Non-rigid Iterative Closest Point (non-rigid ICP) paradigm for addressing it. We analyze the failures of non-rigid ICP and propose a novel algorithm for unsupervised surface registration, which works even when the surface undergoes drastic deformation. We describe this Correlated Correspondence algorithm and evaluate it experimentally on several real-world datasets. Finally, we present applications of the algorithm to the problems of partial view completion and interpolation between two registered scans.

Chapter 5. Recovering articulated object models: We define the problem of articulated model recovery. We present a novel algorithm for partitioning the object into approximately rigid parts and evaluate it experimentally. Then we describe a way of recovering the joints between the parts. We present an application of the learned models to the problem of articulated model tracking.

Chapter 6. Learning deformable models of human shape: We describe a method for learning the space of deformations for an entire object class. In particular, we show how to learn the space of human shapes which spans changes in pose and physique. We present applications of the space to the problems of animation and shape completion.

Chapter 7. Conclusions and future directions: We review the main contributions of the thesis and summarize their significance, applicability and limitations. We discuss extensions and future research directions not addressed in the thesis.

1.5 Previously published work

Most of the work described in this thesis has been published in conference proceedings. In particular, the Correlated Correspondence algorithm for surface registration and its application to animation and partial view completion was published in Anguelov, Srinivasan, Koller, Thrun, Pang and Davis [6]. The method for articulated object recovery was published in Anguelov, Koller, Pang, Srinivasan and Thrun [4]. Finally, the method for modeling human deformations and its applications for animation and shape-completion was published in Anguelov, Srinivasan, Koller, Thrun, Rodgers and Davis [7]. The algorithm for articulated object tracking was submitted as a conference abstract [5], along with a separate experimental validation study [84].

Chapter 2

Surfaces and Transformations

In this chapter, we describe the basics of 3D surface models: how to represent surfaces and how to manipulate them in three-dimensional space. This discussion provides the foundation for the shape-modeling algorithms, which are the contribution of this thesis. We also introduce the standard notation for various surface properties, which will be used in the subsequent thesis chapters.

First, we present standard 3D surface representations such as *point clouds* and *meshes*, which are the representations of choice in our learning algorithms. We will motivate briefly their advantages over other surface representations, such as *signed distance maps* and *splines*.

Then we discuss how to apply rigid transformations to our surfaces. We place particular emphasis on different ways of parameterizing rotations in three dimensional space. We introduce the standard *unit quaternion* and *exponential map* representations of rotation and discuss the relative benefits of each. We also describe how to compute the optimal rigid alignment between two clouds of corresponding points.

2.1 3D Surface Representations

Ideally, 3D surfaces are continuous manifolds with an infinite number of degrees of freedom, requiring an equally infinite number of parameters for their representation. However, most practical applications require only a certain degree of modeling accuracy. In addition,

current 3D sensors provide only a finite amount of readings, in the form of point samples describing the surface. Thus, a discretization of the continuous 3D surface at an appropriate resolution is sufficient for our purposes. Below we present several tractable ways of discretizing the surface. In general, there are two classes of discretizations. *Explicit* representations model the surface directly. Recently, *implicit* representations in the form of scalar fields have also gained popularity [33, 69]. These fields assign values to all points in 3D space — surfaces are obtained by looking at the isosurfaces of the field.

2.1.1 Explicit Representations

3D acquisition devices have become a popular source for the creation of 3D geometric data. They provide information about object shape in terms of unstructured clouds of sample surface readings. These readings are obtained either by performing matching in stereo data, or by emitting rays and measuring the time of travel from the sensor to the object and back. Each reading usually contains information about the coordinates of a point on the scanned surface. Usually, an estimate of the normal vector to the surface the point can also be obtained. This can be done either in a post-processing step by interpolation from adjacent sensor readings [79] or by using shape-from-shading and photometric stereo information [125]. The resulting *point cloud*, which contains the surface readings and the point normals, is the simplest representation of the surface.

Definition 2.1.1 *A* **point cloud** *is a description of the surface X as a collection of sensor readings, where each reading contains the coordinate of a surface point, and an estimate of the surface normal at that point. The point cloud is denoted as $\mathcal{P}^X = (\mathcal{V}^X, \mathcal{N}^X)$.*

Here, $\mathcal{V}^X = (x_1, \ldots, x_{N_X})$ is a set of 3D surface point coordinates, while $\mathcal{N}^X = (n_1, \ldots, n_{N_X})$ are the corresponding unit-length normal vectors. An example point cloud can be seen in Fig. 2.1.

Point clouds are very useful for representing 3D sensor data. However, they provide only incomplete information about the underlying continuous surface. Most importantly, point clouds do not explicitly model surface connectivity, and hence, topology. Many different continuous surfaces can be a plausible fit to the samples of a point cloud. The shape

uncertainty is further increased by measurement noise, which cannot be avoided in any physical acquisition process.

We find it useful to define another standard surface representation called a *mesh*, which explicitly models surface connectivity.

Definition 2.1.2 *A* **mesh** $\mathcal{M}^X$ *is a tessellation of a continuous 3D surface* X *into a set of polygons. It can be represented as a collection of points and polygons:* $\mathcal{M}^X = \left(\mathcal{V}^X, \mathcal{P}^X\right)$.

Here, $\mathcal{V}^X = (x_1, \ldots, x_{N_X})$ represents the coordinates of the polygon vertices. The set of polygons covering the surface is denoted by $\mathcal{P}^X = (p_1, \ldots, p_{M_X})$. In general, polygons containing an arbitrary number of vertices can be used. Without loss of representation power, we will assume that $\mathcal{P}^X$ only contains triangles. This assumption simplifies the notation and streamlines the treatment of meshes; its only drawback is a slight increase in model size. Every triangle p_k is a defined as a set of three natural numbers $(p_{1,k}, p_{2,k}, p_{3,k})$, corresponding to the indexes of the triangle points in the list $\mathcal{V}^X$. An example of a surface represented by a triangle mesh is displayed in Fig. 2.1.

The mesh representation defined above is sufficient for estimating the normal vectors to the surface. For example, the normal n_{p_k} at triangle p_k can be estimated by taking the cross-product of the triangle edges:

$$n_{p_k} = \frac{u}{\|u\|}, \quad u = (x_{k,2} - x_{k,1}) \times (x_{k,3} - x_{k,1}). \tag{2.1}$$

To avoid ambiguity, convention requires that the vertices of each triangle p_k are specified in a counter-clockwise order (the opposite order flips the direction of the normal). The normal n_{x_i} at a point x_i can be estimated by simply averaging the adjacent triangle normals.

We will also define the set of mesh *edges* $\mathcal{E}^X$, which contains the edges of all mesh triangles, without repetition. Formally, this can be represented as follows:

$$\mathcal{E}^X \doteq \left(e_{i,j} \mid \exists p_k \in \mathcal{M}^X; i, j \in p_k; i \prec j\right). \tag{2.2}$$

The set of mesh triangles $\mathcal{P}^X$ and edges $\mathcal{E}^X$ both contain essentially the same information about surface connectivity. For several machine learning tasks, either of these sets can be used. For example, surface deformation can be quantified by looking at the deformation

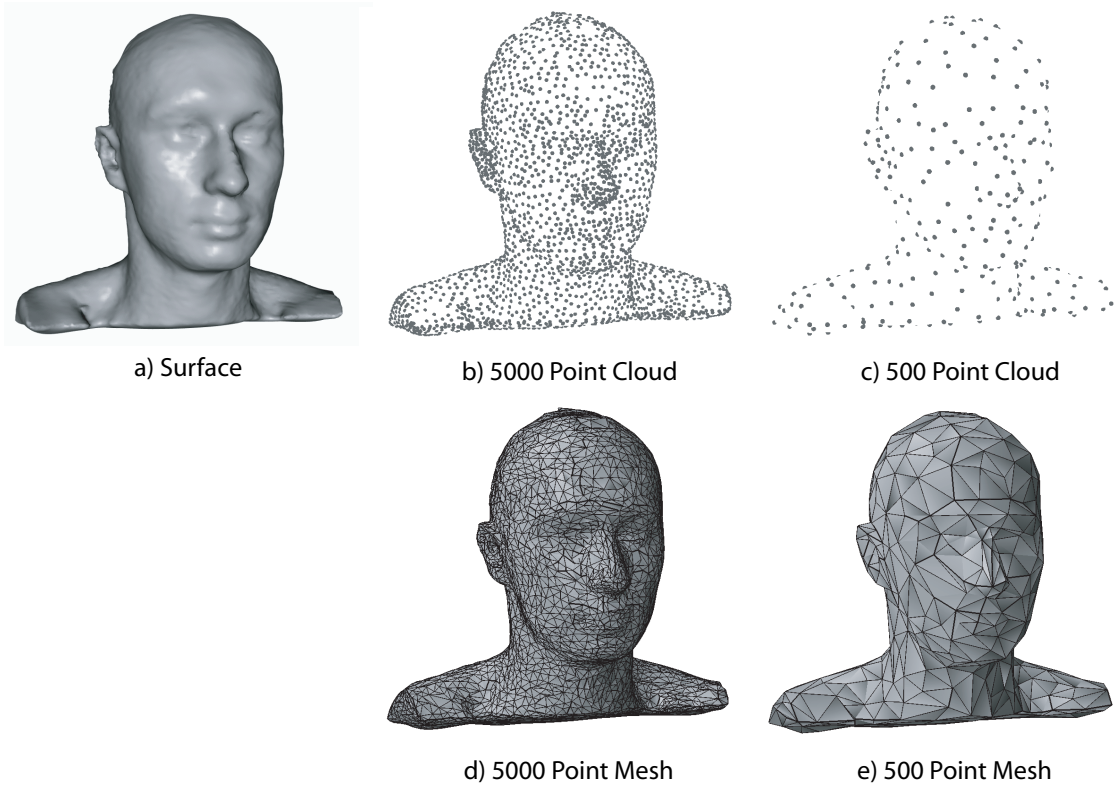


Figure 2.1: Surface discretization using point clouds and meshes.

of the edges $\mathcal{E}^X$ (in the method of Hähnel *et al.* [52]), or by looking at the deformation of the triangles $\mathcal{P}^X$ (in the method of Sumner and Popović [111]). In general, edges connect pairs of points rather than triples, and can be more efficient in combinatorial search methods (see Chapter 4). On the other hand, triangles are more convenient for expressing certain specialized constraints about the mesh surface. These design choices will be elaborated later when we introduce specific probabilistic models.

Meshes are very general representations – they can approximate any continuous surface arbitrarily well given a fine enough polygon tessellation. The surface connectivity information they contain is useful for a variety of purposes, such as extraction of high-level topological information about the surface, visualization, and editing of surface shape or appearance. Still, meshes tile the surface with a set of planar patches, and therefore are piecewise-linear surface representations. The resulting surfaces are only C0-continuous

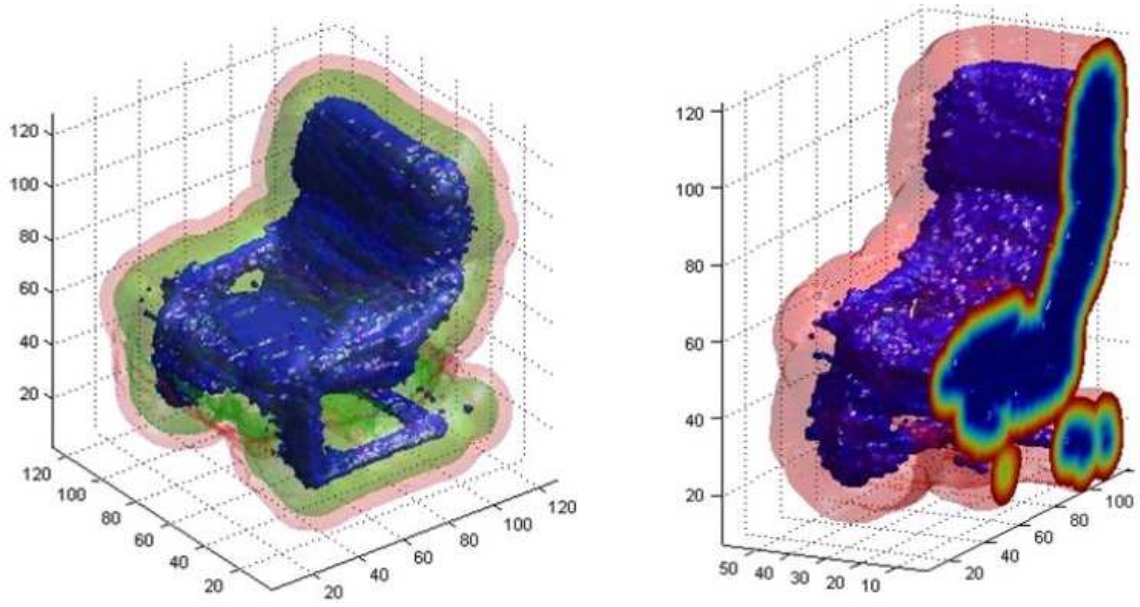


Figure 2.2: Signed distance map of a chair. The image on the right shows a cross-section of the chair, illustrating how the signed distance function protrudes into the object interior.

(non-differentiable at the triangle edges). In order to obtain nice smooth-looking shapes, a large number of triangles may be needed.

For this reason, many of the geometry editing tools use surface models that preserve a higher-order of surface continuity. These models are also known as *splines* – models of piecewise quadratic, cubic or higher-order polynomials that pass through a set of interpolation nodes and keep the derivative (and possibly, the second derivative of the surface) continuous everywhere. Among the many spline models, *b-splines* and *Bezier surfaces* are especially popular [46]. How to use these representations successfully in a machine learning setting is largely an open research question. These models come with an increased degree of complexity, as higher degree polynomials are used to represent the surface model. This causes the optimization problems for important tasks such as shape-completion and animation to become considerably more difficult for splines than for meshes.

2.1.2 Implicit Representations

The most popular implicit surface representation is the *signed distance map (SDM)* [103]. The SDM is a function that measures, for every point in 3D space, the distance to the nearest surface of the object. The distance is positive on the outside, and negative on the inside. Fig. 2.2 illustrates the SDM of a particular object. The SDM is represented over a discrete grid and can be efficiently computed from a mesh using just two passes over the grid [69]. It is also easy to extract any of the SDM isosurfaces [74]. For example, the isosurface containing all points in space for which the signed distance is zero corresponds to the original mesh surface.

The advantage of the SDM for representing object models is twofold. First, the SDM is defined everywhere in 3D space relative to an object. This property facilitates generalization, as it makes it straightforward to relate SDM representations of different objects to each other. Second, the SDM is smooth, which is essential for smooth interpolation between objects, and for well-behaved shape averaging. For the above reasons, SDMs have been used for a very popular scan merging algorithm by Curless and Levoy [33], as well as for CT-scan segmentation [69].

Unfortunately, signed distance maps also have several drawbacks. First, the SDM has to be defined for the entire 3D space, which tends to make it a more computation and memory-intensive representation than meshes. Second, averaging of SDMs produces reasonable surfaces only for largely convex shapes, and does not work for articulated objects such as humans. Finally, SDMs do not explicitly model the surface, making it difficult to model and enforce surface properties such as smoothness and contiguity.

For all of the reasons stated above, we will focus on meshes as our surface representation of choice for the rest of this thesis.

2.2 Rigid Body Transformations

Consider an object moving in the three-dimensional world. This movement can be described in terms of a series of *transformations*. Each transformation is a map, which describes the displacement of all object points between two moments in time. *Rotation* and

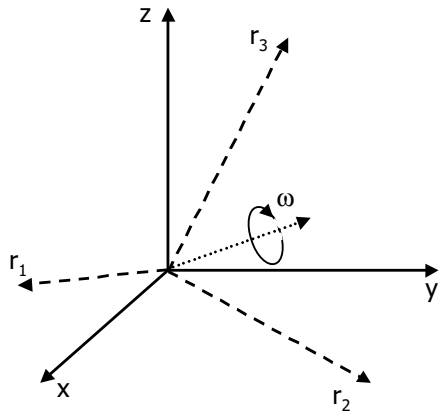


Figure 2.3: Rotation of a coordinate system around its origin and around the axis ω. The directions of the rotated main axes become r_1, r_2, r_3. These three orthonormal vectors comprise the columns of the rotation matrix R which was applied to the coordinate system.

translation are the simplest transformations, which preserve the object shape. We will use the term *rigid transformation* to denote any combination of rotational and translational motion. Because rigid transformations preserve the object shape, they can be expressed very compactly. Each object point x_i is rigidly transformed into point y_i as follows:

$$y_i = T(x_i) = R \cdot x_i + t, \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}. \tag{2.3}$$

Above, T denotes the rigid transformation, R is a matrix that accounts for the object rotation, and t is a vector corresponding to the object translation.

The rotation matrix R above has special properties. The best way to visualize this matrix is to consider the rotation of a particular coordinate frame around its origin (Fig. 2.3). The columns of R are three orthonormal vectors r_1, r_2, r_3 which correspond to the directions of the three principal coordinate axes after the rotation. Since the vectors r_1, r_2, r_3 form a right-handed frame, we further have the condition that the determinant of R must be 1. Formally, the space of all rotation matrices is denoted as follows:

$$SO(3) \doteq \{R \in \mathbb{R}^{3\times 3} \mid R^T R = I, \det(R) = 1\}. \tag{2.4}$$

The matrix representation of rotation, which we introduced so far, contains $3 \times 3 = 9$ entries. However, these 9 entries are not free parameters because they have to satisfy the orthonormality constraints $R^T R = I$. There are in fact 6 such constraints for the 9 entries, suggesting that we need a total of 3 free parameters in order to represent rotation.

2.2.1 Exponential Coordinates for Rotation

Here we will introduce an explicit parametrization for the space of rotations which uses only three parameters. Each rotation can be encoded with a three-dimensional vector $\omega = [\omega_1, \omega_2, \omega_3]^T$. Such a vector, known as an *exponential map*, has a clear and intuitive interpretation. The direction of the vector ω represents the axis around which the rotation will take place (see Fig. 2.3). The magnitude $\|\omega\|$ of the vector corresponds to the angle of rotation around that axis.

Given an exponential map ω, the corresponding rotation matrix $R(\omega)$ can be uniquely determined. In order to define this mapping, we will first introduce the concept of a *skew-symmetric matrix*.

Definition 2.2.1 *A* **skew-symmetric** *matrix U is a matrix of size 3×3 for which the equation $U^T = -U$ holds. Each skew-symmetric matrix has 3 free parameters. If $u = [u_1, u_2, u_3]^T$ is a 3-dimensional vector containing these parameters, the corresponding skew-symmetric matrix is defined as follows:*

$$\widehat{u} \doteq \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{2.5}$$

The space of all skew-symmetric matrices is commonly denoted as $so(3) \doteq \{\widehat{u} \in \mathbb{R}^{3\times 3} \mid u \in \mathbb{R}^3\}$.

A 3-dimensional vector ω can be mapped to a rotation matrix $R(\omega)$ by taking the exponent of its corresponding skew-symmetric matrix $\widehat{\omega}$:

$$R(\omega) = \exp(\widehat{\omega}) = I + \widehat{\omega} + \frac{\widehat{\omega}^2}{2!} + ... + \frac{\widehat{\omega}^n}{n!} + \dots \tag{2.6}$$

This equation is the reason for the vector ω to be called *exponential map*. It is also sometimes referred to as the *exponential coordinates* of the rotation.

Of course, an infinite series is not a practical way of obtaining rotation matrices from the exponential coordinates. An efficient way of doing this is provided by the following useful theorem:

Theorem 2.2.2 (Rodrigues' formula) *Given $\omega \in \mathbb{R}^3$, the matrix exponential $R(\omega)$ is given by:*

$$R(\omega) = \exp(\widehat{\omega}) = I + \frac{\widehat{\omega}}{\|\omega\|}\sin(\|\omega\|) + \frac{\widehat{\omega}^2}{\|\omega\|^2}(1 - \cos(\|\omega\|), \tag{2.7}$$

where $\widehat{\omega}$ is a 3×3 skew-symmetric matrix, defined as in Eqn. (2.5).

The reverse is also possible: we can express each rotation matrix R in its exponential form. Intuitively, this is true because each rotation matrix can be realized by rotating around some axis ω by angle $\|\omega\|$. However, many ways of doing this are possible, therefore the mapping from R to some parameters $\widehat{\omega} \in so(3)$ is not one-to-one. To see why, recall that rotation by angle $2\pi + \theta$ has the same effect as rotation by θ. A compact way of performing this inverse mapping is provided in the next theorem.

Theorem 2.2.3 (Logarithm of SO(3)) *For any $R \in SO(3)$, there exists (a not necessarily unique) $\omega \in \mathbb{R}^3$ such that $R = \exp(\widehat{\omega})$. The inverse of the exponential map is denoted as $\widehat{\omega} = \log(R)$. The exponential map parameters are given by*

$$\|\omega\| = \cos^{-1}\left(\frac{\mathrm{trace(R)} - 1}{2}\right), \frac{\omega}{\|\omega\|} = \frac{1}{2\sin(\|\omega\|)}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \tag{2.8}$$

where r_{ij} are corresponding entries from the rotation matrix R, as defined in Eqn. (2.3).

The exponential coordinates introduced in this section provide a simple and intuitive parametrization of rotation matrices. One of its important uses of this representation is that it provides a simple algebraic way of "linearizing" the rotation matrix (in other words, for obtaining a locally linear estimate of the space of rotation matrices). Recall Eqn. (2.6),

which maps exponential coordinates to rotation matrix parameters. If we take only the first two terms from that series, we obtain the following linear approximation:

$$R_0(\omega) \approx I + \widehat{\omega}. \tag{2.9}$$

This approximation is most accurate for matrices numerically similar to the identity matrix $I = R(0)$, and its predictions worsen as $\|\omega\|$ increases. To obtain a good local estimate of rotation in the vicinity of some other set of exponential coordinates ϑ, we can compose the rotation matrices as follows:

$$R_\vartheta(\omega) \approx (I + \widehat{\omega})R(\vartheta). \tag{2.10}$$

This algebraically simple formula is frequently used in problems where optimization over rotation is required. The gracious reader will see examples of this further on in this thesis.

2.2.2 Quaternions

Here we will briefly describe an alternative representation of rotation in terms of *unit quaternions*. A more extensive discussion of quaternions can be found in Ma *et al.* [75].

Definition 2.2.4 *The group of unit quaternions contains all vectors from a 4-dimensional unit sphere:*

$$\mathbb{S}^3 \doteq \{q \in \mathbb{R}^4 \mid \|q\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1\}. \tag{2.11}$$

Unit quaternions are a popular representation of rotation in terms of 4 parameters and a constraint. A rotation around axis $\omega = [\omega_1, \omega_2, \omega_3]$ by angle r is represented by the unit quaternion

$$q = [q_0\ q_1\ q_2\ q_3]^T = [\cos(r/2)\ \ \sin(r/2)\omega_1\ \ \sin(r/2)\omega_2\ \ \sin(r/2)\omega_3]^T. \tag{2.12}$$

In intuitive terms, the relative size of coefficient q_0 encodes the amount of rotation, while the vector $[q_1\ q_2\ q_3]^T$ points along the axis of rotation. Because of this, it is easy to see that the unit quaternions q and $-q$ correspond to the same rotation matrix. In fact, it can be shown that each rotation matrix can be described with exactly 2 such quaternions, and

consequently that the group $\mathbb{S}^3$ is a double covering of the group of rotations $SO(3)$. Compare this to the exponential coordinate representation, where each rotation matrix can be associated with an infinite amount of exponential maps (all related by periodicity).

There exists a straightforward mapping from unit quaternions to rotation matrix parameters:

Theorem 2.2.5 *A unit quaternion q is associated with the following rotation matrix:*

$$R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{2.13}$$

The most remarkable thing about this mapping is that each matrix coefficient can be represented in as a second degree polynomial function in terms of the quaternion parameters. This property comes in very handy in some optimization problems.

2.3 Aligning Two Corresponding Point Clouds

For example, consider the very useful problem of aligning two point clouds, when the point-to-point correspondences between them are known. It turns out that using the unit quaternion representation of rotation, this problem can be solved in a straightforward manner [13]. Below we briefly describe the solution. For simplicity, we will assume that point clouds $\mathcal{P}^X$ and $\mathcal{P}^Y$ have the same number of points N and that we know point x_1 corresponds to y_1, x_2 to y_2 and so on. The objective is to find the rigid transformation T (unit quaternions q and translation t) that best aligns the corresponding points. This objective can be written as:

$$\min F(q,t) = \sum_{i=1}^{N} \|R(q)x_i + t - y_i\|^2, \ \text{s.t.} \ \ \|q\|^2 = 1. \tag{2.14}$$

First, the translation vector t can be expressed simply as the difference between the point cloud centroids

$$t = \frac{1}{N}\sum_{i=1}^{N} y_i - R(q)\frac{1}{N}\sum_{i=1}^{N} x_i = \mu_Y - R(q)\mu_X. \tag{2.15}$$

We can substitute for t in the resulting Lagrangian function:

$$\begin{aligned} L(q) &= F(q,t) + \lambda(1 - \|q\|^2) = \\ &= \sum_{i=1}^{N} \|R(q)(x_i - \mu_X) + (\mu_Y - y_i)\|^2 + \lambda(1 - q_0^2 - q_1^2 - q_2^2 - q_3^2). \end{aligned}$$

We expand the first term in the Lagrangian:

$$\begin{aligned} & \sum_{i=1}^{N} \|R(q)(x_i - \mu_X) + (\mu_Y - y_i)\|^2 = \\ = & \sum_{i=1}^{N} \{(x_i - \mu_X)^T R(q)^T R(q)(x_i - \mu_X) + 2(\mu_Y - y_i)(x_i - \mu_X)^T R(q)^T + (\mu_Y - y_i)^T(\mu_Y - y_i)\} = \\ = & \sum_{i=1}^{N} \{(x_i - \mu_X)^T (x_i - \mu_X) + 2(\mu_Y - y_i)(x_i - \mu_X)^T R(q)^T + (\mu_Y - y_i)^T(\mu_Y - y_i)\} \end{aligned}$$

Above we used the fact that rotation matrices are orthonormal and hence $R^T R = I$. The resulting Lagrangian is only *linear* in terms of the rotation matrix parameters, and only *quadratic* in terms of the quaternion parameters q (recall the rotation matrix definition in Eqn. (2.13)):

$$\begin{aligned} L(q) &= \sum_{i=1}^{N} \{2(\mu_Y - y_i)(x_i - \mu_X)^T R(q)^T + (x_i - \mu_X)^T (x_i - \mu_X) + (\mu_Y - y_i)^T(\mu_Y - y_i)\} + \\ &+ \lambda(1 - q_0^2 - q_1^2 - q_2^2 - q_3^2). \end{aligned} \tag{2.16}$$

Now, when we take the derivative $\partial L/\partial q$ we obtain an equation of the form:

$$Aq = \lambda q, \tag{2.17}$$

where A is a symmetric 4×4 matrix and λ is a positive scalar. The largest eigenvector of matrix A corresponds to the quaternions q which minimize our objective.

Chapter 3

Probabilistic Graphical Models

In a world of shifting shapes and imprecise sensors, we need to be able to deal with the underlying uncertainly of everything that surrounds us. Probabilistic graphical models provide a means of encoding the inherent structure of the world's complex environments in a compact fashion. They are representations of the joint probability distribution over a set of variables, and their structure can be utilized to perform efficiently the tasks of reasoning and learning. In particular, the theory of graphical models provides us with the capability to reason about the assignments to a large number of variables simultaneously. This capability allows us to tackle difficult combinatorial problems such as registration and segmentation, which will be explored in depth later in this thesis.

In this chapter, we will present the basics of the probabilistic framework that underlies our models. We will briefly describe two different representations of uncertainty, directed and undirected graphical models, which are both used in our algorithms. Then we present the computational tools which we will employ for reasoning and learning in such networks. We will describe inference methods such as belief propagation, and linear programming relaxations for maximum-a-posteriori inference in an important subclass of Markov networks. We will also review maximum-likelihood learning and describe the expectation-maximization algorithm for learning in networks that contain hidden variables.

3.1 Bayesian Networks

We are often interested in modeling the *joint probability distribution* $P(\mathcal{X})$ over a set of variables $\mathcal{X} = \{X_1, \ldots, X_N\}$. Each variable X_i can take a set of possible values, which is called the *domain* of X_i, and is denoted as $dom(X_i)$. In our notation, the *instantiation* of variable X_i to some value $x_i \in dom(X_i)$ will be denoted as $X_i = x_i$.

Assuming the variables in $\mathcal{X}$ are discrete, the joint distribution can be represented simply as a table, which associates a *probability value* $P(\mathcal{X} = x)$ with each instantiation $\{\mathcal{X} = x\} \equiv \{X_1 = x_1, \ldots, X_n = x_n\}$. Given this table, we can answer different kinds of queries about the variables. Most often, we are interested in asking conditional probability queries, of the form: give me the probability $P(\mathcal{Z} = z \mid \mathcal{Y} = y)$, for any subsets $\mathcal{Y}$ and $\mathcal{Z}$ of the complete set $\mathcal{X}$. The answers can be computed simply by summing the appropriate probabilities in our table:

$$P(\mathcal{Z} = z \mid \mathcal{Y} = y) = \frac{P(\mathcal{Z} = z, \mathcal{Y} = y)}{P(\mathcal{Y} = y)} = \frac{\sum_{x:(z,y \subset x)} P(\mathcal{X} = x)}{\sum_{x:(y \subset x)} P(\mathcal{X} = x)}, \tag{3.1}$$

where the notation $y \subset x$ is used to express that the instantiation y is part of the instantiation x. The problem with this representation is that a distribution over N binary variables requires a table containing $2^N - 1$ independent parameters.

Joint probability distributions can also be represented as a product of conditional probability distributions. A *conditional probability distribution* (CPD) $P(\mathcal{X} \mid \mathcal{Y})$ defines the conditional probabilities $P(\mathcal{X} = x \mid \mathcal{Y} = y)$ for all possible values $x \in dom(\mathcal{X}), y \in dom(\mathcal{Y})$. A CPD $P(\mathcal{X} \mid \mathcal{Y})$ over a set of discrete variables is a table containing the conditional probabilities for all possible assignments to the variables in $\mathcal{X}$ and $\mathcal{Y}$. Joint probability distributions can be represented in terms of conditional probability distributions using the chain rule:

$$P(X_1, X_2, \ldots X_n) = P(X_1)P(X_2 \mid X_1) \ldots P(X_n \mid X_1, \ldots X_{n-1}) \tag{3.2}$$

The chain rule is a mathematical equivalence; for binary discrete variables we still need to specify $2^N - 1$ independent parameters in the conditional probability tables.

Above we showed that as the number of variables in the domain grows, the general distribution representations very quickly become intractable. This can be addressed by using

the *Bayesian Network* (BN) framework, which can encode the qualitative properties of the domain, resulting in much more compact representations of joint probability distributions.

3.1.1 Conditional Independence

The Bayesian network formalism is based on the notion of *conditional independence*. We explain this concept using the classical example from Pearl's book [90]. The setting for the example is a simple domain, in which a house alarm (A) can be triggered either by burglary (B) or by an earthquake (E). If the alarm is triggered by any of these causes or spontaneously, a call (C) from the neighbor can be expected. In addition, an earthquake is usually followed by a radio report (R).

There is inherent structure in this domain, which the general probability distribution representation (joint or conditional probability tables) does not capture. For example, the events of burglary (B) and earthquake (E) are generally deemed to occur independently of each other[1]. Our belief that the neighbor will call (C) is independent of a cause that might trigger the alarm if we already know that the alarm (A) was activated. Similarly, if the alarm (A) has been activated, the radio report (R) of an earthquake may change our belief whether burglary (B) occurred, but is no longer relevant if we actually felt the earthquake (E).

The concept of *conditional independence* allows us to formalize these properties.

Definition 3.1.1 *We say that $\mathcal{X}$ is* **conditionally independent** *of $\mathcal{Y}$ given $\mathcal{Z}$ if*

$$P(\mathcal{X} \mid \mathcal{Y}, \mathcal{Z}) = P(\mathcal{X} \mid \mathcal{Z}) \quad \textit{when } P(\mathcal{Z}) > 0$$

and we denote this statement by $P \models (\mathcal{X} \perp \mathcal{Y} \mid \mathcal{Z})$.

We can represent a subset of conditional independence assumptions associated with a particular domain using a directed graph.

[1]The recent events connected with the hit of hurricane Katrina on New Orleans cast some doubt on the lack of connection between natural disasters and increased criminal activity.

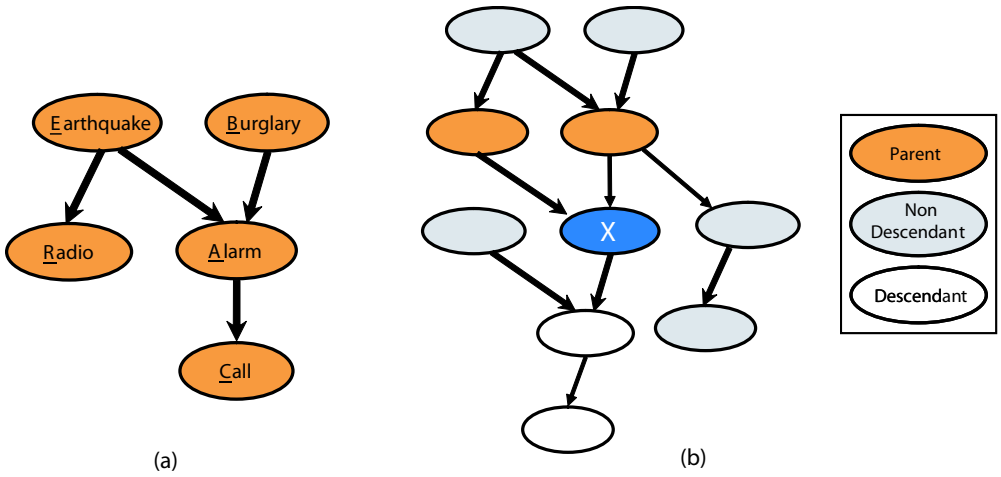


Figure 3.1: (a) An example of a simple Bayesian network structure for the Alarm domain. This network structure implies several conditional independence statements: $(E \perp B)$,$(A \perp R \mid B, E)$,$(R \perp A, B, C \mid E)$, and $(C \perp B, E, R \mid A)$. (b) Markov independence statements in a Bayesian network. X is independent of all its non-descendant nodes in the graph $\mathcal{G}$, given its parent nodes.

Definition 3.1.2 *Let $\mathcal{G}$ be a* **directed acyclic graph** *(DAG) whose vertices correspond to random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$. We say the $\mathcal{G}$ encodes a set of* **Markov independence statements***: Each variable X_i is independent of its non-descendants, given its parents in $\mathcal{G}$.*

$$\forall X_i \ (X_i \perp \text{NonDescendants}_{X_i} \mid \text{Pa}_i) \tag{3.3}$$

and we denote the set of these statements as Markov$(\mathcal{G})$.

The DAG corresponding to our Alarm example is displayed in Fig. 3.1(a), while Fig. 3.1(b) illustrates the concept of the Markov independence statements.

Using the rules of probability, we can infer additional independence statements from *Markov*$(\mathcal{G})$. For example, in Fig. 3.1, we can say that $(A \perp R \mid E)$. This follows from $(R \perp A, B, C \mid E) \Rightarrow (R \perp A \mid E)$ and the *Symmetry of Independence* property of conditional probabilities. Similarly, it is easy to see that all the independence statements

we made in the case of the burglary alarm domain follows directly from the Markov independence statements encoded in the graph from Fig. 3.1(a). The set of conditional independence statements encoded by the DAG structure is fairly easy to elicit using a set of graph-theoretic criteria, known as *d-separation*. We will omit a detailed discussion of d-separation here, but refer the reader to Pearl's book [90].

3.1.2 Model Definition

We can now formally define the Bayesian network model.

Definition 3.1.3 *A* **Bayesian network** $\mathcal{B} = \langle \mathcal{G}, \theta \rangle$ *is a representation of a joint probability distribution over a set of random variables* $\mathcal{X} = \{X_1, \ldots, X_N\}$*, consisting of two components: A directed acyclic graph* $\mathcal{G}$ *whose vertices correspond to the random variables and that encodes the Markov independence assumptions* $\text{Markov}(\mathcal{G})$*; a set of parameters* θ *that describe a conditional probability distribution (CPD)* $P(X_i \mid \text{Pa}_i)$ *for each variable* X_i *given parents in the graph* Pa_i*. The probability distribution defined by the graph* $\mathcal{G}$ *and parameters* θ *can be written as follows:*

$$P(X_1, \ldots, X_N) = \prod_{i=1}^{n} P(X_i \mid \text{Pa}_i^{\mathcal{G}}). \tag{3.4}$$

Eqn. (3.4) is known as the *chain rule for Bayesian networks*. The CPDs in this product are smaller than those in the original chain rule from Eqn. (3.2). As an example, consider the joint probability distribution $P(B, E, R, A, C)$ represented in Fig. 3.1(a). By the chain rule of probability, without any independence assumptions:

$$P(B, E, R, A, C) = P(B)P(E \mid B)P(R \mid B, E)P(A \mid B, E, R)P(C \mid B, E, R, A, C)$$

Assuming we have binary event variables, this representation requires $1+2+4+8+16 = 31$ parameters. Taking the conditional independencies into account we can write

$$P(B, E, R, A, C) = P(B)P(E)P(R \mid E)P(A \mid B, E)P(C \mid A)$$

which only requires $1 + 1 + 2 + 4 + 2 = 10$ parameters. More generally, if $\mathcal{G}$ is defined over N binary variables and its in-degree (i.e., maximal number of parents) is bounded by K, then instead of representing the joint distribution with $2^N - 1$ independent parameters we can represent it with at most $(2^K - 1)N$ independent parameters.

3.2 Markov Networks

Bayesian networks are an appropriate fit for many domains, which can benefit from their directed graph structure. This is often the case for domains whose variables are representing cause and effect relationships between events. Bayesian networks whose links follows the causal structure are usually compact and intuitive (although Baysian networks themselves make no claim to encode causal relationships). In other domains, there is no natural way to exploit the link directionality of Bayesian networks. For example, surfaces are complex manifolds in 3D space, for which the surface directionality notion is largely irrelevant in the vast majority of cases, and can even be a nuisance.

In this section, we review undirected graphical models known as *Markov networks* or *Markov random fields* (MRFs) [53, 90]. MRFs offer an alternative approach for encoding independence structure in joint probability distributions. We introduce them using a simple example, and will then generalize it into a formal MRF definition.

In our example, we will define a distribution over the possible segmentations of a surface into K regions. We want this distribution to prefer segmentations in which adjacent points p_i and p_j are assigned to the same region. The surface is represented as a discrete sampling of points, in which adjacent points are connected by links. Each point p_i is associated with a discrete variable X_i which can take K values, assigning the point to one of the respective regions. To represent our preference that adjacent points are assigned to the same region formally in the model, we will associate a measure, called a *potential*, with pairs of variables (X_i, X_j).

Definition 3.2.1 *Let $\mathcal{Y}$ be a set of random variables, and let $dom(\mathcal{Y})$ be their joint domain. A* **potential** *(or* **factor***) $\psi(\mathcal{Y})$ is a mapping from $dom(\mathcal{Y})$ to $\mathbb{R}^+$.*

This notion is very similar to that of a probability function, in the sense that each assignment

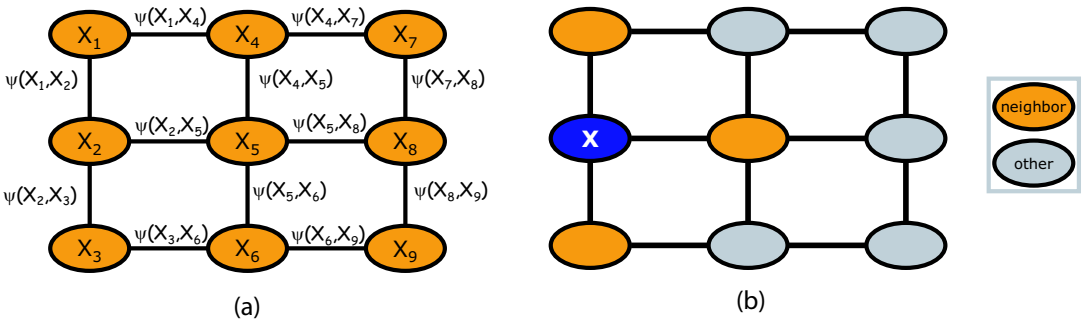


Figure 3.2: (a) A Markov network for our surface partitioning problem. The surface is represented by 9 points, connected in a grid. (b) Conditional independence in the Markov network: variable X is independent of the rest of the network variables, given its immediate neighbors.

is mapped to a score correlated with our belief about its likelihood. Unlike probability distributions, potentials do not need to be normalized. In our example, the potential assigns higher preference values to cases when adjacent points p_i and p_j belong to the same region. A plausible potential for the $K = 2$ case is displayed below:

X_i	X_j	$\psi(X_i, X_j)$
1	1	10
1	2	1
2	1	1
2	2	10

(3.5)

As we will see shortly, Markov random fields represent joint probability distributions in terms of a product of such potentials. Each potential is associated with a fully-connected group of variables, called a *clique*. The set of connections between the variables of all cliques induces an undirected graph $\mathcal{U}$. This graph can be used to encode a set of conditional independence assumptions in the underlying distribution.

Definition 3.2.2 *Let $\mathcal{U}$ be an undirected graph whose vertices correspond to random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$. We say the $\mathcal{U}$ encodes a set of* **Markov independence statements***: Each variable X_i is independent of all other variables in the network, given all its neighbors in $\mathcal{U}$:*

$$\forall X_i \ (X_i \perp \{\mathcal{X} \setminus X_i\} \mid \text{Neighbors}(X_i)) \tag{3.6}$$

We denote the set of all such independence properties as Markov($\mathcal{U}$).

As a consequence of this definition, the *Markov blanket* $\mathbf{MB_{X_i}}$ of each variable X_i in the graph is simply the set of its neighbors in the graph. Fig. 3.2(b) illustrates a specific example of the Markov blanket concept in undirected graphs.

We are now ready to formally define the concept of Markov networks.

Definition 3.2.3 *A* **Markov network** *(or* **Markov random field***) $\mathcal{F} = \langle \mathcal{U}, \Psi \rangle$ is a representation of a joint probability distribution over a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$, consisting of two components. The undirected graph $\mathcal{U}$ has vertices that correspond to the variables, and encodes a set of Markov independence assumptions* Markov($\mathcal{U}$)*. The set of potentials Ψ is associated with cliques in the graph, and is used to define the distribution*

$$P(X) = \frac{1}{Z} \prod_c \psi_c(X_c) \tag{3.7}$$

The value Z is the normalization factor (also known as the **partition function***), defined as*

$$Z = \sum_X \prod_c \psi_c(X_c). \tag{3.8}$$

A Markov network is a factored representation of a joint probability distribution as a product of small local factors. A subset of the class of Markov networks, called *pairwise* Markov networks, contains factors only over single variables and pairs of variables. Pairwise Markov networks will be sufficient for the problems tackled in this thesis. An example of a pairwise Markov network for our surface partitioning example is displayed in Fig. 3.2(a).

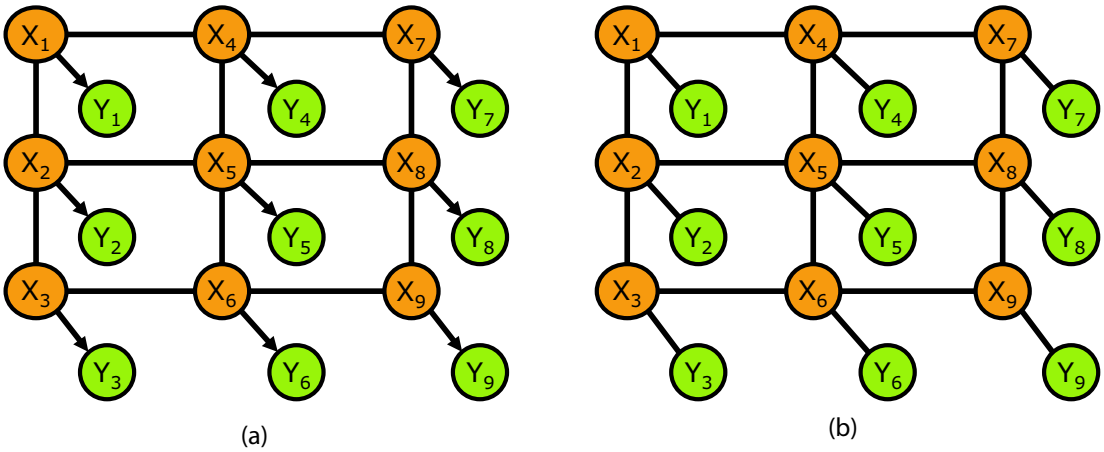


Figure 3.3: (a) A model combining directed and undirected links, for our *surface partitioning with evidence* example. Different evidence Y is generated depending on which region of the surface it is linked to (determined by the variables X). (b) The Markov network encoding the same conditional independencies.

3.2.1 Incorporating Directed Potentials

Despite the fact that in this dissertation we will be reasoning about surfaces, which are more intuitively modeled with Markov networks, there is cause-and-effect structure in our models as well. All the algorithms in this thesis are based on a *probabilistic generative framework* — for each problem, we define a probabilistic model which describes our beliefs about how the world works and how the evidence provided to the algorithm was obtained. The cause-and-effect relationships between events in the world and the evidence are naturally represented using conditional distributions (which are usually part of the Bayesian network definition). For this reason, throughout this thesis our graphical models will be defined in terms of both undirected potentials and conditional distributions. Probabilistic graphical models that combine directed and undirected interactions are called *chain graphs* [20]. However, the mixed models that we will be exploring in this thesis are easily converted to Markov networks — hence, we will not be exploring in detail the machinery of chain graphs.

In our models, a set of variables $\mathcal{X}$ that define a surface prior are connected with undirected edges. We have another set of variables $\mathcal{Y}$, in which each variable Y_i is connected using a set of directed links a single parent $X_i \in \mathcal{X}$. An example of such a model is displayed in Fig. 3.3(a). In this extension of the original surface partitioning example from Fig. 3.2(a), each evidence variable Y_i is connected to only a single surface variable X_i in the graph. The conditional probabilities $P(Y_i \mid X_i)$ capture the fact that different regions of the surface tend to produce different evidence. The joint probability distribution becomes

$$\begin{aligned} P(\mathcal{X} = x, \mathcal{Y} = y) &= P(\mathcal{X} = x) P(\mathcal{Y} = y \mid \mathcal{X} = x) \\ &= \left[\frac{1}{Z_x} \prod_c \psi_c(x_c) \right] \left[\prod_i P(y_i \mid x_i) \right] \end{aligned} \tag{3.9}$$

Note that the partition function Z_x is defined by sum over all assignments to $\mathcal{X}$ (as in Eqn. (3.8)). The conditional probabilities $P(Y_i \mid X_i)$ do not contribute to the partition function, because $\sum_{y_i} P(y_i \mid X_i) = 1$, and can be factored out. The joint probability distribution remains unchanged if these conditional probability tables are treated as potentials in a Markov network. The equivalent Markov network for our example on surface partitioning with evidence is displayed in Fig. 3.3(b).

3.3 Inference

Inference is a fundamental task in graphical models. Both the Bayesian and the Markov network formalisms represent joint probability distributions, and contain sufficient information to answer any question about these distributions. We can ask *conditional probability queries*, in which we want to compute $P(\mathcal{Y} \mid \mathcal{Z} = z)$, the probability of a set of variables $\mathcal{Y}$ given some evidence $\mathcal{Z} = z$. We can also ask *maximum a-posteriori (MAP)* queries, which ask for the most likely assignment to all the non-evidence variables. Here we want $\arg\max_{\mathcal{Y}} P(\mathcal{Y} \mid \mathcal{Z})$, and we will assume that if there are several most-likely assignments any of them will suffice.

In this section, we will describe several inference algorithms that are used in this thesis.

Because the algorithms for answering conditional probabilities and MAP queries differ, we will introduce them separately. First, in Sec. 3.3.1 we give a general overview of the problem of answering conditional probability queries, and review a specific inference algorithm for solving the problem. In Sec. 3.3.2 we review a specific inference algorithm for the problem, called *Belief Propagation*. The problem of answering MAP queries is discussed separately in Sec. 3.3.3. There we introduce a special subclass of Markov networks, called *associative* Markov networks, for which MAP-inference can be performed efficiently. We describe two specific algorithms for inference in associative Markov networks (based on linear programming relaxation and minimum-cuts in a graph, respectively).

3.3.1 Answering Conditional Probability Queries

As an example of a conditional probability query, consider the task of evaluating the probability of getting a call from our neighbor $P(C)$ in the Alarm network from Fig. 3.1. By the complete probability formula

$$P(C) = \sum_{b,e,a,r} P(b, e, a, r, C)$$

We can improve on this by utilizing the decomposition of the joint probability in the Bayesian network:

$$P(C) = \sum_{a} P(C|a) \sum_{e} P(e) \sum_{b} P(b) P(a|b, e) \sum_{r} P(r|e). \tag{3.10}$$

The resulting equation allows us to compute the probability much more efficiently, using *dynamic programming*. We can sum the variables in order (from right to left), and keep the summation results in intermediate cliques. This process is known as *variable elimination*, and it allows us to avoid computing the same summations multiple times. Most exact inference algorithms (e.g., *Junction Trees* (e.g., [59] and *Bucket Elimination* [38]) exploit the above idea.

It has been shown, however, that the general inference problem in Bayesian networks is NP-hard [30] (in fact, it is #P-complete). The related problem of performing inference

in a Markov network is also NP-hard. The difficulty of the problem is correlated to the structure of the underlying graphical model. For example, exact inference takes linear time for graphical models whose underlying graph is a tree (such graphs are also called *singly-connected*). The difficulty of the inference task tends to increase as the underlying graph structure gets more complex. In some graphical models with many cycles, exact inference is infeasible.

Sometimes we may be willing to accept an approximate answer. In these cases, we resort to approximate inference methods. These include instance or particle based methods such as *Gibbs sampling* (see [88] for an overview of inference sampling techniques), variational approximation method such as the *Mean Field* approximation (see [61] for an introduction) and *Loopy Belief Propagation* (e.g., [86] and references within). While these methods have shown great success in different scenarios, like exact inference, approximate inference in general is also NP-hard [34] and choosing the best method of inference for a particular task remains a challenge. In the next section, we will review the Belief Propagation method which will be used later in this thesis.

3.3.2 Belief Propagation

The *Belief Propagation (BP)* algorithm was originally proposed by Pearl [90]. It performs exact inference in singly-connected graphical models, but also provides good empirical results on graphical models with cycles [86] (in those cases, it is known as *Loopy Belief Propagation (LBP)*). In this section, we describe a special case of the algorithm, which can be executed on pairwise Markov networks. This is not a strong restriction, because all graphical models can be converted to such Markov networks [127]. Furthermore, it will be sufficient to restrict our attention to cases when all variables in the network are discrete.

The input to this variant of the Belief Propagation (BP) algorithm is a Markov network $\mathcal{F}$ with discrete variables, containing single and pairwise potentials $\psi_i(X_i)$ and $\psi_{i,j}(X_i, X_j)$, respectively. We are interested in obtaining the marginal beliefs $P(X_i)$ for all variables $X_i \in \mathcal{X}$. In the BP algorithm, this goal is accomplished by scheduling a set of local computations, in which variables send *messages* to their neighbors in the network in order to update their beliefs. These messages are tables associated with the edges of the Markov

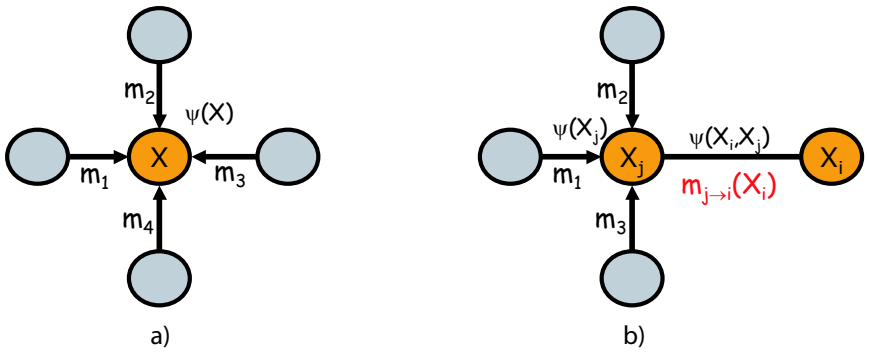


Figure 3.4: Belief Propagation operations. a) Illustration of the belief update $b_i(x_i) = k\psi_i(x_i)\prod_{j\in\text{Neighbors}(X_i)} m_{j\to i}(x_i)$. b) Illustration of the message update $m'_{j\to i}(x_i) = \sum_j \psi_j(x_j)\psi_{i,j}(x_i,x_j)\prod_{k\in\{\text{Neighbors}(X_j)\setminus i\}} m_{k\to j}(x_j)$.

network. We will use $m_{i\to j}(X_j)$ to denote the message sent by node X_i to update the belief of its neighbor X_j.

At any point in time, the BP algorithm maintains an estimate of the marginal probabilities at all Markov network nodes. We call such estimates *beliefs* and use $b_i(x_i)$ to denote the estimate of the probability that $X_i = x_i$. Naturally, we require that $\sum_{x_i} b_i(x_i) = 1$. The entire set of beliefs associated with a node X_i is denoted as $b_i(X_i)$. The beliefs can be computed from the incoming messages sent by the node's neighbors using the following *belief update* rule:

$$b_i(x_i) = k\psi_i(x_i) \prod_{j\in\text{Neighbors}(X_i)} m_{j\to i}(x_i). \tag{3.11}$$

Here, $\psi_i(x_i)$ denotes the single potential associated with node X_i, which also needs to be factored in. The normalization constant k ensures that the beliefs sum to 1. The belief computation process is illustrated in Fig. 3.4(a).

The messages themselves are computed from the following recursive message update rule (which is also known as the *Sum-product* rule):

$$m_{j\to i}(x_i) = \sum_j \psi_j(x_j)\psi_{i,j}(x_i,x_j) \prod_{k\in\{\text{Neighbors}(X_j)\setminus X_i\}} m_{k\to j}(x_j) \tag{3.12}$$

Algorithm **Belief Propagation**

Input: Pairwise Markov network $\mathcal{F}$.

Output: $P(X_i)$ for all variables X_i in $\mathcal{F}$.

1: Start with uniform messages $m_{j\to i}(x_i) = 1$ and beliefs $b_i(x_i) = 1$.

2: Recompute the messages:

$$m'_{j\to i}(x_i) = \sum_j \psi_j(x_j)\psi_{i,j}(x_i, x_j) \prod_{k\in\{\text{Neighbors}(X_j)\setminus i\}} m_{k\to j}(x_j)$$

3: Compute the new node beliefs $b'_i(X_i)$:

$$b'_i(x_i) = k\psi_i(x_i) \prod_{j\in\text{Neighbors}(X_i)} m'_{j\to i}(x_i).$$

4: If for some x_i, $|b'(x_i) - b_i(x_i)| > \epsilon$, repeat steps 2 and 3 using the new messages and beliefs.

5: **return** Beliefs $b(X_i)$.

Figure 3.5: Belief Propagation algorithm with parallel message updates.

In this equation, the right-hand size includes the messages from all neighbors of X_j except X_i. This is illustrated in Fig. 3.4(b). Intuitively, the message $m_{j\to i}(x_i)$ conveys information from node X_j and its neighbors to node X_i.

The Belief Propagation algorithm is defined in Fig. 3.5. In each BP stage, we compute the messages in the network using the update rule from Eqn. (3.12). Because the rule is recursive, during the computation we use the old estimates of the messages remaining from the previous stage. Given a new set of messages, we can estimate the beliefs at the network nodes using Eqn. (3.12). The process is repeated until the updates stop changing the beliefs in the network. The algorithm may not converge in some cases, in which case we stop after a predefined maximum number of iterations. A discussion of the some of the algorithm's convergence properties can be found in Yedidia *et al.* [127].

Different versions of the algorithm are possible, depending on the way message updates are scheduled. The updates can be done in parallel (where all messages are estimated at the same time), or in a sequence (where the each subsequent update uses the message estimates available at that moment). Sequential message updates offer some computational savings as well as better convergence, but require the algorithm designer to come up with

an appropriate order for computing the messages. In this thesis, we use BP with parallel message updates.

3.3.3 MAP Inference

MAP inference is the problem of answering *maximum a-posteriori* queries $\arg\max_{\mathcal{Y}} P(\mathcal{Y} \mid \{\mathcal{X} \backslash \mathcal{Y}\})$. Here we give a brief overview of the problem, and describe a special subclass of Markov networks, called *associative* Markov networks, in which efficient inference with performance guarantees is possible. Then we describe two different algorithms for inference in associative Markov networks, which are used in this thesis.

First, we revisit our favorite Alarm network example. Assume we know that the Alarm is on $(A = a_1)$, and the neighbor didn't call $(C = c_0)$. The most likely joint assignment to the remaining variables can be computed in a straightforward manner:

$$\arg\max_{b,r,e} P(b, r, e \mid a_1, c_0) = \arg\max_{b,r,e} \frac{P(b, r, e, a_1, c_0)}{P(a_1, c_0)} = \arg\max_{b,r,e} P(b, r, e, a_1, c_0)$$

Above we used Bayes' rule and the fact that $P(a_1, c_0)$ is constant relative to the $\arg\max$ variables. As a result, the answer can be obtained from a full joint probability table, by looking up the most probable entry consistent with the evidence $A = a_1, C = c_0$.

In general Markov and Bayesian networks, MAP inference is NP-hard [32]. Because the problems are related, many inference techniques for answering conditional probability queries can be modified to answer MAP queries. For example, recall the *Sum-product* message update rule for BP, defined in Eqn. (3.12). It can be changed to the following *max-product* rule:

$$m_{j \to i}(x_i) = \max_j \psi_j(x_j)\psi_{i,j}(x_i, x_j) \prod_{k \in \{\text{Neighbors}(X_j) \backslash i\}} m_{k \to j}(x_j) \tag{3.13}$$

The BP algorithm using this rule is known as *max-product* BP, and produces exact answers to MAP queries for singly-connected networks. While it can also be used to obtain in approximate answers in general loopy graphs, in our experience it is less likely to converge to a good local minimum than the original *sum-product* BP from Sec. 3.3.2.

Another way to keep inference tractable and efficient is to limit the form of the potentials in the network. Below we describe an important subclass of models, called *associative Markov networks* [114].

Definition 3.3.1 *An* **associative Markov network** *(AMN) contains discrete variables with K labels and arbitrary-size clique potentials with K parameters that favor the same labels for all variables in the clique. In particular, for a clique c over variables $X_1, \ldots, X_n$, we have*

$$\psi_c(x_c) = \begin{cases} \lambda_c^k & \text{if } x_1 = \ldots = x_n = k; \\ 1 & \text{otherwise} \end{cases}$$

with the additional requirement that $\forall c, k \;\; \lambda_c^k \geq 1$. Potentials of this form will be called **attractive potentials**.

AMN potentials can be used to capture positive interactions, in which connected (associated) variables tend to have the same label. In particular, they are a natural fit for our surface partitioning model from Fig. 3.2(a). The model for our surface partitioning example, which contains only pairwise potentials, is also known as the *generalized Potts model* [93].

Inference in such networks has been studied extensively. For binary-valued Potts models, Greig *et al.* [50] show that the MAP problem can be formulated as a min-cut in an appropriately constructed graph. Thus, the MAP problem can be solved exactly for this class of models in polynomial time. For $K > 2$, the MAP problem is NP-hard but a method based on a relaxed linear program guarantees a factor 2 approximation of the optimal solution [17, 63]. Below we describe two different inference methods for MAP inference in AMNs, which are used further in this thesis.

LP Inference

The MAP problem in AMNs can be expressed as an integer linear program [63]. We associate binary variables μ_i^k with each node X_i and label k. The case when node X_i has value k is expressed by setting $\mu_i^k = 1$, and $\forall_{k' \neq k} \;\; \mu_i^{k'} = 0$. We also associate binary variables μ_c^k with each clique c and label k, which represent the case when all variables in

the clique are assigned that label. The MAP objective (expressed in terms of log-likelihood) can be represented by the following *integer program* (IP):

$$\begin{aligned} \max \quad & \sum_{x_i} \sum_{k=1}^{K} \mu_i^k \log \psi_i(x_i = k) + \sum_{c \in \mathcal{C}} \sum_{k=1}^{K} \mu_c^k \log \psi_c(x_c = k) \\ \text{s.t.} \quad & \mu_c^k \in \{0, 1\}, \quad \forall c \in \mathcal{C},\ k; \quad \sum_{k=1}^{K} \mu_i^k = 1, \quad \forall x_i \in X; \\ & \mu_c^k \leq \mu_i^k, \quad \forall c \in \mathcal{C},\ x_i \in X,\ k. \end{aligned} \tag{3.14}$$

Note that, in the definition above, the natural constraint $\mu_c^k = \bigwedge_{i \in c} \mu_i^k$ is replaced with the linear inequality constraints $\mu_c^k \leq \mu_i^k$. This works because we have only attractive potentials in the network cliques, for which $\log \psi_c(x_c = k) \geq 0$. Therefore, at optimum we have $\mu_c^k = \min_i \mu_i^k$, but since the μ_i^k variables are binary and discrete, this is equivalent to the statement $\mu_c^k = \bigwedge_{i \in c} \mu_i^k$.

We can obtain a linear relaxation of the integer program from Eqn. (3.14), by replacing the integer constraints $\mu_c^k \in \{0, 1\}$ with the linear constraints $\mu_c^k \geq 0$. The resulting *linear program* (LP) can be solved using any standard LP package, such as CPLEX [92]. The LP solution can be used to obtain solutions for the original IP. It can be shown that in the binary-valued case, the LP is guaranteed to produce an integer solution.

Theorem 3.3.2 *If $K = 2$, for any associative Markov network $\mathcal{F}$, the LP relaxation of Eqn. (3.14) is guaranteed to produce an integer solution. Therefore, optimizing the LP produces the optimal solution for the MAP inference problem in that network.*

See [114] for the proof. This result states that the MAP problem in binary AMNs is tractable, regardless of network topology or clique size. In the non-binary case ($K > 2$), the linear program can produce fractional solutions and we use a rounding procedure to get an integral solution.

Theorem 3.3.3 *If $K > 2$, and ℓ is the log-likelihood obtained by solving the LP relaxation of Eq. (3.14), there exists a rounding procedure for the LP result that produces integer solutions with log-likelihood of at least $\ell/2$ for pairwise AMNs, and ℓ/T for AMNs whose*

largest cliques are of size T. Therefore, rounding the LP solution produces a solution within a factor of $1/T$ of the optimal MAP objective.

See [114] for the proof and a description of the rounding procedure. Although artificial examples with fractional solutions can be easily constructed by using symmetry, it seems that in real data such symmetries are often broken. In fact, in all our experiments with $K > 2$ on real data, we have not encountered fractional solutions.

Min-cut Inference

Instead of using an LP solver, the objective in Eqn. (3.14) can be optimized using efficient min-cut algorithms. It has been long known that for pairwise AMNs with $K = 2$, the MAP inference problem can be reduced to the problem of finding the minimum-cut in an appropriately constructed graph [50]. Thus, the MAP problem can be solved exactly for this class of models using efficient minimum-cut algorithms in polynomial time.

In pairwise AMNs with $K > 2$, inference can be performed using the local search algorithm called α*-expansion* by Boykov *et al.* [16]. Here we give a brief overview of the approach; we refer the reader to the work of Boykov *et al.* [16] for a detailed discussion. For the purposes of this thesis, it is sufficient to limit the discussion to pairwise AMNs, although the α-expansion method can be generalized to AMNs with arbitrary-sized cliques [114].

The α-expansion algorithm performs a series of expansion moves in order to optimize the MAP objective. In particular, consider an existing labeling μ of the variables and a particular label $k \in \{1, \ldots, K\}$. A k-expansion from the current labeling μ is allowed to reassign some of the those labels to k. The k-expansion move is essentially an optimization of the MAP-objective over two labels — it either allows a variable to retain its current label, or to switch to label k. This optimization is similar to performing MAP inference in a two-class AMN. Similarly, it can be also reduced to the problem of finding a minimum-cut in a graph [16]. The α-expansion algorithm cycles through all labels k in either a fixed or random order, performing expansion moves which find new labelings of higher log-likelihood. It terminates when there is no expansion move for any label k that produces a labeling with higher log-likelihood.

Theorem 3.3.4 *The α-expansion algorithm converges in $\mathcal{O}(N)$ iterations, where N is the*

number of AMN variables.

The proof of this theorem can be found in Veksler's thesis [118]. As noted by Boykov *et al.* [16], and as we observed in our experiments, the algorithm terminates after a few iterations with most of the improvement occurring in the first $2-4$ iterations.

The α-expansion algorithm converges to a local minimum of the MAP objective. The quality of this local minimum is described in the following theorem [118]:

Theorem 3.3.5 *For MAP inference in pairwise AMNs with $K > 2$, the α-expansion algorithm converges to a factor of $1/2$ of the optimal MAP objective.*

This guarantee is identical to the one that can be obtained for the results of the linear programming relaxation method, which we described earlier in this section.

In our experiments, we found the α-expansion algorithm to be very efficient. For our problem,the α-expansion method performed up to 15 times faster than direct optimization of the linear program using the CPLEX solver, and is therefore our preferred method of MAP inference in associative Markov networks.

3.4 Parameter Learning

In many problems, we are given a set of observations, and want to learn about the structure of the domain that generated these observations. More formally, we are interested in inducing the underlying distribution P^* over properties, events and our evidence in the domain. Probabilistic graphical models are a natural and compact way of representing the distribution P^*. Hence, the task of learning reduces to one of estimating the structure and parameters of a probabilistic graphical model from data.

During learning, we face a fundamental problem: rather than having access to P^*, or equivalently to an infinite number of samples generated by it, we are given a finite *training set* of samples $\mathcal{D} = \{x[1], \ldots, x[M]\}$, that are independently drawn from P^*. Using the limited knowledge available to us via $\mathcal{D}$, our goal is to somehow learn a model that best approximates P^*. This may require us to take into account particular phenomena that arise in $\mathcal{D}$ and are solely due to its finite nature. In particular, one should be very careful about

the problem of *overfitting*: we can learn a model that fits the training data perfectly and yet has poor generalization performance on new samples from the distribution.

The problems which we address in this thesis are a subset of the complete learning task, because the structure of the probabilistic model is known beforehand. Therefore, here we need to discuss only methods for *parameter learning* of models with known structure. Moreover, it will be sufficient to limit our discussion to the learning of Bayesian networks with discrete variables. First we describe a *maximum likelihood* approach for learning the conditional probability parameters θ, which assumes that a *complete* training set is available (all variables in all examples were observed). Then we describe the *Expectation-Maximization* algorithm, which can be used for parameter estimation in cases with missing data and hidden variables, and discuss the complications that arise in that scenario.

3.4.1 Maximum Likelihood

The *maximum likelihood estimation* (MLE) approach is widely used in all fields of learning. At its core is the intuition that a good model is one that fits the data $\mathcal{D}$ well. In other words, we prefer models that are likely to have generated the data.

Definition 3.4.1 *The* **likelihood function**, $L(\theta : \mathcal{D})$, *is the probability of the independently sampled instances of* $\mathcal{D}$ *given the parameterization* θ

$$L(\theta : \mathcal{D}) = \prod_{m=1}^{M} P(x[m] \mid \theta) \tag{3.15}$$

where $P(x[m] \mid \theta)$ *is the probability of the* m*'th complete instance given the parameter of the network. The log of this function is known as the* **log-likelihood***:*

$$\ell(\theta : \mathcal{D}) = \sum_{m=1}^{M} \log P(x[m] \mid \theta) \tag{3.16}$$

In the MLE approach, we want to choose parameters $\hat{\theta}$ that maximize the likelihood of the data:

$$\hat{\theta} = \max_{\theta} L(\theta : \mathcal{D}) \tag{3.17}$$

Eqn. (3.17) describes optimization in a high dimensional space even for relatively simple network structures since we need to jointly optimize over the parameters of all the conditional probability distributions. As in the case of representation and inference, the Bayesian network representation offers a decomposition of this optimization task. We can use the decomposition property of Eqn. (3.4) to write

$$\begin{aligned} L(\theta : \mathcal{D}) &= \prod_{m=1}^{M} P(x[m] \mid \theta) && (3.18)\\ &= \prod_{m=1}^{M} \prod_{i=1}^{N} P(x_i[m] \mid \mathbf{u_i}[m], \theta_{X_i|\mathrm{Pa}_i}) && (3.19)\\ &= \prod_{i=1}^{N} \left[\prod_{m=1}^{M} P(x_i[m] \mid \mathbf{u_i}[m], \theta_{X_i|\mathrm{Pa}_i}) \right] && (3.20)\\ &= \prod_{i=1}^{N} L_i(\theta_{X_i|\mathrm{Pa}_i} : \mathcal{D}) && (3.21) \end{aligned}$$

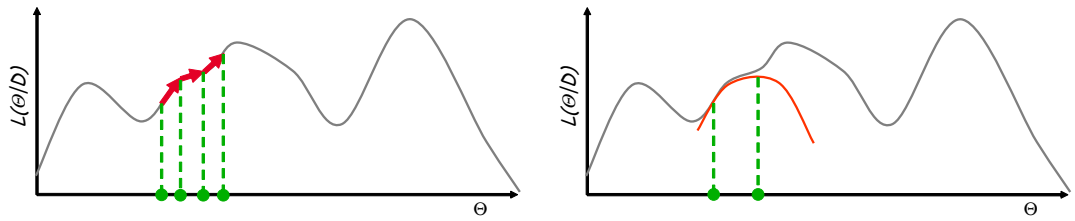
where $\theta_{X_i|\mathrm{Pa}_i}$ are the parameters that encode the conditional probability distribution of X_i given its parents Pa_i and

$$L_i(\theta_{X_i|\mathrm{Pa}_i} : \mathcal{D}) \equiv \prod_{m=1}^{M} P(x_i[m] \mid \mathbf{u_i}[m], \theta_{X_i|\mathrm{Pa}_i}) \tag{3.22}$$

is the *local likelihood function* for X_i. Thus, the global optimization problem is decomposed into significantly smaller problems, where we optimize the parameters of each conditional probability distribution $P(X_i \mid \mathrm{Pa}_i)$ independently of the rest. In the case of Bayesian networks with discrete variables, the parameters can be estimated in closed form:

$$\hat{\theta}_{x_i|\mathbf{u_i}} = \frac{S[x_i, \mathbf{u_i}]}{\sum_{x_i} S[x_i, \mathbf{u_i}]}, \tag{3.23}$$

where $S[x_i, \mathbf{u_i}]$ denotes the empirical counts of the instances $\{x_i[m] = x_i, \mathbf{u_i}[m] = \mathbf{u_i}\}$ in the dataset.



(a) (b)

Figure 3.6: Illustration of likelihood optimization using: (a) Gradient Ascent that proceeds in the direction of maximal change; (b) the Expectation Maximization (EM) algorithm that locally approximates the likelihood using a concave function and then optimizes this concave lower-bound.

3.4.2 Expectation-Maximization

In the case of missing data, each training sample $x[m] = \{o[m], \mathcal{H}[m]\}$ contains a set of observed variables $\mathcal{O}[m] = o[m]$, but also a set of variables $\mathcal{H}[m]$ whose values are not provided to us. In this section, we assume that the variables in $\mathcal{H}[m]$ are discrete. It is usually assumed that these values are *missing at random* – given the observations $o[m]$, there is no correlation between the fact that the variables $\mathcal{H}[m]$ are not observed, and their actual values.

Definition 3.4.2 *The log-likelihood function in the case of missing data is:*

$$\ell(\theta : \mathcal{D}) = \sum_{m=1}^{M} \log \sum_{h[m]} P(h[m], o[m] \mid \theta). \tag{3.24}$$

This is the objective we are interested in optimizing. Unfortunately, here there is no closed-form solution for the maximum-likelihood estimate of the parameters θ. This results from the fact that the different parameters become correlated in the presence of missing data. The correlations between the parameters preclude us from decomposing the general problem into local estimation problems. Consequently, the optimization needs to be performed in very high-dimensional space. Furthermore, the parameter space typically contains many local maxima. When some of the variables are hidden (not observed in any of the training instances), we also face the problem of multiplicity of both global and local

maxima that arises from possible permutations of the values of these variables.

The *Expectation Maximization* (EM) algorithm [39, 67] is a popular approach for learning with missing data. EM and its variants are typically used when the local distribution functions are in the *exponential family* [29] and sufficient statistics exist. The idea of EM in Bayesian networks with discrete hidden variables is straightforward: since parameter estimation is easy when the data is complete, we first complete the training set using the current model parameters, and then use this completed set to update the parameters themselves. The resulting model is then used to repeat the data completion step and the process is repeated until convergence.

Before we introduce the EM objective function, we define two helper quantities. We use $Q(\mathcal{H})$ to denote some posterior distribution over the missing variables:

$$Q(\mathcal{H}) = \prod_{m=1}^{M} Q(\mathcal{H}[m]) = \prod_{m=1}^{M} P(\mathcal{H} \mid \mathcal{O} = o, \theta), \tag{3.25}$$

where $\mathcal{H} = \{\mathcal{H}[1], \ldots, \mathcal{H}[M]\}$ is the set of missing variables, while $\mathcal{O} = \{\mathcal{O}[1], \ldots, \mathcal{O}[M]\}$ are the observed variables. We also define the log-likelihood of a completed data set $\{o, h\}$, in which all hidden variables are instantiated to some set of values $h = \{h[1], \ldots, h[M]\}$:

$$\ell(\theta : o, h) = \sum_{m=1}^{M} \log P(h[m], o[m] \mid \theta). \tag{3.26}$$

Given these definitions, it is easy to write the EM objective, which is the expected log-likelihood:

$$E_{Q(\mathcal{H})}\left[\log \ell(\theta : o, h)\right] = \sum_{m=1}^{M} \sum_{h[m]} Q(h[m]) \log P(h[m], o[m] \mid \theta). \tag{3.27}$$

In optimizing this objective, the EM algorithm begins with some initial parameter assignment θ^0, which can either be chosen randomly or using some other approach. Then it alternates between the following two steps:

- In the *Expectation step* (E-step), we use the current parameter estimates θ^t to compute the posterior distributions $Q^{t+1}(h[m]) = P(h[m] \mid o[m], \theta^t)$.

- In the *Maximization step* (M-step), we find the parameters θ^{t+1} which maximize the expected log-likelihood $\sum_{m=1}^{M} \sum_{h[m]} Q^{t+1}(h[m]) \log P(h[m], o[m] \mid \theta)$. Because in the M-step we are using a completed dataset, this optimization reduces to computing a maximum-likelihood parameter estimate, which can usually be done in closed form. The details of the optimization depend on the specific definition of the conditional probability distributions in the model.

It is fairly straightforward to show that both the E-step and the M-step cannot decrease the expected log-likelihood objective from Eqn. (3.27). Furthermore, a fundamental link between this objective, and the log-likelihood objective in Eqn. (3.24) was proven by Dempster *et al.* [39]:

Theorem 3.4.3 (Dempster *et al.*) *During any consecutive iterations t and $t+1$ of the EM procedure we have*

$$\ell(\theta^{t+1} : \mathcal{D}) \geq \ell(\theta^{t} : \mathcal{D}).$$

Moreover,

$$\ell(\theta^{t+1} : \mathcal{D}) - \ell(\theta^{t} : \mathcal{D}) \geq E_{Q^{t+1}(\mathcal{H})}\left[\ell(\theta^{t+1} : o, h)\right] - E_{Q^{t}(\mathcal{H})}\left[\ell(\theta^{t} : o, h)\right].$$

This theorem shows that each iteration of EM improves the log-likelihood until convergence to a (typically) local maximum. Furthermore, we are guaranteed that an increase in expected log-likelihood results in at least as large an increase of the log-likelihood. It can also be shown (see [39]), that the expected log-likelihood is a concave function which lower-bounds the actual log-likelihood. For each setting of the model parameters θ^t, the EM algorithm maximizes this lower bound, as illustrated in Fig. 3.6.

In this thesis we will also mention a version of EM, called *hard-EM*, which is sometimes used for simplicity and speed. The algorithm makes hard assignments for the variables $\mathcal{H}$, unlike the soft posterior assignments $Q(\mathcal{H})$ done in standard EM. Hard-EM maximizes the following log-likelihood function over both the parameters θ and the data-set completions h of $\mathcal{H}$:

$$\ell(\theta : o, h) = \sum_{m=1}^{M} \log P(h[m], o[m] \mid \theta). \tag{3.28}$$

In the hard E-step, we compute the most likely assignments for the missing variables $\mathcal{H}$ using the current parameters:

$$h^{t+1} = \arg\max_{h} \ell(\theta^t : o, h). \tag{3.29}$$

Then, in the M-step we use the assignments h^{t+1}, to re-estimate the parameters θ as follows:

$$\theta^{t+1} = \arg\max_{\theta} \ell(\theta : o, h^{t+1}). \tag{3.30}$$

The well-known K-means clustering algorithm is an instance of hard-EM, while its soft EM counterpart is used to learn Gaussian mixture models [14].

Similar to standard gradient descent methods [14], the EM algorithm also suffers from the problem of local maxima and its performance depends on the initial starting conditions. A straightforward method often used to cope with this is simply to run EM from multiple starting points and choose the best of the local maxima solutions. While there are no guarantees as to the number of random restarts needed for an effective solution, this method, and EM in general, can be surprisingly effective and is being used in a large number of practical applications.

Chapter 4

Correlated Correspondence Algorithm

In this chapter, we address the fundamental problem of *non-rigid 3D registration*, which is the problem of finding the point-to-point correspondences between two deforming surfaces. Non-rigid registration is an essential capability for the task of example-based learning of deformable models. The examples are usually meshes (or point clouds) produced by 3D scanners, which capture the deformable shapes from which we are to derive a model. These meshes are often missing parts of the surface, usually have different topologies, and can capture a variety of shapes in different configurations. The registration task is crucial for obtaining a common parametrization of all the scan examples before information from all of them can be aggregated by the learning process. Because we are often interested in learning shape models for object classes with significant intra-class deformations, or models of objects with significant pose changes, it is desirable that the registration algorithm can handle those cases successfully with minimal human intervention.

The registration problem requires search in the space of possible alignments between two surfaces. This problem is easiest when we deal with rigid surfaces, because the space of alignments has only six degrees of freedom. Another relatively easy case occurs when the deformation between the two surfaces is rather small — then we can assume that a point in one surface can only match to a few nearby points in the other. In the presence of large deformations, such an assumption is no longer warranted, and the set of potential matches for a point in one surface includes all points in the other. Local surface appearance can be used to prune these large sets of potential point-to-point matches. However, usually it is

not distinctive enough, and not persistent enough in the presence of deformation to allow significant pruning. Therefore, in the general case, determining the correspondence for all object points results in a combinatorially large search problem.

The existing algorithms for deformable surface registration make this problem tractable by assuming significant prior knowledge about the objects being registered. Some rely on the presence of markers on the object [2, 111], while others assume prior knowledge about the object dynamics [72], or about the space of nonrigid deformations [69, 15]. Algorithms that make neither restriction [105, 52] simplify the problem by de-correlating the choice of correspondences for the different points in the scan. However, this approximation is only good in the case when the object deformation is small; otherwise, it results in poor local maxima as nearby points in one scan are allowed to map to far-away points in the other.

In this chapter, we present a novel algorithm called *Correlated Correspondence*, which can be used to register significantly deforming surfaces in an unsupervised manner. The algorithm can produce reasonable results even in the absence of prior knowledge about object deformations and initial surface alignment. Its only limitation is its assumption that the surfaces being registered do not undergo significant topology changes. The algorithm is based on a probabilistic model over the set of possible point-to-point correspondences, which prefers registrations that match similar-looking surface areas, minimize surface deformation and preserve distances along the surface. The search in the combinatorial space of correspondence assignments is done using probabilistic inference [128], which produces a registration which (approximately) maximizes the score of the probabilistic model.

The rest of this chapter is organized as follows. First, we formally define the problem of non-rigid registration. Then we describe an important class of non-rigid registration algorithms, called *Non-rigid Iterative Closest Point (Non-rigid ICP)* algorithms[1]. These algorithms (e.g. [105, 28, 2, 52, 111]) simplify the registration problem by assuming that the choice of correspondences for different scan points can be de-correlated. We examine the cases when this assumption leads to poor registrations, and derive insights which will be used to obtain the *Correlated Correspondence (CC)* algorithm. Then we describe in detail the CC algorithm, and demonstrate successful registration in several datasets containing

[1]The moniker 'Iterative Closest Point' is borrowed from the famous algorithm for registration of rigid bodies by Besl *et al.* [13].

different objects and exhibiting different kinds of deformation.

Finally, we demonstrate two applications for our unsupervised registration capability. In our first application, we show how a partial scan of an object can be registered onto a fully specified model in a different configuration. The resulting registration allows us to use the model to "complete" the partial scan in a way that preserves local surface geometry. In our second application, we produce believable animation sequences by interpolating between two poses of an object. Both of these applications can be done in an unsupervised way, using the results of the Correlated Correspondence algorithm.

4.1 Traditional Non-rigid Surface Registration

4.1.1 Problem Definition

The registration problem is one of determining point-to-point correspondences between two surfaces. We assume we are given a complete model of the surface of the object, which we will call the *model mesh* and will denote as $\mathcal{M}^X$. We are also given a model of the surface in a different configuration. This surface is usually a mesh acquired with a 3D range scanner. We refer to it as a *scan mesh* and denote it $\mathcal{M}^Z$. The scan mesh can be a complete, or partial model of the surface (scanners generally fail to acquire the entire surface due to occlusion and reflectance problems).

The goal of registration is to match the corresponding parts of the two meshes, and to bring those parts together with minimal deformation. The registration problem can be defined formally in terms of a generative process, which is displayed in Fig. 4.1. The model mesh $\mathcal{M}^X$ is first deformed using a non-rigid transformation Θ. This transformation places the points and polygons of the mesh in a new configuration producing the transformed mesh $\mathcal{M}^Y = \Theta(\mathcal{M}^X)$. The points $\mathcal{V}^Y$ of the transformed mesh are then resampled to generate the scan mesh points $\mathcal{V}^Z$. The resampling process is guided by a set of correspondence variables C. Each point z_k in the scan mesh $\mathcal{M}^Z$ is associated with a correspondence variable c_k. This variable c_k has a discrete domain containing all model point indexes $\{1, \ldots, N_X\}$. Setting $c_k = i$ picks point y_i to generate scan point z_k.

Our generative model of registration is fully specified when we define the probability

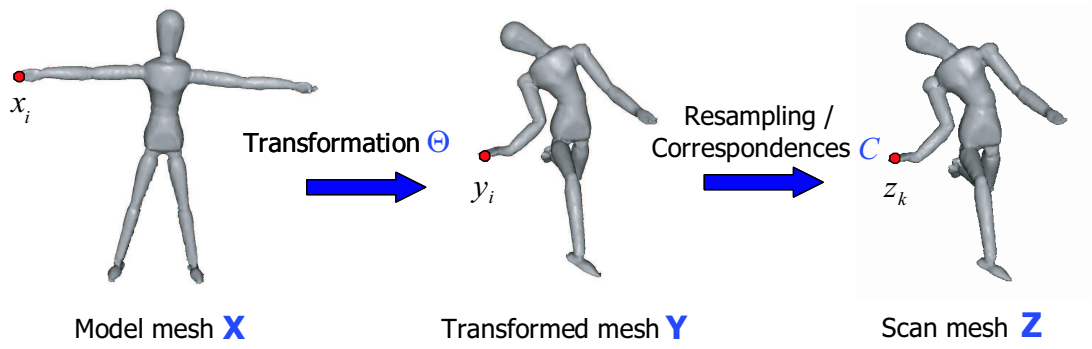


Figure 4.1: The generative process of registration. First, a non-rigid transformation is applied to the model mesh $\mathcal{M}^X$, producing the transformed version $\mathcal{M}^Y$, which contains the same points and polygons, but placed in new locations. Then, the transformed surface is resampled, producing the scan mesh $\mathcal{M}^Z$. The resampling is guided by the set of correspondence variables C — each point z_k in mesh Z is associated with a correspondence variable c_k that specifies which point in $\mathcal{M}^Y$ generated it.

distributions involving the deformation Θ and correspondences C. First, we need to assign a penalty for deformation, in the form of the conditional distribution $P(\Theta \mid \mathcal{M}^X)$. This distribution will assign high likelihood to cases with small deformation, and low likelihood otherwise. Many different choices are possible [105, 52, 2], but we omit giving a precise definition of this probability score until the next section. Second, we need to associate a probability distribution with the resampling step. Most existing approaches assume each scan point is generated from its corresponding point in the transformed model with Gaussian noise. In particular, we define the conditional probability:

$$P(z_k \mid c_k = i, y_i) = \mathcal{N}(z_k; y_i, \Sigma_C) \tag{4.1}$$

where Σ_C is a diagonal 3×3 matrix specifying the Gaussian variance. Finally, we assume uniform prior distribution $P(C)$ over the correspondence variables, reflecting the lack of prior information about the correspondences.

Given the probabilistic model described above, the registration problem can be cast as likelihood maximization — finding the most likely set of values for the Θ and C given the

original meshes $\mathcal{M}^X$ and $\mathcal{M}^Z$:

$$\arg\max_{\Theta,C} P(\Theta, C \mid \mathcal{M}^X, \mathcal{M}^Z) = \arg\max_{\Theta,C} P(\mathcal{M}^Z \mid C, \Theta) P(\Theta \mid \mathcal{M}^X) P(C). \quad (4.2)$$

The equality above follows from Bayes' rule. Now we are ready to summarize our discussion in the following definition:

Definition 4.1.1 *The* **non-rigid registration** *of model mesh $\mathcal{M}^X$ and scan mesh $\mathcal{M}^Z$ recovers correspondences C between the meshes and the non-rigid transformation Θ that align the meshes and minimizes the amount of the necessary deformation of model mesh $\mathcal{M}^X$.*

4.1.2 Non-rigid Iterative Closest Point Algorithm

Defining the generative model allows us to score different instantiations of the deformation Θ and correspondences C. However, the space of possible deformations is infinite, while the set of possible assignments to the correspondence variables is exponential. A standard way to search this space is the *Non-rigid Iterative Closest Point (non-rigid ICP)* algorithm [52, 111, 105], which is an adaptation of the Iterative Closest Point method of Besl *et al.* [13] used for aligning rigid objects.

This algorithm starts with a reasonable initial estimate of the deformation Θ, and tries to maximize the objective $\log P(C, \Theta \mid \mathcal{M}^X, \mathcal{M}^Z)$. Unfortunately, this is difficult to do in closed form, because of the complex correlations between the variables Θ and C. Instead, all non-rigid ICP algorithms are based on the insight that it is much easier to optimize the log-likelihood by iteratively optimizing either C or Θ, while keeping the other set of variables fixed.

The algorithm can be viewed as an instance of hard Expectation-maximization (for a refresher please refer to Sec. 3.4.2). It aims to maximize the log-likelihood $\log P(C, \Theta \mid \mathcal{M}^X, \mathcal{M}^Z)$. This objective is optimized by iterating between two steps. The hard E-step solves for the most likely assignment for the correspondences C, assuming Θ is fixed; in other words, for $\arg\max_C P(C \mid \Theta, \mathcal{M}^X, \mathcal{M}^Z)$. Given the transformation Θ, the transformed mesh $\mathcal{M}^Y$ is uniquely determined. With this in mind, it is easy to infer from the

Algorithm **Non-rigid Iterative Closest Point**

Input: Meshes $\mathcal{M}^X$ and $\mathcal{M}^Z$, initial alignment Θ^*.

1: **while** $\mathcal{M}^Y = \Theta(\mathcal{M}^X)$ and $\mathcal{M}^Z$ not sufficiently close **do**
2: **Hard E-Step:** Given Θ, compute the set of correspondences C:
For each scan point z_k, find the nearest point y_i in $\mathcal{M}^Y$, and set $c_k = i$.
3: **M-step:** Given C, compute a new transformation estimate Θ:
Solve for $\arg\max_\Theta \log P(\Theta \mid C, \mathcal{M}^X, \mathcal{M}^Z)$.
4: **end while**
5: **return** deformed mesh $\mathcal{M}^Y = \Theta(\mathcal{M}^X)$ and correspondences C.

Figure 4.2: A description of the Non-rigid ICP algorithm.

probabilistic model that the correspondence variable assignments are conditionally independent of each other given a known deformation Θ and the scan mesh $\mathcal{M}^Z$. Thus, the E-step objective can be optimized independently for each correspondence variable. In the M-step, the correspondences computed in the E-step are used to update the deformation $\Theta = \arg\max_\Theta P(\Theta \mid C, \mathcal{M}^X, \mathcal{M}^Z)$, and the process is iterated until convergence. The Non-rigid ICP algorithm is displayed in Fig. 4.2. The algorithms is often used in conjunction with a simulated-annealing style strategy which starts with a strong penalty on deformation and gradually decreases it in subsequent iterations. Such a strategy is more likely to produce a good solution.

A soft version of EM has also been explored by the work of Chui and Rangarajan [28], which maintains distributions over the variables C (instead of taking the most likely assignment) at an increased computational expense. Both the soft and the hard EM versions converge to a point where changing correspondences or deformation alone cannot improve the joint likelihood.

A specific instance of the Non-rigid ICP algorithm is described in the paper of Hähnel *et al.* [52].

4.1.3 Local Maxima of Non-rigid ICP

Non-rigid ICP is only guaranteed to get to a local maximum of the energy in Eqn. (4.2). As displayed in Fig. 4.15, an attempt to register two different puppet poses yields a monstrous

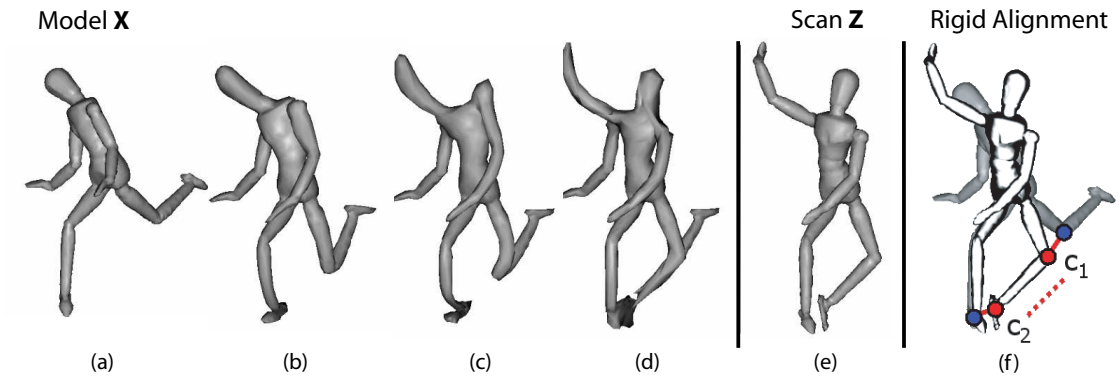


Figure 4.3: Non-rigid ICP fails to deal with large deformations. Model mesh (a) is registered with Scan mesh (e). Meshes (b)-(d) are intermediate Non-rigid ICP results, each subsequent result was produced by weakening the prior on link deformations. The algorithm is initialized with the best rigid alignment between the model and scan meshes, shown in (f). De-correlating the correspondence assignments in the E-step causes points on the same leg in the model mesh to be mapped to two different legs in the scan mesh.

result — the right puppet arm turns into a head, and a new head grows from the left shoulder. Such a poor local maximum is due in large part to an inability to obtain a good initial transformation estimate. The best rigid alignment between the two meshes is displayed on the right of Fig. 4.15. Using it in conjunction with the Non-rigid ICP algorithm causes the poor result.

Non-rigid ICP uses the transformation estimate to decorrelate the correspondence variables — each scan point is assigned its nearest transformed model point. The de-correlation assumption makes non-rigid ICP computationally tractable even for large model meshes. However, this assumption is clearly incorrect when the transformation estimate is poor. The rightmost picture in Fig. 4.15 demonstrates that two points which lie on the same leg in the scan mesh get associated with two *different* legs in the model mesh. This problem is never corrected in subsequent iteration of non-rigid ICP, as poor transformation and correspondence estimates reinforce each other.

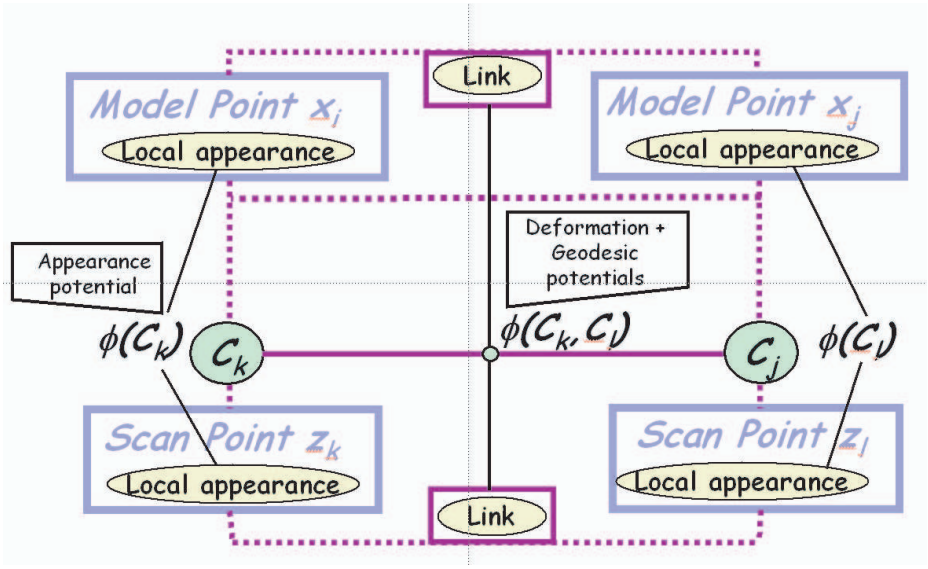


Figure 4.4: Illustration of the Correlated Correspondence model. Setting the values $c_k = i$ and $c_l = j$ picks a model mesh link that matches the scan mesh link (z_k, z_l). Single potentials $\phi(c_k)$ compare the local surface appearance of the possible point matches. Pairwise potentials $\phi(c_k, c_l)$ quantify the deformation of suitable matching links, and ensure they satisfy geodesic distance constraints.

4.2 Correlated Correspondence Algorithm

The main insight from Sec. 4.1.3 is that in the absence of a good alignment hypothesis, the correspondence variables associated with the scan mesh points are *correlated*. An example of this correlation is the requirement that nearby points on the scan mesh should be mapped to nearby points on the model mesh. Determining the correspondence for all object points, while taking into account their correlations, results in a combinatorially large search problem. The idea behind our algorithm is to explicitly model the correlations between the correspondence variables, and to search for a consistent solution directly in the resulting combinatorial space.

We view non-rigid registration as the task of finding a deformable embedding of the scan mesh $\mathcal{M}^Z$ into the model mesh $\mathcal{M}^X$. Unlike most other embedding methods [40, 116, 10], our algorithm can be used to register a partial scan to a complete model. Usually the meshes obtained by 3D scanners have holes caused by occlusion and self-occlusion, which makes this property desirable in practice. The embedding is defined by providing a

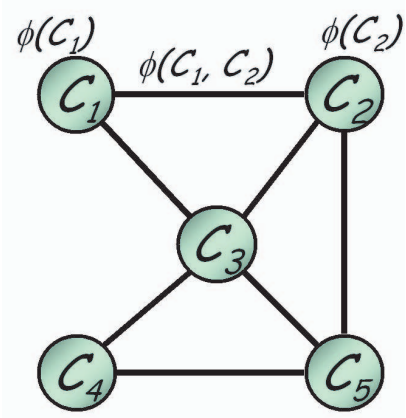


Figure 4.5: The induced Markov network encoding the correlations between the correspondence variables.

complete assignment to all correspondence variables $C = (c_1, \ldots, c_K)$. If the dependencies between the correspondence assignment are modeled correctly, we can make sure that the embedding avoids the problems exhibited by Non-rigid ICP in Fig. 4.3.

In order to find a consistent embedding, we need to define a model that captures the correspondence variable correlations. For this task, we use a pairwise Markov network (see Chapter 3). The network contains single potentials $\psi(c_k)$ which prefer embeddings that match similar-looking areas in the two surfaces. The network also contains probabilistic potentials $\psi(c_k, c_l)$ associated with pairs of correspondence variables (c_k, c_l). These potentials model the variable correlations that enforce a preference for embeddings that minimize surface deformation, and conform to geodesic distance constraints. The resulting Markov network is a model of a joint probability distribution of the form $p(C) = \frac{1}{Z} \prod_k \psi(c_k) \prod_{k,l} \psi(c_k, c_l)$ which contains only single and pairwise potentials. A sketch of the Correlated Correspondence model is displayed in Fig. 4.4, while the induced pairwise Markov network is displayed in Fig. 4.5.

The task of registration is thus reduced to one of performing probabilistic inference in the Markov network, in order to find the most likely joint assignment to the entire set of correspondence variables C. The inference problem for a general Markov network is NP-hard, but extensive literature on approximate Markov net inference is available. We apply the algorithm called *loopy belief propagation (LBP)* [128], which is an efficient search in

the exponential space of correlated variable assignments (see Chapter 3.3.2). In contrast, the Non-rigid ICP algorithm requires that an initial alignment hypothesis for the entire surface is given to the algorithm. Generally, there are exponentially many such hypotheses, and the non-rigid ICP algorithm lacks a mechanism to perform efficient search in that space.

4.3 Probabilistic Model

In this section, we describe in detail our probabilistic model of registration, which takes the form of a Markov network over the correspondence variables. This network contains the following kinds of pairwise and single potentials:

1. Single *local surface signature potentials* $\psi_s(c_k, c_l)$, which prefer to match similar-looking parts of the surface.

2. Pairwise *deformation potentials* $\psi_d(c_k, c_l)$, which encode a preference for small deformation during the embedding.

3. Pairwise *geodesic distance potentials* $\psi_n(c_k, c_l)$ and $\psi_f(c_k, c_l)$, which enforce approximate preservation of geodesic distance during the embedding.

Below we describe how to define and compute each of these potentials in detail.

4.3.1 Local Surface Signatures

We encode a set of potentials that correspond to the preservation of local surface properties between the model mesh and scan mesh. The use of local surface signatures is important, because it helps guide the optimization in the exponential space of assignments. We mainly experimented with spin-image features [60], although other features can be used as well. A spin-image is an oriented histogram associated with a point p on the surface; an example is displayed in Fig. 4.6. The point normal n defines the plane T_p, tangent to the surface at p. Each point q in the neighborhood of p is associated with two statistics: the signed distance β between q and the plane T_p and the distance α from p to q's projection in the plane T_p. The spin-image is a 2D histogram, which partitions the space of possible α and

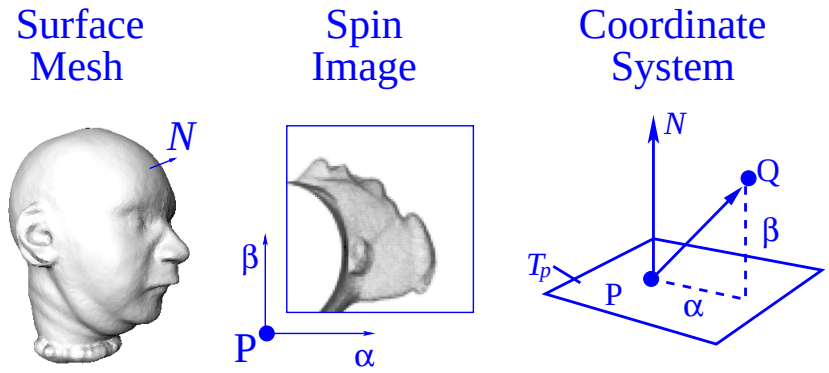


Figure 4.6: Spin images are two-dimensional histograms computed at an oriented point P on the surface mesh of an object.

β values into bins, and counts how many neighboring surface points fall in each bin. When the surfaces around scan and model points are similar, we expect their spin-images to be similar as well.

The spin-image signatures are invariant to surface rotations around the point normal n (since the statistics α and β are invariant to such rotations). As a result, spin-images offer an efficient way of comparing the surfaces around two points without requiring their rotational alignment, which is usually unknown. We compress the spin-images using principal component analysis (PCA) to produce a low-dimensional *signature* s_x of the local surface geometry around a point x. Two low-dimensional signatures s_{x_i} and s_{z_k} can be compared simply by using their L2 distance: $d_{i,k} = \|s_{x_i} - s_{z_k}\|$. Our surface similarity potentials are defined as a Gaussian distribution over these distances:

$$\psi_s(c_k = i) = \mathcal{N}(d_{i,k}; 0, \Sigma_S),$$

where σ_S is a diagonal covariance matrix.

In our experience, we found spin-images to be highly efficient features, which performed well for registration of articulated objects. In that case, the surface around the joints undergoes significant deformation, but the rest of the surface is usually deformed only slightly, comparable to the spin-image resolution. The spin-images were sufficient to obtain high-quality registrations in this case.

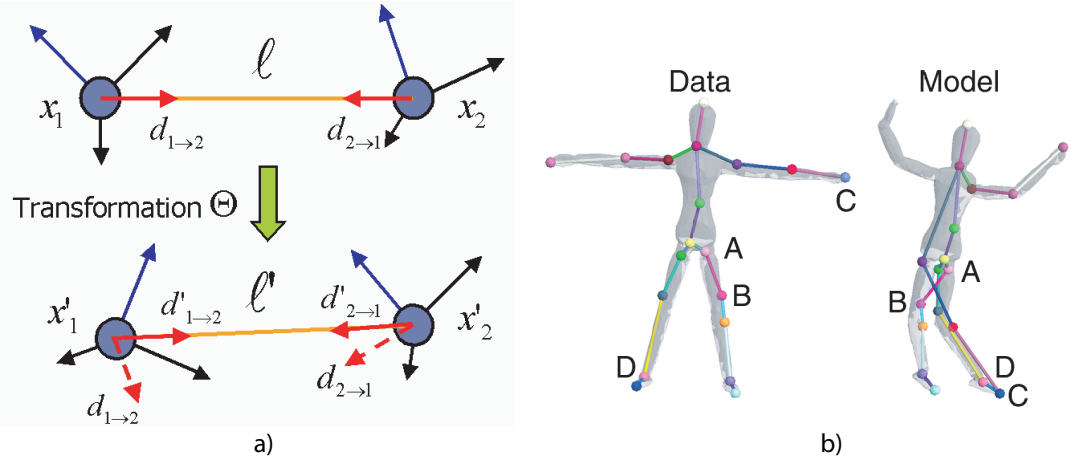


Figure 4.7: a) Illustration of the link deformation process b) The CC algorithm which uses only deformation potentials can violate mesh geometry. Near regions can map to far ones (segment AB) and far regions can map to near ones (points C,D).

There are limits as to how well features, such as spin-images, can perform in a completely unsupervised setting. For example, when we register scans of human bodies, we need to deal with large deformations in some parts of the shape, and small deformations in others. Whenever the resolution of the spin-image bins was small relative to the deformation, we got poor results. Increasing the spin-image resolution impaired the feature accuracy in areas which don't deform as much (human head and fists). This suggests that in the presence of additional knowledge, we can adapt the size and scale of the local surface signatures for different parts of the scan and achieve even better registration results.

4.3.2 Deformation Potentials

We want our model needs to encode a preference for embeddings of mesh $\mathcal{M}^Z$ into mesh $\mathcal{M}^X$ that minimize the amount of deformation Θ induced by the embedding. In order to quantify the deformation amount, we borrow ideas from the model of Hähnel *et al.* [52].

We model deformation using pairwise potentials $\psi_d(c_k, c_l)$ between the correspondence variables associated with adjacent scan mesh points (points connected by an edge in $\mathcal{M}^Z$).

We introduce a separate potential associate with each edge[2] in the scan mesh. The deformation potential is a table, assigning values for each possible *joint* assignment to its correspondence variables. Because of this, there is a distinct benefit of using pairs of variables (instead of triples or larger groups). A particular value $\psi_d(c_k = i, c_l = j)$ in the potential table denotes the preference given to the assignment $c_k = i, c_l = j$. Intuitively, the value corresponds to the amount of deformation that model edge (i, j) incurs to transform into scan edge (k, l).

Importantly, the set of possible matches for scan edge (k, l) is not limited only to the set of model mesh edges $\mathcal{E}^X$. That set of edges is sparse and local, and therefore insufficient to cover the space of deformations we want. Instead, we will allow any two points in the model mesh $\mathcal{M}^X$ to implicitly define a matching link for edge (k, l).

To quantify the amount of deformation, we treat the model mesh links as springs, which resist stretching and twisting at their endpoints. Consider a particular model link (i, j). Its stretching is easily defined by looking at changes in the link length $l_{i,j}$. Link twisting, however, is ill-specified by looking only at the Cartesian coordinates of the points alone. Similar to Hähnel *et al.* [52], we attach an imaginary *local coordinate system* to each point on the model (see Fig. 4.7(a)). This local coordinate system allows us to quantify the amount of link twisting: no twisting occurs if the orientation of the link endpoints in their neighbors' coordinate systems is preserved. This orientation will be captured by defining the unit vector $d_{i\to j}$, which describes the orientation of point x_j in the local coordinate system of point x_i (and similarly, we can define $d_{j\to i}$):

$$d_{i\to j} = \frac{u}{\|u\|}, \quad u = R_i \cdot (x_j - x_i). \tag{4.3}$$

Above, R_i is the matrix describing the rotation of the coordinate system centered on point x_i. For simplicity, we will assume that in the original model mesh the rotation is simply the identity matrix I.

The set of deformation-related parameters for a particular model link are denoted as $e^X_{i,j} = (l_{i,j}, d_{i\to j}, d_{j\to i})$, and are displayed in Fig. 4.7. After applying a non-rigid deformation Θ to the mesh, the local coordinate systems associated with the mesh points are

[2] We will also refer to mesh edges as *links*, to emphasize the variable correlations they entail.

rotated, and the edge parameters are transformed into $e^Z_{i,j} = (\tilde{l}_{i,j}, \tilde{d}_{i\to j}, \tilde{d}_{j\to i})$. Our model penalizes stretching and twisting independently:

$$P(e^Z_{i,j} \mid e^X_{i,j}) = P(\tilde{l}_{i,j} \mid l_{i,j})\ P(\tilde{d}_{i\to j} \mid d_{i\to j})\ P(\tilde{d}_{j\to i} \mid d_{j\to i}). \tag{4.4}$$

Furthermore, we assume a zero-mean Gaussian noise model for each parameter:

$$P(\tilde{l}_{i,j} \mid l_{i,j}) = \mathcal{N}(\tilde{l}_{i,j}; l_{i,j}, \sigma^2_L), \qquad P(\tilde{d}_{i\to j} \mid d_{i\to j}) = \mathcal{N}(\tilde{d}_{i\to j}; d_{i\to j}, \Sigma_D). \tag{4.5}$$

Prior information can be introduced into this model in the form of link-specific standard deviation parameters σ_L and covariances Σ_D. However, such information is usually not readily available, so in our experiments we assume all links share the same parameter values.

In order to quantify the deformation induced by the embedding C, we need to include a potential $\psi_d(c_k, c_l)$ for each link $e^Z_{k,l} \in \mathcal{E}^Z$. Every potential value $\psi_d(c_k = i, c_l = j)$ captures the amount of deformation needed to transform link $e^X_{i,j}$ into link $e^Z_{k,l}$. The precise value is defined as follows:

$$\psi_d(c_k = i, c_l = j) = P(e^Z_{k,l} \mid e^X_{i,j}). \tag{4.6}$$

Unfortunately, we cannot directly estimate the quantity $P(e^Z_{k,l} \mid e^X_{i,j})$, since the link parameters $e^Z_{k,l}$ depend on extra information about the local coordinate systems, which is not given as part of the input. The key issue is estimating the (unknown) coordinate system rotations. In effect, this rotation is an additional latent variable, which must also be part of the probabilistic model. To remain within the realm of discrete Markov networks, allowing the application of standard probabilistic inference algorithms, we discretize the space of the possible rotations, and fold it into the domains of the correspondence variables.

For each candidate match $c_k = i$, we need to select a small set of candidate rotations, that are consistent with local geometry. We do this by aligning the local surface patches around the points x_i and z_k. For each patch, we run PCA on its point cloud to obtain the direction of its largest eigenvector. For a particular candidate point match (x_i, z_k), we align the point normals and eigenvector directions to obtain two opposite candidate alignments.

We additionally refine the alignments using rigid ICP. The current implementation performs well using only two candidate rotation alignments per pair. A larger number of candidate alignments per pair can be easily computed, but we did not see the need for doing this in practice.

Using these alignment estimates, we extend the domain of each correspondence variable c_k. Each value in the domain encodes a matching point *and* a particular rotation from the precomputed set for that point. Given assignments to the correspondence variables in this extended domain, the quantities $P(e^Z_{k,l} \mid e^X_{i,j})$ and all the values of the deformation potentials can be computed.

We point out that the Correlated Correspondence algorithm can easily incorporate different deformation models. Most implementations of Non-rigid ICP use a carefully chosen definition of deformation, which can be easily linearized and results in a least-squares optimization objective. The probabilistic inference employed by the Correlated Correspondence algorithm does not rely on the continuity or differentiability of the deformation-scoring function. Different models of deformation can be introduced without need for changing the optimization algorithm, simply by replacing the values in the deformation potentials.

4.3.3 Geodesic Distances

Our proposed approach raises the question as to what constitutes the best constraint between neighboring correspondence variables. The literature on scan registration — for rigid and non-rigid models alike — relies on preserving the Euclidean distance. While the Euclidean distance is meaningful for rigid objects, it is very sensitive to deformations, especially those induced by moving parts. For example, in Fig. 4.7(b), we see that the two puppet legs are fairly close together, allowing the algorithm to map adjacent points in the scan mesh to the two separate legs, with minimal deformation penalty. In the complementary situation, especially when object symmetries are present, two distant yet similar points in one scan might get mapped to the same region in the other. For example, in the same figure, we see that points in both an arm and a leg in the scan mesh get mapped to a single leg in the model mesh.

We therefore want to introduce constraints that preserve the distance along the mesh surface (geodesic distance). Our probabilistic framework can treat such constraints as correlations between pairs of correspondence variables. We encode a *nearness preservation constraint* which prevents adjacent points in mesh $\mathcal{M}^Z$ to be mapped to geodesically distant points in $\mathcal{M}^X$. For *adjacent* points z_k, z_l in the scan mesh, we define the following potential:

$$\psi_n(c_k = i, c_l = j) = \begin{cases} 0 & \text{dist}_{\text{Geodesic}}(x_i, x_j) > \alpha\rho \\ 1 & \text{otherwise} \end{cases} \tag{4.7}$$

where ρ is the scan mesh resolution and α is a constant, chosen to be 3.5.

The *farness preservation* potentials encode the complementary constraint. For *every* pair of points z_k, z_l whose geodesic distance is more than 5ρ on the scan mesh, we have a potential:

$$\psi_f(c_k = i, c_l = j) = \begin{cases} 0 & \text{dist}_{\text{Geodesic}}(x_i, x_j) < \beta\rho \\ 1 & \text{otherwise} \end{cases} \tag{4.8}$$

where β is also a constant, chosen to be 2 in our implementation. The intuition behind this constraint is fairly clear: if z_k, z_l are far apart on the scan mesh, then their corresponding points must be far apart on the model mesh.

4.4 Optimization

In the previous section, we defined a Markov network, which encodes a joint probability distribution over the correspondence variables as a product of single and pairwise potentials. Our goal is to find a joint assignment to these variables that maximizes this probability. This problem is one of standard probabilistic inference over the Markov network. However, the Markov network is quite large, and contains a large number of loops, so that exact inference is computationally infeasible. We therefore apply *Sum-product loopy belief propagation (LBP)*(see Sec. 3.3.2), which is an approximate inference method that has been shown to work in a wide variety of applications. Running LBP until convergence results in a set of probabilistic assignments to the different correspondence variables, which are locally consistent. We then simply extract the most likely assignment for each variable to obtain a correspondence.

4.4.1 Dealing with Farness Preservation Potentials

One complication arises from the form of our farness preservation constraints. In general, most pairs of points in the mesh are not close, so that the total number of such potentials grows as $\mathcal{O}({N_Z}^2)$, where N_Z is the number of points in the scan mesh. This can easily be the bottleneck of the algorithm, because the number of all other potentials scales linearly with the number of scan mesh points. However, rather than introducing all these potentials into the Markov net from the start, we can introduce them as needed. First, we run LBP without any farness preservation potentials. We can easily check whether the solution violates a set of farness preservation constraints. This is done efficiently by checking if nearby points in our solution on the model mesh are indeed nearby on the scan mesh as well. If this is not true in all cases, we add the geodesic potentials associated with the violated constraints and rerun BP. In practice, this approach adds a small number of farness preservation constraints.

4.4.2 Dealing with Local Minima of Loopy Belief Propagation

In some cases LBP inference may converge to a local minimum of the energy defined by the Markov network — there may be another solution with a higher log-likelihood than the one found by the algorithm. In practice we observed that this can happen when the object shape contains symmetries. These cases most frequently include the presence of several identical object parts such as chair legs or car tires. Another popular case is that of planar (or mirror) symmetry, which is also frequently observed in many objects such as people, cars and chairs. To deal with the local minima problem, we need to run LBP with different starting conditions, which will cause the algorithm to explore different parts of the energy landscape. Below we are going to describe how to provide these different starting conditions in a completely unsupervised manner. In general, the algorithm below is not very efficient, hence we ran it only in the case when straightforward application of LBP got stuck in a local minimum.

Our CC algorithm provides the capability to embed the scan mesh, which can contain any subset of the surface, into the model mesh. Thus, it also provides the capability to embed any subset of the scan mesh into the model mesh, as well. We consider breaking up

Algorithm **MultipleStartingHypotheses**

Require: Part size R, K embeddings per part.
Output: A set of N starting hypotheses for LBP.

1: Find a set of scan mesh parts $P_Z = (p_1, \ldots, p_S)$.
2: Find the point z_C on the scan mesh surface, which is nearest to the mesh centroid.
3: Compute the geodesic distance map $\mathcal{D}_\mathcal{G}$ from z_C to all mesh points.
4: Get a set of extrema points $V = (v_1, \ldots, v_S)$ that correspond to local maxima of $\mathcal{D}_\mathcal{G}$ with non-trivial support.
5: Each part p_s then contains the subset of the surface within a radius R around extremum point v_s.
6: Use the CC algorithm to find K different embeddings of each part p_s into the model mesh, and their likelihood.
7: Find the set of N highest-likelihood hypotheses which embed all parts into the model in a non-overlapping way. If no such hypotheses exist return $\emptyset$.

Figure 4.8: Algorithm for providing multiple starting hypotheses for loopy belief propagation

the scan mesh into parts, and using CC to find several different embeddings for each part. In the extreme case when the part contains a single point, and the embedding algorithm needs to only consider its local surface signature, there is a lot of ambiguity and hence many possible matches in the model. On the other hand, we expect that object parts of a non-trivial size will have only a few good candidate matches. Our algorithm will thus rely on computing the embeddings of a few medium-sized scan mesh parts. From these, we can obtain several high-scoring hypotheses that embed all parts into the model in a non-overlapping way. Each such hypothesis can then be used to initialize a different run of LBP inference. At a high level of abstraction, this approach is similar to RANSAC-style algorithms [44], only we are looking to use entire subsets of the scan mesh as features. The algorithm is sketched in Fig. 4.8 in more detail.

Each embedding hypothesis H^i, obtained with the algorithm above, contains a candidate match point x_s^i in the model for each extremum point v_s^i. We initialize LBP for each hypothesis, by requiring that the extrema points are embedded in the vicinity of the matches found by the algorithm in Fig. 4.8 (this is accomplished by allowing each point v_s^i to match only points near x_s^i in the model mesh). We compare all LBP results obtained using the different initialization hypotheses, and pick the solution with the highest log-likelihood.

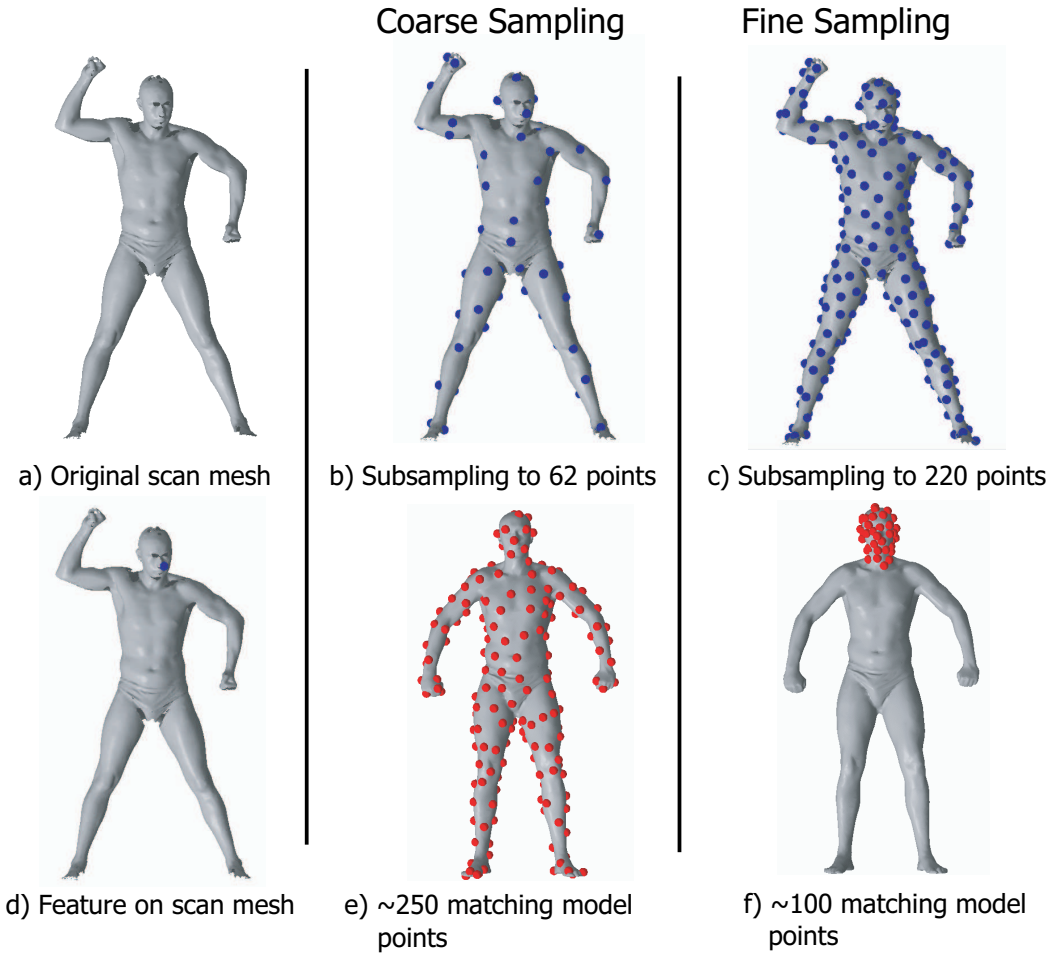


Figure 4.9: Illustration of the mesh subsampling process. The top row displays the subsampling of the (a) scan mesh points at a (b) coarse resolution and (c) fine resolution. The bottom row displays the decimation of the correspondence domain, associated with (d) a point on the subject's nose. (e) The coarse subsampling phase covers the model mesh with 250 possible matches for that point. (f) The fine subsampling phase uses the coarse solution to restrict the correspondence domain to about 100 points on the head.

4.5 Surface Subsampling

In the previous two sections, the problem of non-rigid registration was reduced to one of performing inference in a Markov network over the correspondence variables C. In practice, we are often faced with the task of registering very large meshes that contain tens of thousands of points and polygons. For such meshes, our algorithm can still be directly applied since we can build very large Markov networks, and perform LBP inference to compute the correspondence assignments. However, this direct scheme has the following drawbacks:

1. Large Markov networks occupy a substantial chunk of computer memory, and LBP in them takes a long time and many iterations to converge.

2. The CC algorithm computes an embedding of the scan mesh points into the model mesh points. For this embedding to make sense, we implicitly assumed that the point sampling of the model mesh surface is denser than that of the scan mesh surface. Otherwise, our solution can have many scan points mapped to the same model point.

3. In practice, we observed that as the Markov network size keeps increasing, at some point the solution quality tends to worsen. This is due to the fact that a growing number of variables in the network also causes a growing number of possible local minima where LBP inference can get stuck.

Our solution for this is to subsample the set of scan mesh points and the domains of their associated correspondence variables. In practice the benefits of such an approach outweigh the considerations that in the process we may be ignoring some of the original information about surface geometry. Many algorithms in computer vision (e.g. [64, 105]) have benefited from such a subsampling approach.

Our implementation employs a coarse-to-fine subsampling strategy. First, we subsample the scan mesh $\mathcal{M}^Z$ to about 60-100 points. Then we appropriately subsample the domains of the correspondence variables as well. The LBP inference is run on the resulting Markov network which embeds the subsampled point set into the model mesh. Our second iteration of the algorithm uses a finer-resolution subsampling resulting in about 200-250

scan mesh points. Their correspondence domains are then decimated in a way, which uses the coarse registration results.

Below we describe the subsampling process in more detail. We would like to point out that the approach we present is one of many reasonable choices, and is simply a pre-processing step to our registration algorithm.

4.5.1 Subsampling the Scan Mesh

Our goal is to take the mesh $\mathcal{M}^Z$ and obtain a sparse set of points covering its surface, along with a set of links connecting these points.

First, we describe a simple and efficient way of subsampling the set of mesh points $\mathcal{V}^Z$. We assign a *preference score* b_k to each scan mesh point z_k, which describes the desirability of that point. We would like to keep points where the surface is distinct; by a simple rule of thumb, these are points in areas of high mesh curvature. We also prefer points that are far from the mesh boundaries, where spin-image features and surface normals are not as accurate. We use a very simple method to obtain likely high-curvature points. The preference score for each scan point is defined as the ratio between the area of mesh triangles incident to the point and the length of the triangle edges that do not contain the point. While this simple rule of thumb was enough for our purposes, more sophisticated mesh curvature estimation methods [68, 3] can be used as well. We assign these preference scores to all points z_k that lie at least a distance ρ from the scan mesh boundary. The points that are close to the mesh boundary are assigned a uniform negative score, rendering them most undesirable. Given the scores, we use a simple algorithm *SubsampleMeshPoints*$(\mathcal{M}^Z, B, d_Z)$ (described in Fig. 4.10) that greedily covers the mesh with high-preference points, while ensuring they are spaced at least a distance d_Z apart. The results of applying this algorithm to the scan mesh of Fig. 4.9(a) at two different resolutions d_Z are displayed in Fig. 4.9(b,c).

Having determined the subset $\mathcal{S}$ of points to keep, we also need to determine the links between them. We keep only links between adjacent points in the surface. Connecting non-adjacent surface points can impair our capability to deal with articulated models, and

Algorithm **SubsampleMeshPoints** ($\mathcal{M}, B, d$)

Require: Mesh $\mathcal{M}$, preference scores $B = (b_1, \ldots, b_N)$, distance d.
Produce: A subset $\mathcal{S}$ of the mesh points, spaced at least d apart.

```
Construct a heap H = {(b_1, z_1), ..., (b_N, z_N)} of score-point tuples.
Allocate a bit array U of size N, set all bits to false.
while H not empty do
    Extract the highest-scoring pair (b_k, z_k) from H.
    if U[k] = false then
        S = S ∪ z_k.
        U[k] = true.
        for all points z_l within a distance d from z_k along the surface of M do
            Set U[l] = true.
        end for
    end if
end while
```

Figure 4.10: Simple algorithm for subsampling the mesh points

is more computationally expensive. We chose to keep the links consistent with the Delaunay triangulation over the points in $\mathcal{S}$. Delaunay triangulations result in few sliver triangles [37], which are undesirable because they contain short edges that can in some cases be flipped by the registration algorithm. In obtaining the triangulation, we use the distances along the surface of the scan mesh. We point out that the CC algorithm can also be used with a set of links that does not correspond to a valid triangulation of the mesh surface.

4.5.2 Subsampling the Domains of the Correspondence Variables

Since each correspondence variable c_k has $2 \times N_X$ values in its domain (corresponding to the N_X model mesh points, with two orientations for each), we get a substantial computational benefit from decimating these domains for large meshes. Subsampling the surface of mesh $\mathcal{M}^X$ using standard software such as *Qslim* [47] is not appropriate in this case. The reason for this is that standard mesh subsampling solutions naturally focus on obtaining good polygon tessellations of the surface, which can result in a very non-uniform point sampling. This is a problem for our registration algorithm, which essentially embeds one set of point samples into another. Consider planar areas in the original model mesh, which

will be tessellated with very few polygons(and hence, very few points as well) by the standard approaches. If the areas deform, that may cause the scan mesh to be much more densely sampled in the same regions. This causes a problem for our registration algorithm, which will try to embed a densely sampled surface region into a sparsely sampled one. We previously discussed the requirement that for good algorithm performance, the point sampling of the model surface is denser than that of the scan surface*in all surface areas*. Below we describe a solution that achieves this requirement.

We are going to decimate the values in the domain of each correspondence c_k separately, using the local surface appearance of its associated point. For each possible assignment $c_k = i$, we compute the its local signature score $b_i^k = \log \psi_s(c_k = i)$. We then execute the greedy algorithm *SubsampleMeshPoints*$(\mathcal{M}^X, B^k, \frac{1}{2}d_Z)$ from Sec. 4.5.1. The algorithm greedily prunes the domain of c_k, and as a result, we obtain the a subset of the mesh points which locally maximizes the signature scores, and is spaced at least $\frac{1}{2}d_Z$ distance apart. Recall that the scan mesh was subsampled using the resolution parameter d_Z using the same algorithm. Our decimated domains cover the entire model mesh, to ensure that we do not miss the correct solution as a result of this pre-processing step. In the presence of additional domain knowledge, the correspondence variable domains can be decimated further.

In practice, during the first subsampling phase of the algorithm that leaves 60 scan mesh points, we end up with correspondence domains containing about 250 points, with 2 alignments per point (Fig. 4.9(e)). In the fine subsampling phase, we end up with about 100 possible model point candidates, because we restrict the solution to be close to the correspondences found during the first phase (Fig. 4.9(f)).

4.6 Experimental Results

We applied our registration algorithm to three different datasets, containing meshes of a human arm (7 meshes), wooden puppet (7 meshes) and a CAESAR dataset of whole human bodies (10 meshes), all acquired by a 3D range scanner. The meshes were not complete surfaces, but several techniques exist for filling the holes (e.g., [35]).

We ran the Correlated Correspondence algorithm using the same probabilistic model

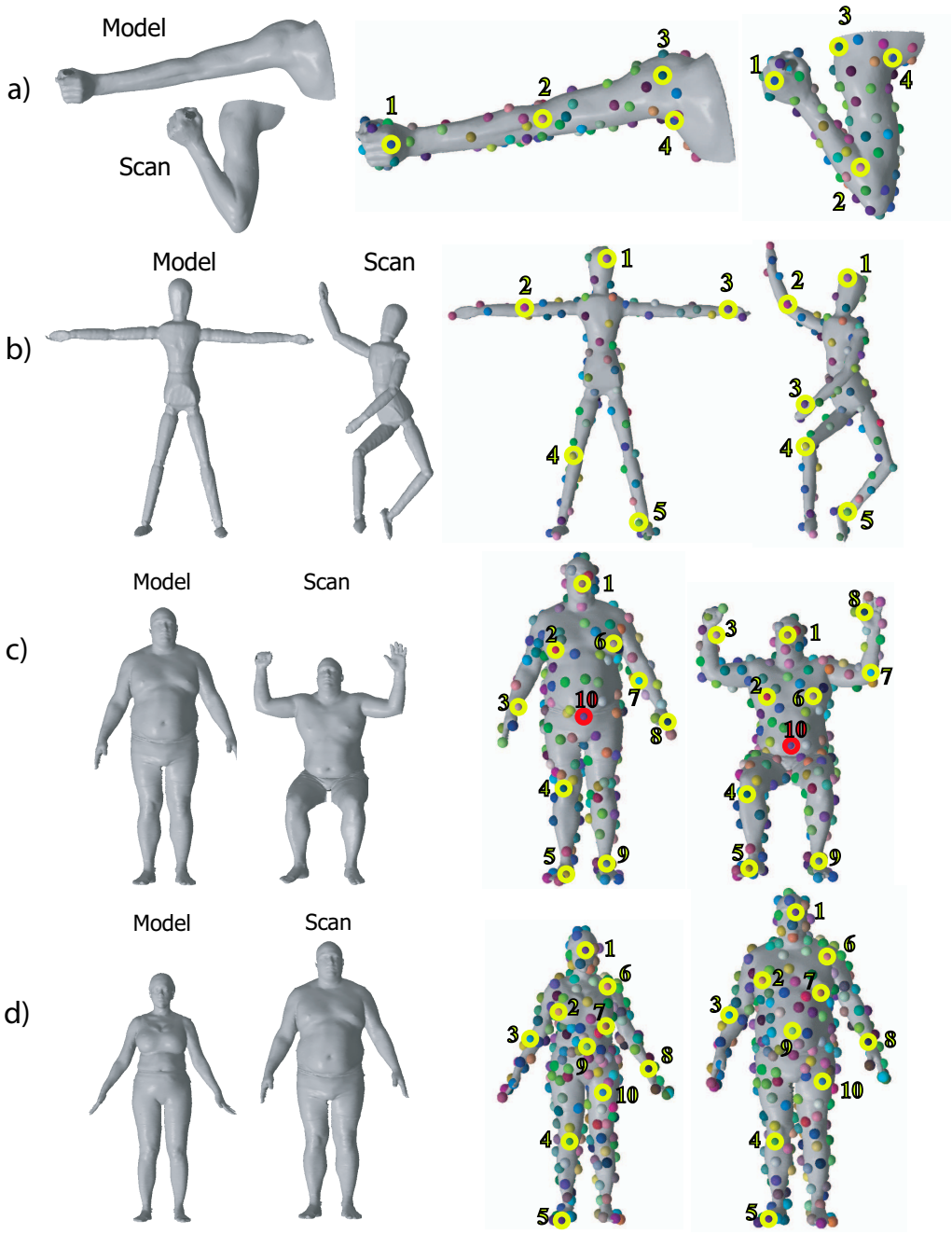


Figure 4.11: Results obtained by running the CC algorithm on pairs of scans in a completely unsupervised fashion. The algorithm finds 200-300 matching point pairs, which are displayed as colored spheres on the meshes — matching points are assigned the same color. Select correct and incorrect correspondences were numbered, and highlighted in yellow or red, respectively.

and the same parameters on all data sets. We use the coarse-to-fine subsampling strategy, outlined in Sec. 4.5. Our spin-image features contain 6×12 bins, the total spin-image size is $1.2 \times d_Z^1$, where d_Z^1 is the resolution of the scan mesh features at the coarse scale of subsampling. The spin-images are compressed using PCA, of which we keep the first 15 principal components. The standard deviations of our Gaussian distributions are set as follows: $\sigma_L = 0.7 \times d_Z, \Sigma_D = 0.7 \cdot I, \Sigma_S = (1.2 \times e_1) \cdot I$, where e_1 is the size of the largest eigenvalue in the PCA used to compress the spin-images, and I is the identity matrix. We run Loopy belief propagation, and as a result obtain the point-to-point correspondences between the two meshes. We did not use the multiple-initializations strategy from Sec. 4.4.2 unless explicitly stated.

The Correlated Correspondence algorithm successfully aligned all mesh pairs in our human arm data set containing 7 arms. The most difficult registration case, when the arm goes from a completely extended to a bent position, is displayed in Fig. 4.11(a). In the puppet data set we picked one of the meshes to be the template mesh, and registered it to the remaining 6 puppets. The algorithm correctly registered 4 out of 6 scan meshes to the model mesh. One of those correct registrations is displayed in Fig. 4.11(b). In the two remaining cases, the algorithm produced a registration where the torso was flipped, so that the front was mapped to the back. This problem arises from ambiguities induced by the puppet symmetry, whose front and back are almost identical; one of these problematic cases is displayed in Fig. 4.12. However, in both of these cases, our probabilistic model assigns a higher likelihood score to the correct solution. Thus, the incorrect registration is a consequence of local maxima in the LBP algorithm. Using the approach in Sec. 4.4.2, we re-ran loopy BP for all puppets several times, with different initializations. This modified algorithm correctly registered the template mesh to all the remaining ones in the dataset.

We also applied the CC algorithm to 6 pairs of human body scans from the CAESAR dataset. It performed well in challenging cases involving both articulated motion and deformations (Fig. 4.11(c)), as well as body shape deformation and (small) changes in scale (Fig. 4.11(d)). The algorithm made only one significant error, which is located around the belly-button in Fig. 4.11(c) and is highlighted in red. No multiple initializations were needed in these cases.

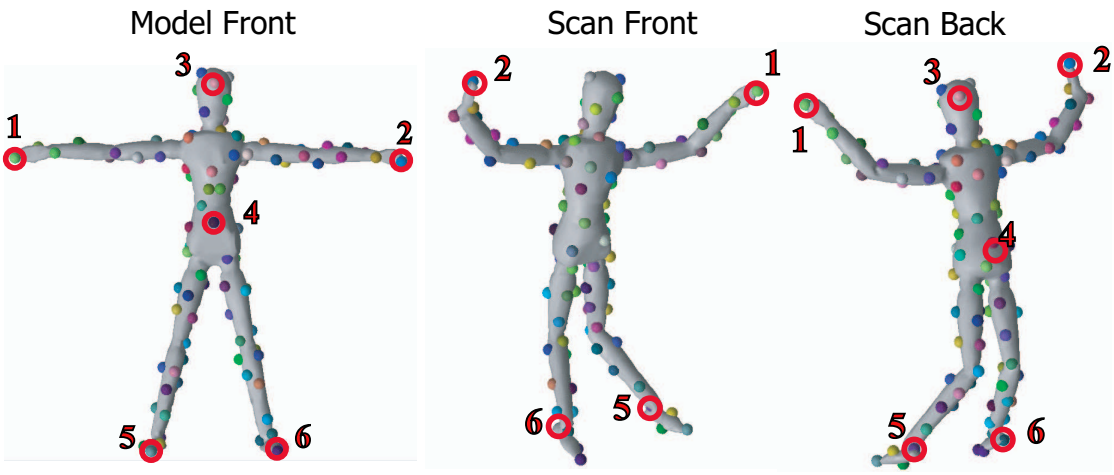


Figure 4.12: Illustration of a local minimum found by the algorithm, which is due to the symmetry of the puppet shape. The front of the puppet in the model mesh to the back of the puppet in the scan mesh.

Overall, the algorithm performed robustly in a variety of cases, producing close-to-optimal registrations even for pairs of meshes that involve large deformations, articulated motion or both. The registration is accomplished in an unsupervised way, without any prior knowledge about object shape, dynamics, or alignment. In the experiments above we demonstrate that the algorithm performance is fairly robust to the parameter settings, as we used the same settings for all the experiments.

Our unoptimized implementation of the CC algorithm ran on an Intel Xeon 2.4 GHz platform. The running times on the examples from Fig. 4.11 are displayed in the table in Fig. 4.13. The results show that it takes significantly longer to compute spin-images and the Markov network potentials (denoted as *Setup* in the table) than to run LBP inference. Also, the variable domains during the *Fine Subsampling* phase are smaller, since they are computed using the result from the coarse phase. Hence, that phase of the algorithm runs faster despite having to deal with many more correspondence variables. We believe that an optimized implementation on a parallel hardware architecture can run in close to real-time.

	Arm	Puppet	Caesar1	Caesar2
Spin Images	97s	49s	42s	48s
Coarse Subsampling				
Setup	145s	99s	118s	121s
LBP	45s	75s	55s	43s
Fine Subsampling				
Setup	82s	53s	70s	108s
LBP	22s	6s	8s	14s

Figure 4.13: Running times of the CC algorithm.

4.7 Applications

4.7.1 Obtaining Morphs

The set of correspondences obtained by the CC algorithm can be used to morph the model mesh onto the scan mesh. We used the corresponding point pairs as markers in the non-rigid ICP algorithm of Hähnel *et al.* [52]. In one example, we ran the CC algorithm to register a pair of puppet scans for which direct application of non-rigid ICP failed (Fig. 4.3). The correspondences obtained with the CC algorithm were sufficient to obtain a good morph (displayed in Fig. 4.15), although with a small defect on the right shoulder (inset). We also computed morphs for a set of arms, and display some of them in Fig. 4.14.

4.7.2 Partial View Completion

The Correlated Correspondence algorithm allows us to register a partial scan of an object to a known complete surface model of the object. We can then use non-rigid ICP to morph the template mesh onto the partial scan. The result is a mesh that matches the scan surface, while completing the unknown portion of the surface using the template geometry.

We take a partial mesh, which is missing the entire back part of the puppet in a particular pose. The resulting partial model is displayed in Fig. 4.16(a); for comparison, the correct complete model in this configuration (which was not available to the algorithm), is shown in Fig. 4.16(b). We register the partial mesh to a model of the object in a different pose (Fig. 4.16(c), and compare the completions we obtain (Fig. 4.16(d), to the ground truth represented in Fig. 4.16(b). The result demonstrates a largely correct reconstruction of the

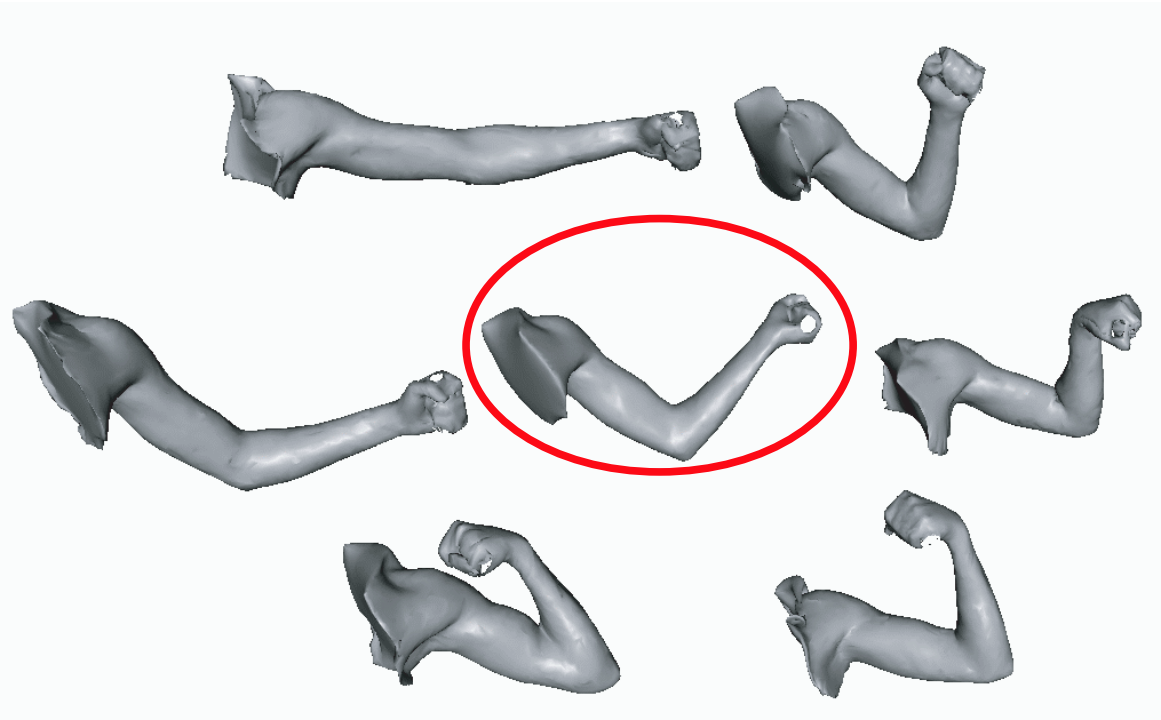


Figure 4.14: Arm morphs obtained automatically by registering the model mesh (center) to a set of meshes representing different arm poses.

complete surface geometry from the partial scan and the deformed template.

We also registered the partial mesh to a different object model (Fig. 4.16(e) in a more extreme pose. This registration results in a different completion (Fig. 4.16(f), which retains some of the model shape in the right shoulder area. The example demonstrates that the choice of model can affect the completion quality, particularly in the places that are occluded in the partial view. The example also demonstrates the limitations of an approach that relies on a single shape template for its completions.

We also used the CC algorithm to hole-fill Cyberware human body scans. Unlike the previous CC results, which were obtained in a completely unsupervised manner, here we used 4 additional markers to resolve the problem of body symmetries. We hole-filled 70 scans of the same person in a variety of poses. Several shape-completions and the model template used to obtain them are displayed in Fig. 4.17. The figure demonstrates that the shape-completion process produces reasonable results for a very rich set of poses using a

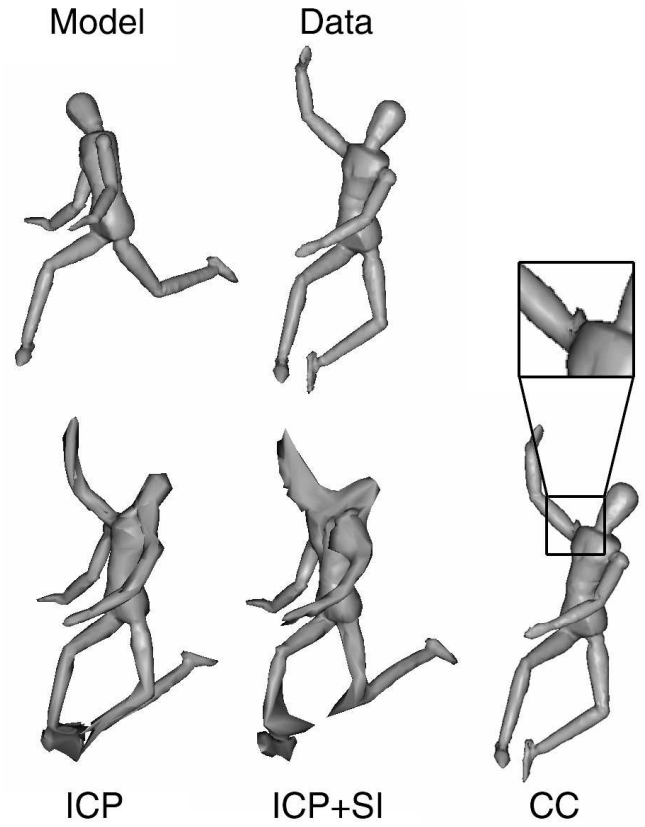


Figure 4.15: Registration results for two meshes. Nonrigid ICP and its variant augmented with spin images get stuck in local maxima. Our CC algorithm produces a largely correct registration, although with an artifact in the right shoulder (inset).

single template model. Cyberware scans are acquired from four different points of view simultaneously and therefore contain only fairly small holes. The extensive partial view surface data minimized the effect of the original template shape prior and constrained it to fit the local surface in practically all cases.

4.7.3 Animation

Our second application generates smooth and believable animations by interpolating between a pair of registered meshes. When the meshes undergo significant deformations, simple linear interpolation of the mesh point locations produces incorrect results (Fig. 4.18).

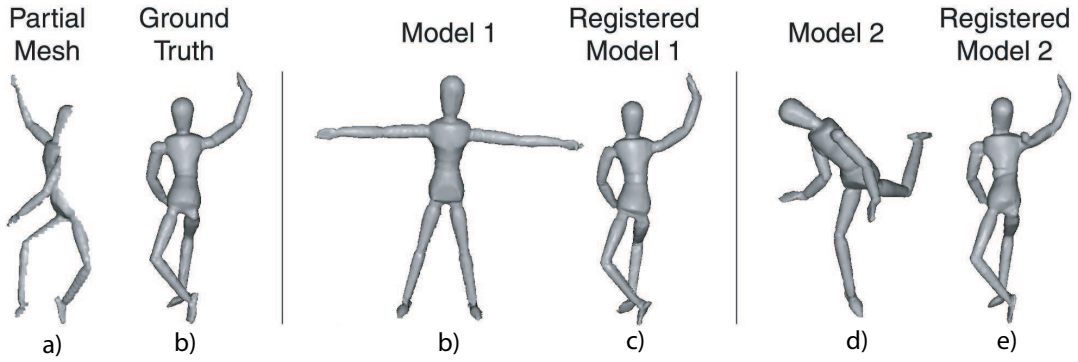


Figure 4.16: Partial view completion results. The missing parts of the surface were estimated by registering the partial view to a complete model of the object in a different configuration.

Traditional animation techniques circumvent the problem by utilizing additional knowledge, usually in the form of an articulated skeleton underlying the surface [1, 123, 80]. Some of those approaches can also end up with significant interpolation artifacts [70]. In this section, we will present an approach that can handle relatively large deformations and yet is purely data-driven, using no external information except the meshes themselves.

We wish to interpolate between two registered meshes: the source mesh $\mathcal{M}^X$ and the target mesh $\mathcal{M}^Y$, which have the same mesh topology. The key idea in our solution is to use an alternative mesh parametrization. As described in Sec. 4.3, meshes with the same topology as $\mathcal{M}^X$ can be described using a set of local coordinate system rotations $(t_1, \ldots, t_{N_X})$, and edge length and twisting parameters $(l_{i,j}, d_{i\to j}, d_{j\to i})$. The above parameters are sufficient to recover the point locations of the original mesh, except for a translational degree of freedom. This degree of freedom can be removed by specifying the location of a single point on the mesh.

A linear interpolation of the above parameters produces believable animations for many articulated objects. The main problem with point location interpolation in Euclidean space is that link lengths get foreshortened in the process. Preserving the link lengths is at the core of the current representation, which tends to produce better animations. Let $\alpha \in [0, 1]$ be the linear interpolation coefficient, where $\alpha = 0$ corresponds to $\mathcal{M}^X$, and $\alpha = 1$ to the target mesh $\mathcal{M}^Y$. The interpolated parameter values are a function of α and are listed

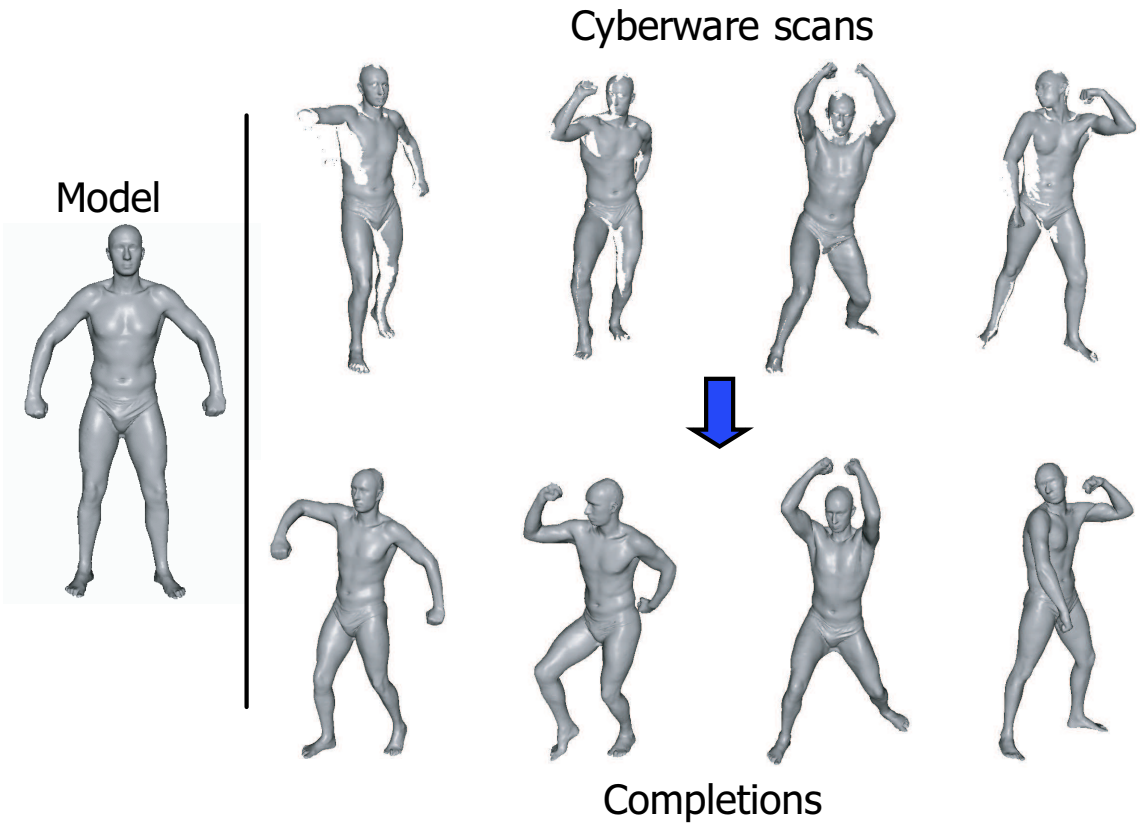


Figure 4.17: Hole-filling of human body scans. The missing parts of the surface were estimated by registering the scans to a complete model of the object in a different configuration.

in Fig. 4.19.

Having described how to interpolate the relevant parameters, we will now discuss how to use them in reconstructing a mesh $\mathcal{M}^\alpha$. First, the values of the interpolated parameters for some $\alpha \in (0, 1)$ do not have to define a consistent mesh. The mesh point locations impose consistency constraints on the parameter values, but our interpolation method treats each parameter separately and essentially ignores these constraints. These constraints thus have to be enforced during the mesh reconstruction step. We do this by solving for the point locations that satisfy the parameter settings in the best least-squares sense. If $\mathcal{V}^\alpha =$

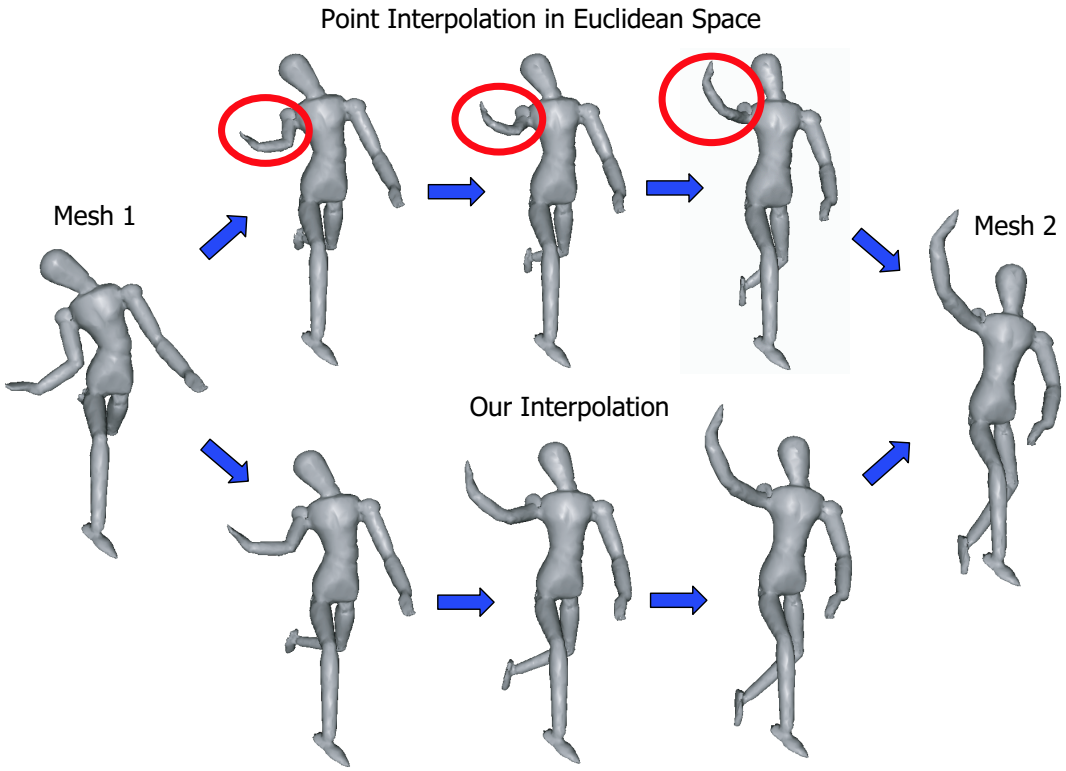


Figure 4.18: Illustration that point interpolation in Euclidean space produces undesirable limb foreshortening effects (top). On the other hand, our interpolation technique produces reasonable results even for strongly deforming scans (bottom).

$\{v_1, \ldots, v_{N_X}\}$ denote these point locations, our objective becomes:

$$\arg\min_{\mathcal{V}^\alpha} \sum_{(i,j)\in\mathcal{E}^X} (v_j - v_{i\to j})^\top (v_j - v_{i\to j}) + (v_i - v_{j\to i})^\top (v_i - v_{j\to i}) \tag{4.9}$$

$$v_{i\to j} = v_i + l^\alpha_{i,j}\, R(t^\alpha_i)\, d^\alpha_{i\to j} \tag{4.10}$$

$$\tag{4.11}$$

where $R(t^\alpha_i)$ is the rotation matrix induced by the twist t^α_i. The objective in (4.9) forces the point locations of the reconstructed mesh to be placed in accordance with the local edge parameter predictions, defined in (4.10). The above objective is similar to the logarithm

Interpolation parameter	Source mesh value	Target mesh value
$t_i^\alpha = \alpha t_i$	$\mathbf{0}$	t_i
$l_{i,j}^\alpha = (1-\alpha) l_{i,j} + \alpha \tilde{l}_{i,j}$	$l_{i,j}$	$\tilde{l}_{i,j}$
$d_{i\to j}^\alpha = \frac{u}{\|u\|}; u = (1-\alpha) d_{i\to j} + \alpha \tilde{d}_{i\to j}$	$d_{i\to j}$	$\tilde{d}_{i\to j}$
$d_{i\to j}^\alpha = \frac{u}{\|u\|}; u = (1-\alpha) d_{i\to j} + \alpha \tilde{d}_{i\to j}$	$d_{i\to j}$	$\tilde{d}_{i\to j}$

Figure 4.19: Interpolated quantities for animation between two scans.

of the non-rigid deformation estimate in (4.4), and can be solved by any suitable least-squares solver. As discussed, we can remove the translational degrees of freedom by setting $v_1 = (0,0,0)^\top$.

Some of our results are displayed in Fig. 4.20. There we demonstrate reasonable animations between two poses of a puppet, an arm and entire human bodies, which undergo significant deformations. All the animations were produced automatically from pairs of meshes, registered with the Correlated Correspondence algorithm. No knowledge of the articulated object structure was used in producing these animations. Our algorithm is applicable whenever the point coordinate system rotations are less that 180 degrees. Our interpolation tries to find the shortest rotation path, and whenever 180 degrees of rotation are exceeded the correct path is no longer the shortest. In this case, although the source and the target meshes are still faithfully reconstructed, the interpolated meshes may cause some object parts to fold upon themselves.

Finally, we note that while our choice of mesh parametrization is one of the simplest options resulting in good animations, several other choices are possible. However, we believe that the information contained in the local coordinate system alignments is essential for obtaining good interpolations and any reasonable method for animation must maintain it.

4.8 Related Work

Surface registration is a fundamental building block in computer graphics. The classical solution for registering rigid surfaces is the Iterative Closest Point algorithm (ICP) [13, 24, 98]. It is based on the insight that while solving for the correspondences and for the

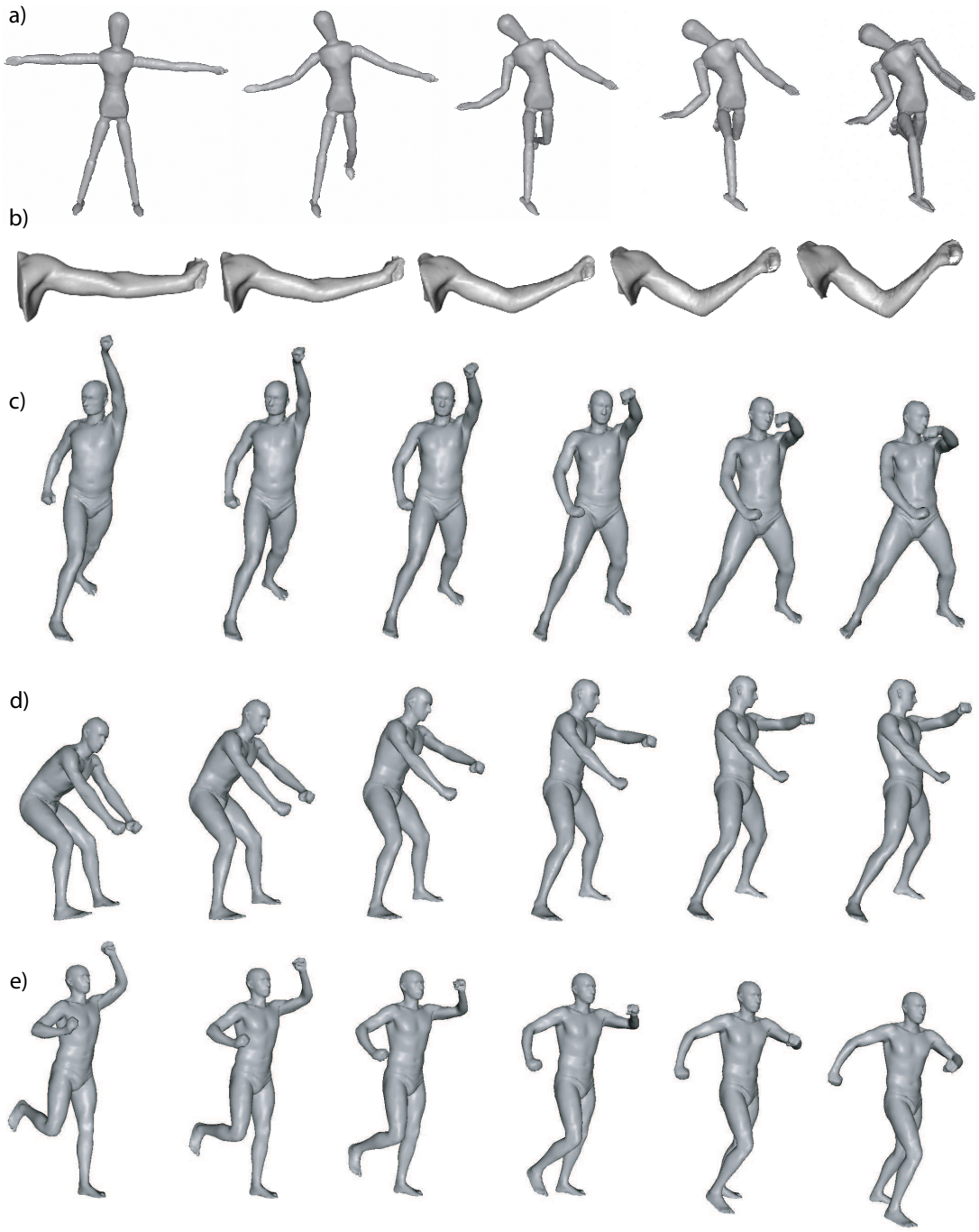


Figure 4.20: Animation frames generated by interpolating pairs of scans, which were registered with the Correlated Correspondence algorithm.

transformation simultaneously is quite difficult, solving for each in turn (while holding the other fixed) is much simpler. Recently, there have been many approaches that extend the ICP paradigm to non-rigid surface and image models [105, 28, 52, 2, 19].

These algorithms differ mainly in their definition of surface deformation. The most related approaches define deformation relative to a template shape, usually represented as a mesh. Allen *et al.*[2] look at the affine transformations that move the model mesh points, and introduce a rather weak prior that requires that the transformations of adjacent points are similar. Sumner *et al.* [111] enforce a similar smoothness on the deformation of adjacent mesh polygons, but strengthen the prior by requiring that the original polygon shape and orientation are also preserved. Shelton [105] treats the model mesh links similar to Hähnel *et al.*, but defines all link displacements in the same global coordinate system. Chui and Rangarajan [28] use a deformation model called a *thin-plate spline* (originally developed by Wahba [121]). This model is defined for a point set (hence does not exploit the topology information contained in a mesh) and defines a measure based on the Euclidean distances between all point pairs. While all the above algorithms perform well for a large set of template deformations, they are not designed to deal with articulated body parts such as arms. Arms can be placed in a variety of different configurations relative to the global coordinate system, but most of the local appearance remains unchanged (and ideally will not be penalized by the deformation model). To capture this, one needs to penalize displacements relative to the local coordinate frames on the surface, which is done by Hähnel *et al.*[52]. None of the above algorithms model local coordinate systems, however, and penalize all parts of the surface for deviating from the original template. This idea of local coordinate systems was only recently explored in the computer graphics community by the work of Lipman *et al.*[73] on rotation-invariant mesh editing.

Another set of approaches assumes that a set of previously registered meshes of the same object is available in order to define a measure of surface deformation. They apply principal component analysis (PCA) either to a set of registered meshes [15, 2] or to aligned volumetric representations such as active level sets [69]. Registration for this class of approaches is also done by iterating between finding the correspondences and finding the rigid alignment and the principal components describing the shape deformation. Unfortunately, the types of deformations that can be encoded through linear PCA interpolation

over points in Euclidean space is quite restricted — these approaches often work well for largely convex objects, but have problems with articulated object parts. All these models are applicable only when a set of registered surfaces is available.

All approaches mentioned above can be viewed as instances of the Non-rigid ICP registration framework. As discussed in Sec. 4.1.3, all these algorithms are likely to end in poor local maxima in the absence of a good initial alignment hypothesis. Several strategies for avoiding these undesirable local minima have been used to address this issue.

The local maximum problem is usually circumvented by assuming that select point-to-point correspondences, or markers, are provided to help the algorithm [1, 2]. These markers can be obtained by placing distinct textures on select areas of the scanned objects, or — even more frequently — by having a human pick corresponding points between the surfaces. In a preprocessing step, the model surface is deformed to fit the markers in order to obtain an acceptable initial transformation estimate. After this, the standard non-rigid ICP algorithm can be applied. A drawback of this solution is that a non-trivial amount of human effort is required for marker placement. For example, the work of Allen *et al.* [2], which registers scans of humans with different physiques, uses more than 70 markers for each pair of scans.

In the absence of markers, several other techniques have been found to alleviate the problem of incorrect initialization. [105] performs registration in a multi-resolution pyramid, and employs local features, such as color (other features such as curvatures, surface normals and spin-images can be helpful as well). Performing soft-EM, which maintains beliefs over the correspondence estimates [28] can lead the algorithm to a better local maximum at an increased computational expense. While generally helpful, these techniques generally cannot resolve the cases when significant object deformation is taking place.

Several algorithms which represent alternative registration paradigms are worth mentioning as well. Belongie *et al.* [11] register 2D shape templates by using shape context features and casting the registration problem as relatively easier bipartite matching problem (rather than the general NP-complete graph matching problem). Since the geometric relationships between the points are only captured via the features (and are ignored in the bipartite matching) this approach can cause poor registrations in challenging cases. Kimmel *et al.* [40] construct bending-invariant surface signatures, by embedding the surfaces

in a low-dimensional Euclidean space, where the original geodesic distances are preserved as much as possible. Two shapes can be compared (and coarsely registered) by aligning of the signatures in the Euclidean space. However, this framework cannot incorporate surface features which help accurate registration, and can only register complete surface models. Finally, we should mention discriminative algorithms for human pose detection in computer vision [109, 81, 104]. These algorithms rely on hundreds of supervised examples to learn a mapping from object appearance to the angles of a known articulated skeleton. Given a new instance of the same object, this mapping function can provide a reasonable estimate of the skeleton pose. Unlike our algorithm, these discriminative approaches can only be applied after a set of training examples have been obtained beforehand.

Our algorithm is most closely related to combinatorial algorithms for deformable template matching in computer vision. The idea that many objects can be represented in terms of a set of parts arranged in deformable configurations has been around since the 1970's [45]. Since then, different kinds of spatial priors (that capture the relationships between the object parts) and associated strategies for object detection in images have been proposed.

A popular set of approaches define a joint Gaussian model over the object part locations in the image, which captures explicit dependencies between all pairs of parts [21, 22, 43]. Detection algorithms that use these models until very recently have relied on search heuristics, which limit the number of parts that computationally feasible models can contain (for example, the models of Fergus *et al.* [43] contain 6 parts).

Tree-structured graphical models have also been very popular. They have primarily been used for detection and localization of articulated objects such as human bodies [56, 58, 107], consisting of rigid parts connected by joints. Such algorithms assume that the articulated model is provided, and that its spatial dependency graph has no loops. Under these conditions, efficient *dynamic programming (DP)* methods can be applied to find the globally optimal (or nearly optimal) part placement. The main challenge for such algorithms lies in keeping the domains of the correspondence variables tractable. This can be done by defining a 2D template model, which has fewer degrees of freedom [56], or by assuming that a set of detectors has been predefined for all articulated parts [107], or both [58]. Tree-structured graphical models have also been used for localizing deformable

2D shapes in images [42]. The algorithm is limited to a special class of 2D shapes that consist of triangles whose adjacency graph is a tree. Unfortunately, general 3D meshes induce graphs that contain loops and do not allow exact optimization methods such as DP to be applied.

Dynamic programming techniques are subsumed by *belief propagation (BP)* — an optimization technique that provides good empirical solutions on a variety of graphical models with loops. Before our publication, Coughlan and Ferreira [31] employed loopy belief propagation for detection of 2D loopy curves in images. At the time of writing this thesis, loopy graphical models are becoming the state-of-the-art for detection of objects and classes of objects in images. These models are optimized either with loopy belief propagation [65, 66], or integer programming [12]. However, all of the approaches address the problem of 2D object detection in images, and cannot be easily extended to the problem of deformable 3D registration. In the field of computer graphics and 3D modeling, discrete graph optimization methods such as loopy belief propagation are not yet popular — we hope that our algorithm contributes toward the adoption of these methods.

4.9 Conclusion

The contribution of this chapter is an algorithm for unsupervised registration of non-rigid 3D surfaces in significantly different configurations. The algorithm was not provided with markers or other cues regarding correspondence, and makes no assumptions about object shape, dynamics, or alignment. Our results show that the algorithm can deal with articulated objects subject to large joint movements and with non-rigid surface deformations. We show the quality and the utility of our registration results by using them as a starting point for compelling computer graphics applications: partial view completions and animations obtained by interpolation between registered scans. Importantly, all these results were generated in a completely unsupervised manner from pairs of input meshes.

The main limitation of our approach is the fact that it makes the assumption of (approximate) preservation of geodesic distance. Although this assumption is desirable in many cases, it is not always warranted. In some cases, the mesh topology may change drastically, for example, when an arm touches the body. We can try to extend our approach to handle

these cases by trying to detect when they arise, and eliminating the associated constraints. However, even this solution is likely to fail on some cases. A second limitation of our approach is that it assumes that the scan mesh is a subset of the model mesh. If the scan mesh contains clutter, our algorithm will attempt to embed the clutter into the model. We feel that the general nonrigid registration problem becomes under-specified when significant clutter and occlusion are present simultaneously. In order to obtain reasonable solutions in such cases, we would need to make some assumptions about the object deformation space.

The most straightforward way of extending our algorithm is to design more sophisticated local surface signatures. Spin-images are features that are very efficient to use, and we demonstrated that they perform well in a variety of cases. However, spin-images are sensitive to choice of histogram resolution and invariant to mirror symmetries, which contributed to our problem of local minima for objects possessing such symmetries, such as humans. In these cases, more sophisticated features such as shape contexts [11] and in particular features based Earthmover's distance [91] are possible. Earthmover's distance (EMD) is a metric which is particularly useful for general shape comparison, and has been used for contour matching [49] and color histogram matching [97]. If used appropriately, it can address the two technical drawbacks of spin-images — difficulty of choosing the appropriate histogram resolution, and inability to directly model occlusion. The main challenge lies in making the feature computationally tractable. Recent work of Indyk *et al.* [57], which shows how to perform efficient lookup of similar EMD shapes using projections into L1-distance space and locality-sensitive hashing, may lead to an efficient EMD feature comparison strategy.

The ability to register pairs of scans without making object-specific assumptions provides an automatic way of dealing with novel object shapes. There are limitations as to what can be accurately learned about an object from just two registered scans. However, this capability is a necessary step for obtaining an entire collection of registered scans of the same object. The set of registered scans can be used as the foundation for learning models of object shape and dynamics. For example, we can learn the deformation penalty associated with the different model links, and bootstrap the algorithm to obtain even better registrations. Also, we can learn the correlations between the deformations of different object parts, which helps the tasks of animation and shape-completion. The possibilities and

implications of learning deformable object models will be explored further on in Chapter 6 of this thesis.

Chapter 5

Recovering Articulated Object Models

Articulated objects consist of approximately rigid parts, which are linked by joints to form an object skeleton; examples include the human body, most animals, office chairs, cars and many others. Articulated models have been used for popular tasks such as character animation [70, 1, 80] and object tracking in video [55, 18, 129, 107] and in 3D data streams [72, 25]. In the vast majority of applications, the articulated skeleton structure and its parameters need to be specified by hand. In this chapter, we describe an algorithm that can recover complex articulated models from 3D scans in a completely unsupervised manner. This capability eliminates the need for human effort during the model construction and provides insight into the structure of different objects.

Given registered 3D scans of an object in different configurations, our algorithm automatically recovers a decomposition of the object into approximately rigid parts, as well as the location of the parts in the different object instances. The joints linking adjacent object parts are then obtained using a post-processing step. An overview of the approach is given in Sec. 5.1. In Sec. 5.2, we show how we can segment the object surface using a graphical model that captures the spatial contiguity of parts. We describe an Expectation-Maximization algorithm, which iterates between finding a decomposition of the object into rigid parts, and finding the location of the parts in the object instances. In Sec. 5.3, we show how to use the resulting segmentation to estimate the locations of the joints connecting the parts. We test the algorithm on real world datasets, and show that we can successfully obtain complex models containing many parts, even in cases when the surfaces undergo

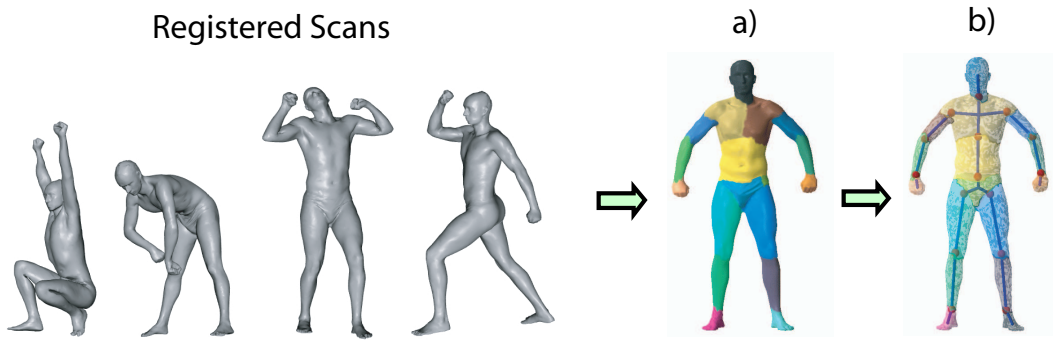


Figure 5.1: Overview of articulated model recovery. The algorithm is given a set of registered scan instances. (a) First, it segments the surface of the template mesh into approximately rigid parts and estimates their location for all instances. (b) Using the segmentation and the part location estimates, the joints between adjacent parts are estimated.

non-trivial deformation. Finally, we describe a simple method for tracking the acquired articulated models in 3D data streams, which is useful for capturing the natural kinematics of humans and animals.

5.1 Framework Overview

In this section, we will introduce our notation for dealing with articulated models, and define the task of articulated model recovery from 3D scans. Then we give a general outline of our approach.

5.1.1 Articulated Models

Articulated models consist of a set of rigid parts connected by joints and provide a convenient representation for a rich set of real-world object shapes. These involve many man-made objects such as vehicles, office chairs, laptops, and flip-top cell phones, which indeed consist of fairly rigid parts. Articulated models are also useful for modeling the shape of many natural objects, such as humans and other animals, for which the part deformations

are relatively small. The use of articulated models for such objects is twofold. In tracking applications we may want a representation that simply ignores subtle part deformation effects for simplicity and computational reasons. In animation applications, articulated models account for a large part of the deformation of an object. The more subtle deformations can be correlated to the parameters of the articulated skeleton, which provide a natural low-dimensional representation for the space of object configurations.

Below we provide precise definitions for concepts related to articulated models. We start with the fundamental notions of *rigid part* and *joint*, then define the terms *articulated model* and *articulated model pose*.

Definition 5.1.1 *A* **rigid part** $\mathcal{P}_n$ *is a subset of the mesh surface whose shape is preserved over time. If* $\mathcal{M}^X$ *is the original mesh defining the object shape,* $\mathcal{P}_n^X = (\mathcal{V}_n^X, \mathcal{E}_n^X)$ *contains a subset* $\mathcal{V}_n^X$ *of the original mesh points, as well as the edges* $\mathcal{E}_n^X \subseteq \mathcal{E}^X$ *connecting these points.*

Definition 5.1.2 *A* **joint** $g_{n,m} \in \mathbb{R}^3$ *is a point constraining the motion of two adjacent rigid parts* $\mathcal{P}_n$ *and* $\mathcal{P}_m$*. The existence of a joint* $g_{n,m}$ *imposes a constraint on the rigid transformations* T_n *and* T_m*, which can be applied to its incident parts:*

$$T_n(g_{n,m}) = T_m(g_{n,m}). \tag{5.1}$$

Intuitively, a joint $g_{n,m}$ *is a point that moves with both part* $\mathcal{P}_n$ *and part* $\mathcal{P}_m$ *simultaneously.*

Definition 5.1.3 *An* **articulated model** *(or* **skeleton***)* $\mathcal{S}^X = (\mathcal{M}^X, \mathcal{P}^X, G^X)$ *is an object shape representation, consisting of the following components:*

- *A mesh* $\mathcal{M}^X$ *defining the object shape.*
- *A set of rigid parts* $\mathcal{P}^X = (\mathcal{P}_1^X, \ldots, \mathcal{P}_{N_P}^X)$ *defined on the mesh* $\mathcal{M}^X$*.*
- *A set of joints* G^X *between pairs of adjacent parts.*

Since at any moment in our discussion we will have no more than a single articulated model, we will simplify the notation, and denote it simply as $\mathcal{S}^X = (\mathcal{M}^X, \mathcal{P}, G)$. We find it useful to define the mapping $\mathcal{B} = (b_1, \ldots, b_{N_X})$, which encodes the assignment of surface mesh points to rigid parts. Setting $b_j = p$ assigns mesh point x_j to rigid part p.

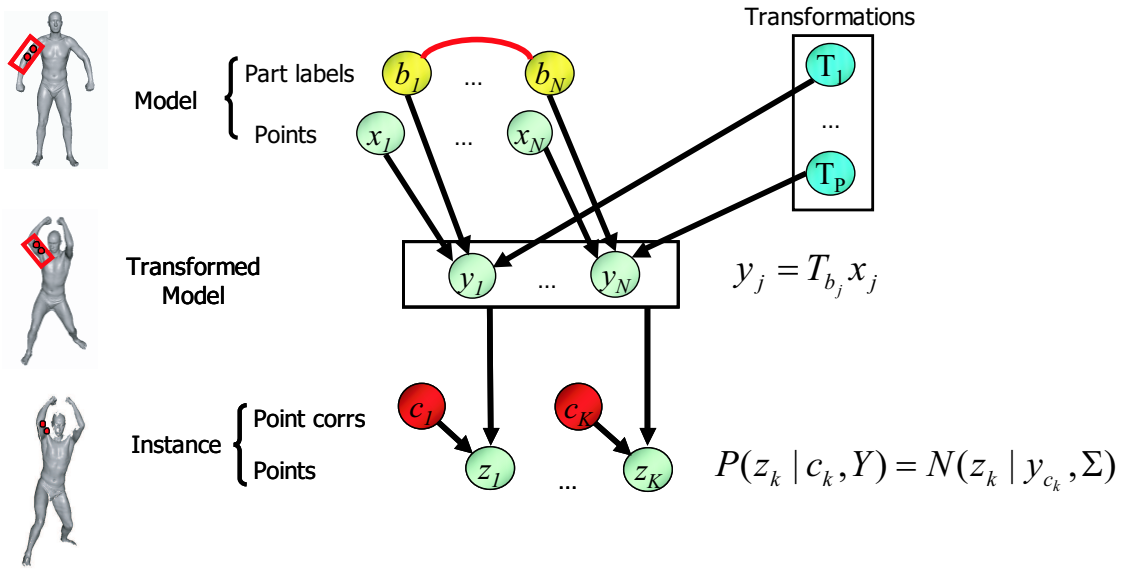


Figure 5.2: Probabilistic generative model for segmenting the template surface into rigid parts. The parts of template shape $\mathcal{M}^X$ are transformed by their respective rigid transforms, and then are resampled to generate the points of mesh $\mathcal{M}^Z$.

Definition 5.1.4 *An articulated model* **pose** *(or* **transformation***)* $T = (T_1, \ldots, T_{N_P})$ *is a placement of all skeleton parts in space. A rigid transformation matrix* T_n *is associated with each rigid part* $\mathcal{P}_n$*; the set of transformations must be consistent with the joint constraints, which are defined in Eqn. (5.1).*

We would like to point out that deformable models are in effect *articulated models with very many rigid parts*. For the deformable models introduced in Sec. 4, each mesh edge can be thought of as a separate rigid part. As a result of this, there are natural parallels in the use of deformable and articulated models. Nevertheless, the language of articulated models will prove useful, as it allows us to define meaningful regions on the object surface and their relationships.

5.1.2 Recovering Articulated Models

Here we will describe the problem of recovering an articulated from a set of scans. We start with a set of regular scans $\mathcal{M}^{D_1}, \ldots, \mathcal{M}^{D_N}$, represented as meshes, and corresponding to

different configurations of the same object. From these we can obtain a set of registered scans $\mathcal{M}^{Z_1}, \ldots \mathcal{M}^{Z_N}$ as follows. First, we pick the scan with the most complete surface $\mathcal{M}^{D_1}$ to be our template mesh $\mathcal{M}^X$. Then, we use the Correlated Correspondence algorithm presented in Chapter 4 to automatically register this template with all the remaining meshes in our dataset. We provide the obtained mesh correspondences to the method of Hähnel *et al.* [52], which then morphs the template mesh onto each scan. As a result, we obtain a set of scan meshes $\mathcal{M}^{Z_1}, \ldots, \mathcal{M}^{Z_N}$ which have the same mesh topology. In particular, each point z_j^i in scan mesh $\mathcal{M}^{Z_i}$ corresponds to point x_i in the template mesh $\mathcal{M}^X$. This step of bringing meshes into correspondence is only a pre-processing step to the algorithm presented in this chapter, which is independent of the particular registration method used.

Our goal is to recover an articulated model $\mathcal{S}^X = (\mathcal{M}^X, \mathcal{P}, G)$ from our registered scans. In doing so, our algorithm has to deal robustly with noisy scan readings and small errors introduced by the registration process. Furthermore, we want to be able to recover articulated models of real-world objects such as humans and animals, for which the articulated model assumptions in Defn. refdefn:articulated-model hold only approximately. Our solution is based on the language of probabilistic models, which allows us to deal with these issues in a principled manner. Our strategy for obtaining an articulated model consists of the following successive steps:

1. Partition the template mesh surface $\mathcal{M}^X$ into contiguous rigid parts, which can be posed to fit well the scan instances.

2. Given the rigid parts and their positions, estimate the set of articulated model joints.

This strategy is visualized in Fig. 5.1. It is motivated by the fact that we do not really need to know the joint locations in order to partition the mesh into rigid parts. The next two sections describe in detail how to perform each of the above steps.

5.2 Segmentation into Rigid Parts

In this section, we describe a solution to the combinatorial problem of segmenting the surface of the template mesh $\mathcal{M}^X$ into approximately rigid parts. We describe a probabilistic

graphical model which defines our objective function and then show how to optimize this function efficiently.

5.2.1 Probabilistic Model

Generating the Instance Meshes

First, we describe the generative process that transforms the template mesh $\mathcal{M}^X$ into the instance meshes $\mathcal{M}^{Z_1}, \ldots, \mathcal{M}^{Z_N}$. We assume that the surface of $\mathcal{M}^X$ is made up of the set of $\mathcal{P} = (1, \ldots, N_P)$ rigid parts. Each template mesh point x_j is associated with a part label b_j which denotes the rigid part to which the point belongs. Each part label b_j can take one of N_P possible values.

The template mesh is associated with a set of articulated model transformations $T^1, \ldots, T^N$. For each scan instance i, there is a different set of rigid part transformations $T^i = (T^i_1, \ldots, T^i_{N_P})$. All points assigned to part p share this set of transformations for that instance. More precisely, $y^i_j = T^i_{b_j}(x_j) = R^i_{b_j} x_j + s^i_{b_j}$, where y^i_j denotes the transformed location of x_j in instance i, and R is a rotation matrix while s is a translation vector.

We want to model objects which are not perfectly rigid, so we allow the point locations z^i_j in the meshes $\mathcal{M}^{Z_i}$ to deviate from these predicted locations. We assume that each point location z^i_j is generated from y^i_j by a Gaussian process:

$$P(z^i_j \mid y^i_j) = \mathcal{N}(z^i_j; y^i_j, \mathbf{diag}(\sigma^2)) \tag{5.2}$$

where σ^2 is the variance, chosen to be a multiple of the resolution of mesh $\mathcal{M}^X$. In the above equation, we used the assumption that meshes $\mathcal{M}^X$ and $\mathcal{M}^{Z_i}$ are registered, hence scan point z^i_j corresponds to template mesh point y^i_j. The part of the generative model described here, which transforms the template mesh into a scan mesh instance, is captured by the black edges in Fig. 5.2.

Introducing Part Contiguity Constraints

So far, our model allows a part to be composed of an arbitrary set of points interspersed throughout the mesh. What we actually want is that each part is comprised of a set of points

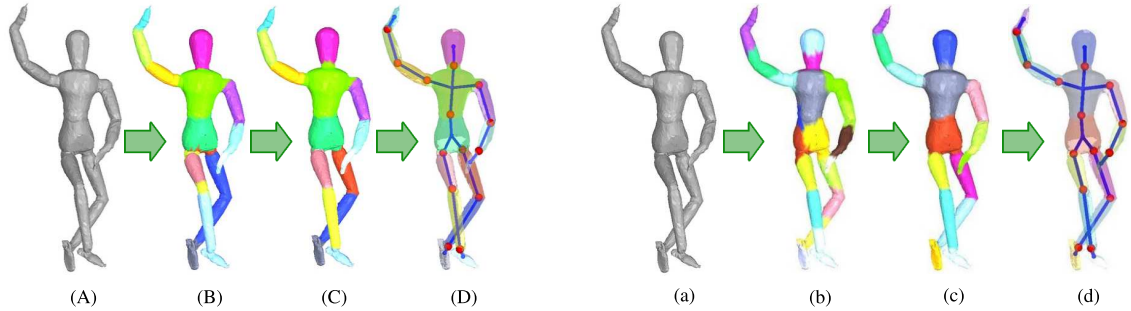


Figure 5.3: Puppet segmentations obtained with two different initialization strategies. (A),(a) Template mesh. (B) Initialization obtained by clustering the rigid point transformations using Gaussian Mixture Modeling, different parts are color-coded. (b) Initialization obtained by randomly dividing the mesh into small patches of similar size. (C),(c) Results of our part segmentation algorithm initialized with the initialization from (B),(b) respectively. (D),(d) Estimated skeleton joints.

in a connected region.

We choose to enforce this preference by using *soft contiguity constraints*. These constraints penalize cases when neighboring points in the template mesh have different part labels. More formally, we define two labels b_j and b_k to be neighboring if their corresponding points x_j and x_k are connected by an edge in $\mathcal{M}^X$. Soft contiguity constraints are probabilistic potentials $\phi(b_j, b_k)$ between all neighboring pairs of part labels b_j and b_k. They are displayed with red edges in Fig. 5.2.

The simplest way to enforce contiguity constraints is with the following potential:

$$\phi_1(b_j, b_k) = \begin{cases} \exp\{N(1-\tau)\} & : \quad b_j = b_k \\ \exp\{N\tau\} & : \quad b_j \neq b_k \end{cases} \tag{5.3}$$

where N is the number of scan instances and $\tau < 0.5$. These potentials introduce a separate penalty for each mesh edge that spans different rigid parts. We chose that the potential strength grows with the number of example scans N, with the goal of balancing it against the likelihood terms in Eqn. (5.2), whose number grows linearly with N. In all experimental results, except when explicitly stated, we will be using this kind of soft potential.

More sophisticated contiguity potentials are also possible. For example, we can enforce

a preference that rigid part boundaries are placed in the areas that undergo larger surface deformation. Consider the model mesh link (x_j, x_k). Since the template mesh is registered to all instance meshes, link deformation is easy to estimate. We could use the link deformation measure defined in Eqn. (4.4), but we opted for an even simpler measure. We look at the dot-product $n_{x_j}^T n_{x_k}$ of the link endpoint normals. The same quantity can be estimated in all scan instances. Let $\sigma_{j,k}$ denote its variance over the meshes. Our contiguity potential will prefer to place boundaries at a link whose point normals twist relative to each other, resulting in a large variance $\sigma_{j,k}$:

$$\phi_1(b_j, b_k) = \begin{cases} \exp\{N \max(1 - \lambda\sigma_{j,k}, \tau)\} & : \quad b_j = b_k \\ \exp\{N\tau\} & : \quad b_j \neq b_k \end{cases} \tag{5.4}$$

where $\tau < 0.5$.

The soft contiguity constraints (from either of the alternative definitions above) bias us toward a partitioning of the template mesh into contiguous regions. Importantly, they introduce an implicit preference for models which have fewer parts: the more parts there are, the more edges there are between mesh parts, the larger the penalty introduced by the pairwise contiguity potentials. Finally, the soft contiguity constraints induce a probabilistic model which can be optimized efficiently (as we will shortly discuss).

However, the soft contiguity constraints can still allow each part to be comprised of several disjoint components. For example, if the arms of an office chair always get raised and lowered together, they can be assigned to the same part by this model. Such results can be preferable in some situations but they are not appropriate for recovering an articulated object skeleton: the notion of a joint between parts is not well-defined when each part consists of several disconnected regions on the template mesh. In order to model the object articulation correctly, we need to disallow such cases. We explicitly detect them during the optimization, and assign a different part identity to each separate region. We discuss this straightforward procedure in more detail in Sec. 5.2.2.

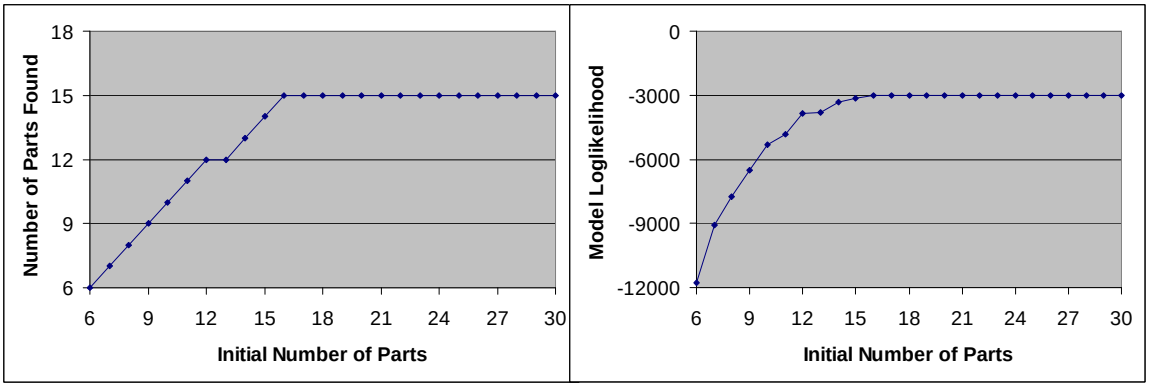


Figure 5.4: Graphs showing the number of parts of the final model and the log-likelihood score using initialization with different number of parts in the puppet dataset.

Expressing the Model as a Markov Network

Ignoring the hard contiguity constraints, the framework described in Sec. 5.2 defines a *Markov network* over the part labels $\mathcal{B}$. The Markov network encodes the joint distribution over these variables as a product of single and pairwise potentials:

$$P_T(\mathcal{B}) = \frac{1}{Z} \prod_j \phi_T(b_j) \prod_{j,k} \phi(b_j, b_k) \tag{5.5}$$

where Z is a normalization constant.

The singleton potentials $\phi_T(b_j)$ correspond to the probabilities that a template point x_j generates its corresponding points $z_{1,j}, \ldots, z_{N,j}$, as follows:

$$\phi_T(b_j = p) = \prod_{i=1}^{N} P(z_j^i \mid b_j = p, T_p^i). \tag{5.6}$$

The potential values depend on T, the set of rigid part transformations. The pairwise potentials in the Markov network correspond to the soft contiguity constraints, which are defined in (5.3) or (5.4).

5.2.2 Optimization

We want to find a joint assignment to the part labels $\mathcal{B}$ and the transformations T which maximizes the log-likelihood of the model:

$$\log P(\mathcal{B}, T) = \text{const} + \sum_{(j,k)\in\mathcal{E}^X} \log \phi(b_j, b_k) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \sum_{j=1}^{N_X} \|z_j^i - T_{b_j}^i[x_j]\|^2 \qquad (5.7)$$

where N_X is the number of points in meshes $\mathcal{M}^X, \mathcal{M}^{Z_1}, \ldots, \mathcal{M}^{Z_N}$. Note that our objective is defined as optimizing both the part assignment and transformations simultaneously, rather than marginalizing over the (hidden) part assignment variables. A hard assignment of points into parts is very appropriate for our application, and it also allows the use of efficient global optimization steps, as we discuss below. Note that the hard contiguity constraints are not accounted for in the above equation, and have to be enforced separately.

The objective in (5.7) is non-convex in the set of variables $\mathcal{B}, T$. We optimize it using hard *Expectation-Minimization (EM)* to find a good assignment for $\mathcal{B}, T$ in an iterative fashion. EM iterates between two steps: the *E-step* calculates a hard assignment for all part labels $\mathcal{B}$ given an estimate of the transformations T. The *M-step* improves the estimate for the parameters T using the labels $\mathcal{B}$ obtained in the E-step.

E-Step

Our goal in the E-step is to find the MAP assignment to the part labels maximizing (5.7) for a given set of transformations T. It turns out that this is an instance of the Uniform Labeling problem [63], which can be expressed as an integer program. Following Kleinberg and Tardos [63], we introduce indicator variables b_{jp} for the event $b_j = p$, and associated the constraints $b_{jp} \in \{0, 1\}$ and $\sum_{p=1}^{P} b_{jp} = 1$ with them. These integer constraints imply that we have only a single p for which $b_{jp} = 1$, and the others are all 0. The log-cost associated with a particular single potential can then be expressed as $\sum_{p=1}^{P} c(j, p) \cdot b_{jp}$ where

$$c(j, p) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \|z_j^i - T_{i,p}[x_j]\|^2 \qquad (5.8)$$

The *separation cost* of an edge in mesh $\mathcal{M}^X$ can also be defined in terms of the variables b_{jp}. The difference between the labels of the edge endpoints can be expressed as

$$\alpha_{jk} = \frac{1}{2}\sum_{p=1}^{P} |b_{jp} - b_{kp}| = \frac{1}{2}\sum_{p=1}^{P} \alpha_{jkp}$$

where $\alpha_{jkp} = |b_{jp} - b_{kp}|$. The cost associated with an edge is therefore $s(j,k) \cdot \alpha_{jk}$, where depending on the definition of the soft contiguity potential, we can have $s_1(j,k) = N(1-2\tau)$ or $s_2(j,k) = N\max(1-\lambda_{j,k}-\tau, 0)$.

We can now rewrite our optimization problem as an integer program:

$$\begin{aligned}
\max \quad & \sum_{j=1}^{N_X}\sum_{p=1}^{P} c(j,p)\cdot b_{jp} + \sum_{(j,k)\in\mathcal{E}^X} s(j,k)\cdot\alpha_{jk} \\
\text{s.t.} \quad & \sum_{p=1}^{P} b_{jp} = 1, \qquad j = \{1,\ldots,N_X\} \\
& \alpha_{jk} = \frac{1}{2}\sum_{p=1}^{P}\alpha_{jkp}, \quad (j,k)\in\mathcal{E}^X,
\end{aligned}$$

$$\begin{aligned}
& \alpha_{jkp} \geq b_{jp} - b_{kp}, \quad (j,k)\in\mathcal{E}^X, p\in\mathcal{P} \\
& \alpha_{jkp} \geq b_{kp} - b_{jp}, \quad (j,k)\in\mathcal{E}^X, p\in\mathcal{P} \\
& b_{jp} \in \{0,1\}, \qquad j = \{1,\ldots,N_X\}, p\in\mathcal{P}
\end{aligned}$$

In general, solving an integer program optimally is NP-hard. However, we can define a linear programming relaxation of the above problem by replacing the integrality constraints $b_{jp} \in \{0,1\}$ with $b_{jp} \geq 0$. This relaxation allows fractional solutions for the labels b_j. The linear program can be solved very efficiently by a solver such as CPLEX.

For problems of this type, Kleinberg and Tardos [63] describe a method for rounding the fractional solution, losing at most a factor of 2 in the objective function. In our experiments we did not need to perform this rounding because the relaxed linear formulation always returned integer solutions. In this case, we are guaranteed that our solution is the optimal

assignment of template mesh points to parts, which maximizes (5.7) given a set of rigid transformations T.

The inference can also be performed using a procedure proposed by Boykov *et al.* [16], which employs a multiway-cut algorithm augmented with an iterative procedure called *alpha-expansion*. This algorithm still provides the quality guarantees enjoyed by the linear programming formulation above, and our experiments showed that it performs about 10 times faster in practice. Our final implementation uses this algorithm for maximum efficiency. More information about how to perform inference in Markov networks is provided in Sec. 3.3.3.

As we discussed in Sec. 5.2.1, our soft contiguity constraints allow a part to consist of several disconnected regions on the surface of the template mesh. However, we can easily detect such cases by examining the part labels returned by the Markov network inference. Whenever a part is made of several disconnected regions, we break it up and assign each region to a separate part. This step satisfies our preference that each rigid part is a single connected component of the surface, while preserving the value of the objective function in Eqn. (5.7).

M-Step

The goal of the M-step is to find the set of rigid part transformations T which maximize the log-likelihood in (5.7), given the part label assignments $\mathcal{B}$ supplied by the E-step. The objective function decomposes into a separate equation for each T_p^i:

$$\underset{T_p^i}{\operatorname{argmin}} \sum_{j=1}^{N_X} I(b_j = p) \cdot \|z_{ij} - T_p^i(x_j)\|^2 \tag{5.9}$$

where $I(\cdot)$ is the indicator function. This problem is isomorphic to the registration problem studied extensively in the ICP literature. We adapt the canonical solution to this problem, proposed by Besl and Mckay [13], where 3D rotations are represented as quaternions, and a closed form estimate of T_p^i is obtained by solving a small system of linear equations.

Figure 5.5: Four different poses from the puppet dataset display the 15 rigid parts and the articulated skeleton, both of which are recovered automatically.

5.2.3 Initializing the Model

The optimization criterion for our model is a complex non-convex function in terms of the transformations T and part labels $\mathcal{B}$. Our hard EM algorithm is only capable of getting to a local minimum of this function. Therefore, it is dependent on a good starting point. Here we address the problem of providing the EM algorithm with a good starting point.

Obtaining Transformation Estimates

One way of initializing the algorithm is by performing clustering in the space of rigid transformations, as suggested by Cheung *et al.* [25]. Since the correspondences between the template mesh $\mathcal{M}^X$ and all instance meshes $\mathcal{M}^{Z_i}$ are known, we can estimate the local rigid transformation between a point x_j and its counterpart z_j^i. To do so, we look at small local patches centered at x_j and z_j^i, and assume that the local transformation between the patches is rigid. Using ICP [13], the optimal rigid transformation t_j^i registering these patches can be computed. Every t_j^i can be represented as a vector in 6 dimensional Euclidean space.

Each point x_j becomes associated with N such vectors, corresponding to its transformation in each instance mesh. The resulting stacked vectors can then be clustered by applying adaptive PCA [8], a variant of Gaussian Mixture Modeling. The cluster labels serve as an initial set of part labels to the points x_j. A result of this step is demonstrated in Fig. 5.3(B). As it does not exploit the connectivity of the mesh surface, it can serve as initialization to our main algorithm, but is not good enough by itself.

Using a Matlab implementation of adaptive PCA available on the web [8], clustering a set of 7 puppet poses (4000 points each) into 15 rigid parts takes about an hour on a Sun Blade 2000 dual-processor machine. Surprisingly, this pre-processing step becomes the bottleneck of the whole part-finding pipeline. Here we propose a more efficient way of initializing the model that gives comparable or even better results than clustering.

The main insight we will exploit is that the soft contiguity constraints introduce a preference for models which have fewer parts: The more parts there are, the more edges there are between mesh parts, the larger the penalty introduced by the pairwise contiguity potentials. Thus, we can start the model with a large number of possible parts, and redundant part hypotheses will be automatically pruned.

We therefore initialize the model by dividing the mesh into small surface patches, all of which have approximately the same area. This is done by uniformly subsampling the mesh, and assigning each point on the original mesh to the nearest point on the subsampled mesh. All points on the original mesh that are given the same assignment are grouped together to form a patch. This process takes a fraction of a second, compared to an hour for our previous initialization scheme.

When the subdivision into patches is fine enough, some rigid parts will contain patches that lie completely inside them, and the transformations for those patches from the model mesh $\mathcal{M}^X$ to the morphed meshes $\mathcal{M}^Z$ will closely approximate the corresponding transformations for the actual rigid parts. Using the patches as initial part assignments for our algorithm, we get a good starting point for the first M-Step.

Determining the Initial Number of Parts

The idea of initializing the algorithm by subdividing the surface of mesh $\mathcal{M}^X$ into patches leads to the question of how many initial patches are necessary. In principle, it is sensible

to choose an initial number of patches N_P to be larger than the actual number of rigid parts we expect. The larger N_P is, the more likely it is to get a patch that lies completely inside a rigid part. As we discussed, our model encodes a preference for having fewer parts, so that redundant part hypotheses are pruned automatically. Indeed, our experiments (Fig. 5.4) show that, initially, as we increase the number of model parts N_P, the number of selected parts increases; but once the optimal number of parts is reached, increasing N_P further does not increase the number of parts found.

In the case of rigid objects, the final number of parts found by our algorithm is generally the correct number of parts in the articulated objects. When the object parts undergo some non-rigid deformations, the number of parts found depends on the tradeoff between allowing more deformation within a part and splitting into more parts to preserve part rigidity. These preferences depend on the edge potential τ and the variance σ^2.[1] As their ratio $\kappa = \sigma^2/\log(\tau)$ increases, we allow instance mesh point to deviate more from their predicted locations.

5.2.4 Simulated Annealing

When κ is large, the problem becomes less constrained, with multiple possible solutions that are plausible and have similar scores. This large hypothesis space makes the relaxed integer program considerably more difficult to solve, especially in the absence of good transformation estimates. To address the problem, we start with a low value of κ, and gradually increase it in subsequent algorithm iterations. The intuition behind this approach is that we separate the error due to random initialization from the error due to non-rigidity. During the early stages of the algorithm, there is a great deal of error due to random initialization; we therefore start with a smaller-than-desired κ, heavily penalizing discrepancies from the rigid part assumption. As a side effect, our algorithm will tend to split a non-rigid part into several rigid parts, resulting in more parts than we want. As the algorithm converges, the noise from random initialization becomes less significant, so we can gradually relax the rigidity assumptions and anneal the value of κ; this process results in the merging

[1] For this discussion, we will assume that we are using the simpler soft contiguity potential from Eqn. (5.4). In the case when our contiguity potential depends on link deformation, there are additional link deformation parameters to be considered.

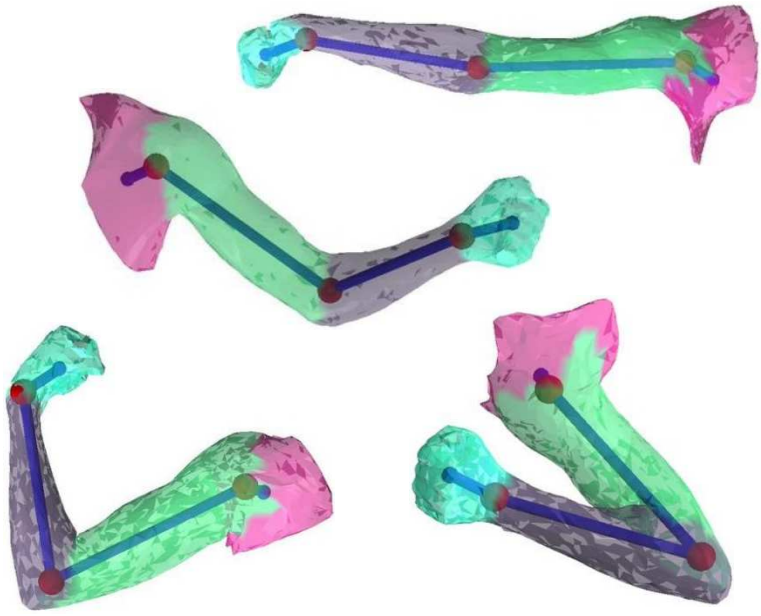


Figure 5.6: Four different poses from the arm dataset display four (approximately) rigid parts and the articulated skeleton, both of which are recovered automatically

(and modification) of parts, and the elimination of unnecessary part hypotheses.

5.3 Estimating the Skeleton Joints

Once we obtain the part labels for every point in a mesh, it is easy to recover the joint between two adjacent rigid parts. We adapt the solution by Cheung *et al.* [25]. Suppose we want to find the joint between two adjacent parts p and q. Let the coordinates of the joint in the model mesh be $g_{p,q}$. Since the joint belongs to two object parts simultaneously, it should satisfy the equation:

$$T_p^i(g_{p,q}) = T_q^i(g_{p,q}), \qquad i = 1 \ldots N \tag{5.10}$$

Putting together the equations for all instance meshes, $g_{p,q}$ is the solution to the following problem:

$$\arg\min_{g_{p,q}} \sum_{i=1}^{N} \|T_p^i(g_{p,q}) - T_q^i(g_{p,q})\|^2 \tag{5.11}$$

The rigid part transformation T_p^i, T_q^i are known from the previous stage in our algorithm, this equation is a particularly easy least-squares problem.

Sometimes, the solution to the above equation can be an entire subspace of points. Suppose the joint only allows one degree of movement, such as the knee joint of a human leg. Then any point on the line perpendicular to the plane of allowed movement is a solution candidate. We chose to resolve this problem by introducing an additional regularization term, which likes to place the joint close to where the two parts meet (and inside the body). We require that the joint is close to the centroid $c_{p,q}$ of the set of points that lie on the boundary between the two parts p and q in mesh $\mathcal{M}^X$. Then $g_{p,q}$ is the solution to the following least-squares problem:

$$\arg\min_{g_{p,q}} \sum_{i=1}^{N} \|T_{ip}(g_{p,q}) - T_{iq}(g_{p,q})\|^2 + \gamma \|g_{p,q} - c_{p,q}\|^2 \tag{5.12}$$

With the above formulation, we can compute the joint between any two adjacent parts on the template surface, which completes our automatic recovery of the skeleton $\mathcal{S}^X = (\mathcal{M}^X, \mathcal{P}, G)$.

5.4 Experimental Results

We applied our algorithm to meshes from three different datasets. In one data set, we used a range scanner based on temporal stereo [36] to acquire a set of seven different complete surface meshes of a wooden puppet in different positions. Each mesh was constructed from ten range scans taken from different viewing angles, composed using the method of Curless and Levoy [33], and subsampled to contain $\sim$4000 points and 8000 triangles.

We automatically aligned one puppet mesh to the remaining six meshes in our puppet

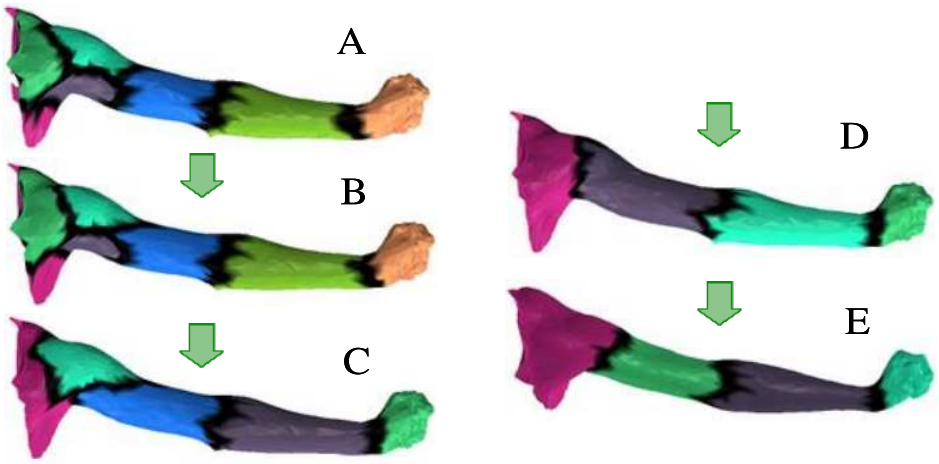


Figure 5.7: Illustration of annealing on the Arm dataset. The sequence above is obtained by starting with low κ, and gradually increasing it after each iteration of EM, until our desired κ is reached. The algorithm anneals part hypotheses and eventually converges at four parts (D). Setting the value of κ to be too large (E) focuses more on the soft contiguity constraints and less on the underlying geometric structure. This results in partitions which reduce the number of links between parts.

dataset using the Correlated Correspondences algorithm (Chapter 4). We experimented with both initialization approaches described in Sec. 5.2.3. Results shown in Fig. 5.3 demonstrate that both initialization methods performed equally well. However, the method where we initialize the M-step by partitioning the mesh $\mathcal{M}^X$ into small surface patches is preferable because of its simplicity and overwhelming computational advantage. The correct model containing 15 parts was found whenever the number of surface patches in the initialization was equal to or greater than 16 (Fig. 5.4). More instances of the final model superposed onto the recovered articulated skeleton are displayed in Fig. 5.5. To our knowledge, this is the first implementation that estimates such a complex skeleton from real world data with very few poses, in a completely unsupervised way.

Our second data set consisted of eight meshes of a human arm, acquired and used by Allen *et al.* [1]. We used a standard hole filling technique (unpublished implementation similar to [35, 71]) to fill the scan holes in a pre-processing step. This dataset is more challenging than the puppet dataset because the arm undergoes significant deformations as it bends, so that it is not purely an articulated model composed of rigid parts. Fig. 5.7

demonstrates the progress of our algorithm as the parameter κ is increased, until we end up with four parts, which is the intuitively correct number of parts for the arm (see Fig. 5.6). The partition of the object depends on the exact setting of the parameter κ (see Fig. 5.7 D and E). Setting the value of κ to be too large over-emphasizes the soft contiguity constraints. The part boundaries are shifted to a configuration minimizing the number of links between parts, ignoring the underlying geometric structure (Fig. 5.7 E). Our results on the arm dataset suggest that, even in the presence of significant non-rigidity like twisting of the forearm and bulging of the biceps, our algorithm performs quite well.

Our third dataset was the most challenging, as it contains 65 scans of a particular individual placed in a variety of poses, which was acquired with a Cyberware WRX scanner. These meshes were all registered with the Correlated Correspondence algorithm, aided with a few handpicked markers for each mesh. Each of the registered meshes contains 12K points and 25K polygons, and exhibits non-trivial muscle deformation (see Fig. 5.1 for examples). Our unoptimized implementation easily scales to a dataset of such size, and takes about three minutes to obtain the articulated model. We experimented with our two different soft contiguity potentials, which were defined in equations (5.3) and (5.4). In Fig. 5.8 (1a) and (1b), we show the rigid parts obtained by using the first potential, which penalizes all links that lie on part boundaries by the same amount. The algorithm automatically finds an intuitive decomposition of the object into 17 articulated parts, including a compelling decomposition of the human torso itself. Upon closer inspection, however, one can see that there are non-intuitive artifacts near some part boundaries (top left insets). The main cause is that our rigid transformation estimates T are most inaccurate near part boundaries, where the deformation is usually the largest. As a result, points on the top of the thigh and at the bottom of the neck get assigned to the closer, yet intuitively incorrect articulated part. (We confirmed this hypothesis, by starting from the result in Fig. 5.8 (1a) and (1b), and executing several more iterations by a version of our EM algorithm which ignores the soft contiguity potentials. The boundary was still placed in a similar, visually suboptimal place.) In addition, our first soft contiguity potential introduces a bias toward shorter part boundaries. This is confirmed by the results — the part boundaries in the top insets are shorter than those of their counterparts at the bottom.

Our second soft contiguity potential prefers to place the part boundaries in areas which

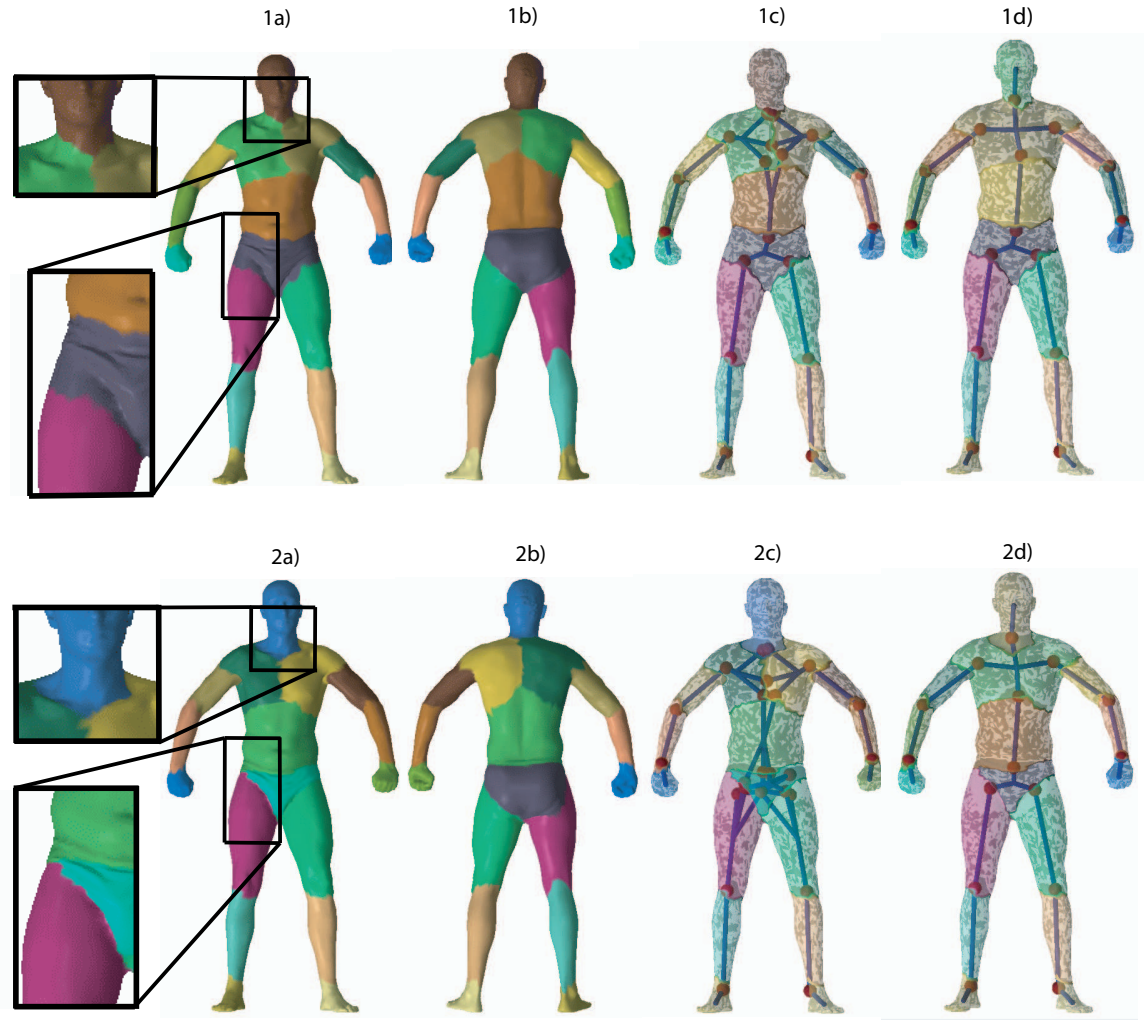


Figure 5.8: Results of the algorithm on the human dataset. (1a) and (1b) Segmentation into parts obtained using the soft contiguity potential from Eqn. (5.3). The different parts are color-coded. (1c) Recovered joints between adjacent parts on the surface. (1d) Recovered tree-shaped skeleton. (2a) and (2b) Segmentation into parts using the soft contiguity potential from Eqn. (5.4). (2c) Recovered joints between adjacent parts on the surface. (2d) Recovered tree-shaped skeleton.

undergo larger deformation. Using this potential with a setting of $\lambda = 2$ recovers an articulated skeleton consisting of 18 parts — Fig. 5.8 (2a) and (2b). In particular, the buttock and crotch area are split into two separate parts. From the bottom row insets we can see that this potential produces an intuitively better-looking segmentation. In particular, this potential clearly fixes the problem with the upper thigh boundaries and produces a considerably more symmetric partitioning of the upper chest. It is not as biased towards short part boundaries, which can be seen by looking at the neck and the upper thighs. The resulting model contains the parts one would expect, and nice intuitive boundaries between them.

We applied our joint estimation algorithm to the two segmentation results. In Fig. 5.8 (1c) and 2c, we show the joints between all adjacent parts on the template surfaces. In computing the joints, we use a very small value for the parameter γ to show the locations of the joints as predicted by the rigid transformations. Interestingly, the resulting skeleton is not tree-shaped in both cases. While this is not necessarily a problem in example Fig. 5.8 (1c), it may be undesirable for two reasons. As the result in Fig. 5.8 (2c) shows, some parts can become adjacent as a result of noise in the meshes, and small errors by the algorithm, resulting in extra joints. Whether these extra joints are meaningful depends on the particular application (for example, they might be useful for animating articulated models). The precise criteria on when to keep certain joints should be defined with a particular application in mind, and are outside the scope of this chapter.

Still, many tracking and detection algorithms (e.g., [18]) require a tree-shaped articulated structure, which allows the definition of part motion in terms of kinematic chains. By introducing a post-processing step, which is allowed to merge parts, and to remove joints which cause a fairly large error in Eqn. (5.12), we obtain the tree-shaped models shown in Fig. 5.8 (1d) and (2d). The spine is not completely straight in both cases — the small error most likely is a result of the fact that the algorithm was run on a set of random (and hence non-symmetric) poses. But overall, the results are very satisfactory. To the best of our knowledge, this is the first algorithm to recover such complex articulated models in a completely unsupervised manner.

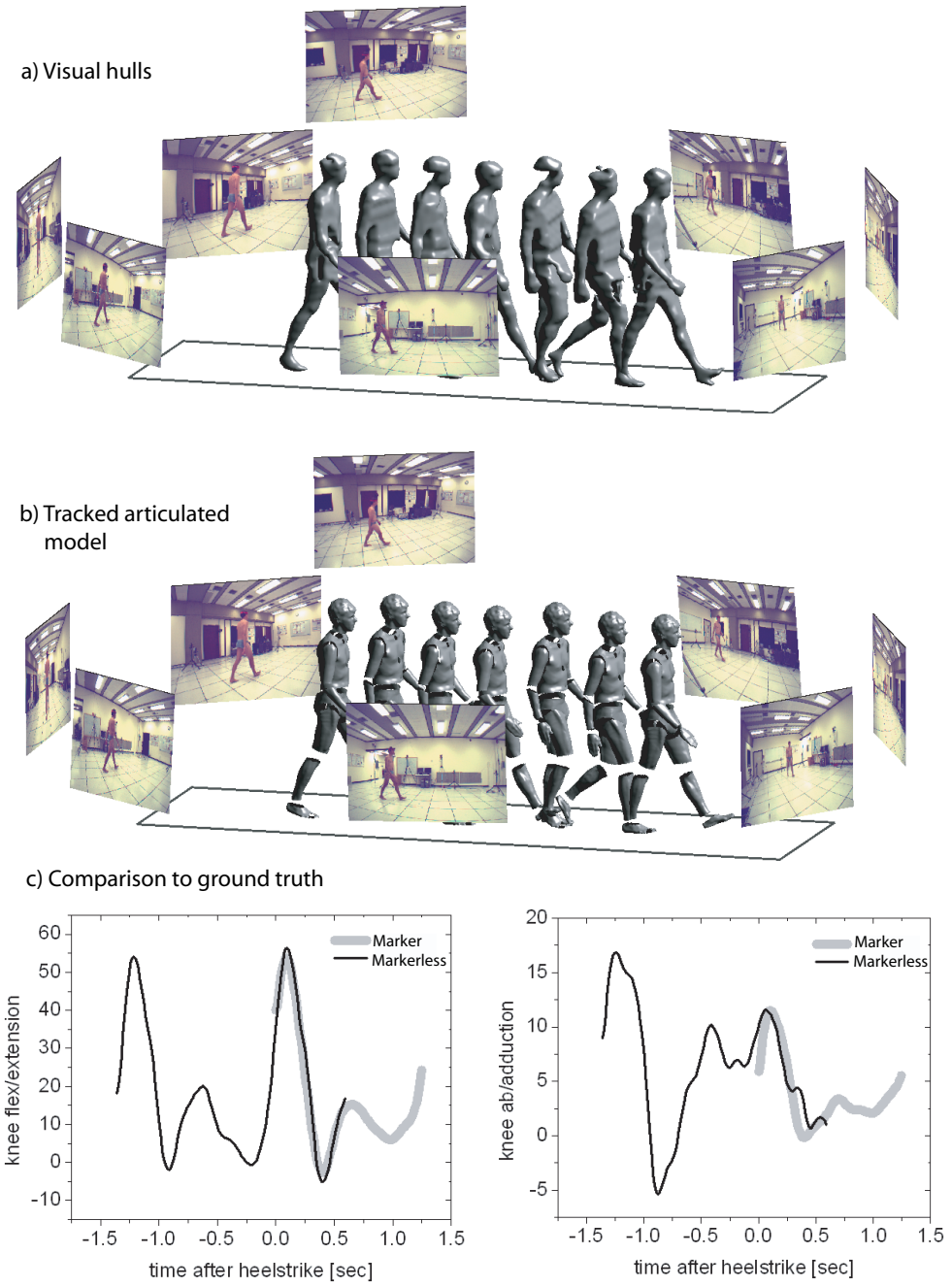


Figure 5.9: Tracking an articulated model in visual hull sequences. (a) The camera setup and the visual hulls obtained by shape-from-silhouette computation. (b) Tracking results by our algorithm c) The algorithm quality was compared to the ground truth obtained by tracking photoreflective markers on the body of the walking subject. The algorithm recovers well both degrees of freedom at the knee.

5.5 Articulated Model Tracking

The capability to capture articulated motion has many possible applications. They include biomechanical applications such as human gait analysis, where tracked human motions can be examined to diagnose gait abnormalities and joint diseases. They also span entertainment applications, where the movement of popular actors can be captured for use in movies and games. Currently, these tasks are done with commercial motion capture systems[82, 83], which attach optical or magnetic markers on the person whose motion is to be tracked and use triangulation on the positions of the markers to achieve tracking. Although these systems generally produce very good results, they are invasive and difficult to use.

In recent years, researchers have proposed a variety of vision-based systems for tracking human body motion in video sequences and silhouette image sequences (e.g., [95, 48, 18, 23, 106, 78, 107] among many others). All of these algorithms (with the exception of the work by Cheung *et al.* [25]) assume that an articulated model of the tracked object is available before the start of the algorithm. Such models are usually human-designed, and often use cylinders to approximate the shape of the human body parts. The method in this chapter provides an automatic way of recovering accurate articulated models, which can reduce human effort substantially. In this section, we will demonstrate the suitability of our recovered articulated models for the task of capturing articulated object kinematics.

We describe a simple algorithm which can be used to track our articulated models. The algorithm is an extension of the standard Iterative Closest Point algorithm [13, 52], which enforces the joint constraints during tracking. While it is a variation of existing approaches, the exact formulation, to the best of our knowledge, is novel. Some previous approaches for tracking articulated models (e.g., [25, 18]) enforce hard constraints on the kinematic structure (joints of the skeleton must be preserved). Our approach allows small movement at the joint, which is penalized in least-squares terms. As a result we obtain a more anatomically correct model, and an objective function that can be optimized in an efficient and straightforward manner. Our formulation can be used in a straightforward way even when the skeleton structure contains loops. In contrast, the vast majority of the tracking algorithms assume that the skeleton is tree-shaped and would require significant

modification if that assumption is violated.

5.5.1 Probabilistic Model

We are interested in tracking an articulated model $\mathcal{S}^X$ in 3D shape-from-silhouette data (for a detailed discussion of shape-from-silhouette estimation, please refer to Cheung *et al.* [26]). The output from the shape-from-silhouette estimation is a sampling of the object's surface at each point in time i, which can be represented as a 3D point cloud $\mathcal{P}^{Z_i}$. These point clouds can be obtained using different shape-from-silhouette techniques [26, 85].

Our goal is to compute an alignment T^i that brings the articulated model surface into close alignment with the point cloud $\mathcal{P}^{Z_i}$ (from this point on, we omit the subscript i for the benefit of clear notation). Our generative model assumes that each scan point z_k is generated from its corresponding point x_j in the appropriately posed articulated model. To express this formally, we associate a correspondence variable c_k with each point z_k in the point cloud $\mathcal{P}^{Z_i}$. Setting $c_k = j$ selects a corresponding point x_j in the skeleton mesh $\mathcal{M}^X$. Because the shape-from-silhouette techniques are based on background subtraction in video [26], they tend to produce many spurious readings, which have to be explicitly dealt with in the model. We do this by associating an additional *validity variable* v_k with each point z_k, which accounts for cases when the reading is generated by noise in the acquisition process. If $v_k = 1$, the point z_k is generated from some matching point x_j as follows:

$$P(z_k \mid v_k = 1, c_k = j, T_{b_j}, x_j) = \mathcal{N}(z_k; T_{b_k}(x_j), \Sigma_C). \tag{5.13}$$

Intuitively, this equation specifies that point z_k is generated from its corresponding template mesh point x_j, transformed by T_{b_k} in accordance with the current skeleton pose. If $v_k = 0$, the point z_k is generated from our noise model, which is a uniform distribution on point locations:

$$P(z_k \mid v_k = 0) = \alpha \tag{5.14}$$

In addition, we assume a uniform prior $P(v_k)$ for all validity variables and a uniform prior $P(c_k)$ for all correspondence variables.

In posing the articulated model, our generative model needs to enforce the joint constraints as well. We treat the joints as elastic bands, which are allowed to stretch at a cost. Our joint mismatch is associated with the amount of band stretching at the joint, and is defined as

$$\psi_J(T_p, T_q \mid \mathcal{S}^X) = \exp\{-\frac{1}{2}\left(T_p(g_{p,q}) - T_q(g_{p,q})\right)^T \Sigma_J^{-1} \left(T_p(g_{p,q}) - T_q(g_{p,q})\right)\} \quad (5.15)$$

The penalty is related to our definition of a joint in Eqn. (5.10). The covariance matrix Σ_J can be estimated from data, but in our experiments, we set it to the same value for all joints.

5.5.2 Optimization

The goal of tracking is to recover the pose T^i of the model that best first the point cloud $\mathcal{P}^{Z_i}$ in each sequence frame i. Our Expectation-Maximization algorithm aims to optimize the following expected log-likelihood for each frame (superscripts i omitted):

$$E_{P(C,V|\mathcal{S}^X,\mathcal{P}^Z,T)}\left[\log P(C, V, T \mid \mathcal{S}^X, \mathcal{P}^Z)\right] \quad (5.16)$$

Our iterative solution will alternate between an E-step which computes the probabilities $P(C, V \mid \mathcal{S}^X, \mathcal{P}^Z, T)$, and an M-step which uses these probabilities to optimize the objective in Eqn. (5.16).

E-step

It is fairly straightforward to see that the probability $P(C, V \mid \mathcal{S}^X, \mathcal{P}^Z, T)$ decomposes into a separate estimation for each scan point z_k:

$$\begin{aligned} P(v_k, c_k \mid z_k, T, \mathcal{S}^X) &\propto P(z_k \mid v_k, c_k, T, \mathcal{S}^X) P(v_k, c_k \mid \mathcal{S}^X, T) && (5.17)\\ &= P(z_k \mid v_k, c_k, T, \mathcal{S}^X) P(v_k) P(c_k) && (5.18)\\ &= P(z_k \mid v_k, c_k, T, \mathcal{S}^X) && (5.19) \end{aligned}$$

In this derivation, we used Bayes' rule to obtain (5.17) and conditional independence assumptions in the probabilistic model to obtain (5.18). The final expression in (5.19) follows

from the fact that the distributions $P(v_k)$ and $P(c_k)$ are uniform in our model.

Now, we can define the E-step probabilities:

$$\begin{aligned} q_{k,j} &= P(v_k = 1, c_k = j \mid z_k, T, \mathcal{S}^X) \propto P(z_k \mid v_k = 1, c_k = j, T_{b_j}, x_j) \\ q_{k,0} &= P(v_k = 0 \mid z_k, T, \mathcal{S}^X) \propto P(z_k \mid v_k = 0) \end{aligned} \quad (5.20)$$

The conditional distributions on the right hand side are defined in Eqn. (5.13) and Eqn. (5.14) and can be readily computed. The resulting values can be normalized to obtain $q_{k,0}$ and $q_{k,j}$ for $j = \{1, \ldots, N_X\}$.

The straightforward treatment of the E-step probabilities above ignores the issue of efficiency. Naively computing N_X probability values for each scan point is unnecessary, especially because most of the probability mass is captured by $\max_j q_{k,j}$, corresponding to the nearest model point, and $q_{k,0}$. The vast majority of the remaining $q_{k,j}$ values are infinitesimally small and can be ignored.

M-step

In the M-step, we are given the expectations q, and are interested in finding the pose parameters that optimize the objective in Eqn. (5.16). First we expand the quantity inside the expectation:

$$\begin{aligned} P(T, V, C \mid \mathcal{S}^X, \mathcal{P}^Z) &\propto P(\mathcal{P}^Z \mid V, C, T, \mathcal{S}^X) P(V, C \mid T, \mathcal{S}^X) P(T \mid \mathcal{S}^X) \\ &= P(\mathcal{P}^Z \mid V, C, T, \mathcal{S}^X) P(T \mid \mathcal{S}^X). \end{aligned} \quad (5.21)$$

Here we made the same assumptions as the ones discussed in the derivation of Eqn. (5.17).

Using this result, and expanding Eqn. (5.16), the M-step objective becomes:

$$\begin{aligned} \arg\min_T \quad & \sum_k \sum_j q_{k,j} \left(z_k - T_{b_k}(x_j)\right)^T \Sigma_C^{-1} \left(z_k - T_{b_k}(x_j)\right) \\ + \quad & \sum_{(p,q) \in G^X} \left(T_p(g_{p,q}) - T_q(g_{p,q})\right)^T \Sigma_J^{-1} \left(T_p(g_{p,q}) - T_q(g_{p,q})\right) \end{aligned} \quad (5.22)$$

Using the definition of rigid transformation $T_p(x_j) = R(r_p)x_j + t_p$, we can express the

objective directly in terms of the twist rotation parameters r_p and translation vectors t_p:

$$\begin{aligned}\arg\min_{r,s} \quad & \sum_k \max_j q_{k,j} \left(z_k - R(r_{b_k})x_j - t_{b_k}\right)^T \Sigma_C^{-1} \left(z_k - R(r_{b_k})x_j - t_{b_k}\right) \\ + \quad & \sum_{(p,q)\in G^X} \left(R(r_p)g_{p,q} + t_p - R(r_q)g_{p,q} - t_q\right)^T \Sigma_J^{-1} \left(R(r_p)g_{p,q} + t_p - R(r_q)g_{p,q} - t_q\right)\end{aligned} \tag{5.23}$$

When we use the standard linearization of rotation $R(r_p) \approx (I+\widehat{\omega_p})R(r_p^{\text{old}})$ (see Sec. 2.2.1), the above objective reduces to a least squares problem over the translation parameters t and the exponential map parameters ω, for a total of $6 \times N_P$ variables, and can be solved very efficiently.

5.5.3 Experimental results

We experimented with this algorithm on several human movement sequences. The sequences were provided to us by the Stanford Biomotion Laboratory, which specializes in studying diseases of the human knee. In our first experiment, the articulated model was obtained from a single scan of the subject. The 15 rigid parts, and the 14 joints connecting them were defined by a human (the process took about an hour). Seven cameras placed around the viewing volume (shown in Fig. 5.9(a) provided images, in which background differencing was performed in order to obtain silhouettes. The silhouettes were used to construct a 3D visual hull [85]. Our algorithm was applied to the point clouds that were obtained from the visual hulls. For each new frame in the sequence, the algorithm was initialized with the position obtained in the previous frame. The algorithm produced excellent results, shown in Fig. 5.9(b). The knee movement in that sequence was compared to ground truth, obtained using photoreflective markers and standard motion capture hardware. Our estimates proved close to the ground truth, especially for the main degree of freedom at the knee, denoted as flex/extension in Fig. 5.9(c)-1. The algorithm performed reasonably well in recovering even the second degree of freedom at the knee, denoted as abduction in Fig. 5.9(c)-2. In contrast, some methods (e.g., Cheung *et al.* [27]) assume that there is a single degree of freedom at the knee.

We then acquired a more difficult data set using the subject, whose articulated model

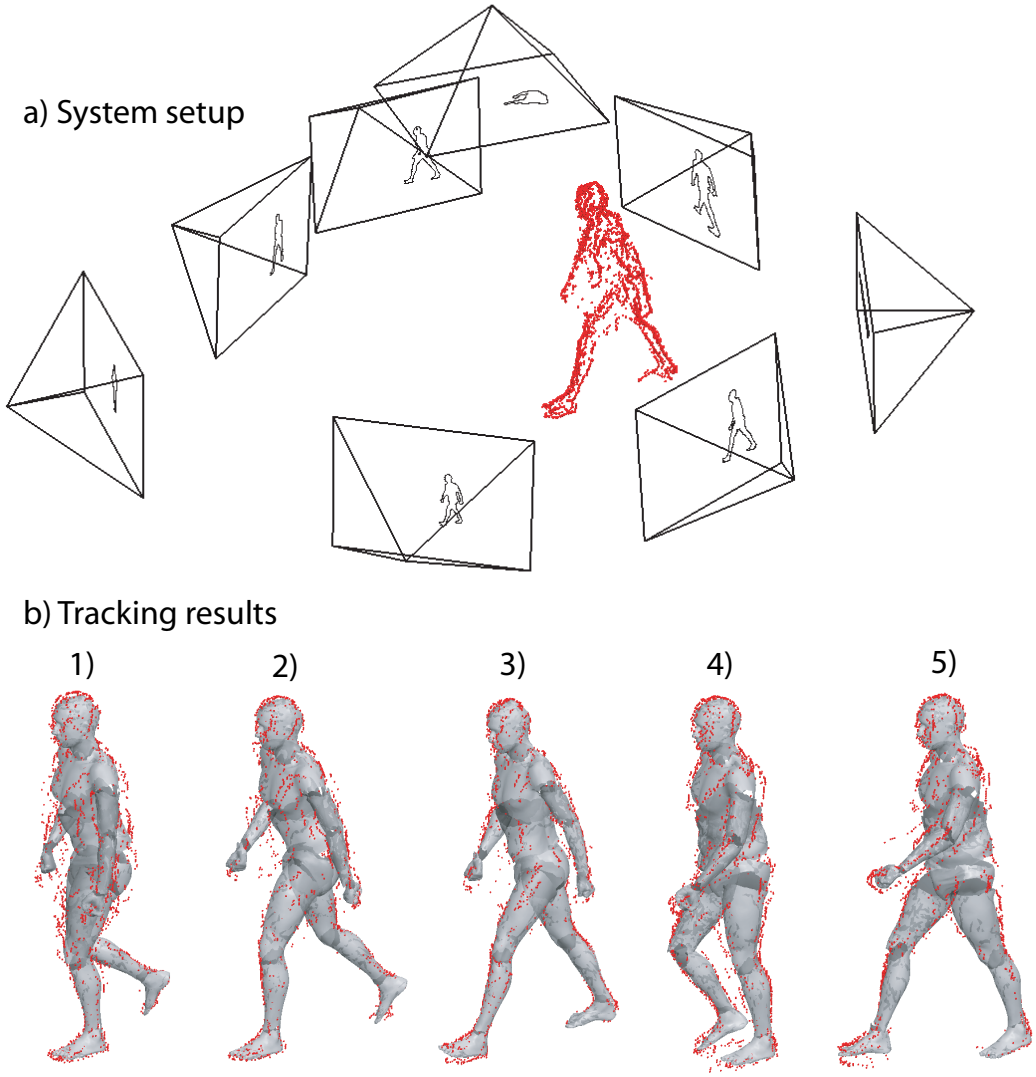


Figure 5.10: Tracking an automatically recovered articulated model in point cloud data. (a) The camera setup and the point cloud obtained for a particular frame by using the shape-from-silhouette method of [27]. b) The results of our algorithm for some frames in the sequence, superposed on the point clouds in those frames (shown in red).

was recovered automatically in Sec. 5.4. This sequence was obtained using an 8-camera setup, shown in Fig. 5.10a. Background subtraction was less successful in this dataset, due to faster camera shutter speed settings, and use of dark clothing. As a result, the computed visual hulls exhibited large errors. We opted for the shape-from-silhouette method of Cheung *et al.* [26], which uses additional image color cues. Our automatically recovered model was tracked in the resulting point cloud sequence. Unfortunately, ground truth was not available for this dataset, therefore we could only verify the results by observation. The results are shown in Fig. 5.10(b), and demonstrate that even when the point clouds are fairly sparse and inaccurate, our simple algorithm can track reasonably well. The only significant error in this algorithm can be seen in the left wrist, whose joint performs some unnatural rotations. This problem arises because our model does not constrain the rotations of the joints, and this can become a problem with noisy data. Introducing such a joint rotation constraint is a subject of future work.

5.6 Related Work

Because articulated models are used widely in animation and tracking applications, there has been work in the past to automate their acquisition. For example, many applications can take a predefined skeleton, and update the parameters of the individual joints using image sequences [18, 78] or 3D data [48, 1]. The work of Taycher *et al.* [115] can estimate tree-shaped skeleton models, but assumes that the rigid parts and their transformations are provided. The work of Song *et al.* [110] demonstrates recovery of articulated human models from tracked 2D features in video streams. These models are represented in terms of decomposable triangulated graphs — a limited class of graphs, unsuitable for representing 3D shapes and articulation in 3D. Additionally, recovering articulation in 2D is considerably more challenging, due to the information loss arising from the projection of a 3D scene to 2D. As a consequence, the models recovered using such procedures tend to be very sparse (containing about a dozen points and triangles), and are fairly far from being realistic human models [110].

Our approach is most directly related to the work of Cheung *et al.* [25], which shows

how to estimate articulated object models from 3D Shape-From-Silhouette (SFS) data, augmented with information about object color. They report recovering an articulated human model with 9 parts. However, their algorithm was applied only to sequences where a single body part is moving at a time. Each sequence contains two articulated parts, and allows the estimation of a single human joint. The final articulated model is generated by combining the joints estimated in all the two-part sequences. Despite the important cue of color information, they did not demonstrate simultaneous recovery of multiple parts.

We believe that the reason for this limitation is the presence of local maxima in their approach, arising for two reasons. First, they solve for the point correspondences between the input meshes while solving for the articulated model. The approach is a generalization of the *Iterative Closest Point (ICP)* algorithm [13] to multiple rigid parts. However, ICP is known to be prone to local maxima (see a discussion of the problems with ICP in Chapter 2). The additional degrees of freedom provided by the possible part decompositions make the problem more severe. By contrast, we take a two-phase approach, first solving the correspondence problem using a non-rigid registration technique that allows large deformations, and then learning the articulated object model. Our approach has the potential limitation of ignoring information about coherent part motion in solving the registration problem. Nevertheless, its ability to circumvent many local maxima appears to significantly offset that potential disadvantage.

A second source for local maxima arises from the choice of constraints enforcing that parts are contiguous regions of the object surface. The approach of Cheung *et al.* [25] enforces part contiguity with discrete constraints between assignments to mesh points and their neighbors. This type of model does not allow the application of efficient global optimization steps. By contrast, our algorithm enforces part contiguity using soft probabilistic constraints, which allow us to violate these constraints locally as long as it is maximizing the log-likelihood of the model as a whole. Moreover, we can apply efficient global optimization methods to determine the optimal part decomposition. These two properties allow us to be less sensitive to initialization, and to avoid local maxima even if a large number of parts is present.

Our algorithm is also related to segmentation approaches, designed to partition image (or video) input from a scene into coherent regions. Standard segmentation approaches

compute local features at all image points, and cluster these features to obtain a segmentation [122, 87, 76]. While good image segmentations usually contain spatially contiguous clusters, they do not usually correspond to a highly distinctive group of feature vectors, and can be missed by the above feature-space clustering approaches. Some early methods attempt to enforce a measure of spatial contiguity by using a strategy called superpixels. They perform a finely-grained initial segmentation, and compute feature vectors directly from these regions [96, 41]. This can be problematic if the superpixels themselves cross boundaries of objects, and still exhibits the original problem because clustering in superpixel feature space may not produce contiguous regions.

Expectation-Maximization methods have been applied to the problem of clustering with spatial constraints. The technique proposed by Weiss and Adelson [126] appears to be one of the first such approaches. More recently, Markov Random Fields have been applied to the problem as well (e.g., [77, 132]). Most of the existing approaches use soft class membership in the E-step, however this causes the inference task to become intractable. This has forced the use of approximate inference methods such as mean field approximations [126], Iterated Conditional Modes [132] and Hidden Markov Measure Fields [77]. In contrast, the algorithm presented in this chapter computes the *maximum-a-posteriori* estimate of the E-step part labels. This allows inference to be performed with the multiway graph-cuts algorithm, which is very efficient and produces solutions whose score is within a factor of the optimal score. The algorithm most similar to our approach is the work of Zabih and Kolmogorov [130] on image segmentation. Their method, developed at the same time as ours, models spatial image contiguity with pairwise associative potentials. The image segmentation is obtained using an EM-algorithm, whose E-step uses a graph cut algorithm to segment the image. The difference between the two methods is mainly in the model optimized in the M-step: our algorithm is designed to segment 3D objects into rigid parts, while theirs focuses on 2D image segmentation.

5.7 Conclusion

We describe an algorithm which automatically recovers articulated object models given a set of registered 3D meshes of the object in different configurations. The algorithm iteratively estimates the part assignments for all points on the template surface, and the rigid transformations of all object parts. Once the part assignments are recovered, the joints are estimated by articulation constraints. We apply the algorithm to three challenging real-world datasets, containing a large number of parts, deforming parts, or both. We demonstrate that the algorithm can recover complex articulated models even from a small number of example meshes, and that it easily scales to large datasets as well. In all experiments, our algorithm not only recovers the parts and joints, but also figures out the optimal number of parts automatically. For the first two datasets (puppet and arm), the articulated models were constructed completely automatically, starting with the original scans, and without making any object-specific assumptions.

In our approach, we have decoupled the registration algorithm from the algorithm which recovers the articulated object structure. While ideally both steps could be executed simultaneously, this decoupling allows us to apply robust global inference strategies during the registration process (e.g., the Correlated Correspondence algorithm) and during the inference step partitioning the object surface into parts. The ability to perform robust and efficient global inference is very important, because it helps us to circumvent many local maxima during both steps. Our approach can be bootstrapped in a fairly straightforward way to use the computed rigid parts and their transformations to improve the registration. However, there was little to be gained from such bootstrapping on these data sets given the quality of our initial registration results.

There are many interesting directions in which this work can be extended. For example, it would be interesting to automatically learn a model of the allowable deformations at different joints, and to incorporate these joint limits in our markerless motion capture application. Probably the most compelling extension would be to model the deformations that the object parts undergo, in addition to their rigid motion. This extension would allow us to obtain realistic-looking shapes, and is addressed in the next chapter of this thesis.

Chapter 6

Learning Deformable Models of Human Shape

In this chapter, we describe a data-driven method for constructing a deformable model of the human body. Such a model, which spans deformations due to changes in both human pose and physique, has a variety of possible uses. It can be used for producing compelling animations of different realistic characters in computer games and movies. It can also be used for automatic tracking and analysis of the movement of different people, with applications to clinical diagnosis and rehabilitation. While human bodies will be our particular example, we attempt to keep the methodology general enough so that many other shapes can be modeled in a similar manner.

Learning a complex deformable model from range scans presents several difficult challenges. The fundamental tasks of scan registration and articulated model acquisition have already been explored in depth in Chapter 4 and Chapter 5 of this thesis, respectively. The main focus of this chapter is on modeling deformation. The deformation space which we are interested in modeling is complex — we come in a variety of sizes and girths, can be male and female, and our bodies themselves deform as a result of changing our pose or flexing our muscles. This raises the following interesting question: what is a suitable shape representation that can capture all this variation?

In the previous chapter on articulated models, we represented shape in terms of a collection of rigid body parts, connected by joints. While the resulting models are compact

and easy to use, they lack the power to model muscle deformations, and changes in shape due to different body physiques. On the other hand, the deformable models we presented in Chapter 2 and Chapter 4 are based on the appearance of a single shape template. In these models, each link is allowed to deform independently of the others, and as a result, they cannot capture the correlations between the deformation of different parts on the surface. Some examples of correlations that we would like to capture are that long legs probably mean long arms, bodies tend to be symmetric, and lifting the arm causes the pectoral muscles to stretch.

The problem of modeling human bodies from examples has already received significant attention in the computer graphics literature. Recent approaches represent the deformation of an example shape in terms of the displacements of its points from some generic template shape. For example, approaches that model deformations due to changes in pose [70, 108, 1, 123, 80] represent deformation as point displacements relative to an underlying articulated model. On the other hand, approaches that model body shape deformation across different humans [2, 102] compute point displacements relative to an average shape. Unfortunately, it is difficult to combine two of the above approaches in order to obtain an integrated model of human pose *and* body shape variation (which may explain why no such model had been proposed up to this point). The main challenge lies in finding a good way to combine two distinct deformation models based on point displacements. Point displacements are vectors and cannot be multiplied in a meaningful way. Adding them ignores an important notion of scale — pose displacements learned on a large individual cannot be added to the shape of a small individual without undesirable artifacts.

In this chapter, we introduce the SCAPE method (Shape Completion and Animation of PEople) — a data-driven method for building a human shape model. Our model is based on a representation of deformation that allows us to model pose and body shape deformation separately, and combine them in a natural way to produce 3D surface models with realistic muscle deformations of different people in different poses.

The pose deformation component of our model is acquired from a set of dense 3D scans of a single person in multiple poses. A key aspect of our pose model is that it decouples deformation into a rigid and a non-rigid component. The rigid component of deformation

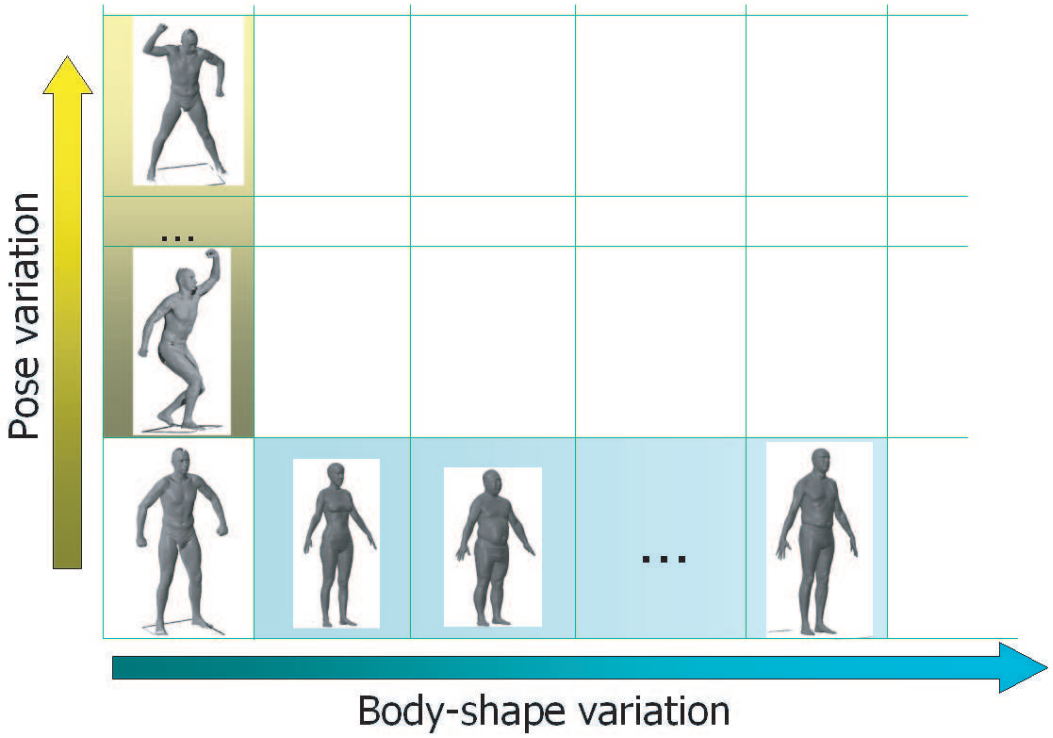


Figure 6.1: Human shape deformations can be decomposed along two axes, corresponding to variations due to pose and body shape. These can be arranged in a matrix as shown. We use examples from a column and a row from this matrix to train separate models of pose and body shape deformation. The two models can be combined, which allows us to generate shapes corresponding to various people in various poses.

is described in terms of a low degree-of-freedom rigid body skeleton. The non-rigid component captures the remaining deformation such as flexing of the muscles. In our model, the deformation for a body part is dependent only on the adjacent joints. Therefore, it is relatively low dimensional, allowing the shape deformation to be learned automatically, from limited training data.

Our representation also models shape variation that occurs across different individuals. This model component can be acquired from a set of 3D scans of different people in different poses. The shape variation is represented by using principal component analysis (PCA), which induces a low-dimensional subspace of body shape deformations. Importantly, the

model of shape variation does not get confounded by deformations due to pose, as those are accounted for separately. The two parts of the model form a single unified framework for shape variability of people. The framework can be used to generate a realistic complete surface mesh given only a succinct specification of the desired shape — in our case, 33 angles of the human skeleton and 20 eigen-coefficients describing the body shape.

We apply our model to two important graphics tasks. The first is partial view completion. Most scanned surface models of humans have significant missing regions. Given a partial mesh of a person for whom we have no previous data, our method finds the shape that best fits the observed partial data in the space of human shapes. The model can then be used to predict a full 3D mesh. Importantly, because our model also accounts for non-rigid pose variability, muscle deformations associated with the particular pose are predicted well even for unobserved parts of the body.

The second task is producing a full 3D animation of a moving person from marker motion capture data. We approach this problem as a shape completion task. The input to our algorithm is a single scan of the person and a time series of extremely sparse data — the locations of a limited set of markers (usually between 50 and 60) placed on the body. For each frame in the sequence, we predict the full 3D shape of the person, in a pose consistent with the observed marker positions. Applying this technique to sequences of motion capture data produces full-body human 3D animations. We show that our method is capable of constructing high-quality animations, with realistic muscle deformation, for people of whom we have a single range scan.

The rest of this chapter proceeds as follows. In Sec. 6.1 we briefly describe our pipeline for acquisition and pre-processing of the example meshes which are used to train our models. In Sec. 6.2 we show how human body deformations are represented and learned in our model. Finally, in Sec. 6.3, we demonstrate how our model can be applied for compelling shape-completion applications, including partial view completion and producing 3D animations from marker motion capture sequences.

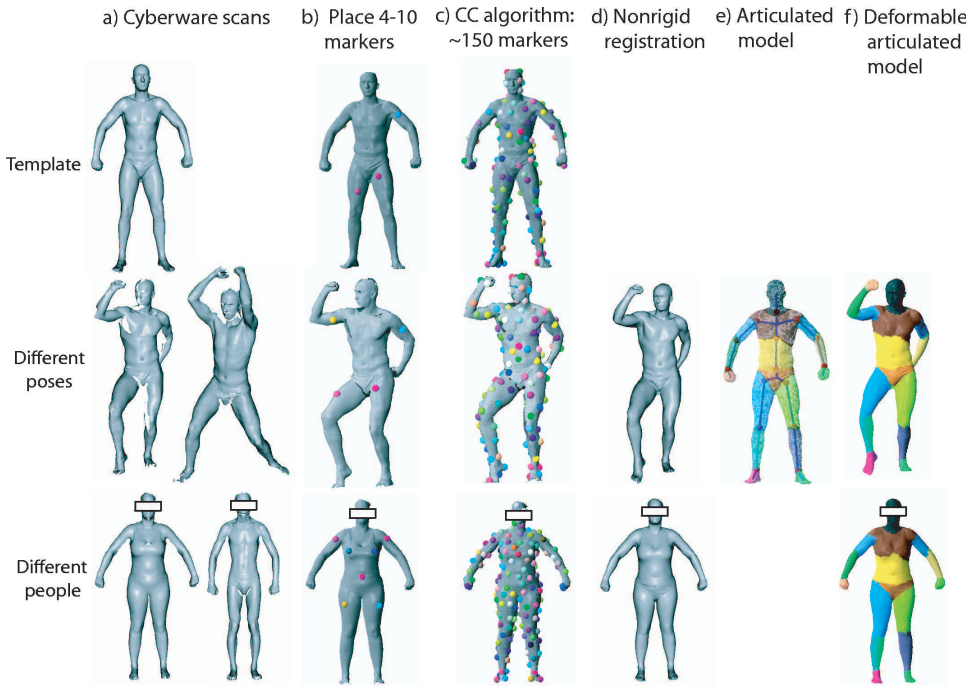


Figure 6.2: The mesh processing pipeline used to generate our training set. (a) We acquired two data sets spanning the shape variability due to different human poses and different physiques. (b) We select a few markers by hand, mapping the template mesh and each of the range scans. (c) We apply the Correlated Correspondence algorithm, which computes numerous additional markers. (d) We use the markers as input to a non-rigid registration algorithm, producing fully registered meshes. (e) We apply a skeleton reconstruction algorithm to recover an articulated skeleton from the registered meshes. (f) We learn the space of deformations due to pose and physique.

6.1 Data Acquisition and Preprocessing

The SCAPE model acquisition is data driven, and all the information about the shape is derived from a set of range scans. This section describes the basic pipeline for data acquisition and pre-processing of the data meshes. This pipeline, displayed in Fig. 6.2, consists largely of a combination of previously published methods. The specific design of the pipeline is inessential for the main contribution of this chapter; however, we demonstrate that most of the processing in the pipeline can be done using methods presented in this thesis, which minimize the need for human involvement.

Range Scanning We acquired our surface data using a Cyberware WBX whole-body scanner. The scanner captures range scans from four directions simultaneously and the models contain about 200K points. We used this scanner to construct full-body instance meshes by merging the four scan views [33] and subsampling the instances to about 50,000 triangles [47]. Using the process above, we obtained two data sets : a *pose data set*, which contains scans of 70 poses of a particular person in a wide variety of poses, and a *body shape data set*, which contains scans of 37 different people in a similar (but not identical) pose. We also added eight publicly available models from the CAESAR data set [2] to our data set of individuals.

We selected one of the meshes in the pose data set to be the *template mesh*; all other meshes will be called *instance meshes*. The function of the template mesh is to serve as a point of reference for all other scans. The template mesh is hole-filled using an algorithm by Davis *et al.* [35]. In acquiring the template mesh, we ensured that only minor holes remained mostly between the legs and around the armpits. The template mesh and some sample instance meshes are displayed in Fig. 6.2(a). Note that the head region is smoothed in some of the figures, in order to hide the identity of the scan subjects; the complete scans were used in the learning algorithm.

Correspondence The next step in the data acquisition pipeline brings the template mesh into *correspondence* with each of the other mesh instances. Current non-rigid registration algorithms require that a set of corresponding markers between each instance mesh and the template is available (the work of Allen *et al.* [2] uses about 70 markers for registration). We obtain the markers using the Correlated Correspondence algorithm (Chapter 4). The CC algorithm computes the consistent embedding of each instance mesh into the template mesh, which minimizes deformation, and matches similar-looking surface regions. To break the scan symmetries, we initialize the CC algorithm by placing 4–10 markers by hand on each pair of scans. The result of the algorithm is a set of 140–200 (approximate) correspondence markers between the two surfaces, as illustrated in Fig. 6.2(c).

Non-rigid Registration Given a set of markers between two meshes, the task of non-rigid registration is well understood and a variety of algorithms exist [1, 52, 111]. The task is to bring the meshes into close alignment, while simultaneously aligning the markers. We apply the non-rigid ICP algorithm described in Chapter 2 to register the template mesh

with all of the meshes in our data set. As a result, we obtain a set of meshes with the same topology, whose shape approximates well the surface in the original Cyberware scans. Several of the resulting meshes are displayed in Fig. 6.2(d).

Recovering the Articulated Skeleton As discussed in the introduction, our model uses a low degree-of-freedom skeleton to model the articulated motion. We construct a skeleton $\mathcal{S}^X$ for our template mesh automatically, using only the meshes in our data set. We use the algorithm from Chapter 5, which automatically recovers a decomposition of the object into approximately rigid parts, the location of the parts in the different object instances, and the articulated object skeleton linking the parts. The experiment was already discussed at length in that chapter, and the results are shown in Fig. 5.8. As discussed, we obtain an articulated model with 18 parts, which is displayed in Fig. 5.8(2c). The algorithm broke both the crotch area and the chest area into two symmetric parts, resulting in a skeleton which was not tree-structured. To facilitate pose editing, we combined the two parts in each of these regions into one. The result was a tree-structured articulated skeleton with 16 parts, displayed in Fig. 6.2(e).

Data Format and Assumptions As a result of our preprocessing, we obtain a data set consisting of a model mesh X and a set of instance meshes $\mathcal{M}^Y = \{\mathcal{M}^{Y_1}, \ldots, \mathcal{M}^{Y_N}\}$. All of these meshes have the same set of points and triangles as the model mesh, albeit in different configurations. We also compute the alignments of the rigid parts of our model $\mathcal{S}^X$ to all the mesh instances. For each part p, we compute its rigid alignment T_p^i in instance i, simply by using the known point correspondences in the meshes. The rigid alignment for each part is computed separately using the alignment method originally proposed by Besl *et al.* [13] and described in Sec. 2.3. In computing the alignment, the joint constraints between the parts are ignored. As will be seen shortly, we will only be using the rotation component R_p^i of these transformations for our model of human deformation.

The data acquisition and pre-processing pipeline provides us with exactly this type of data; however, any other technique for generating similar data will also be applicable to our learning and shape completion approach.

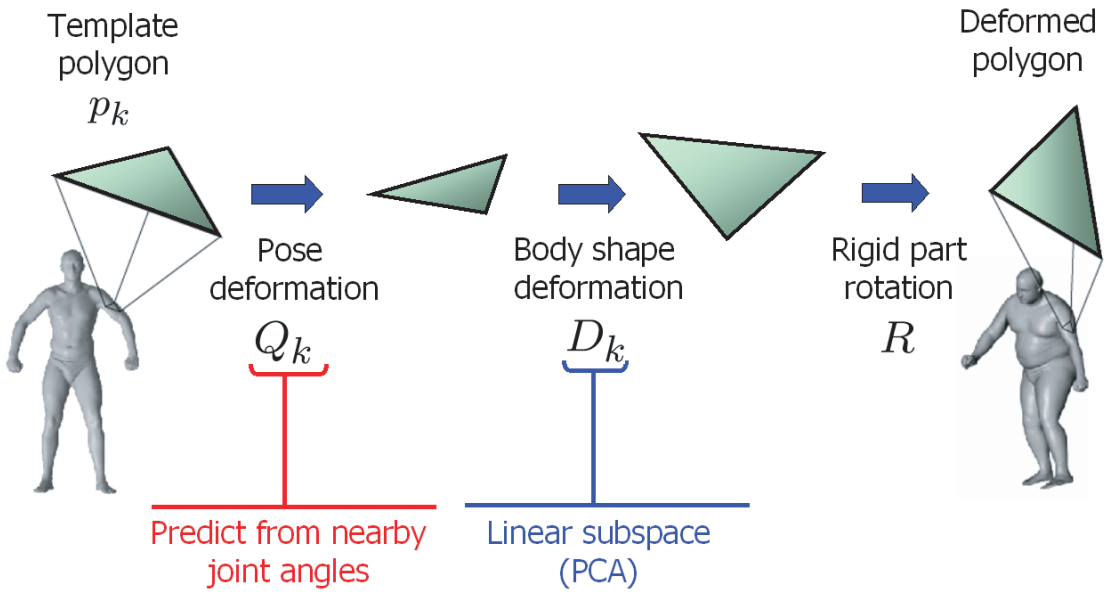


Figure 6.3: Overview of the SCAPE model. We model how template triangles p_k get transformed to generate a new human shape instance. Our model accounts for several sources of triangle deformation: non-rigid muscle deformations Q_k due to changes in pose, non-rigid body shape deformations D_k accounting for changes between different humans, and articulated part rotations R. The transformations are represented as 3×3 matrices and are applied in order, which preserves proper deformation scaling. Our model correlates the values of matrices Q_k to the nearby joint angles, and leans a low-dimensional linear manifold for the values of matrices D_k.

6.2 Human Shape Model

This section describes our model of human shape, in which deformations due to changes in pose and body shape are modeled separately. First, we show how these models can be combined to produce complete surface meshes of different people in different poses. Then, we describe in detail how to represent and learn the two models, accounting for pose and body shape variation.

6.2.1 Model Overview

We want to model the deformations which align the template $\mathcal{M}^X$ with each mesh $\mathcal{M}^{Y_i}$ in the data set, which corresponds to some pose of a particular human. We will model the deformations for each triangle p_k of the template in a way, which was inspired by the work of Sumner and Popović [111] on mesh deformation transfer. In our model, we will account for the polygon deformations arising from three separate sources:

1. Rigid transformations R resulting from placing the articulated model in a different pose.

2. Non-rigid transformations Q resulting from changes in pose, such as bulging of the muscles, and sticking out of elbows.

3. Non-rigid transformations D accounting for changes in body shape between different individuals.

The above transformations are modeled in terms of 3×3 matrices, which are applied in order to transform the template triangle into its counterpart in mesh instance $\mathcal{M}^{Y_i}$. Let triangle p_k contain the points $(x_{k,1}, x_{k,2}, x_{k,3})$. We apply our deformations in terms of the triangle's local coordinate system, obtained by translating point $x_{k,1}$ to the global origin. Thus, the deformations will be applied to the triangle edges $\widehat{v}_{k,j} = x_{k,j} - x_{k,1}$, $j = 2, 3$.

First, we apply a 3×3 linear transformation matrix Q_k^i to the triangle. This matrix, which corresponds to a non-rigid pose-induced deformation, is specific to each triangle p_k and each pose $\mathcal{M}^{Y_i}$. Then, we apply a linear transformation matrix D_k^i, which is also triangle-specific, and accounts for deformations between the shape of different individuals. The deformed triangle is then rotated by $R_{b[k]}^i$, the rotation of its rigid part in the articulated skeleton. Here, $b[k]$ denotes the body part associated with triangle p_k. The same rotation will be applied to all triangles that belong to that part. Combining the three transformation matrices, we write:

$$v_{k,j}^i = R_{b[k]}^i D_k^i Q_k^i \widehat{v}_{k,j}, \quad j = 2, 3. \tag{6.1}$$

The triangle deformation process is sketched in Fig. 6.3. A key feature of this model is that it combines an element modeling the movement of the rigid skeleton, with an element

that allows for arbitrary local deformations. Importantly, the application of consecutive transformation matrices to each triangle maintains proper scaling of pose and body shape deformation.

The order in which the matrices are applied matters. Multiplying a vector $\widehat{v}$ by a linear matrix produces different results depending on the direction of the vector. Let's examine the order of deformation $Q_k D_k \widehat{v}_{k,j}$. If $D_k \widehat{v}_{k,j}$ is a vector with a significantly different direction than $\widehat{v}_{k,j}$, the effect of applying matrix Q_k to $D_k \widehat{v}_{k,j}$ is different from applying Q_k to $\widehat{v}_{k,j}$. This is undesirable, as it means that changes in body shape influence the effects of the pose model. Because it is advantageous to multiply the vector $\widehat{v}_{k,j}$ first by the matrices that change its direction least, and body shape deformations are usually orders of magnitude larger than pose deformations, we chose the order $D_k Q_k$ in the paper. At this stage, we do not have conclusive experimental evidence as to which matrix order yields better results. The rotation matrix $R_{b[k]}$, associated with the articulated model orientation, can change the direction of a vector it multiplies arbitrarily much, therefore it is important for $R_{b[k]}$ to be the last one in the multiplication order.

Given a set of transformation matrices Q, D for all template triangles, and the rotations of all the articulated parts R, our method's predictions can be used to synthesize a mesh for that pose. For each individual triangle, our method makes a prediction for the edges of p_k as $R_{b[k]} D_k Q_k \widehat{v}_{k,j}$. In general, the predictions for the edges in adjacent triangles are not consistent. We solve for the set of point locations $y_1, \ldots, y_N$ that minimize the reconstruction error for the predicted triangle edges:

$$\arg\min_{y_1,\ldots,y_N} \sum_k \sum_{j=2,3} \| R^i_{b[k]} D^i_k Q^i_k \widehat{v}_{j,k} - (y_{j,k} - y_{1,k}) \|^2 \tag{6.2}$$

Note that, as translation is not directly modeled, the problem has a translational degree of freedom. By anchoring one of the points y (in each connected component of the mesh) to a particular location, we can make the problem well-conditioned, and reconstruct the mesh.[1] The objective can be decomposed along each dimension of the points y, resulting in three separate least-squares problems. In particular, let the column vectors $\mathbf{y}_x, \mathbf{y}_y, \mathbf{y}_z$ contain the

[1] See the paper of Sumner and Popović [111] for a very related discussion on mesh reconstruction from a set of deformation matrices. In their reconstruction formulation, the reconstruction error is defined over the transformation matrices themselves, which results in a much larger least-squares problem.

x, y and z coordinates of the points $\mathcal{V}^y$, respectively. The objective can be easily expressed in terms of three separate problems of the following form:

$$\arg\min_{\mathbf{y}_x} \|A\mathbf{y}_x - b_x\|^2, \quad \arg\min_{\mathbf{y}_y} \|A\mathbf{y}_y - b_y\|^2, \qquad \arg\min_{\mathbf{y}_z} \|A\mathbf{y}_z - b_z\|^2. \tag{6.3}$$

Importantly, the sparse matrix A is the same for all three subproblems; furthermore, the values of A does not depend on the values of D, Q, or R. The vectors b_x, b_y, b_z are the only quantities above that depend on the values of the transformation matrices D, Q and R.

In order to solve the above subproblems, we need to compute

$$\mathbf{y}_x = (A^T A)^{-1} A^T b_x, \quad \mathbf{y}_y = (A^T A)^{-1} A^T b_y, \quad \mathbf{y}_z = (A^T A)^{-1} A^T b_z \tag{6.4}$$

These three subproblems require performing the same matrix inversion $(A^T A)^{-1}$, which needs to be computed only once. Computing the inverse, using the sparse matrix package *umfpack*, takes approximately 1 second on a 2.4 GHz Intel Xeon processor for meshes consisting of 25K polygons.

Once the inverse has been computed, we can animate in close to real time, because a change in the matrices R, D and Q only induces a change in the vectors b_x, b_y, b_z. The new shape can be computed by multiplying those vectors with the precomputed matrix $(A^T A)^{-1} A^T$. Thus, our model can be used for real-time animation of synthesized or cached motion sequences.

6.2.2 Pose Deformation Model

So far, we showed that given a set of transformation estimates Q and D, we can reconstruct complete human meshes. Now we show how we can learn deformation models, which can provide these estimates.

Learning the Model

We showed how to model pose-induced deformations using a set of matrices Q_k^i for the template triangles p_k. We want to predict these deformations from the articulated human pose, which is represented as a set of relative joint rotations. We learn a regression function

for each triangle p_k which predicts the transformation matrices Q_k^i as a function of the rotations of its two nearest joints $\triangle r^i_{b[k],1}$ and $\triangle r^i_{b[k],2}$. These two joints can be grouped to form the 6-dimensional vector $\triangle r^i_{b[k]} = (\triangle r^i_{b[k],1}, \triangle r^i_{b[k],2})^T$. By assuming that a matrix Q_k^i can be predicted in terms of these two joints only, we greatly reduce the dimensionality of the learning problem.

The joint rotations are computed easily from the absolute rotation matrices of the two rigid parts adjacent to that joint. If those rotations are R_i and R_j, then the relative joint rotation matrix is simply $\triangle R_{i,j} = R_i^T R_j$. Joint rotations are conveniently represented with their exponential map coordinates (see Sec. 2.2.1). Briefly, let M denote any 3×3 rotation matrix, and let m_{ij} be its entry in i-th row and j-th column. The exponential map t for the joint angle is a 3D vector, and can be computed from the following formula [75]:

$$t \;=\; \frac{\theta}{2\sin(\theta)} \begin{bmatrix} m_{32} - m_{23} \\ m_{13} - m_{31} \\ m_{21} - m_{12} \end{bmatrix}, \quad \theta = \cos^{-1}\left(\frac{\mathrm{tr}(M) - 1}{2}\right).$$

The direction of the exponential map t represents the axis of rotation, and its magnitude represents the rotation angle about that axis. We denote the exponential map parameters of a joint rotation matrix $\triangle R$ as $\triangle r$.

Each joint rotation can be thus specified using three parameters, so altogether we are predicting from the vector $\triangle r^i_{b[k]}$, which has six parameters. Adding a term for the constant bias, we associate a 7×1 regression vector $\mathbf{a}_{k,lm}$ with each of the 9 values of the matrix Q, and write:

$$q^i_{k,lm} = \mathbf{a}^T_{k,lm} \cdot \begin{bmatrix} \triangle r^i_{b[k]} \\ 1 \end{bmatrix} \qquad l, m = 1, 2, 3 \tag{6.5}$$

Thus, for each triangle p_k, we have to fit 9×7 entries $\mathbf{a}_k = (\mathbf{a}_{k,lm} \;:\; l, m = 1, 2, 3)$. The reconstruction of the entire matrix is denoted as $Q^i_k = \mathcal{Q}_{\mathbf{a}_k}(\triangle r^i_{b[k]})$, where $\mathcal{Q}_{\mathbf{a}_k}(\triangle r)$ is a shorthand for the operation which predicts all the entries of a matrix Q from the rotation parameters $\triangle r$, and arranges them appropriately.

Our goal now is to learn these parameters $a_{k,lm}$. If we are given the transformation Q^i_k for each instance $\mathcal{M}^{Y_i}$ and the rigid part rotations R^i, solving for the regression values

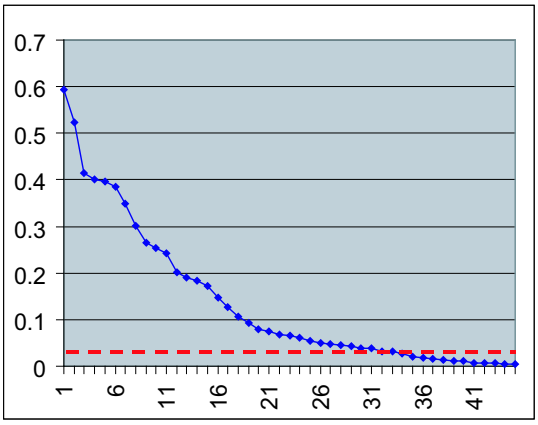


Figure 6.4: A plot of the eigenvalues obtained by performing PCA on the joint angles of the human pose examples. Each joint contributes three different eigenvalues, whose magnitude describes the amount of joint rotation, and whose associated eigenvectors describe the joint rotation axes. The red line shows our chosen cut-off point for ignoring rotational degrees of freedom, for which observed rotation was small.

(using a quadratic cost function) is straightforward. It can be carried out for each triangle k and matrix value $q_{k,lm}$ separately:

$$\arg\min_{\mathbf{a}_{k,lm}} \sum_i \left([\triangle r^i \;\; 1]\mathbf{a}_{k,lm} - q^i_{k,lm}\right)^2 . \tag{6.6}$$

In practice, we can save on model size and computation by identifying joints which have only one or two degrees of freedom. Allowing those joints to have three degrees of freedom can also cause overfitting in some cases. We performed PCA on the observed angles of the joints $\triangle r^i$, removing axes of rotation whose eigenvalues are smaller than 0.03. The associated entries in the vector $\mathbf{a}_{k,lm}$ are then not estimated. The value 0.03 was obtained by observing a plot of the sorted eigenvalues (Fig. 6.4). We found that the pruned model minimally increased cross-validation error, while decreasing the number of parameters by roughly one third.

To train our pose model, we only use the instance meshes $\mathcal{M}^{Y_i}$ which correspond to different poses of the human in the template (hence, all matrices D^i will be identity and will be ignored in the discussion below). The rigid part rotations, and hence the joint rotations

are computed as part of our preprocessing step. Unfortunately, the transformations Q_k^i for the individual triangles are not known. We estimate these matrices by fitting them to the transformations observed in the data. However, the problem is generally underconstrained. We follow Sumner *et al.* [111] and Allen *et al.* [2], and introduce a smoothness constraint which prefers similar deformations in adjacent polygons that belong to the same rigid part. Specifically, we solve for the correct set of linear transformations with the following equation for each mesh $\mathcal{M}^{Y_i}$:

$$\begin{aligned} \underset{\{Q_1^i,\ldots,Q_P^i\}}{\arg\min} \quad & \sum_k \sum_{j=2,3} \|R_{b[k]}^i Q_k^i \widehat{v}_{k,j} - v_{k,j}^i\|^2 + \\ & w_s \sum_{k_1,k_2 \text{ adj}} I(b[k_1] = b[k_2]) \cdot \|Q_{k_1}^i - Q_{k_2}^i\|^2, \end{aligned} \tag{6.7}$$

where $w_s = 0.001\rho$ and ρ is the resolution of the model mesh X. Above, $I(\cdot)$ is the indicator function. The equation can be solved separately for each rigid part and for each row of the Q matrices.

Given the estimated Q matrices, we can solve for the (at most) 9×7 regression parameters $\mathbf{a}_k$ for each triangle k, as described in Eqn. (6.6).

Application to Our Data Set

We applied this method to learn a SCAPE pose deformation model using 65 training instances from our pose data set. The learned model contains 33 free joint angle parameters (out of possible 45). Fig. 6.5 shows examples of meshes that can be represented by our learned model. Note that these examples do not correspond to meshes in the training data set; they are new poses synthesized completely from a vector of joint rotations R, using Eqn. (6.5) to define the Q matrices, and Eqn. (6.2) to generate the mesh.

The model captures well the shoulder deformations, the bulging of the biceps and the twisting of the spine. It deals reasonably well with the elbow and knee joints, although example (k) illustrates a small amount of elbow smoothing that occurs in some poses. The model exhibits an artifact (flat surface) in the armpit area (l), which is caused by hole-filling in the template mesh.

Figure 6.5: Examples of muscle deformations that can be captured in the SCAPE pose model.

Comparison to Non-linear Regression

Our simple regression performs remarkably well in reconstructing the body shape for a large variety of poses. However, it is a model with limited expressivity, and may produce visibly suboptimal shapes in some cases. Intuitively, these cases should include some extreme angle configurations, which are close to the joint limits. We examined all the reconstructions produced by our model, and found such an instance, in which the arm was close to the body, and twisted about 45 degrees. This instance was already displayed in Fig. 6.5(l), and exhibits an unnatural widening of the shoulders.

This instance motivated us to compare our results to a more complicated regression model, which can capture non-linear deformation effects. Many model choices are possible, but we picked Support Vector Machine (SVM) regression. For detailed description of the regression model, please refer to the tutorial of Schölkopf and Smola [101]. Just like in the linear case, we learn predictors $f(\cdot)$, which map joint angles to the values of the matrices Q:

$$q^i_{k,lm} = f\left(\triangle r^i_{b[k]}\right) \qquad l, m = 1, 2, 3. \tag{6.8}$$

In our implementation, this mapping is done in non-linear feature space, using a radial basis function kernel $K(x, x') = e^{-\frac{\|x-x'\|^2}{2\gamma^2}}$ with a setting of $\gamma = 2.5$. We use a separate regressor for each value $q_{k,lm}$, although a solution that regresses on entire groups of parameters is also possible.

Fig. 6.6(a) shows that the non-linear regression model fixes the widening of the shoulders artefact. However, in most cases the difference between the shapes reconstructed by the non-linear and the linear models is hardly visible (Fig. 6.6(b)). Fig. 6.7(a)-(b) show that the non-linear regression model, which has a greater expressive power, produces smaller reconstruction errors (defined in Eqn. (6.2)) on the training set. However, using 5-fold cross-validation, we determined that the non-linear regression model provides only minimal improvements in the prediction for unseen shapes. Fig. 6.7(c)-(d) shows the point distance errors (distance between corresponding points on the ground truth and reconstructed shape) decrease minimally when non-linear regression is used. In those cases, reconstruction errors even tended to slightly increase when non-linear regression was used. The reasons for such limited generalization capability of the non-linear regression model

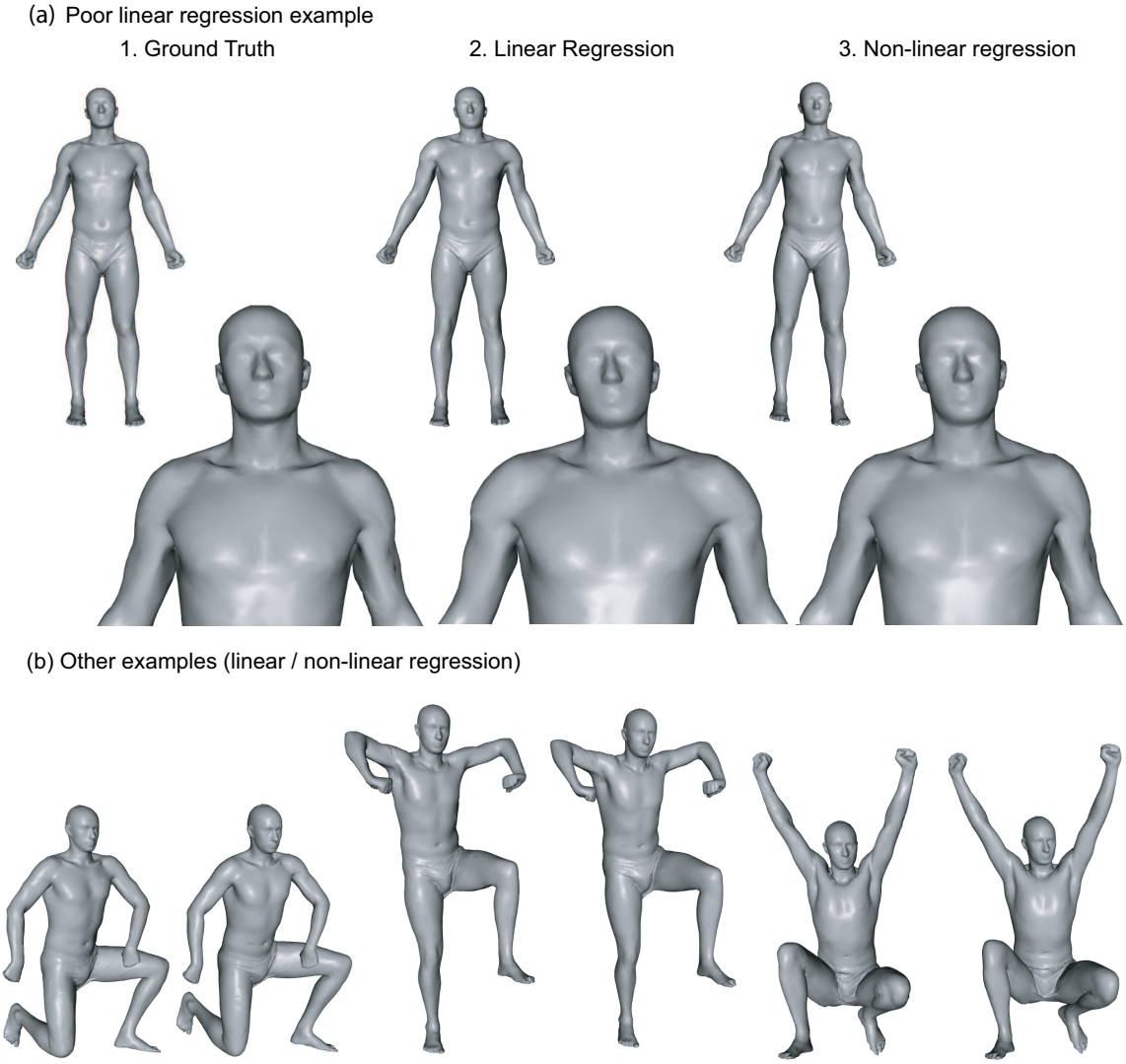
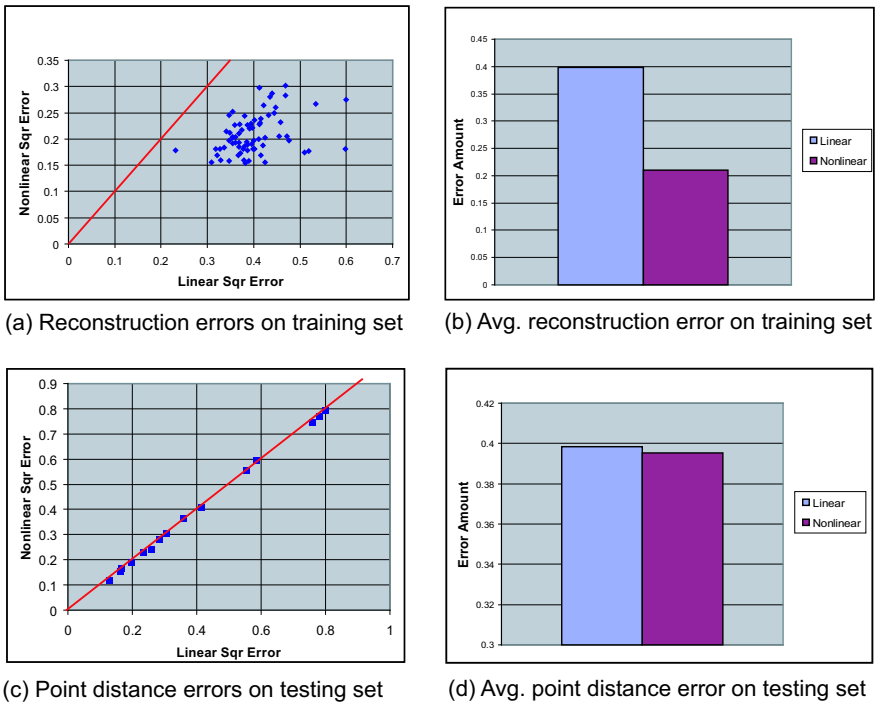


Figure 6.6: Comparison of linear and non-linear pose regression. In this particular pose example (a)-1, the linear regression model produces a ”widening” of the shoulders (a)-2, which is fixed by a non-linear model (a)-3. (b) In the vast majority of reconstructions the difference between the linear and the non-linear regression is hardly visible.



(a) Reconstruction errors on training set

(b) Avg. reconstruction error on training set

(c) Point distance errors on testing set

(d) Avg. point distance error on testing set

Figure 6.7: Numerical comparison of linear and non-linear regression for modeling pose deformation. (a)-(b) Non-linear regression performs noticeably better in decreasing the reconstruction errors on the training set. (c)-(d) Non-linear regression did not contribute to a noticeable improvement in generalizing to new examples.

can be twofold. One reason is that our regression model needs to be trained with more examples. Another, and more likely reason, is that the shape examples used for training and testing have some errors introduced by our data-processing. These data-processing errors prevent the model from achieving higher scores on unseen testing examples.

Overall, for all reconstructions except the one we discussed above, the differences between the linear and non-linear models are barely discernible with a naked eye. Because our linear model results in a formulation which is considerably more efficient to train and to use for the tasks of animation and shape-completion, the discussion in the rest of the chapter will focus on that model.

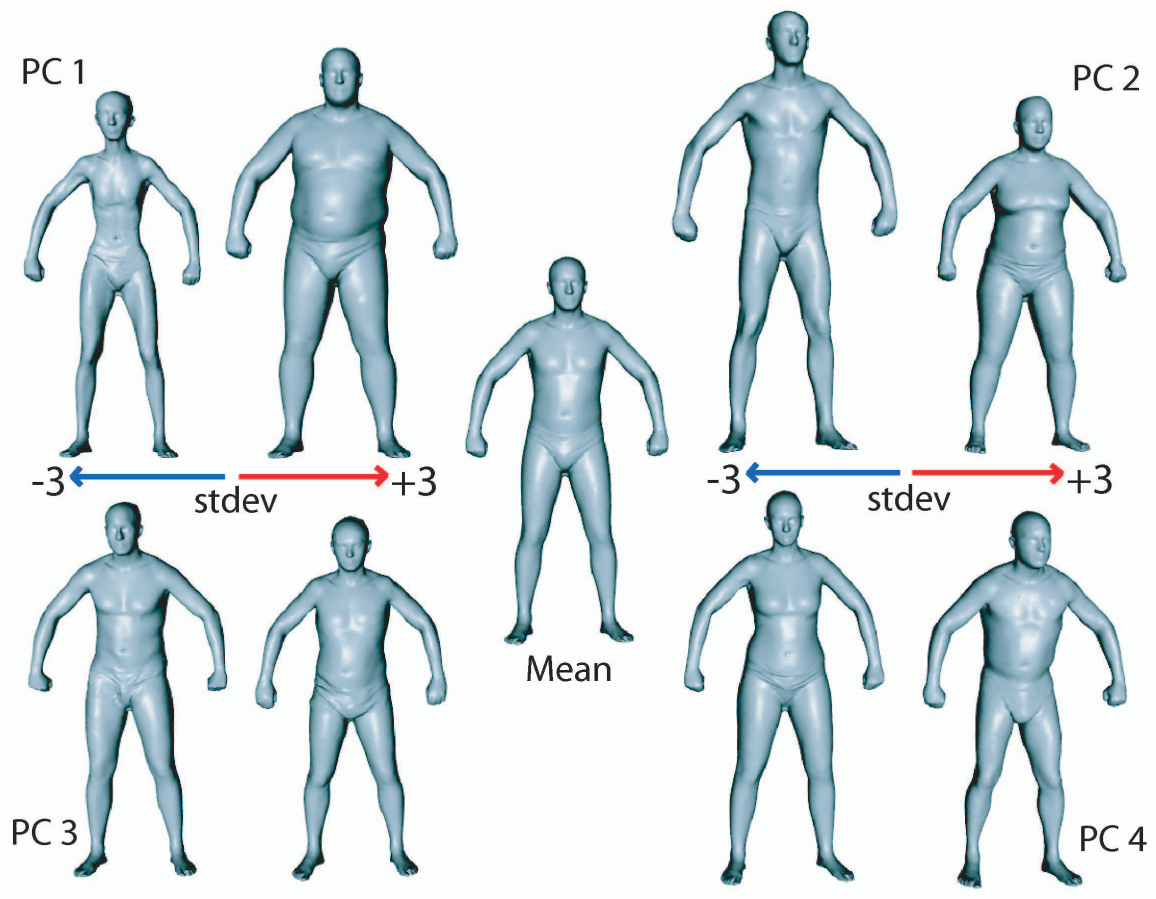


Figure 6.8: The first four principal components in the space of body shape deformation

6.2.3 Body-Shape Deformation

The SCAPE model also encodes variability due to body shape across different individuals in terms of the transformation matrices D^i. We now assume that the scans of our training set $\mathcal{M}^{Y_i}$ correspond to different individuals.

Learning the Model

To map out the space of body shape deformations, we view the different matrices D^i as arising from a lower dimensional subspace. For each example mesh, we create a vector of size $9 \times N$ containing the parameters of matrices D^i. We assume that these vectors are

generated from a simple linear subspace, which can be estimated by using PCA:

$$D^i = \mathcal{D}_{U,\mu}(\beta^i) = \overline{U\beta^i + \mu} \tag{6.9}$$

where $U\beta^i + \mu$ is a (vector form) reconstruction of the $9 \times N$ matrix coefficients from the PCA, and $\overline{U\beta^i + \mu}$ is the representation of this vector as a set of matrices. PCA is appropriate for modeling the matrix entries, because body shape variation is consistent and not too strong. We found that even shapes which are three standard deviations from the mean still look very much like humans (see Fig. 6.8).

If we are given the affine matrices D_k^i for each i, k we can easily solve for the PCA parameters U, μ, and the mesh-specific coefficients β^i. However, as in the case of pose deformation, the individual shape deformation matrices D_k^i are not given, and need to be estimated. We use the same idea as above, and solve directly for D_k^i, with the same smoothing term as in Eqn. (6.7):

$$\arg\min_{D^i} \sum_k \sum_{j=2,3} \|R^i_{b[k]} D^i_k Q^i_k \widehat{v}_{k,j} - v^i_{k,j}\|^2 + w_s \sum_{k_1,k_2 \text{ adj}} \|D^i_{k_1} - D^i_{k_2}\|^2. \tag{6.10}$$

Importantly, recall that our data preprocessing phase provides us with an estimate R^i for the joint rotations in each instance mesh, and therefore the joint angles $\triangle r^i$. From these we can compute the predicted pose deformations $Q_k^i = \mathcal{Q}_{\mathbf{a}_k}(\triangle r^i_{b[k]})$ using our learned pose deformation model. Thus, the only unknowns in Eqn. (6.10) are the shape deformation matrices D_k^i. The equation is quadratic in these unknowns, and therefore can be solved using a straightforward least-squares optimization.

Application to Our Data Set

We applied this method to learn a SCAPE body shape deformation model using the 45 instances in the body shape data set, and taking as a starting point the pose deformation model learned as described in Sec. 6.2.2. We used only the top 20 principal components found in our PCA decomposition of the shape space. Fig. 6.8 shows the mean shape and the variation of the first four principal components. These components represent very reasonable variation in weight and height, gender, abdominal fat and chest muscles, and bulkiness

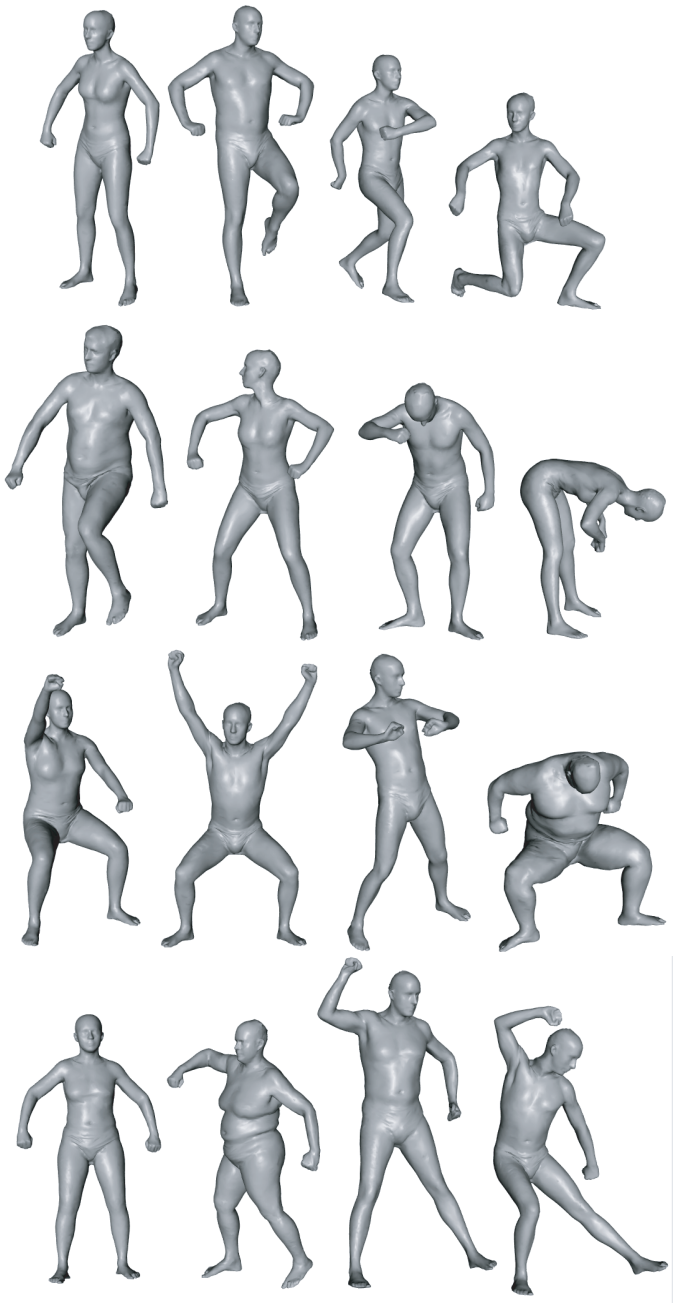


Figure 6.9: A variety of body shapes produced by the SCAPE model. The input to the model is a concise description of the shape in the form of joint angles and PCA body coefficients.

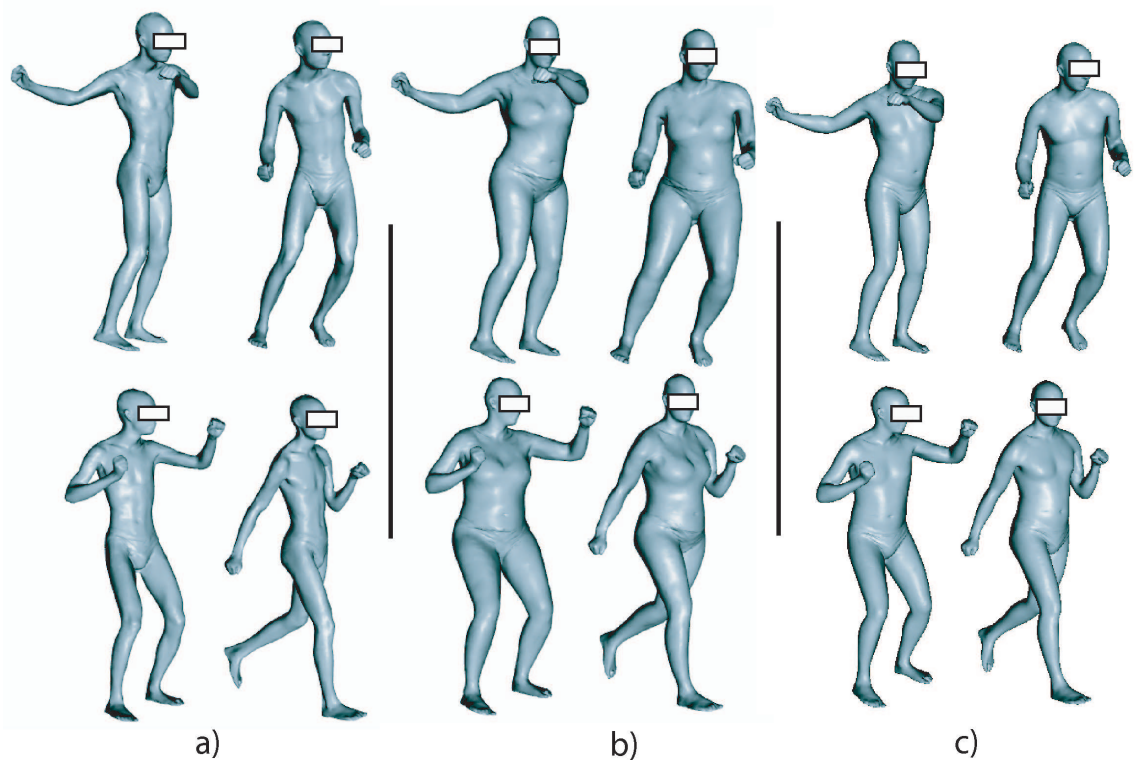


Figure 6.10: Deformation transfer by the SCAPE model. The figure shows three subjects, each in four different poses. Each subject was seen in a single reference pose only

of the chest versus the hips.

Our PCA space spans a wide variety of human body shapes. Put together with our pose model, we can now synthesize realistic scans of various people in a broad range of poses. Assume that we are given a set of rigid part rotations R and person body shape parameters β. The joint rotations R determine the joint angles $\triangle R$. For a given triangle p_k, the pose model now defines a deformation matrix $Q_k = \mathcal{Q}_{\mathbf{a}_k}(\triangle r_{b[k]})$. The body shape model defines a deformation matrix $D_k = \mathcal{D}_{U,\mu}(\beta)$. As in Eqn. (6.2), we solve for the vertices $\mathcal{M}^Y$ that minimize the reconstruction error:

$$E_H[Y] = \sum_k \sum_{j=2,3} \| R_{b[k]} \mathcal{D}_{U,\mu}(\beta) \mathcal{Q}_{\mathbf{a}_k}(\triangle r_{b[k]}) \widehat{v}_{j,k} - (y_{j,k} - y_{1,k}) \|^2 \qquad (6.11)$$

Using this approach, we can generate a mesh for any body shape in our PCA space in any

pose. Fig. 6.9 shows some examples of different synthesized scans, illustrating variation in both body shape and pose. We can also apply pose deformations to shape of subjects, for whom we have a single scan only. This capability, called deformation transfer, is demonstrated in Fig. 6.10. The above figures show that realistic muscle deformation is achieved for very different subjects, and for a broad range of poses.

6.3 Shape Completion

6.3.1 Shape Completion Overview

So far, we have focused on the problem of constructing the two components of our SCAPE model from the training data: the regression parameters $\{\mathbf{a}_k : k = 1, \ldots, M_X\}$ of the pose model, and the PCA parameters U, μ of our body shape model. We now show how to use the SCAPE model to address the task of shape completion, which is the main focus of our work. We are given sparse information about an instance mesh, and wish to construct a full mesh consistent with this information; the SCAPE model defines a prior on the deformations associated with human shape, and therefore provides us with guidance on how to complete the mesh in a realistic way.

Assume we have a set of markers $Z = z_1, \ldots, z_L$ which specify known positions in 3D for some points $x_1, \ldots, x_L$ on the model mesh. We want to find the set of points $\mathcal{M}^Y$ that best fits these known positions, and is also consistent with the SCAPE model. In this setting, the joint rotations R and the body shape parameters β are also not known. We therefore need to solve simultaneously for $\mathcal{M}^Y$, R, and β minimizing the objective:

$$\sum_k \sum_{j=2,3} \|R_{b[k]} \mathcal{D}_{U,\mu}(\beta) \mathcal{Q}_{\mathbf{a}_k}(\triangle r_{b[k]}) \widehat{v}_{j,k} - (y_{j,k} - y_{1,k})\|^2 + w_Z \sum_{l=1}^{L} \|y_l - z_l\|^2, \quad (6.12)$$

where the first term represents the reconstruction error, and w_Z is a weighting term that trades off the fit to the markers against the reconstruction error.

A solution to this optimization problem is a *completed mesh* $\mathcal{M}^Y[Z]$ that both fits the observed marker locations and is consistent with the predictions of our learned SCAPE model. It also produces a set of joint rotations R and shape parameters β. Note that these

parameters can also be used to produce a *predicted mesh* $\mathcal{M}^{\tilde{Y}}[Z]$, as in Sec. 6.2. This predicted mesh is (by definition) constrained to be within our PCA subspace of shapes; thus it generally does not encode some of the details unique to the new (partial) instance mesh to be completed. As we shall see, the predicted mesh $\mathcal{M}^{\tilde{Y}}[Z]$ can also be useful for smoothing certain undesirable artifacts.

Eqn. (6.12) is a general non-linear optimization problem to which a number of existing optimization techniques can be applied. The approximate solution method, which we used for our SIGGRAPH paper and in our implementation, is described below. Then in Sec. 6.4 we describe another, mathematically more elegant way of optimizing this objective.

Our specific implementation of the optimization is intended to address the fact that Eqn. (6.12) is non-linear and non-convex, hence is subject to local minima. Empirically, we find that care has to be taken to avoid local minima. Hence, we devise an optimization routine that slows the adaptation of certain parameters in the optimization, thereby avoiding the danger of converging to sub-optimal shape completions. In particular, optimizing over all of the variables in this equation using standard non-linear optimization methods is not a good idea. Our method uses an iterative process, where it optimizes each of the three sets of parameters (R, β, and $\mathcal{M}^Y$) separately, keeping the others fixed.

The resulting optimization problem still contains a non-linear optimization step, due to the correlation between the absolute part rotations R and the joint rotations $\triangle R$, both of which appear in the objective of Eqn. (6.12). We use an approximate method to deal with this problem. Our approach is based on the observation that the actual joint rotations R influence the point locations much more than their (fairly subtle) effect on the pose deformation matrices via $\triangle R$. Thus, we can solve for R while ignoring the effect on $\triangle R$, and then update $\triangle R$ and the associated matrices $\mathcal{Q}_{\mathbf{a}}(\triangle R)$. This approximation gives excellent results, as long as the value of $\triangle R$ does not change much during each optimization step. To prevent this from happening, we add an additional term to the objective in Eqn. (6.12). The term penalizes steps where adjacent parts (parts that share a joint) move too differently from each other.

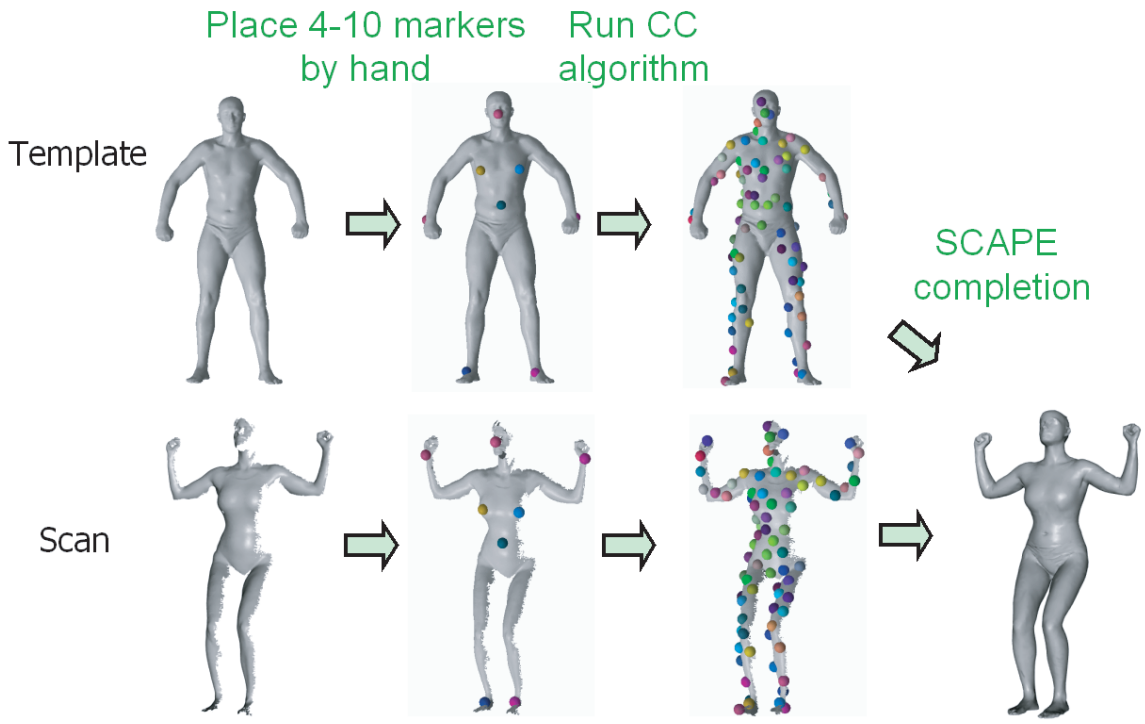


Figure 6.11: Obtaining a reasonable starting point for the shape-completion process. We place a few markers by hand and then run the Correlated Correspondence algorithm, which produces $\sim 100 - 150$ markers. We use the markers on each object part to solve for a good rigid alignment for that part. As a result, we obtain an initial set of part rotations, which are used to initialize our shape-completion optimization.

Specifically, when optimizing R, we approximate rotation using the standard approximation $R^{new} \approx (I + \hat{t})R^{old}$, where $t = (t_1, t_2, t_3)$ is an *exponential map*, and

$$\hat{t} = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix} \tag{6.13}$$

Let t_b denote the exponential map for a part b. The term preventing large joint rotations then is simply $\sum_{\{b[1],b[2] \text{ adj}\}} \|t_{b[1]} - t_{b[2]}\|^2$.

We are now ready to describe the overall optimization technique applied in our work. This techniques iteratively repeats three steps:

- We update R, using the following equation:

$$\begin{aligned}\arg\min_{t} \quad & \sum_{k}\sum_{j=2,3} \|(I+\hat{t})R^{old}DQ\widehat{v}_{j,k} - (y_{j,k}-y_{1,k})\|^2 \\ & + w_T \sum_{b[1],b[2]\ \mathrm{adj}} \|t_{b[1]} - t_{b[2]}\|^2\end{aligned} \tag{6.14}$$

  Here $D = \mathcal{D}_{U,\mu}(\beta)$ according to the current value of β, $Q = \mathcal{Q}_{\mathbf{a}_k}(\triangle r)$ where $\triangle r$ is computed from R^{old}, and w_T is an appropriate trade-off parameter.

  After all rotations R are thus updated, we recompute the joint angles $\triangle r$ accordingly.

- We update $\mathcal{M}^Y$ to optimize Eqn. (6.12), with R and β fixed. In this case, the D and Q matrices are determined, and the result is a simple quadratic objective that can be solved efficiently using standard methods:

$$\arg\min_{y_1,\ldots,y_{N_X}} \quad \sum_{k}\sum_{j=2,3} \|RDQ\widehat{v}_{j,k} - (y_{j,k}-y_{1,k})\|^2 + w_Z \sum_{l=1}^{L} \|y_l - z_l\|^2 \tag{6.15}$$

- We update β to optimize Eqn. (6.12). In this case, R and the Q matrices are fixed, as are the point positions $\mathcal{M}^Y$, so that the objective reduces to a simple quadratic function of β:

$$\arg\min_{\beta} \sum_{k}\sum_{j=2,3} \|R_{b[k]}\overline{(U\beta+\mu)}_k Q\widehat{v}_{j,k} - (y_{j,k}-y_{1,k})\|^2 \tag{6.16}$$

This optimization process converges to a local optimum of the objective in Eqn. (6.12). The surface reconstruction step from Eqn. (6.15) can be executed very efficiently, as long as the set of matching pairs (y_l, z_l) remains unchanged during the iterations (as is the case described here). Similar to our approach for solving Eqn. (6.2) earlier, this objective decomposes into three subproblems, requiring the computation of the same matrix inverse, which can be precomputed once and then used in all iterations of the above algorithm.

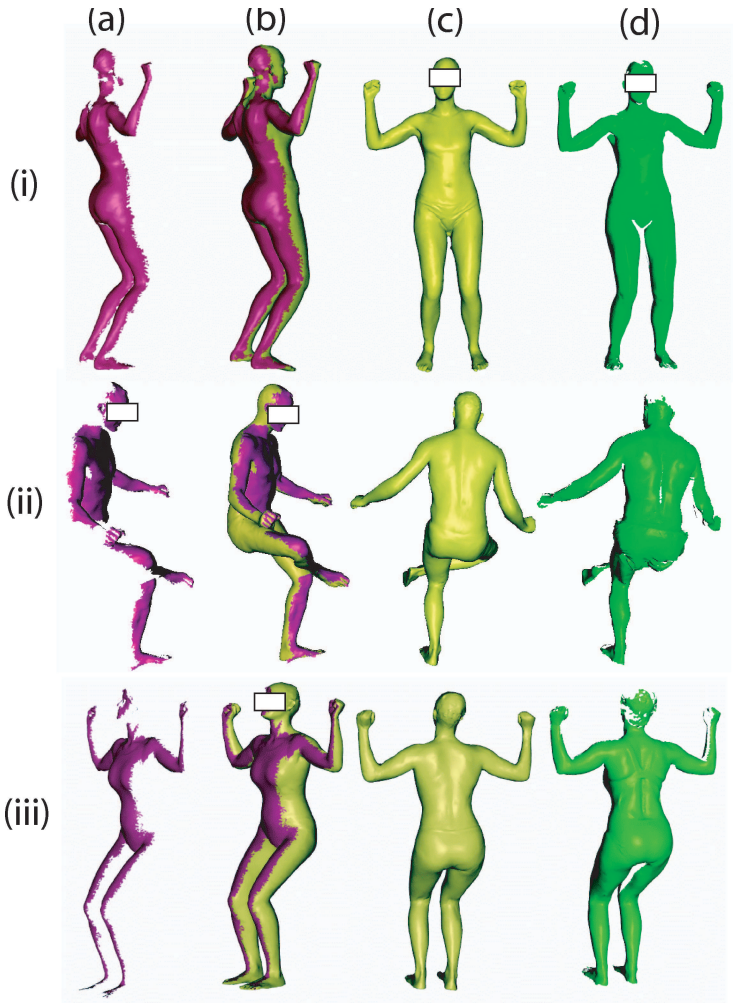


Figure 6.12: Examples of view completion, where each row represents a different partial view scan. Subject (i) is in our data set but not in the this pose; neither subjects (ii) and (iii) nor their poses are represented in our data set. (a) The original partial view. (b) The completed mesh from the same perspective as (a), with the completed portion in yellow. (c) The completed mesh from a view showing the completed portion. (d) A true scan of the same subject from the view in (c).

6.4 An Alternative Optimization Approach

Here we describe an alternative optimization scheme for the shape-completion objective defined in Eqn. (6.12). Briefly, the shape-completion task is: given a set of markers $Z = z_1, \ldots, z_L$ which specify known positions in 3D for some points $x_1, \ldots, x_L$ on the model mesh, recover the body shape parameters β, rigid part rotations R and the mesh point locations Y which minimize the objective:

$$\sum_k \sum_{j=2,3} \|R_{b[k]} \mathcal{D}_{U,\mu}(\beta) \mathcal{Q}_{\mathbf{a}_k}(\triangle r_{b[k]}) \widehat{v}_{j,k} - (y_{k,j} - y_{k,1})\|^2 + w_Z \sum_{l=1}^{L} \|y_l - z_l\|^2. \quad (6.17)$$

The main difficulty in the above equation is that it features both the absolute rigid part rotations $R_{b[k]}$, and the relative joint rotations $\triangle r_{b[k]}$ in the same product. These entities are correlated, since changing the absolute rotations also changes the joint angles. Our optimization scheme in Sec. 6.3.1 ignores the effect of this correlation, and solves directly for the absolute rotations while assuming the angles get preserved.

The key to avoiding that approximation is to use only the relative joint rotations. Here we assume that our articulated model is tree-structured. Then, we can pick one of our parts as a root, and represent the absolute rotations in terms of a sequence of joint angle rotations (going outwards from that root). For example, for a part which is M joints removed from the root, we can write this as follows:

$$R_{b[k]} = R(\triangle r_{b[k],M}) \ldots R(\triangle r_{b[k],1}) R_{root} \quad (6.18)$$

In such a manner, all absolute rotations can be replaced in the objective. Now, let us solve for an update of the exponential map parameters of a specific joint rotation $\triangle r_i$. To simply the notation here we will just denote it as $u_i = \triangle r_i$. The update can be expressed as $u_i' = u_i + t_i$, where u_i' is the new estimate of that rotation, and t_i is the update for which we will be solving.

The rotation matrix around that joint can be expressed as follows [75]:

$$R(u_i') = I + \widehat{u_i'} + \frac{(\widehat{u_i'})^2}{2!} + \ldots + \frac{(\widehat{u_i'})^n}{n!} + \ldots \quad (6.19)$$

$$= I + \widehat{u_i + t_i} + \frac{(\widehat{u_i + t_i})^2}{2!} + \ldots \quad (6.20)$$

$$= \{I + \widehat{u_i} + \frac{(\widehat{u_i})^2}{2!} + \ldots\} + \widehat{t_i} + \frac{\widehat{u_i}\widehat{t_i} + (\widehat{t_i})^2}{2!} + \ldots \quad (6.21)$$

$$= R(u_i) + \widehat{t_i} + \frac{\widehat{u_i}\widehat{t_i} + (\widehat{t_i})^2}{2!} + \ldots \quad (6.22)$$

Since our update step is chosen to be small, we can use the following linear approximation:

$$R(u_i') \approx R(u_i) + \widehat{t_i} + \frac{1}{2}\widehat{u_i}\widehat{t_i}. \quad (6.23)$$

Our goal is to iteratively optimize the rotation parameters u_i' of each joint, while holding the others fixed. We will define several quantities in the objective from Eqn. (6.17) to make the dependence on u_i' more explicit. First, we will assume that the rotation matrix associated with parameters u_i' appears m-th in the kinematic chain $R(\triangle r_{b[k],M}) \ldots R(\triangle r_{b[k],1}) R_{\text{root}}$ for polygon p_k. We denote this rotation matrix as $R(u_i')$ to emphasize this dependence. We also define:

$$A_k = R(\triangle r_{b[k],m-1}) \ldots R(\triangle r_{b[k],1}) R_{root} \mathcal{D}_{U,\mu}(\beta) \quad (6.24)$$

$$B_k = R(\triangle r_{b[k],M}) \ldots R(\triangle r_{b[k],m+1}). \quad (6.25)$$

Intuitively, matrix A_k denotes the product of rotation matrices that come before $R(u_i')$ in the kinematic chain (pre-multiplied by the body shape deformation matrix, which does not depend on u_i'). Matrix B_k, on the other hand, contains product the rotation matrices that come after $R(u_i')$. These quantities can be substituted into the shape completion objective, as follows:

$$\sum_k \sum_{j=2,3} \|B_k R(u_i') A_k \mathcal{Q}_{\mathbf{a}_k}(u_i') \widehat{v}_{j,k} - (y_{j,k} - y_{1,k})\|^2 + w_Z \sum_{l=1}^{L} \|y_l - z_l\|^2. \quad (6.26)$$

Above we introduce another expression $\mathcal{Q}_{\mathbf{a}_k}(u_i')$ to show that the pose deformation matrix may be (a linear) function of the rotation parameters u_i'. In many cases, in will be independent of u_i' altogether, but we will ignore this in the current notation.

Replacing $u_i' = u_i + t_i$ and $d_{k,j} = (y_{j,k} - y_{1,k})$ in the above equation, we obtain our new objective:

$$\sum_k \sum_{j=2,3} \|B_k R(u_i + t_i) A_k \mathcal{Q}_{\mathbf{a}_k}(u_i + t_i)\widehat{v}_{j,k} - d_{k,j}\|^2 + w_Z \sum_{l=1}^{L} \|y_l - z_l\|^2. \tag{6.27}$$

This new objective needs to be minimized for the parameters t_i, as follows:

$$\arg\min_{t_i} \sum_k \sum_{j=2,3} \|B_k R(u_i + t_i) A_k \mathcal{Q}_{\mathbf{a}_k}(u_i + t_i)\widehat{v}_{j,k} - d_{k,j}\|^2. \tag{6.28}$$

We use the identity $R^T R = I$, which is valid for all rotation matrices, in expanding this expression. In particular, $(A_k R(u_i'))^T A_k R(u_i') = I$, which eliminates second order dependence on the matrix $R(u_i')$. After some tedious algebraic transformations resulting from expanding the norm, we obtain the following expression:

$$\arg\min_{t_i} \sum_k \sum_{j=2,3} \quad \widehat{v}_{j,k}^T \mathcal{Q}_{\mathbf{a}_k}(u_i + t_i)^T A_k^T A_k \mathcal{Q}_{\mathbf{a}_k}(u_i + t_i)\widehat{v}_{j,k} + \tag{6.29}$$

$$2d_{k,j}^T (R(u_i + t_i) A_k \mathcal{Q}_{\mathbf{a}_k}(u_i + t_i)\widehat{v}_{j,k}. \tag{6.30}$$

The important thing to notice in this complicated expression, is the fact that it is at most a quadratic expression in terms of t_i. First of all, we use the linear approximation of the rotation matrix $R(u_i + t_i)$ from Eqn. (6.23). Second, $\mathcal{Q}_{\mathbf{a}_k}(u_i + t_i)$ is a linear function of the update t_i. Therefore, we can obtain the value of t_i using simple least-squares optimization of the equation above.

Given that we now have a way to solve for the rotation of each joint separately, our entire optimization schedule consists of a set of phases, where we solve for each joint rotation u_i (as well as for the orientation of the root R_{root}) separately, then for parameters β, and finally for the shape Y, each time while holding all other parameters fixed.

6.4.1 Partial View Completion

An obvious application of our shape completion method is to the task of partial view completion. Here, we are given a partial scan of a human body; our task is to produce a full 3D mesh which is consistent with the observed partial scan, and provides a realistic completion for the unseen parts.

Our shape completion algorithm of Sec. 6.3 applies directly to this task. We take the partial scan, and manually annotate it with a small number of markers (4–10 markers, 7 on average). We then apply the CC algorithm [4] to register the partial scan to the template mesh. The result is a set of 100–150 markers, mapping points on the scan to corresponding points on the template mesh. This number of markers is sufficient to obtain a reasonable initial hypothesis for the rotations R of the rigid skeleton. We then iterate between two phases. First, we find point-to-point correspondences between the partial view and their nearest neighbor points in our current estimate of the surface $\mathcal{M}^Y[Z]$. Then we use these correspondences as markers and solve Eqn. (6.12) to obtain a new estimate $\mathcal{M}^Y[Z]$ of the surface. Upon convergence, we obtain a completion mesh $\mathcal{M}^Y[Z]$, which fits the partial view surface as well as the SCAPE model. The steps of the partial view completion process are shown in Fig. 6.11.

We would like to point out that every time we re-compute the point-to-point correspondences between the meshes, the set of matching pairs (y_l, z_l) that are provided to the shape-completion algorithm changes. This change necessitates the re-computation of a matrix inverse necessary for the solution of Eqn. (6.15), which takes about one second in our implementation. Thus, our partial view completion implementation is not real-time.

In Fig. 6.12, we show the results of this algorithm in completing three partial views of different humans. Row (i) shows partial view completion results for a subject who is present in our data set, but in a pose that is not in our data set. The prediction for the shoulder blade deformation is very realistic; a similar deformation is not present in the training pose for this subject. Rows (ii) and (iii) show completion for subjects who are not in our data set, in poses that are not in our data set. The task in row (ii) is particularly challenging, both because the pose is very different from any pose in our data set, and because the subject was wearing pants, which we cut out (see Fig. 6.12(ii)-(d)), leading to the large hole in the

original scan. Nevertheless, the completed mesh contains realistic deformations in both the back and the legs.

6.4.2 Motion Capture Animation

Our shape completion framework can also be applied to produce animations from marker motion capture sequences. In this case, we have a sequence of frames, each specifying the 3D positions for some set of markers. We can view the set of markers observed in each frame as our input Z to the algorithm of Sec. 6.3, and use the algorithm to produce a mesh. The sequence of meshes produced for the different frames can be strung together to produce a full 3D animation of the motion capture sequence.

Note that, in many motion capture systems, the markers protrude from the body, so that a reconstructed mesh that achieves the exact marker positions observed may contain unrealistic deformations. Therefore, rather than using the completed mesh $\mathcal{M}^Y[Z]$ (as in our partial view completion task), we use the predicted mesh $\mathcal{M}^{\tilde{Y}}[Z]$. As this mesh is constrained to lie within the space of body shapes encoded by our PCA model, it tends to avoid these unrealistic deformations.

We applied this data to two motion capture sequences, both for the same subject S. Notably, our data set only contains a single scan for subject S, in the standard position shown in the third row of Fig. 6.2(a). Each of the sequences used 56 markers per frame, distributed over the entire body. We took a 3D scan of subject S with the markers, and used it to establish the correspondence between the observed markers and points on the subject's surface. We completed this scan using the algorithm from Sec. 6.4.1 to obtain the body shape parameters β for that subject; these parameters will be held constant during the rest of the optimization. We then applied the algorithm of Sec. 6.3 to each sequence frame. In each frame, we used the previous frame's estimated pose R as a starting point for the shape-completion optimization.

The animation was generated from the sequence of predicted scans $\mathcal{M}^{\tilde{Y}}[Z_f]$. Using our (unoptimized) implementation, it took approximately 3 minutes to generate each frame. Fig. 6.13 demonstrates some of our results. We show that realistic muscle deformation was obtained for subject S (Fig. 6.13(c)). Additionally, we show that motion transfer can be

performed onto a different subject in our data set (Fig. 6.13(d)) and that the subject can be changed during the motion sequence (Fig. 6.13(e)).

6.5 Related Work

The recent example-based approaches for learning deformable human models represent deformation by point displacements of the example surfaces, relative to a generic template shape. For modeling pose deformation, the template shape is usually assumed to be an articulated model. A popular animation approach called *skinning* (described in Lewis *et al.* [70]) assumes that the point displacements are generated by a weighted set of (usually linear) influences from neighboring joints. A more sophisticated method was presented by Allen *et al.* [1], who register an articulated model (represented as a posable subdivision template) to scans of a human in different poses. The displacements for a new pose are predicted by interpolating from a set of example scans with similar joint angles. A variety of related methods [70, 108, 123, 80] differ only in the details of representing the point displacements, and in the particular interpolation method used. Models of pose deformation are learned not only from 3D scans, but also by combining shape-from-silhouette and marker motion capture sequences [99]. However, none of the above approaches learn a model of the shape changes between different individuals.

To model body shape variation across different people, Allen *et al.* [2] morph a generic template shape into 250 scans of different humans in the same pose. The variability of human shape is captured by performing principal component analysis (PCA) over the displacements of the template points. The model is used for hole-filling of scans and fitting a set of sparse markers for people captured in the standard pose. Another approach, by Seo and Thalmann [102], decomposes the body shape deformation into a rigid and a non-rigid component, of which the latter is also represented using PCA over point displacements. Neither approach learns a model of pose deformation. However, they demonstrate preliminary animation results by using expert-designed skinning models. Animation is done by bringing the space of body shapes and the skinning model into correspondence (this can be done in a manual or semi-automatic way [54]), and adding the point displacements accounting for pose deformation to the human shape. Such skinning models are part of

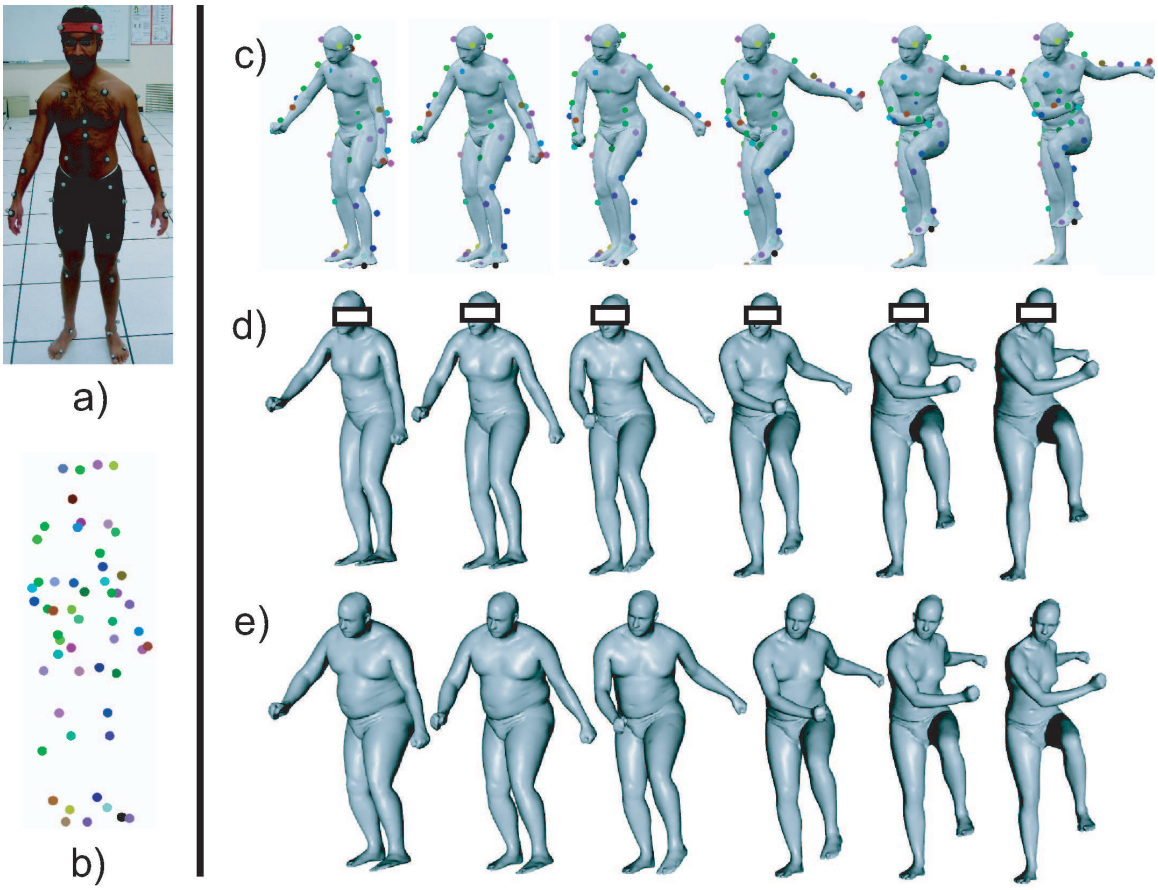


Figure 6.13: Motion capture animation. (a) Subject wearing motion capture markers (b) motion capture markers in a single frame (c) An animation of a subject based on a motion capture sequence, with the markers from which the animation was derived superimposed on the meshes. (d) An example of motion transfer to a different subject in our data set. (e) Animation based on motion capture, but where we change the body shape parameters in PCA space as we move through the sequence.

standard animation packages, but since they are usually not learned from scan data, they usually do not model muscle deformation accurately.

An obvious approach for building a data-driven model of pose and body shape deformation would be to integrate two existing methods in a similar way. The main challenge lies in finding a good way to combine two distinct deformation models based on point displacements. Point displacements cannot be multiplied in a meaningful way; adding them ignores an important notion of scale. For example, pose displacements learned on a large individual cannot be added to the shape of a small individual without undesirable artifacts. This problem has long been known in the fields of deformation transfer and expression cloning [89]. In thinking how to address it, we were inspired by the deformation transfer method of Sumner and Popović [111], which shows how to retarget the deformation of one mesh to another, assuming point-to-point correspondences between them are available. The transfer maintains proper scaling of deformation, by representing the deformation of each polygon using a 3×3 matrix. It suggests a way of mapping pose deformations onto a variety of human physiques. However, it does not address the task of representing and learning a deformable human model, which is tackled in our work.

Multilinear models, which are closely related to our work, have been applied for modeling face variation in images [117]. A generative model of human faces has to address multiple factors of image creation such as illumination, expression and viewpoint. The face is modeled as a product of linear appearance models, corresponding to influences of the various factors. Multilinear approaches have also been used to model 3D face deformation [120]. However, this work uses point displacements from a template shape as a representation of face deformation. We believe our representation of deformation, based on modeling the polygon transformation matrices, is more suitable for the task. It is the subject of interesting future work to compare the two representations in the context of face expression modeling. Of course, our method cannot be applied directly for face modeling, because we correlate the deformations of an individual to the underlying skeleton angles, while a significant part of the face deformations is purely muscle-based (the jaw movement being the exception). However, it is not difficult to modify our approach by learning a suitable space of facial expression deformations (by doing PCA over the face expressions of a particular individual, for example).

Our shape-completion application is related to work in the area of hole-filling. Surfaces acquired with scanners are typically incomplete and contain holes. A common way to complete these holes is to fill them with a smooth surface patch that meets the boundary conditions of the hole [33, 35, 71]. These approaches work well when the holes are small compared to the geometric variation of the surface. Our application, by contrast, requires the filling of huge holes (e.g., in some experiments more than half of the surface was not observed; in others we are only provided with sparse motion capture data) and we address it with a model-based method. Other model-based solutions for hole filling were proposed in the past. Kähler *et al.* [62] and Szeliski and Lavallée [113] use volumetric template-based methods for this problem. These approaches work well for largely convex objects, such as a human head, but are not easily applied to objects with branching parts, such as the human body. While the work of Allen *et al.* [2] can be used for hole-filling of human bodies, it can only do so if the humans are captured in a particular pose.

Marker motion capture systems are widely available, and can be used for obtaining high-quality 3D models of a moving person. Existing animation methods (e.g. [1, 102]) do not utilize the marker data and assume the system directly outputs the appropriate skeleton angles. They also do not handle body shape variation well, as previously discussed. Both of these limitations are lifted in our work.

6.6 Discussion and Limitations

This chapter presents the SCAPE model, which captures human shape deformation due to both pose variation and to body shape variation over different subjects. Our results demonstrate that the model can generate realistic meshes for a wide range of subjects and poses. We showed how the SCAPE model can be used for shape completion, and cast two important graphics tasks — partial view completion and motion capture animation — as applications of our shape completion algorithm.

Our current approach requires a set of scans of a single person in different poses to learn the space of pose deformations. Once we have done this, we can use scans of different people in different poses to learn the space of body shapes. We currently do not provide a method to learn both spaces from a random mix of scans from different people

in different poses. Our assumption on the training set structure is not particularly restrictive, and it simplifies our data collection and learning procedures. We could try to learn our model from a non-uniform data set, by iterating between estimating either the pose or the body shape model while keeping the other one fixed. This process would result in a local minimum in the joint space of deformations. We cannot predict how good this local minimum would be; it depends specifically on the training data we are given, and on the search method used.

The pose deformation in our model is determined by regression from adjacent joint angles. We found that the linear regression model provides surprisingly good animation results, and simplifies the task of shape completion. For many instances of partial view completion, a more accurate model may not be necessary, because our solution is allowed to deform outside of SCAPE space in order to fit the observed surface. Thus, partial view data can correct some of the (fairly small) errors resulting from the assumption of a linear regression model. When the SCAPE model is used purely for animation, the linear regression model is not sufficient for obtaining high-quality meshes in all cases. We demonstrated that in such cases, non-linear regression approaches can be used.

The SCAPE model is focused on representing muscle deformations resulting from articulated body motion. Deformations resulting from other factors are not encoded. One such factor is deformation resulting from pure muscle activity. Thus, the model is not expressive enough to distinguish between a flexed bicep muscle and a lax one in cases where the joint angle is the same. For the same reason, it is not appropriate to deal with faces, where most of the motion is purely muscle-based. Another factor leading to muscle deformation is tissue perturbations due to motion (e.g., fat wiggling), which our model also does not represent.

Currently, our framework includes no prior over poses. Thus, when encountering occlusions, we cannot use the observed position of some body parts to constrain the likely location of others. Our model can easily be extended to encompass such a prior, in a modular way. For example, in the case of static scans, a kinematic prior such as that of Grochow *et al.*[51] could simply be introduced as an additional term into our optimization. When animating dynamic sequences, we can use a tracking algorithm (e.g., a Kalman filter) to generate a pose prior for any frame given all or part of the observation sequence.

Finally, we note that our approach is purely data driven, generating the entire model from a set of data scans. Human intervention is required only for placing a small set of markers on the scans, as a starting point for registration. Thus, the model can easily be applied to other data sets, allowing us to generate models specific to certain types of body shapes or certain poses. Moreover, the framework applies more generally to cases where surface deformation is derived from articulated motion. Thus, if we could solve the data acquisition problem (e.g., using shape from silhouette [131]), we could use this framework to learn realistic deformation models for creatures other than humans.

Chapter 7

Conclusions and Future Directions

In this final chapter, we summarize the contributions of this thesis, discuss a number of its limitations, and present some challenges and future research directions that build on top of the work in this thesis.

7.1 Summary

We present a framework for learning complex shape models from range scan data. The framework consists of several algorithms, based on the theory of probabilistic graphical models, which allow us to learn complex shape models of different objects and object classes with minimal human intervention. We also describe applications of these algorithms, as well as the learned models, to the tasks of tracking, animation and shape-completion.

7.1.1 Unsupervised Registration

We present an algorithm for unsupervised registration of two non-rigid 3D surfaces. Our Correlated Correspondence algorithm can register surfaces that undergo significant deformations, without making prior assumptions about initial alignment, or object shape and dynamics. It performs efficient combinatorial search in the space of possible surface alignments, preferring registrations that preserve the surface appearance and geometry. We

demonstrate successful registration for articulated objects subject to large joint movements, as well as for other kinds of non-rigid surface deformations. In contrast, previous surface registration algorithms avoid tackling the combinatorial nature of the non-rigid registration problem, and as a result are more prone to becoming stuck in poor local minima in such cases. We show the quality and the utility of our registration results by using them as a starting point for compelling computer graphics applications: partial view completion and interpolation between pairs of scans.

7.1.2 Recovering Articulated Models

We address the problem of learning a complex articulated object models from registered 3D scans. The algorithm automatically recovers a decomposition of the object into approximately rigid parts, the location of the parts in the different object instances, and the articulated object skeleton linking the parts. The decomposition into parts is obtained by using the Expectation-Maximization algorithm, using a graphical model that explicitly enforces the spatial contiguity of the object's parts. Although the graphical model is densely connected, the object decomposition step can be performed optimally and efficiently, allowing us to identify a large number of object parts while avoiding local maxima. We demonstrate the algorithm on three real world datasets, recovering complex models with up to 18 parts, even in the presence of non-trivial part deformations. Our algorithm not only recovers the parts and joints, but also figures out the optimal number of parts automatically. We also describe an efficient algorithm, which can be used to track the recovered models in shape-from-silhouette data.

7.1.3 Learning the Space of Human Body Shapes

Finally, we present a method named SCAPE, which learns a model of human shape deformation due to both pose variation and to body shape variation over different subjects. Most methods for modeling deformations represent them in terms of point displacements from a shape template. However, it is difficult to combine such displacement-based models in a way that scales deformations correctly. We address this problem by representing deformations as consecutive 3×3 matrices that deform the polygons of the mesh. Our

pose deformation model derives the values of these deformation matrices as a function of the skeleton pose. Our body shape deformation model induces a low-dimensional space over another set of polygon transformation matrices, associated with the deformations occurring between people with different physiques. The two models can be combined in a natural way to produce 3D surface models with realistic muscle deformation for different people in different poses, when neither appear in the original set of examples. We also show how the SCAPE model can be used for shape completion, and cast two important graphics tasks — partial view completion and motion capture animation — as applications of our shape completion algorithm. We demonstrate shape-completion and motion capture animation results for a variety of different people and poses.

7.2 Extensions and Open Problems

It is our hope that this thesis demonstrates the utility of probabilistic models for studying key problems in the shape modeling domain. Here we describe several possible extensions to our work, and discuss some exciting directions for future research. Some of the most straightforward extensions were already discussed in the relevant chapters.

7.2.1 Real-time Implementations

Our current unoptimized implementation of the Correlated Correspondence algorithm for non-rigid registration takes about two minutes to register a pair of scans. Because of the size of the induced Markov network in which we perform inference, the algorithm cannot be made to run in close to real time on current single-processor machines. However, the algorithm is ideally suited for a distributed multi-processor architecture. The computation of the Markov node and edge potentials can be executed in parallel. Moreover, loopy belief propagation consists of simple local updates of the beliefs over each variable, which can be executed in parallel as well, even asynchronously. Therefore we believe, that an implementation of the Correlated Correspondence algorithm on a distributed parallel system can run in close to real-time.

Another interesting direction is to optimize the partial view completion application of

SCAPE from Chapter 6.4.1. In our current implementation, updating the point-to-point correspondences between the human body model and the scan, or the strength of these correspondences, leads to a recomputation of the matrix inverse, necessary for solving the equation Eqn. (6.15). In practice, the cost of this computation is about one second. Intuitively, this cost is excessive, because even though point-to-point correspondences tend to change little between iterations, we are doing the entire work of computing the matrix inverse from scratch. Therefore, an interesting direction of exploration is to find a suitable way of exploiting the previous matrix values in the new computation. The most straightforward way is to use the conjugate gradient algorithm, although in some cases it tends to converge too slowly. A different direction would try to exploit the structure of the matrix, as Sumner *et al.* [112] have done for a related problem.

7.2.2 Registration in the Presence of Clutter and Occlusion

One of the main limitations of our Correlated Correspondence algorithm is its assumption that the scan mesh is a subset of the model mesh. This *no-clutter* assumption is essential in making the algorithm tractable, because it allows us to avoid reasoning about cases when points or edges in the scan mesh have no counterparts in the model. It is also important for another reason. When we refrain from imposing an object-specific shape prior, and both occlusion and clutter are present in the scene simultaneously, the registration problem becomes ill-defined. In the presence of significant deformation, there are too many different possible ways of aligning surfaces in such cases. We consciously chose the no-clutter assumption as a way to constrain this space, without making the algorithm object-specific.

When prior object-specific knowledge is available, we can tackle cases when both occlusion and clutter are present in the scene. Of particular interest is the problem of detecting the pose of an articulated model in a range scan that contains both clutter and occlusions. The Correlated Correspondence methodology cannot be used directly in such cases, because in the presence of occlusion, articulated parts can be placed even in parts of the scene where no corresponding surface is available. Thus, enumerating all possible part locations is no longer feasible. In a separate work (currently in submission), we show how to extend our methodology in order to address this case. Our approach starts by using low-level

spin-image based detectors to suggest possible part placement hypotheses. However, the detectors are not guaranteed to find good hypotheses for all object parts. Therefore, we introduce a separate hypothesis-enrichment phase, in which the original part location hypotheses are used to generate likely placement suggestions for their neighboring parts. The set of expanded part domains can be used to construct a Markov network, which scores the quality of the part placements and enforces the articulated model constraints. Unlike the Correlated Correspondence model, here we explicitly allow some parts to be completely missing in the scene. The resulting model can be optimized using loopy belief propagation, to obtain the most likely object con figuration in the scene.

7.2.3 SCAPE for Markerless Motion Capture

Markerless motion capture is a compelling application, enabling the acquisition of human motion trajectories for use in entertainment (games and movies) and clinical applications (human movement analysis). Current applications require the placement of photoreflective markers on the tracked object, which is a precise and very time-consuming activity, as well as installation of specialized hardware (infrared cameras), which makes the systems expensive.

In Chapter 5 we described an algorithm for tracking rigid bodies in shape-from-silhouette data, which can be used for human motion acquisition. However, the algorithm has several important limitations. The most important one is that the algorithm currently requires that the articulated model for the specific person being tracked is available. Obtaining such an articulated model is very time-consuming in most cases, limiting the impact of the application. Another limitation of that model is that it treats the human body parts as completely rigid, and cannot account properly for their deformations.

Both of these problems are addressed in our SCAPE model of the human body. We believe that using the SCAPE template for tracking in shape-from-silhouette data will enable the automatic motion capture of different people, and increase the accuracy of the tracking relative to that produced by tracking purely articulated models. The optimization of the SCAPE model in this case is very similar to our partial view completion application from Sec. 6.4.1. Further work on making the partial view completion run in real time will

only make this application even more compelling.

7.2.4 Towards an Integrated Model of the Human Body

In Chapter 6 of this thesis, we learn a human appearance model that captures the body deformations due to changes in physique and pose. However, many exciting aspects of the human modeling task are yet to be addressed. Below we describe briefly some of the possible new directions.

- **Modeling the correlations between pose and body shape deformations**

  Our SCAPE model decouples the pose deformation model and the body shape deformation model. This design choice greatly simplifies the mathematical formulation, improves the identifiability of the model from data, and makes the learning algorithm more efficient. However, it also prevents us from capturing phenomena where there is a strong correlation between body shape and muscle deformation. For example, as the same muscle deformation model is used for all people, we do not capture the fact that more muscular people are likely to exhibit greater muscle deformation than others, and, conversely, that muscle deformation may be obscured in people with significant body fat. To address this, we need to learn a model that explicitly captures this dependence between body shape and pose deformations. There are different possible ways for modeling these correlations, the most straightforward of which is an extension of our model, where the pose deformations are dependent on the body shape parameters. A necessary prerequisite for learning such a model is the availability of a dataset that contains scans of multiple people, in which each person is captured in several different poses.

- **Integrating body and face models**

  The space of human shapes is not complete without incorporating an accurate model of the most expressive body part — the face. There is an extensive body of work on shape modeling, both in image and 3D data. One of the latest methods, by Vlasić *et al.* [119], uses multi-linear models to model 3D face deformation. Their work uses point displacements from a template shape as a representation of face deformation.

Our representation of deformation, based on modeling the polygon transformation matrices, could be more suitable for the task. It would be very interesting to use it in conjunction with multi-linear models, and to compare the results to those obtained by Vlasić *et al.*

- **Informed models of kinematics and dynamics**

  Another interesting direction is to add a temporal dimension to the shape modeling task. The movement of a person causes a set of body deformations, such as fat wiggling and muscle contractions. Knowledge about the movement can help in predicting the body deformations more accurately. The main limitation to exploring this idea is the difficulty of real-time range data acquisition. One avenue for exploration would be to track and refine our SCAPE model in shape-from-silhouette data [26]. Another possibility is to use recent advances in real-time scanning technology, such as the work of Zhang *et al.* [131] on real-time acquisition of face scans.

  Also, our SCAPE framework currently does not include a temporal prior over poses. Such a prior can be very useful in tracking scenarios, where we can reason about the position of occluded body parts using the position and velocity of the observed parts. Learning of such kinematic priors for 3D articulated models has been addressed in the work of Grochow *et al.*[51], as well as the work of Sminchisescu *et al.*[109]. Our current model can be easily extended to encompass such a prior, in a modular way. The benefits of such a combination for the tracking task are yet to be explored.

7.3 The Challenge Ahead

We have presented a framework for learning complex shape models, which consists of several algorithms for performing key shape modeling tasks with minimal human intervention. We hope that our methods demonstrate the utility and applicability of probabilistic graphical models for addressing key problems related to reasoning about object shape. We are looking forward to applying our methods for other different object and object class modeling problems.

In the future, we expect an explosion in the amount of available range scan data, and

an integration of range data with camera input, which will bring about an expanded set of learning problems and applications. We hope that continued research, of which the methods presented in this thesis are but a start, will help tackle these evermore sophisticated learning problems on the way to creating autonomous robotic agents, and compelling virtual realities.

Bibliography

[1] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics*, 21(3):612–619, 2002.

[2] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics*, 22(3):587–594, 2003.

[3] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM SIGGRAPH*, 22(3), 2003.

[4] D. Anguelov, D. Koller, H. Pang, P. Srinivasan, and S. Thrun. Recovering articulated object models from 3d range data. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 18–26, 2004.

[5] D. Anguelov, L. Mündermann, and S. Corazza. An iterative closest point algorithm for tracking articulated models in 3d range scans. In *ASME/Summer Bioengineering Conference*, Vail, Colorado, 2005.

[6] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, H. Pang, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Advances in Neural Information Processing Systems 17*, pages 33–40, 2005.

[7] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *ACM Transactions on Graphics*, 24(3), 2005.

[8] C. Archer and T.Leen. Adaptive principal component analysis. In *Technical Report CSE-02-008*. Department of Computer Science and Engineering, Oregon Health and Science University, 2002.

[9] A. Aubel and D. Thalmann. Interactive modeling of the human musculature. In *Proc. Computer Animation*, 2001.

[10] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems*, pages 585–591, 2001.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1), 2002.

[12] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[13] P. Besl and N. McKay. A method for registration of 3d shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[14] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, United Kingdom, 1995.

[15] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV*, 1999.

[17] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *CVPR*, 1999.

[18] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. IEEE CVPR*, 1998.

[19] Benedict Brown and Szymon Rusinkiewicz. Non-rigid range-scan alignment using thin-plate splines. In *Symposium on 3D Data Processing, Visualization, and Transmission.*, September 2004.

[20] W. Buntine. Chain graphs for learning. In *Proc. Eleventh Conference on Uncertainty in Artificial Intelligence (UAI '95)*, 1995.

[21] M. C. Burl and P. Perona. Recognition of planar object classes. In *Computer Vision and Patter Recognition(CVPR)*, 1996.

[22] M. C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proceedings of European Conference on Computer Vision (ECCV)*, 1998.

[23] T. Cham and J. Rehg. A multiple hypothesis approach to figure tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 1999.

[24] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proc. IEEE Conf. on Robotics and Automation*, 1991.

[25] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–84, 2003.

[26] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, 62(3):221 – 247, May 2005.

[27] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time: Part ii: Applications to human modeling and markerless motion tracking. *International Journal of Computer Vision*, 63(3):225 – 245, August 2005.

[28] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[29] D. Clark and C. Thayer. A primer on the exponential family of distributions. In *Call Paper Program on Generalized Linear Models*, pages 118–148, 2004.

[30] G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[31] J. Coughlan and S. Ferreira. Finding deformable shapes using loopy belief propagation. In *Proc. ECCV*, volume 3, pages 453–468, 2002.

[32] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, 1999.

[33] B. Curless and M. Levoy. A volumetric method of building complex models from range images. *Proceedings of SIGGRAPH 1996*, pages 303–312, 1996.

[34] P. Dagum and M. Luby. An optimal approximation algorithm for bayesian inference. *Artificial Intelligence*, 93(1–2):1–27, 1997.

[35] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Symposium on 3D Data Processing, Visualization, and Transmission*, 2002.

[36] J. Davis, R. Ramamoothi, and S. Rusinkiewicz. Spacetime stereo : A unifying framework for depth from triangulation. In *Proc. CVPR*, 2003.

[37] M. de Berg, M. van Kreveld, O. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer Verlag, 1997.

[38] R. Dechter. Bucket elimination: a unifying framework for probabilistic inference. In *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 211–219, 1996.

[39] 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, B*, 39:1–39, 1977.

[40] A. Elad (Elbaz) and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Trans. on PAMI*, 25(10):1285–1295, 2003.

[41] M. Etoh and Y. Shirai. Segmentation and 2d motion estimation by region fragments. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 192–199, 1993.

[42] Pedro Felzenszwalb. Representation and detection of shapes in images. *PhD Thesis*, 2003.

[43] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of IEEE Conference on Computer Vision and Patter Recognition(CVPR)*, 2003.

[44] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24:381–395, 1981.

[45] M.A. Fischler and R.A. Erschlager. The representation and matching of pictorial structures. *IEEE Trans. Computers*, C-22, 1973.

[46] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice, 2nd ed.* Addison Wesley, 1996.

[47] Michael Garland and Paul S.Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 97*, pages 209–216, August 1997.

[48] D. Gavrila and L. Davis. Tracking of humans in action: a 3-d model-based approach. In *In ARPA Image Understanding Workshop*, 1996.

[49] K. Grauman and T. Darrell. Fast contour matching using approximate earth mover's distance. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2004.

[50] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binar images. *J. R. Statist. Soc. B*, 51:271–279, 1989.

[51] Keith Grochow, Steve L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, 2004.

[52] D. Hähnel, S. Thrun, and W. Burgard. An extension of the ICP algorithm for modeling nonrigid objects with mobile robots. In *Proc. IJCAI*, Acapulco, Mexico, 2003.

[53] J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. In *Unpublished Manuscript*, 1971.

[54] A. Hilton, J. Starck, and G. Collins. From 3d shape capture to animated models. In *First International Symposion on 3D Data Processing, Visualization and Transmission (3DVPT2002)*, 2002.

[55] D. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[56] D. Huttenlocher and P. Felzenszwalb. Efficient matching of pictorial structures. In *CVPR*, 2003.

[57] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *Third International Workshop on Statistical and Computational Theories of Vision*, Nice, France, 2003.

[58] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision*, 43(1), 2001.

[59] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 5(4):269–282, 1990.

[60] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[61] M.I. Jordan, Z. Ghahramani, T. Jaakkola, and K.G. Olesen. An introduction to variational approximation methods for graphical models. In M.I. Jordan, editor, *Learning in graphical models*, Dordrecht, Netherlands, 1998. Kluwer.

[62] Kolja Kähler, Jörg Haber, Hitoshi Yamauchi, and Hans-Peter Seidel. Head shop: generating animated head models with anatomical structure. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 55–64, 2002.

[63] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *Proc. IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1999.

[64] Stefan Kruger and Andrew Calway. Motion estimation and tracking using multiresolution affine models. In *Proceedings of the IEEE Colloquium on Motion Analysis and Tracking*, pages 50–55, May 1999.

[65] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proceedings of the British Machine Vision Conference*, pages 789–798, 2004.

[66] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Obj cut. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 18–25, 2005.

[67] S. L. Lauritzen. The em algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[68] C. H. Lee, A. Varshney, and David Jacobs. Mesh saliency. *ACM SIGGRAPH*, 24(3), 2005.

[69] Michael Leventon. Statistic models in medical image analysis. *PhD Thesis*, 2000.

[70] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. *Proceedings of ACM SIGGRAPH 2000*, pages 165–172, 2000.

[71] P. Liepa. Filling holes in meshes. In *Proc. of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 200–205. Eurographics Association, 2003.

[72] Michael H. Lin. Tracking articulated objects in real-time range image sequences. In *ICCV*, volume 1, pages 648–653, 1999.

[73] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen. Linear rotation-invariant coordinates for meshes. In *ACM SIGGRAPH*, pages 479–487, 2005.

[74] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. In *Computer Graphics Proceedings, Annual Conference Series, Proceedings of SIGGRAPH 87*, pages 163–169, 1989.

[75] Y. Ma, Stefano Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision*. Springer Verlag, 2004.

[76] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1), June 2001.

[77] J. Marroquin, E. Santana, and E. Botello. Hidden markov measure fields for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11), 2003.

[78] I. Mikic, M. Triverdi, E. Hunter, and P. Cosman. Articulated body posture estimation from multi-camera voxel data. In *In Proc. CVPR*, 2001.

[79] Niloy J. Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proc. of Symposium on Computational Geometry*, pages 322 – 328, 2003.

[80] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics*, 22(3):562–568, 2003.

[81] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2002.

[82] Meta motion. http://www.metamotion.com.

[83] Vicon motion systems. http://www.vicon.com.

[84] L. Mündermann, S. Corazza, D. Anguelov, and T. P. Andriacchi. Estimation of the accuracy and precision of 3d human body kinematics using markerless motion capture and articulated icp. In *ASME/Summer Bioengineering Conference*, Vail, Colorado, 2005.

[85] Lars Mündermann, Stefano Corazza, Ajit M. Chaudhari, Eugene J. Alexander, and Thomas P. Andriacchi. Most favorable camera configuration for a shape-from-silhouette markerless motion capture system for biomechanical analysis. In *Videometrics VIII*, pages 278–287, January 2005.

[86] K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. In *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 467–475, 1999.

[87] David Murray and Bernard Buxton. Scene segmentation from visual motion using global optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2), 1987.

[88] R. M. Neal. Probabilistic inference using markov chain monte carlo methods. In *Technical Report CRG-TR-93-1*, Department of Computer Science, University of Toronto, 1993.

[89] Jun Noh and Ulrich Neumann. Expression cloning. *Proceedings of ACM SIGGRAPH 2001*, pages 277–288, 2001.

[90] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[91] S. Peleg, M. Werman, and H. Rom. A unified approach to the change of resolution: Space and gray-level. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:739–742, 1989.

[92] ILOG CPLEX: High performance software for mathematical programming and optimization. http://www.ilog.com/products/cplex/.

[93] R. B. Potts. Some generalized order-disorder transformations. *Proc. Cambridge Phil. Soc.*, 48, 1952.

[94] Poser: The premier 3D figure design and animation solution. http://www.e-frontier.com/go/poser_hpl.

[95] J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 612–617, June 1995.

[96] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2003.

[97] Y. Rubner, C. Tomasi, and L. Guibas. The earth movers distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[98] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3DIM*, Quebec City, Canada, 2001. IEEEComputer Society.

[99] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. *ACM Transactions on Graphics*, 22(3):578–586, 2003.

[100] F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy-based modeling of the human musculature. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997.

[101] Bernhard Schölkopf and Alex Smola. A tutorial on support vector regression. In *Technical Report NC2-TR-1998-030*. NeuroCOLT2, 1998.

[102] Hyewon Seo and Nadia Magnenat-Thalmann. An automatic modeling of human bodies from sizing parameters. In *ACM Symposium on Interactive 3D Graphics*, pages 19–26, 2003.

[103] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambriedge Univ. Press, 1999.

[104] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2002.

[105] Christian Shelton. Morphable surface models. In *International Journal of Computer Vision*, 2000.

[106] H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proceedings of European Conference on Computer Vision (ECCV)*, June 2000.

[107] Leonid Sigal, Michael Isard, Benjamin H. Sigelman, and Michael J. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *NIPS*, 2003.

[108] Peter-Pike J. Sloan, Charles F. Rose, and Michael F. Cohen. Shape by example. In *2001 Symposium on Interactive 3D Graphics*, pages 135–144, 2001.

[109] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discriminative density propagation for 3d human motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[110] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[111] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *Proceedings of ACM SIGGRAPH 2004*, 23(2):399–405, 2004.

[112] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *Proceedings of ACM SIGGRAPH 2005*, 24(3), 2005.

[113] R. Szeliski and S. Lavallee. Matching 3-d anatomical surfaces with non-rigid deformations using using octree-splines. *International Journal of Computer Vision*, 18(2):171–186, 1996.

[114] Ben Taskar. Learning structured prediction models: A large margin approach. In *PhD Thesis*. Stanford University, 2004.

[115] L. Taycher, J. Fisher III, and Trevor Darrell. Recovering articulated model topology from observed motion. In *Proc. NIPS*, 2002.

[116] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, pages 2319–2323, Dec 2000.

[117] M. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision (ECCV)*, pages 447–460, 2002.

[118] O. Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, 1999.

[119] D. Vlasic, H. Pfister, M. Brand, and J. Popović. Face transfer with multilinear models. *ACM Transactions on Graphics*, 24(3), 2005.

[120] Daniel Vlasić, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. *ACM Transactions on Graphics*, 24(3), 2005.

[121] G. Wahba. Spline models for observational data. In *SIAM*, 1990.

[122] J. Wang and E. Adelson. Representing moving images with layers. In *Proc. IEEE on Image Processing Special Issue: Image Sequence Compression*, 1994.

[123] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 129–138, 2002.

[124] J. Wilhelms and A. V. Gelder. Anatomically based modeling. *Proceedings of ACM SIGGRAPH 97*, pages 173–180, 1997.

[125] R. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[126] Weiss Y and E.H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[127] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. In *Technical Report TR-2002-35*. Mitsubishi Electric Research Laboratories, 2002.

[128] J. Yedidia, W. Freeman, and Y Weiss. Understanding belief propagation and its generalizations. In *Exploring Artificial Intelligence in the New Millennium*. Science & Technology Books, 2003.

[129] S. Yu, R. Gross, and J. Shi. Concurrent object recognition and segmentation with graph partitioning. In *Proc. NIPS*, 2002.

[130] Ramin Zabih and Vladimir Kolmogorov. Spatially coherent clustering using graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[131] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: High-resolution capture for modeling and animation. In *ACM Annual Conference on Computer Graphics*, pages 548–558, August 2004.

[132] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20(1), 2001.