TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Echtzeitsysteme und Robotik

Stochastic and Deterministic Methods for 3D Shape Registration

Chavdar Papazov

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Thomas Huckle
Prüfer der Dissertation: 1. Univ.-Prof. Dr. Darius Burschka
2. Associate Prof. Dr. Antonis A. Argyros, Univ. of Crete/Griechenland

Die Dissertation wurde am 26.09.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 02.09.2013 angenommen.

Abstract

In this thesis, we develop deterministic and stochastic approaches for rigid and deformable 3D shape registration and the closely related problem of 3D object recognition and pose estimation.

We introduce an algorithm for the rigid registration of point clouds without any correspondence information available and without making any assumptions about the initial alignment of the input data. The problem is converted to the minimization of a non-linear cost function which we design to be especially robust to noise and outliers. In order to compute a globally optimal solution independent of the initial alignment of the point clouds, we present a new global optimization approach. It is an adaptive stochastic sampler which uses a generalized binary space partitioning tree and allows to efficiently optimize non-linear objective functions over complex shaped search spaces.

Rigid registration, as discussed in this thesis, assumes that there are only two shapes, e.g., two different views of the same physical object. If we want to simultaneously register multiple shapes to a shape which represents several objects, we end up with the problem of 3D object recognition and pose estimation. We develop a solution based on a robust geometric descriptor, a hashing technique and an efficient localized RANSAC-like sampling strategy. We show that the proposed sampling strategy significantly reduces the complexity of the method, namely, from cubic to linear. This allows to integrate the recognition in a robotic object grasping system without causing noticeable delays in the manipulation process.

Furthermore, we drop the restriction of rigidity and develop an algorithm for pairwise deformable 3D shape registration. We introduce a unified framework to study deformable registration together with deformation-based shape modeling which is a highly relevant research topic in computer graphics. Thus, both problems are put on the same theoretical footing which allows to investigate convergence issues and computational aspects of both problems in a unified way. Representing a shape by a collection of rigid cells connected to each other by elastic strings makes it possible to model elastic deformations and to develop provably numerically stable energy minimization methods.

Finally, the presented algorithms are evaluated on a variety of real data sets and compared with several state-of-the-art approaches. Furthermore, some of the methods developed in this thesis are currently employed in an industrial setting as part of an automated truck engine painting pipeline and others are part of a real-world robotic object grasping system.

Acknowledgements

First of all, I want to thank my supervisor Prof. Darius Burschka for giving me the opportunity to do my PhD in his group. I want to thank him for all the insightful discussions, for his advice and especially for giving me the freedom to build up and develop my own ideas.

The GRASP project provided the funding for the major part of my stay at TUM and thus made the development of this thesis possible. Many thanks go to all the colleagues involved in this project for making it an exciting forum for exchanging scientific ideas in the areas of robotics and computer vision.

Furthermore, I want to thank all the members of our chair for Robotics and Embedded Systems at TUM for being such great colleagues. Special thanks go to my colleagues and passionate cooks from the Hochbrück division, namely (sorted alphabetically), Elmar Mair, Susanne Petsch, Juan Carlos Ramirez de la Cruz and Oliver Ruepp, for the funny lunch breaks and epic table soccer battles.

I wish to thank my family, Mariya Papazova, Chavdar Petrov Papazov and Yavor Papazov as well as my girlfriend Martina Ivanova for their love and endless support. I also want to thank all my friends which I neglected so often especially in the tough phases of writing this thesis.

Contents

Chapter 1

Introduction

3D shape registration is a long-standing highly relevant research topic in the field of geometry processing with many important applications in domains as diverse as life sciences, reverse engineering, image processing and entertainment and multimedia. The closely related problem of 3D object recognition can be seen as the "Holy Grail" of computer vision: it is one of the oldest, most fundamental and still widely studied problems in this area. A computer system able to robustly and efficiently recognize its surrounding is very useful in robotics, manufacturing and human-machine interaction. In fact, the development of most of the algorithms presented here is motivated by questions arising in these fields.

Two major classes of shape registration and object recognition approaches can be distinguished: appearance-based methods (operating on 2D images) and methods working with 3D data (point clouds, meshes, etc). As the name suggests, the appearance-based algorithms use only the object's appearance usually captured by different 2D images [7]. This class can be further subdivided into local and global methods. The local approaches deal with so-called features which are small, distinctive parts of the object's projection onto the image, e.g., corners, edges or small patches with characteristic form or texture properties. Provided a set of input images showing the object(s) of interest, the usual strategy of the local methods (e.g., [8]) is first to detect a set of features, describe and store them in a database. Next, given an image representing the scene in which the objects are supposed to be recognized, the same procedure is repeated. However, the detected scene features are not stored but rather matched to those in the database in order to create object hypotheses. The last step consists of verifying the hypotheses using, e.g., RANSAC [9] or the generalized Hough transform [10]. Describing each of these steps in detail is beyond the scope of this thesis—the interested reader is referred to [11, 7].

The global appearance-based methods operate with the whole image or at least with a large portion of it. In a training phase, the information contained in the region of interest is projected to a lower-dimensional subspace in which a comparison can be performed more efficiently [7]. This is often achieved using dimensionality reduction techniques like principle

(a) (b)

Figure 1.1: (a) A good match between a template image representing the object of interest (left) and a scene (right). (b) However, it turns out, that the beer bottle is a sticker put on the surface of a mug (image courtesy of Radu Bogdan Rusu [16]).

component analysis (PCA) [12], independent component analysis (ICA) [13] and non-negative matrix factorization (NMF) [14]. The online phase can be performed by sliding a window over the scene image and testing each part whether it is consistent with some of the patches processed in the training phase. Refer to [15, 7] for more details.

Since the global approaches use more information than the local ones, the former are considered as more discriminative whereas the latter can better deal with occluded objects [7]. However, in order to be useful, all appearance-based methods should be invariant to translation, rotation, scale and perspective projection as well as to changes in illumination. Although significant progress has been made in this direction this still remains a very challenging task. In contrast, methods working with 3D data do not have to deal with these problems. As already mentioned above, a great deal of this work is motivated by applications in the field of robotic object manipulation. Vision-based robotic object grasping, in particular, needs object recognition and pose estimation methods which operate on 3D data and not on 2D images.

The reasons are twofold. First, in order to perform a stable grasp, it is very important to know the true 3D geometric shape of the object. The 2D appearance of an object may provide completely wrong hints about its shape. This is illustrated in Figure 1.1. The beer bottle is apparently successfully recognized and localized in the scene (Figure 1.1(a)). However, after zooming out (Figure 1.1(b)), it turns out that the bottle is actually a sticker put on the surface of a mug. A robot trying to grasp the bottle will most probably fail since the mug has a different geometry.

The second reason in favour of 3D techniques has to do with the pose of the object in space. Clearly, not only the 3D shape but also the position and orientation of the object is crucial for a successful grasp. The 2D object appearance may provide misleading information about the object pose since texture elements may be misaligned with respect to the object surface. This often happens to labels of household objects like grocery items.

These are the reasons why we put the focus of this thesis on the 3D processing techniques. However, there are also difficulties which go along with this paradigm. The acquisition of 3D data is anything but trivial. Compared to the 2D image formation in digital cameras, the process of 3D reconstruction is computationally more demanding and much more error-prone. Reviewing all available techniques is far beyond the scope of this thesis. Instead, we provide a short discussion on two common approaches along with their pros and cons.

The non-invasive (or passive) 3D reconstruction from a set of 2D images is a major class of 3D scene digitization approaches. Stereo reconstruction [17] and bundle adjustment [18] are two well-known representatives. These techniques have a great potential and very diverse application domains ranging from indoor robot navigation and exploration [19, 20] to building reconstruction from satellite image data [21]. Unfortunately, the methods tend to be noisy and produce rather sparse reconstructions in the case of textureless objects which are often encountered in home environments.

The invasive (or active) approaches belong to another class of reconstruction algorithms. They capture the scene geometry in an active way by, e.g., projecting a known light pattern in the scene. Using a digital camera the projection is captured and the depth information is inferred from the way the pattern is deformed [22, 23]. This active process makes the reconstruction more robust and precise and more independent on the particular surface properties of the scene objects. However, the light emitted by natural sources, like the sun, might interfere with the projected light pattern and prevent a successful reconstruction. This limits the usage of many active sensors to indoor close range scenarios with no direct sun irradiation. A further limitation of the active sensors is that only one can be active at a time since otherwise they could interfere each other.

No matter what 3D data acquisition technique is employed there are difficulties common to all of them: (i) noise and outliers are inevitable for any physical sensor, (ii) holes in the reconstruction are common, even for active sensors, in the case of highly specular surfaces and (iii) usually, only partial reconstructions are available since the sensors capture only the facing side of an object. For a shape registration or object recognition algorithm to be useful it has to deal adequately with all these problems. This is taken into account while developing the methods in this thesis.

Several approaches to rigid 3D shape registration convert the problem to the minimization of the sum of squared distances between corresponding points on the input shapes [24, 25, 26]. This results in methods which are sensitive to noise and outliers. We design a cost function which is much more robust to noise, outliers and missing data. Furthermore, a common feature of most existing algorithms [24, 27, 25, 26] is to use a local optimization method which makes the registration result strongly dependent on the initial alignment of the shapes. To avoid this dependency, we develop a new stochastic approach for global optimization.

Many 3D object recognition approaches rely on finding corresponding features between the model shapes and the scene [28, 29, 30, 31, 32, 33]. This, however, is based on the assumption

Figure 1.2: Rigid registration of two range scans.

that the objects of interest have distinctive points, e.g., the tip of a nose of a face or the corners of the eyes. Unfortunately, many objects, like cylinders or spheres, do not possess such unique parts. This makes the feature-based methods to degenerate to brute force search [34]. A further strategy is to model the objects as an assembly of simple parametric shapes (primitives) and to detect those shapes and their spatial relationships in an input scene [35, 36, 37]. This approach limits the recognition system to objects which can be approximated reasonably well by the chosen type of primitives. In contrast, we do not pose any restrictions on the geometry of the objects—they can be as simple as spheres or as complicated as children's toys like teddy bears, rabbits etc.

Similar to the rigid version, the deformable 3D shape registration problem is also often solved by minimizing a cost function defined on a suitable deformation space. Several authors parametrize the shape deformation by using one affine matrix per shape point [38, 39, 40]. This gives rise to highly underdetermined systems which are usually regularized by introducing additional stiffness terms. The resulting cost functions have many unknowns which leads to high-dimensional and computationally heavy optimization problems usually solved with general-purpose optimizers. Instead, we design a framework tailored to the problem at hand and develop an energy minimization algorithm which is very efficient in terms of both computational complexity and memory.

More detailed discussions on related work will be given in Chapters 2, 3 and 4.

1.1 What is 3D Shape Registration and 3D Object Recognition?

To put it briefly, 3D shape registration is the problem of finding a spatial transform which aligns one shape to another. This "definition" gives rise to several questions: (i) what is a *shape*, (ii) what *type of transform* are we looking for and (iii) how do we measure *alignment* quality? Precise answers will be given later in the chapters of the thesis. At this point, we will provide an intuitive idea to the reader:

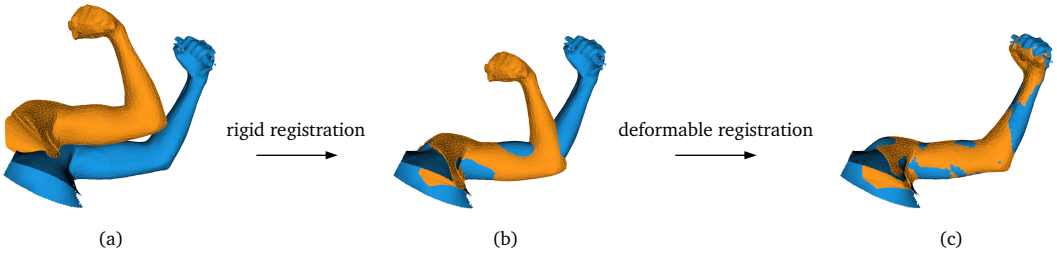


Figure 1.3: Rigid and deformable registration of two 3D shapes.

(i) A 3D shape is a bounded subset of 3D space representing a real, physical object. Examples of shapes are point clouds, surface meshes, volumetric meshes and voxel grids, just to name a few

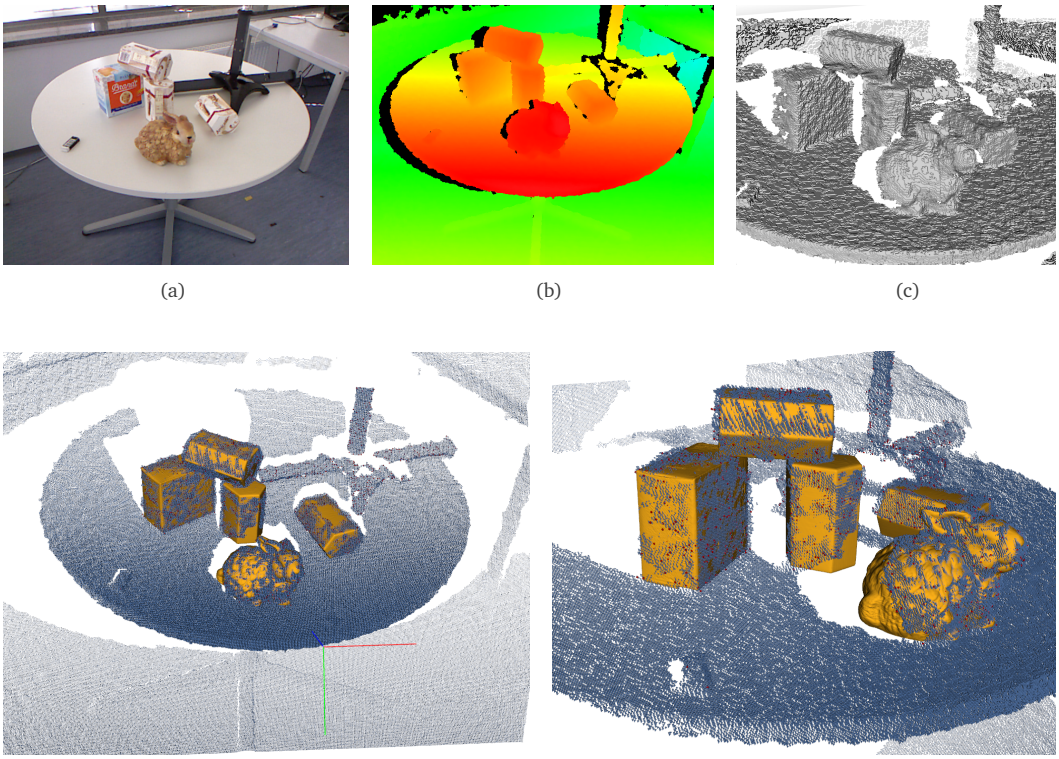
(ii) We are looking for a spatial transform which brings one of the two input shapes close to the other. In the case of rigid shape registration, it is a rigid transform, whereas in the more general case of deformable registration, it is a not too distortive mapping from a shape to 3D space.

(iii) Alignment quality is measured by means of a real-valued cost (or energy) function which is defined on the shape to be aligned.

The concept of rigid/deformable 3D shape registration is best explained by examples. Figure 1.2 shows two scans of a doll head. After capturing the front side, the model was rotated and scanned again in order to be reconstructed more completely. However, the scans are misaligned when rendered in the coordinate system of the scanner (Figure 1.2(a)). This is due to the fact that the rotation was not taken into account. A rigid registration of the scans leads to the desired result (Figure 1.2(b)).

In some situations, like scanning an object while it undergoes a non-rigid motion, solely applying rigid transforms is not sufficient to bring the shapes in a close correspondence. Figure 1.3(a) shows two arm scans which differ by an articulated motion. Obviously, a rigid registration does not lead to a precise alignment (Figure 1.3(b)), whereas a deformable registration does (Figure 1.3(c)).

A problem closely related to 3D shape registration is the one of 3D object recognition and pose estimation. The setting is the following: instead of having two shapes (partially) representing one and the same physical object, we have multiple shapes, called models, and a further data set, called scene, which contains parts (of some) of the models. The task is to identify the models that are present in the scene and to determine their position and orientation. In other words, we have the task to register several shapes (object models) to another shape (the scene) which contains subsets of the models. In this sense, object recognition and pose estimation can be viewed as many-to-one shape registration. Figure 1.4 provides an example.



(a) (b) (c)

(d)

Figure 1.4: 3D object recognition and pose estimation. (a) The scene consisting of several objects on a table. In this particular example, the objects the system is supposed to recognize and localize are the boxes and the rabbit. The black objects on the right and the rubber on the left are not of interest. (b) Depth map of the scene computed by a Kinect sensor [41]. Red corresponds to scene points close to the sensor and blue to those which are further away. (c) The 3D scene reconstruction provided by Kinect. (d) The recognition and pose estimation result shown from two different viewpoints. The scene points are shown in blue and the recognized models are shown as yellow meshes and superimposed over the scene.

1.1.1 Applications in Science and Technology

As already mentioned at the beginning, both 3D shape registration and 3D object recognition and pose estimation have a variety of important applications in many areas of science and technology. We include a brief list of specific applications in several different domains.

- *Life Sciences:*
  - A topic that receives much attention in biology and medicine is the creation of so-called standard spatial atlases. Such an atlas is a geometrical model of, e.g., a human organ built by averaging multiple surfaces each one representing an organ of a particular individual. In order to compute the atlas, the input surfaces have to be

aligned to each other in a rigid or deformable manner in order to establish pointwise correspondences needed for the average computation [42].

  - Medical image segmentation can also benefit from robust rigid/deformable shape registration techniques. For example, in order to segment a human organ in a computer tomography (CT) image set, an atlas can be rigidly or deformably registered to the image stack by defining a force field attracting the atlas surface to the organ boundaries in the CT scan.

  - A further topic of interest in the life sciences is locomotion analysis. The first step in this process is to extract the motion of a particular patient. This can be done using deformable shape registration techniques as follows. Given a mesh model of the object of interest and an input stream of range scans capturing the motion, the model can be sequentially registered to the scans. An appropriate temporal interpolation of the mesh vertex positions yields a trajectory from the first to the last frame in the scan sequence. This information can be used to extract the underlying skeletal motion which is the one needed for locomotion analysis.

- *Robotics:* Vision-based object manipulation is an active research topic in robotics. In order to operate safely and robustly in chaotic, uncontrolled and dynamically changing environments, robots can not rely on hard-coded knowledge about that environment. Efficient and robust 3D object recognition techniques are very useful to provide the necessary updates of the scene knowledge.

- *Human-Machine Interaction:* A direction of research within the area of human-machine interaction deals with new interaction methods which go beyond the desktop metaphor such as multi-touch tabletops. One example is the re-appropriating of everyday objects as interactive devices: suppose a speaker is about to give a talk but she forgot to take a laser pointer. Is there any non-technical reason for not being able to pickup a pen, point towards the projection screen and let a red dot appear at the right spot? A 3D object recognition and pose estimation algorithm can be of great help for the technical realization of this scenario: the pen in the hand of the speaker and the projection screen can be localized and the pointing direction of the pen can be determined in order to compute the intersection point with screen.

- *Entertainment Industry:* Motion capturing is the process of digitizing the movements of one or more real actors. In the context of film and game production, the recorded moves are transferred to a virtual character to make its motion natural and realistic looking. Similar to the locomotion analysis discussed above, a deformable shape registration technique can be used to capture the actor's motion.

1.2 Contributions and Overview

The contributions of this thesis are the following:

1. We develop a new method for robust rigid 3D shape registration.
   (a) Its distinctive feature is extreme robustness to noise and outliers in the data sets. This is achieved by carefully designing a cost function based on an inverse distance kernel known to be insensitive to noise and outliers.
   (b) A globally optimal alignment is achieved by a new efficient stochastic global optimization algorithm.
   (c) A new technique of decomposing the space of rotations in equally sized parts is presented and used in conjunction with the stochastic optimization method.
2. A new efficient 3D object recognition and pose estimation algorithm is introduced. It is based on a robust geometric descriptor, a hashing technique and an efficient localized RANSAC-like sampling strategy. No assumptions about the geometric complexity of the shapes are made. Furthermore, the main procedure in the algorithm has a linear time complexity and is highly parallelizable which makes it possible to use the algorithm for, e.g., robotic object manipulation tasks without causing noticeable delays in the overall process.
3. We introduce a unified framework for deformation-based shape modeling and deformable 3D shape registration. Using the same theoretical framework allows to simultaneously solve both problems and prove the convergence of the resulting energy minimization algorithms.

The rest of the thesis is organized as follows. In Chapter 2, we introduce the rigid 3D shape registration method. Chapter 3 is about 3D object recognition and pose estimation. Chapter 4 introduces the unified framework for 3D shape modeling and deformable 3D shape registration. In Chapter 5, the proposed methods are experimentally validated on a variety of real data and compared with several state-of-the-art approaches. Chapter 6 draws some conclusions and discusses possible directions of future research. Appendix A demonstrates how our 3D object recognition and pose estimation method can be used to support the robotic grasping of known objects in occluded, dynamically changing environments. Appendix B includes an application of our deformable registration algorithm to the problem of knowledge transfer between geometric models.

Chapter 2

Stochastic Optimization for Rigid 3D Shape Registration

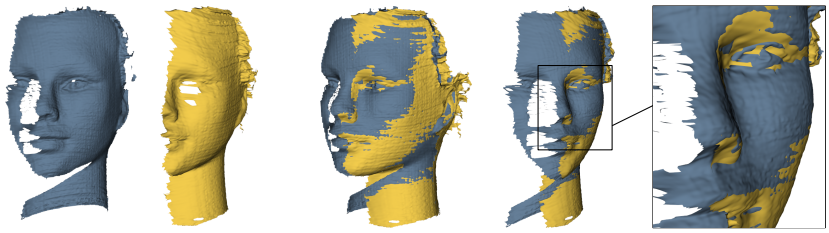
In this chapter, we introduce a rigid 3D shape registration algorithm based on global stochastic optimization of a noise and outlier robust cost function. The problem at hand is formulated as follows. Given two finite point sets $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$, find a rigid transform $T : \mathbb{R}^3 \to \mathbb{R}^3$, such that the point set $T(\mathbf{D}) = \{T(\mathbf{y}_1), \ldots, T(\mathbf{y}_n)\}$ is optimally aligned in some sense to $\mathbf{M}$. $\mathbf{M}$ is referred to as the model and $\mathbf{D}$ as the data (point) set. Note that in this chapter, we do not need any connectivity information, i.e., the input shapes are represented by point clouds. Points from $\mathbf{M}$ and $\mathbf{D}$ are called model points and data points, respectively. In Figure 2.1, a model and a data set are shown before and after rigid registration.

Even though T is restricted to be a rigid transform, i.e., $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$, with R being a rotation matrix and $\mathbf{t}$ a translation vector, the problem remains hard because of several reasons: (i) no initial pose estimation is available, (ii) no correspondences between the points are known, (iii) the point sets are incomplete and (iv) corrupted by noise and outliers.

Contributions and Chapter Overview

The algorithm we propose in this chapter aims to robustly solve the rigid point set registration problem in the case of noisy, outlier corrupted and incomplete point sets with unknown correspondences between the points. Our main contributions are (i) a noise and outlier resistant cost function, (ii) a stochastic approach for its global minimization, (iii) a technique for a hierarchical decomposition of the rotation space in disjoint parts of equal volume and (iv) a procedure for uniform sampling from spherical boxes.

The rest of the chapter is organized as follows. After reviewing related work in Section 2.1, we define, in Section 2.2, the task of aligning two point sets as a minimization problem and introduce our cost function. Section 2.3 presents a stochastic approach for global minimization.



(a) (b) (c) (d) (e)

Figure 2.1: Rigid point set registration obtained with our method. The input point sets, model and data, are shown in (a) and (b), respectively. Although rendered as meshes no surface information (like normals) is used for the registration. Note that the scans are noisy and only partially overlapping. (c), (d) Our registration result (shown from two different viewpoints) obtained without noise filtering, local ICP refinement [24] or any assumptions about the initial pose of the input scans. (e) A closer view of the part marked by the rectangle in (d). Observe the high quality of the alignment.

In Section 2.4, we motivate the choice of the rotation space parametrization we use in combination with our minimization approach and introduce a technique for a hierarchical rotation space decomposition. Furthermore, a procedure for uniform sampling from spherical boxes is described.

2.1 Related Work

2.1.1 Rigid Point Set Registration

One class of rigid point set registration approaches consists of methods designed to solve the initial pose estimation problem[1]. These methods compute a more or less coarse alignment between the point sets without making any assumptions about their initial position and orientation in space. Classic initial pose estimators are the generalized Hough transform [43], geometric hashing [44] and pose clustering [45]. These algorithms are guaranteed to find the optimal alignment between the input point sets. However, because of their high computational cost and/or high memory requirements, these methods are only applicable to small data sets.

Johnson et al. introduced in their work [28] local geometric descriptors, called spin images, and used them for pose estimation and object recognition. The presented results are impressive, but no tests with noisy or outlier corrupted data were performed. Gelfand et al. [29] developed a local descriptor which performs well under artificially created noisy conditions, but still, defining robust local descriptors in the presence of significant noise and a large amount of outliers remains a difficult task.

A more recent approach to the initial pose estimation problem is the robust 4PCS algorithm introduced by Aiger et al. [34]. It is an efficient randomized generate-and-test approach. It

[1] Pose = position (translation) + orientation (rotation).

selects an appropriate quadruple $\mathbf{Q}$ (called a basis) of nearly coplanar points from the model set $\mathbf{M}$ and computes the optimal rigid transform between $\mathbf{Q}$ and each of the potential bases in the data set $\mathbf{D}$ and chooses the best one. In order to achieve a high success probability, the procedure is repeated several times for different bases $\mathbf{Q} \subset \mathbf{M}$. Note, however, that the rigid transform, found by the algorithm, is optimal only for the two bases, i.e., for eight points. In contrast to this, the rigid transform computed by our rigid registration method is optimal for a large amount of points of the input shapes and thus we expect to achieve higher accuracy than the 4PCS algorithm. This is further validated in the experimental results in Section 5.1.

Since the accuracy of the pose computed by the above mentioned methods is insufficient for many applications, an additional pose refinement step needs to be performed. The pose refining algorithms represent another class of registration approaches. The most popular one is the Iterative Closest Point (ICP) algorithm. Since its introduction by Chen and Medioni [46], and Besl and McKay [24], a variety of improvements has been proposed in the literature. A good summary as well as results in acceleration of ICP algorithms have been given by Rusinkiewicz and Levoy [47]. A major drawback of ICP and all its variants is that they assume a good initial guess for the pose of the data point set (with respect to the model). This pose is improved in an iterative fashion until an optimal rigid transform is found. The quality of the solution heavily depends on the initial guess.

Recently, a variety of registration algorithms based on robust statistics has been proposed. Granger and Pennec [48] formulated the rigid point set registration as a general maximum likelihood estimation problem which they solved using expectation maximization principles. Tsin and Kanade [49] introduced the kernel correlation approach as an extension of the well-known 2D image correlation technique to point sets. The shapes are represented by a collection of kernel functions each one centered at a model/data point. If each point in the model set has a close counterpart in the data set the kernel correlation value is large. Thus the registration problem is converted to the maximization of the kernel correlation of the input point sets. An extension of this approach through a Gaussian mixture model was proposed by Jian and Vemuri [50]. Instead of using one-to-one correspondences between the points of the input sets, the above cited methods work with multiple, weighted correspondences. Although this significantly widens the basin of convergence the resulting computational cost limits the applicability of the algorithms to small point sets only [51].

A further class of rigid registration methods is based on particle filtering. Ma and Ellis [52] pioneered the use of the unscented particle filter for registration of surfaces in the context of computer-assisted surgery. A major limitation of the method is its running time: it takes 1.5 seconds for a data set consisting of 15 points. Moreover, outlier robustness was not addressed by the authors. Further interesting approaches from this class are the algorithm of Moghari and Abolmaesumi [53] which is based on the unscented Kalman filter and the point set registration method via particle filtering and stochastic dynamics introduced by Sandhu et al. [54]. Although

these algorithms have a band of convergence significantly wider than the one of local optimizers, they still depend on the initial alignment of the point sets.

2.1.2 Optimization-Based Point Set Registration

Solving the registration problem by minimizing a cost function with a general-purpose optimizer has already been introduced in the literature. Depending on the choice of either a global or a local optimization procedure the corresponding registration approach belongs to the class of initial pose estimators or pose refining methods, respectively.

Breuel [55] used a deterministic branch-and-bound method to globally maximize a quality measure which counts the number of data points a given rigid transform brings within an ϵ-neighborhood of some model point. Although this method always finds the global optimal solution its computational cost seems to be very high since only planar rigid transforms (with three degrees of freedom) were considered.

Olsson et al. [56] also used a deterministic branch-and-bound algorithm to globally minimize the sum of squared distances between corresponding entities (points, lines or planes) in the model and data sets. This method is guaranteed to find the global optimal solution, however, at a high computational cost: a problem consisting of 10 point-to-plane, 4 point-to-line and 4 point-to-point correspondences is solved in about 10 seconds. Furthermore, when applied in the case of point set registration, the correspondences between the points have to be known in advance which is seldom the case in a real world setting.

Another deterministic solver based on Lipschitz global optimization theory was introduced by Li and Hartley [57]. On the positive side, the method does not assume any known correspondences across the point sets and it always solves the problem in a globally optimal way. Unfortunately, the algorithm is very costly (about 18 minutes for input sets consisting of 200 points each) and it is based on some unrealistic assumptions: (i) the model and data sets have exactly the same number of points, (ii) there are no outliers and (iii) there is no missing data, i.e., there is a 100% overlap between model and data.

Mitra et al. [25], Pottmann et al. [26] and Fitzgibbon [27] also formulated the registration problem as a minimization of a geometric cost function. For its minimization, however, a local optimization method is used. This results in the already mentioned strong dependence on a good initial transform estimation.

2.1.3 Stochastic Optimization

Stochastic optimization has received considerable attention in the literature over the last three decades. Much of the work has been devoted to the theory and applications of simulated annealing (SA in the following) as a minimization technique [58, 59, 60]. A comprehensive overview of this field is given in [61]. A major property of SA algorithms is their "willingness" to explore regions in the search space in which the objective function apparently takes values

greater than the current minimum [62]. This is what makes SA algorithms able to escape from local minima and makes them suitable for global minimization. A known drawback of SA algorithms is the fact that they waste a lot of iterations in generating candidate points, evaluating the objective function at these points, and finally rejecting them [61]. In order to reduce the number of rejections, Bilbro and Snyder [63] select candidate points from "promising" regions of the search space, i.e., from regions in which the objective function is likely to have low values. They achieve this by adapting a k-d tree to the objective function each time a new candidate point is accepted. If, however, the current point is rejected, the tree remains unchanged. This is a considerable waste of computation time since the information gained by the (expensive) evaluation of the objective function is not used. In contrast to this, our algorithm adapts a generalized BSP tree at every iteration and thus uses all the information collected during the minimization. Furthermore, the use of a generalized BSP tree allows for a minimization over complex shaped spaces and not only over rectangular regions as in the case of [63].

2.2 Registration as a Minimization Problem

Consider, we are given a model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and a data set $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$. Suppose, we have a continuous function $\mathbf{g} : \mathbb{R}^3 \to \mathbb{R}$, called the model scalar field, which attains a small value at the model points $\mathbf{x}_i,\ i \in \{1, \ldots, m\}$ and increases with increasing distance between the evaluation point and the closest model point. Our aim is to find a rigid transform $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix R and a translation vector $\mathbf{t}$ such that the function

$$\mathbf{f}(T) = \sum_{j=1}^{n} \mathbf{g}(T(\mathbf{y}_j)), \quad \mathbf{y}_j \in \mathbf{D} \tag{2.1}$$

is minimized. This definition of $\mathbf{f}$ is based on the following idea common for many registration algorithms: we seek a rigid transform that brings the data points as close as possible to the model points.

2.2.1 Definition of the Model Scalar Field

Given the model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, we want our model scalar field $\mathbf{g} : \mathbb{R}^3 \to \mathbb{R}$ to attain its minimal value at the model points, i.e.,

$$\mathbf{g}(\mathbf{x}_i) = g_{\min} \in \mathbb{R}, \quad \forall \mathbf{x}_i \in \mathbf{M}, \tag{2.2}$$

and to attain greater values at all other points in $\mathbb{R}^3$, i.e.,

$$\mathbf{g}(\mathbf{x}) > g_{\min}, \quad \forall \mathbf{x} \in \mathbb{R}^3 \setminus \mathbf{M}. \tag{2.3}$$

Define

$$\mathbf{d}_{\mathbf{M}}(\mathbf{x}) = \min_{\mathbf{x}_i \in \mathbf{M}} \|\mathbf{x} - \mathbf{x}_i\| \tag{2.4}$$

to be the distance between a point $\mathbf{x} \in \mathbb{R}^3$ and the set $\mathbf{M}$, where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^3$. If we set

$$\mathbf{g}(\mathbf{x}) = \mathbf{d}_{\mathbf{M}}(\mathbf{x}), \tag{2.5}$$

we get an unsigned distance field which is employed in ICP [24]. It is obvious that this choice for $\mathbf{g}$ fulfills both criteria (2.2) and (2.3).

Mitra et al. [25] and Pottmann et al. [26] considered in their work more sophisticated scalar fields. They assumed that the model point set $\mathbf{M}$ consists of points sampled from an underlying surface Φ. The scalar field $\mathbf{g}$ at a point $\mathbf{x} \in \mathbb{R}^3$ is defined to be the squared distance from $\mathbf{x}$ to Φ. In this context, $\mathbf{g}$ is called the squared distance function to the surface Φ. We refer to [25] for details on computing the squared distance function and its approximation for point sets.

The version of $\mathbf{g}$ given in (2.5) and the ones used by Mitra et al. [25] and Pottmann et al. [26] are essentially distance fields. This means that $\mathbf{g}(\mathbf{x})$ approaches infinity as $\mathbf{x}$ gets infinitely far from the point set. This has the consequence that a registration technique which minimizes a cost function based on an unbounded scalar field will be sensitive to outliers in the data set. This is because data points lying far away from the model point set will have great impact on the sum in (2.1) and thus will prevent the minimization algorithm from converging towards the right alignment. A similar problem arises in the case of model and data sets with low overlap. In this case, there will be a lot of data points which have no corresponding model points and vice versa. The distance between such a data point and the closest model point will be large and thus will deteriorate the sum in (2.1). A simple way to overcome this is just to exclude data points which are too far away from the model set. However, this strategy introduces discontinuities in the cost function which cause a problem for many optimization methods.

Fitzgibbon presented in his work [27] a more convenient way to alleviate these difficulties which does not lead to a discontinuous cost function. He proposed to use either of the following two robust kernels:

$$\mathbf{g}(\mathbf{x}) = \log\left(1 + \frac{\mathbf{d}_{\mathbf{M}}^2(\mathbf{x})}{\sigma}\right) \quad \text{(Lorentzian kernel)} \tag{2.6}$$

or

$$\mathbf{g}(\mathbf{x}) = \begin{cases} \mathbf{d}_{\mathbf{M}}^2(\mathbf{x}) & \text{if } \mathbf{d}_{\mathbf{M}}(\mathbf{x}) < \sigma \\ 2\sigma\mathbf{d}_{\mathbf{M}}(\mathbf{x}) - \sigma^2 & \text{otherwise} \end{cases} \quad \text{(Huber kernel).} \tag{2.7}$$

However, we still have $\lim_{\mathbf{d}_{\mathbf{M}}(\mathbf{x})\to\infty} \mathbf{g}(\mathbf{x}) = \infty$ for both kernels as in the case of (2.5). Thus a cost function based on (2.6) or (2.7) will still be sensitive to outliers. We further validate this in the experimental results presented in Section 5.1.

To avoid this sensitivity, we propose to use a bounded scalar field satisfying (2.2) and (2.3) and having the additional property

$$\lim_{\mathbf{d}_{\mathbf{M}}(\mathbf{x})\to\infty} \mathbf{g}(\mathbf{x}) = 0. \tag{2.8}$$

We set

$$\mathbf{g}(\mathbf{x}) = -\varphi\left(\mathbf{d}_{\mathbf{M}}(\mathbf{x})\right), \tag{2.9}$$

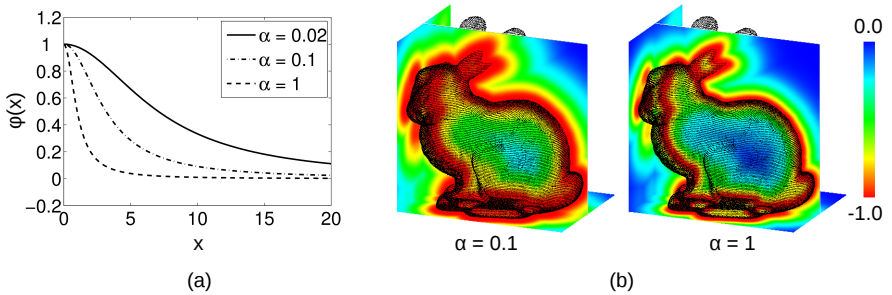


Figure 2.2: (a) The inverse distance kernel (defined in (2.12)) for three different α values. (b) The model scalar field $\mathbf{g}_\alpha$ (defined in (2.13)) based on the inverse distance kernel from (a) for $\alpha = 0.1$ and $\alpha = 1$. In this example, the Stanford bunny is used as the model set. $\mathbf{g}_\alpha$ is visualized by evaluating it at a number of points lying on the three planes and mapping the scalar values to colors.

where $\varphi : \mathbb{R}^* \to \mathbb{R}^*$, for $\mathbb{R}^* = \{x \in \mathbb{R} : x \geq 0\}$, is a strictly monotonically decreasing continuous function with

$$\max_{x \in \mathbb{R}^*} \varphi(x) = \varphi(0) \quad \text{and} \tag{2.10}$$

$$\lim_{x \to \infty} \varphi(x) = 0. \tag{2.11}$$

In our implementation, we use an inverse distance kernel of the form

$$\varphi(x) = \frac{1}{1 + \alpha x^2}, \quad \alpha > 0 \tag{2.12}$$

because it is computationally efficient to evaluate and can be controlled by a single parameter α (see Figure 2.2(a)). This results in the following model scalar field:

$$\mathbf{g}_\alpha(\mathbf{x}) = -\frac{1}{1 + \alpha \left(\mathbf{d}_\mathbf{M}(\mathbf{x})\right)^2}, \quad \alpha > 0. \tag{2.13}$$

It is easy to see that (2.2), (2.3) and (2.8) hold. Different values for α in (2.13) lead to different scalar fields. The greater the value the faster $\mathbf{g}_\alpha(\mathbf{x})$ convergences to zero as $\mathbf{d}_\mathbf{M}(\mathbf{x}) \to \infty$ (see Figure 2.2(b)). In Section 2.2.2, we will discuss how to choose a suitable value for α and why this particular form of $\mathbf{g}_\alpha$ leads to an outlier robust cost function.

2.2.2 Cost Function Definition

The group of all rigid transforms in $\mathbb{R}^3$ is called the special Euclidean group and is denoted by $SE(3)$. At the beginning of Section 2.2, we formulated the rigid point set registration problem as a minimization problem over $SE(3)$. Using a parametrization of $SE(3)$ we convert $\mathbf{f}$ in (2.1) to a real-valued scalar field $\mathbf{f} : \mathbb{R}^6 \to \mathbb{R}$ of the form

$$\mathbf{f}(\varphi, \psi, \theta, x, y, z) = \sum_{j=1}^{n} \mathbf{g}_\alpha(R(\varphi, \psi, \theta)\mathbf{y}_j + (x, y, z)), \tag{2.14}$$

where $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are the data points, $\mathbf{g}_\alpha$ is the model scalar field defined in (2.13), $R(\varphi, \psi, \theta)$ is a rotation matrix parametrized by φ, ψ, θ and $(x, y, z) \in \mathbb{R}^3$ is a translation vector. In order to achieve good optimization performance, it is very important to choose the appropriate parametrization of the rotation group. We employ an axis-angle based parametrization which is especially well suited for our branch and "stochastic bound" minimization method. Furthermore, we introduce a new technique for a hierarchical decomposition of the rotation space in spherical boxes and describe a procedure for uniform sampling from them. Since the advantages of these techniques are best seen in the context of our minimization algorithm we postpone the detailed discussion to Section 2.4 after the introduction of the minimization method in Section 2.3.

A global minimizer of $\mathbf{g}_\alpha$ defines a rigid transform that brings the data points close to the model points. What makes the proposed cost function robust to outliers is the fact that outlier data points have a marginal contribution to the sum in (2.14) depending on α. More precisely, given a positive real number d, we can compute a value for α such that $|\mathbf{g}_\alpha(\mathbf{x})|$ is less than an arbitrary $\delta > 0$, if $\mathbf{d}_\mathbf{M}(\mathbf{x}) > d$ holds. In this way, the contribution of an outlier point to the sum in (2.14) can be made arbitrary close to zero and $\mathbf{g}_\alpha$ will behave like an outlier rejector. However, too large values for α will lead to the rejection of data points which do not have exact counterparts in a sparsely sampled model set, but still are not outliers. In our implementation we set

$$d = \frac{1}{4} \min\{\mathrm{bbox}_x(\mathbf{M}), \mathrm{bbox}_y(\mathbf{M}), \mathrm{bbox}_z(\mathbf{M})\}, \tag{2.15}$$

$$\delta = 0.1, \tag{2.16}$$

where $\mathrm{bbox}(\mathbf{M})$ denotes the bounding box of the model point set and $\mathrm{bbox}_s(\mathbf{M})$, $s \in \{x, y, z\}$ is the extent of the bounding box along the x, y or z axis. Using the absolute value of the right side of (2.13) and solving for α yields

$$\alpha = \frac{1 - \delta}{\delta d^2}. \tag{2.17}$$

The cost function given in (2.14) is nonconvex and has multiple local minima over the search space (see [57] where this is experimentally verified for a similar cost function). Using a local optimization procedure—common for many registration methods—will lead in most cases to a local minimizer of $\mathbf{f}$ and thus will not give the best alignment between model and data. To avoid this, we employ a new stochastic approach for global minimization described in the following section.

2.3 Stochastic Adaptive Search for Global Minimization

Our stochastic minimization approach is inspired by the simulated annealing (SA) method of Bilbro and Snyder [63]. The main difference between their work and a typical SA algorithm is the way how the minimizer candidates are generated. As we already mentioned in Section 2.1.3,

SA algorithms are known to waste many iterations in sampling candidate points in the search space, evaluating the cost function at these points and finally rejecting them [61]. In order to reduce the number of rejections, Bilbro and Snyder [63] sampled the points from a distribution which is modified iteratively during the minimization such that its modes are built around minimizers of the cost function. They achieved this by building a k-d tree and sampling the candidates from those leaves of the tree which cover "promising" regions of the search space, i.e., regions in which the cost function is likely to attain low values. Although this leads to fewer candidate rejections and thus saves computation time the method in [63] still has two drawbacks. First, the candidate points are sampled directly from the tree leaves which are n-dimensional boxes of the form $[a_1, b_1] \times ... \times [a_n, b_n]$, where $[a_i, b_i] \subset \mathbb{R}$ is a closed interval. This strategy is based on the implicit assumption that the search space can be covered efficiently by such boxes. This, however, is not the case if we have a more complex shaped space, e.g., the space of rotations (see Section 2.4). Second, the k-d tree used in [63] is updated only if the generated candidate is accepted. In the case of a rejection, the tree remains unchanged. This is a waste of computation time since the information gained by the (expensive) cost function evaluation is not used.

We account for the first drawback by formulating our minimization algorithm using a more general spatial data structure, namely, a generalized binary space partitioning tree (which we will call a G-BSP tree in the following). As opposed to classic BSP trees (see, e.g., [64]), we do not require that the subspaces represented by the tree nodes are convex sets. Thus we can minimize efficiently over more complex shaped search spaces like, e.g., the space of rotations (details are provided in Section 2.4). To avoid the second drawback, i.e., to use all the information gained by the cost function evaluation, we update the tree at every iteration—even in the cases of bad minimizer candidates. This apparently minor modification leads to a rather different algorithm than [63] and enables a faster rejection of the regions in which the cost function is likely to have high, i.e., poor values and thus speeds up the convergence.

2.3.1 Generalized BSP Trees

A BSP tree is a spatial data structure which decomposes $\mathbb{R}^n$ in a hierarchical manner. At each subdivision stage, the space is subdivided by a hyperplane in two disjoint partitions of arbitrary size. Thus the resulting decomposition consists of arbitrarily shaped convex polygons [64]. Each node of the tree has exactly two or zero child nodes. A node with zero children is called a leaf. If we drop the assumption that the space partitioning is performed by planes we get a generalized BSP tree (G-BSP tree). This results in a decomposition made up of subspaces of arbitrary shape.

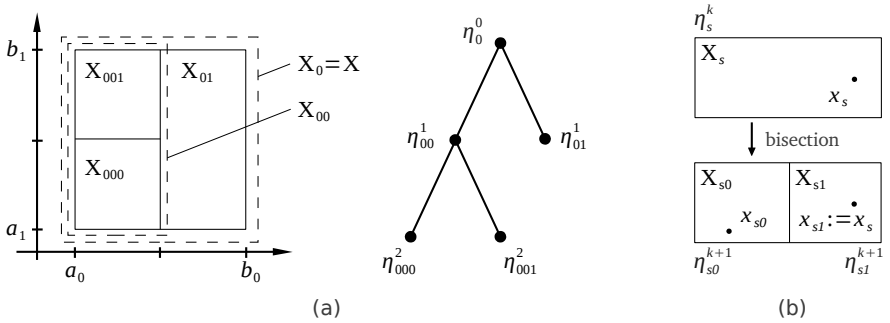


Figure 2.3: (a) An example of a two-dimensional G-BSP tree and a rectangular search space $\mathbf{X}$. In this case, the G-BSP tree is a two-dimensional k-d tree. (b) Expanding the leaf η_s^k. In this example, after the bisection of η_s^k, the point $\mathbf{x}_s$ lies in the box $\mathbf{X}_{s1}$, hence η_{s1}^{k+1} adopts the pair $(\mathbf{x}_s, f(\mathbf{x}_s))$ from η_s^k. For the other child, η_{s0}^{k+1}, a point $\mathbf{x}_{s0}$ is sampled uniformly from $\mathbf{X}_{s0}$ and the objective function is evaluated at that point.

2.3.2 Problem Definition

Given a set $\mathbf{X} \subset \mathbb{R}^n$ (the search space) and a bounded function $\mathbf{f} : \mathbf{X} \to \mathbb{R}$ our aim is to find a global minimizer of $\mathbf{f}$, i.e., an $\mathbf{x}^* \in \mathbf{X}$ such that

$$\mathbf{f}(\mathbf{x}^*) \leq \mathbf{f}(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbf{X}. \tag{2.18}$$

The following assumptions about $\mathbf{X}$ should hold:

- $\mathbf{X} \subset \mathbb{R}^n$ is a bounded set of positive volume (Lebesgue measure in $\mathbb{R}^n$).
- There is an algorithm of acceptable complexity which can build a G-BSP tree for $\mathbf{X}$ such that each two subsets of $\mathbf{X}$ at the same level of the tree are of equal volume (have the same Lebesgue measure in $\mathbb{R}^n$).
- $\mathbf{X}$ is simple enough for sampling algorithms of acceptable complexity to be able to sample uniformly from the G-BSP tree nodes, i.e., from the subsets of $\mathbf{X}$ represented in the G-BSP tree.

2.3.3 Overall Algorithm Description

We use a G-BSP tree to represent the n-dimensional search space $\mathbf{X}$. The root η_0^0 is at the 0th level of the tree and represents the whole space $\mathbf{X}_0 = \mathbf{X}$. η_0^0 has two children, η_{00}^1 and η_{01}^1, which are at the next level. They represent the subsets $\mathbf{X}_{00}$ and $\mathbf{X}_{01}$, respectively, which are disjoint, have equal volume and their union equals $\mathbf{X}_0$. In general, a node η_s^k (where $k \geq 0$ and s is a binary string of length $k+1$) is at the kth level of the tree and has two children, η_{s0}^{k+1} and η_{s1}^{k+1}, which are at the next, $(k+1)$th, level. The volume of η_s^k is $1/2^k$ of the volume of $\mathbf{X}$. This concept is easily visualized in the case $n = 2$ and $\mathbf{X}$ and its subsets being rectangles (see Figure 2.3(a)).

During the minimization, the G-BSP tree is built in an iterative fashion beginning at the root. The algorithm adds more resolution to promising regions in the search space, i.e., the tree is built with greater detail in the vicinity of points in $\mathbf{X}$ at which the objective function attains low values. The overall procedure can be outlined as follows:

1. Initialize the tree (Section 2.3.4) and set an iteration counter $j = 0$.
2. Select a "promising" leaf according to a probabilistic selection scheme (Section 2.3.5).
3. Expand the tree by bisecting the selected leaf. This results in the creation of two new child nodes. Evaluate the objective function at a point which is uniformly sampled from the subset of one of the two children (Section 2.3.6).
4. If a stopping criterion is met (Section 2.3.7) terminate the algorithm, otherwise increment the iteration counter j and go to step 2.

2.3.4 Tree Initialization

For every tree node η_s^k the following items are stored: (i) a set $\mathbf{X}_s \subset \mathbf{X}$ and (ii) a pair $(\mathbf{x}_s, \mathbf{f}(\mathbf{x}_s))$ consisting of a point $\mathbf{x}_s$, uniformly sampled from $\mathbf{X}_s$, and the corresponding function value $\mathbf{f}(\mathbf{x}_s)$. The tree is initialized by storing the whole search space $\mathbf{X}$ and a pair $(\mathbf{x}_0, \mathbf{f}(\mathbf{x}_0))$ in the root.

2.3.5 Leaf Selection

At every iteration, the search for a global minimizer begins at the root and proceeds down the tree until a leaf is reached. In order to reach a leaf, we have to choose a concrete path from the root down to this leaf. At each node, we have to decide whether to take its left or right child as the next station. This decision is made probabilistically. For every node, two numbers $p_0, p_1 \in (0, 1)$ are computed such that $p_0 + p_1 = 1$. Arriving at a node, we choose to descend via either its left or right child with probability p_0 or p_1, respectively. We make these left/right decisions until we reach a leaf.

Computation of the Probabilities p_0 and p_1 The idea is to compute the probabilities in a way such that the "better" child, i.e., the one with the lower function value, has greater chance to be selected. We compute p_0 and p_1 for each node η_s^k based on the function values associated with its children η_{s0}^{k+1} and η_{s1}^{k+1}. Let $\mathbf{f}_{s0}$ and $\mathbf{f}_{s1}$ be the function values associated with η_{s0}^{k+1} and η_{s1}^{k+1}, respectively. The following criterion should be fulfilled:

$$\mathbf{f}_{s0} < \mathbf{f}_{s1} \quad \Leftrightarrow \quad p_0 > p_1. \tag{2.19}$$

If $\mathbf{f}_{s0} < \mathbf{f}_{s1}$ we set

$$p_0 = (t + 1)/(1 + 2t), \quad p_1 = t/(1 + 2t), \tag{2.20}$$

for a parameter $t \geq 0$. For $t \to \infty$ we get $p_0 = p_1 = \frac{1}{2}$ and our minimization algorithm becomes a pure random search. Setting $t = 0$ results in $p_0 = 1$ and $p_1 = 0$ and makes the algorithm deterministically choosing the "better" child of every node which leads to the exclusion of a large portion of the search space and in most cases prevents the algorithm from finding a global minimizer. For $\mathbf{f}_{s1} < \mathbf{f}_{s0}$ we set

$$p_0 = t/(1+2t), \quad p_1 = (t+1)/(1+2t). \tag{2.21}$$

Probabilities Update From the discussion above it becomes evident that t should be chosen from the interval $(0, \infty)$. For our algorithm the parameter t plays a similar role as the temperature parameter for a simulated annealing algorithm [58] so we will refer to t as temperature as well. Like in simulated annealing, the search begins at a high temperature level (large t) such that the algorithm samples the search space quite uniformly. The temperature is decreased gradually during the minimization process so that promising regions of the search space are explored in greater detail. More precisely, we update t according to the following cooling schedule:

$$t = t_{\max} \exp(-vj), \tag{2.22}$$

where $j \in \mathbb{N}$ is the current iteration number, $t_{\max} > 0$ is the temperature at the beginning of the search (for $j = 0$) and $v > 0$ is the cooling speed which determines how fast the temperature decreases.

2.3.6 Tree Expansion

After reaching a leaf η_s^k, the set $\mathbf{X}_s$ associated with it gets bisected in two disjoint subsets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$ of equal volume. The corresponding child nodes are η_{s0}^{k+1} and η_{s1}^{k+1}, respectively. In this way, we add more resolution in this part of the search space. Next, we evaluate the new children, i.e., we assign to the left and right one a pair $(\mathbf{x}_{s0}, \mathbf{f}(\mathbf{x}_{s0}))$ and $(\mathbf{x}_{s1}, \mathbf{f}(\mathbf{x}_{s1}))$, respectively.

Note that the parent of η_{s0}^{k+1} and η_{s1}^{k+1}, namely, the node η_s^k, stores a pair $(\mathbf{x}_s, \mathbf{f}(\mathbf{x}_s))$. Since $\mathbf{X}_s = \mathbf{X}_{s0} \cup \mathbf{X}_{s1}$ and $\mathbf{X}_{s0} \cap \mathbf{X}_{s1} = \emptyset$ it follows that $\mathbf{x}_s$ is contained either in $\mathbf{X}_{s0}$ or in $\mathbf{X}_{s1}$. Thus we set

$$(\mathbf{x}_{s0}, \mathbf{f}(\mathbf{x}_{s0})) = (\mathbf{x}_s, \mathbf{f}(\mathbf{x}_s)) \text{ if } \mathbf{x}_s \in \mathbf{X}_{s0} \text{ or} \tag{2.23}$$

$$(\mathbf{x}_{s1}, \mathbf{f}(\mathbf{x}_{s1})) = (\mathbf{x}_s, \mathbf{f}(\mathbf{x}_s)) \text{ if } \mathbf{x}_s \in \mathbf{X}_{s1}. \tag{2.24}$$

To compute the other pair, we sample a point uniformly from the appropriate remaining set ($\mathbf{X}_{s0}$ or $\mathbf{X}_{s1}$) and evaluate the function at this point (see Figure 2.3(b) for the case $n = 2$ and $\mathbf{X}$ and its subsets being rectangles).

Tree Update During the search we want to compute the random paths from the root down to a certain leaf such that promising regions—leaves with low function values—are visited more often than non-promising ones. Thus, after evaluating a new created leaf, we propagate its

(possibly very low) function value as close as possible to the root. This is done by the following updating procedure. Suppose that the parent point $\mathbf{x}_s$ is contained in the set $\mathbf{X}_{s1}$ belonging to the new created child η_{s1}^{k+1}. Therefore, we randomly generate $\mathbf{x}_{s0} \in \mathbf{X}_{s0}$, compute $\mathbf{f}(\mathbf{x}_{s0})$ and assign the pair $(\mathbf{x}_{s0}, \mathbf{f}(\mathbf{x}_{s0}))$ to the child η_{s0}^{k+1}. Updating the tree consists of ascending from η_{s0}^{k+1} (via its ancestors) to the root and comparing at every parent node η_u^j the function value $\mathbf{f}(\mathbf{x}_{s0})$ with the function value of η_u^j, i.e., with $\mathbf{f}(\mathbf{x}_u)$. If $\mathbf{f}(\mathbf{x}_{s0}) < \mathbf{f}(\mathbf{x}_u)$ we update the current node by setting $(\mathbf{x}_u, \mathbf{f}(\mathbf{x}_u)) = (\mathbf{x}_{s0}, \mathbf{f}(\mathbf{x}_{s0}))$ and proceed to the parent of η_u^j. The updating procedure terminates if we reach the root or no improvement for the current node is possible.

Note that if $\mathbf{f}(\mathbf{x}_{s0})$ is the lowest function value found so far, it will be propagated to the root, otherwise it will be propagated only to a certain level $l \in \{1, \ldots, k+1\}$. This means, that every node contains the minimum function value (and the point at which $\mathbf{f}$ takes this value) found in the subset associated with this node. Since the root represents the whole search space, it contains the point we are interested in, namely, the point at which $\mathbf{f}$ takes the lowest value found up to the current iteration.

2.3.7 Stopping Rule

We break the search if the following two criteria are fulfilled. (i) The leaf η_s^k selected in the current iteration has a volume which is smaller than a user predefined value $\delta_v > 0$. (ii) The absolute difference between the minimal function value found so far and the function value computed in the current iteration is less than a user specified $\delta_f > 0$.

The first condition accounts for the desired precision of the solution and the second one assures that the algorithm makes no significant progress any more.

2.3.8 Remark

We want to emphasize that it is very important that each two nodes at the same tree level are of equal volume. As already mentioned in Section 2.3.2, the points are uniformly sampled within the tree nodes. In this case, if two differently sized nodes at the same tree level are selected equally often, the part of the search space represented by the smaller node will be sampled more densely than the other part. Thus, the algorithm will possibly prefer regions of the search space only because the G-BSP tree is constructed in a particular way and not because of the cost function.

2.4 The Space of Rigid Transforms

As already mentioned in Section 2.2.2, the choice of a parametrization of $SE(3)$ (the group of rigid transforms) is an important issue since different parametrizations lead to different optimization performance. We decompose $SE(3)$ into a translational and a rotational part. While parametrizing translations is straightforward special care is needed when dealing with

rotations since the geometry of the rotation space is more complex than the geometry of $\mathbb{R}^3$. In the following, we concentrate on the rotation space.

In view of our branch and "stochastic bound" minimization method, three specific problems have to be solved. (i) We need to parametrize rotations. (ii) We have to hierarchically decompose the rotation space in disjoint parts of equal volume. In other words, a G-BSP tree has to be computed in which the nodes represent equally sized parts of the rotation space. (iii) We need to sample points (i.e., rotations) uniformly from each leaf of the G-BSP tree. These issues are discussed separately in the next three subsections.

2.4.1 Parametrization of Rotations

There are many ways how to parametrize 3D rotations. Discussing all of them is beyond the scope of this work. An excellent introduction to this topic is included in the books by Kanatani [65] and A. Watt and M. Watt [66] in the context of computer vision and computer graphics, respectively. The set of all 3×3 rotation matrices is a group (under matrix multiplication) which is referred to as $SO(3)$. A parametrization of $SO(3)$ is a mapping $R : \mathbf{U} \to SO(3)$, where $\mathbf{U}$ is a subset of $\mathbb{R}^3$ since every rotation has three degrees of freedom.

Parametrizing rotation matrices using Euler angles is probably the most widely used technique which is, however, inefficient in conjunction with our minimization method. This is due to the fact that Euler angles are a redundant representation of rotations. In order to represent all elements in $SO(3)$ the following range, $\mathbf{E}$, for the three Euler angles is needed: $\mathbf{E} = [0, 2\pi) \times [0, 2\pi) \times [0, \pi]$. However, the corresponding parametrization $R : \mathbf{E} \to SO(3)$, which is given in [65], is not one-to-one. There are infinitely many combinations of Euler angles (within the range $\mathbf{E}$) which lead to the same rotation matrix (see [66]). A minimization method like ours which considers the whole search space will waste computation time exploring regions in $\mathbf{E}$ which should be completely ignored because they do not lead to "new" rotation matrices. The same applies to deterministic branch-and-bound methods (see, e.g., [67]).

In order to avoid this difficulty, we employ a redundant-free rotation space parametrization based on the axis-angle representation of $SO(3)$. According to Euler's theorem (see [65]), each rotation in $\mathbb{R}^3$ can be represented by an axis specified by a unit vector $\mathbf{n}$ and an angle θ of rotation around it. $\mathbf{n}$ can itself be parametrized using spherical coordinates φ and ψ:

$$\mathbf{n} = (\sin(\psi)\cos(\varphi),\ \sin(\psi)\sin(\varphi),\ \cos(\psi)). \tag{2.25}$$

Figure 2.4(a) visualizes this concept. In order to represent all rotation matrices, we need to consider the following range for the spherical coordinates (φ, ψ) and the rotation angle θ:

$$(\varphi, \psi, \theta) \in [0, 2\pi) \times [0, \pi] \times [0, \pi) = \mathbf{R}. \tag{2.26}$$

The resulting parametrization $R : \mathbf{R} \to SO(3)$, which can be found in [65], is a one-to-one mapping between $\mathbf{R}$ and $SO(3)$.

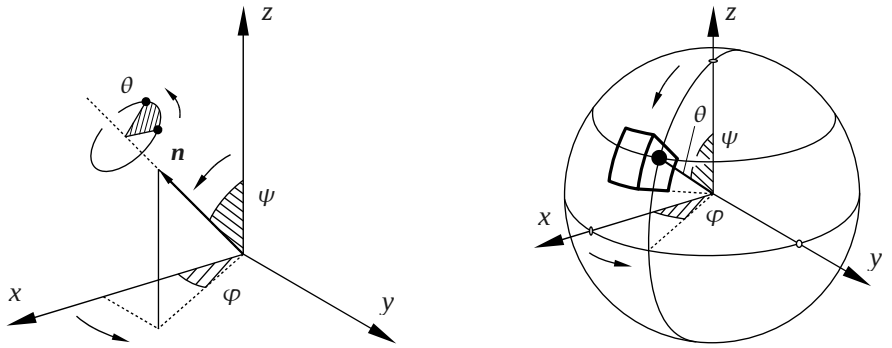


Figure 2.4: (a) The axis-angle based parametrization of $SO(3)$. The two bold dots in the figure represent a point before and after rotation by the angle θ around the axis defined by the unit vector $\mathbf{n}$, which is itself parametrized using spherical coordinates (φ, ψ). (b) The rotation space represented as the open ball in $\mathbb{R}^3$ with radius π. The spherical coordinates (φ, ψ) of the point (shown as a bold dot) define the rotation axis and the distance to the origin gives the angle θ of rotation. The bold lines depict a spherical box.

2.4.2 Hierarchical Decomposition of the Rotation Space

According to the axis-angle representation and to (2.26), it is possible to identify the set of rotations with the open ball $\mathbf{O}(\pi) \subset \mathbb{R}^3$ with radius π located at the origin (see Figure 2.4(b)). Thus a straightforward way to decompose the rotation space is to enclose $\mathbf{O}(\pi)$ in the cube $\mathbf{C}(\pi) = [-\pi, \pi]^3$ and to divide $\mathbf{C}(\pi)$ into smaller cubes by simply bisecting the x, y or z axis. Hartley and Kahl [67] used this technique in conjunction with a deterministic branch-and-bound minimization method to estimate the essential matrix and to solve the relative camera pose problem. However, if combined with our minimization algorithm, this technique leads to two problems. First, the sub-cubes of $\mathbf{C}(\pi)$ which do not lie within $\mathbf{O}(\pi)$ have to be ignored since the rotations they represent are included in other cubes within $\mathbf{O}(\pi)$. This gives rise to nodes in the corresponding G-BSP tree which have only one "legal" child. Second, the sub-cubes of $\mathbf{C}(\pi)$ which are partially intersecting $\mathbf{O}(\pi)$ represent a smaller region of the rotation space than sub-cubes at the same tree level which are fully enclosed in $\mathbf{O}(\pi)$. Thus the minimization algorithm will prefer rotations which are close to the boundary of $\mathbf{O}(\pi)$.

We solve these two problems by changing the shape of the building blocks of the decomposition. Since we are dealing with a three-dimensional ball the most natural shape is the shape of a spherical box (see Figure 2.4(b)). In ball coordinates, we define a spherical box $\mathbf{B}$ to be a point set of the form

$$\mathbf{B} = \{(\varphi, \psi, \theta) : (\varphi, \psi, \theta) \in [\varphi_1, \varphi_2) \times [\psi_1, \psi_2) \times [\theta_1, \theta_2)\}, \tag{2.27}$$

where $[\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$ is the range of the spherical coordinates and $[\theta_1, \theta_2)$ limits the distance of the points to the origin. Decomposing the rotation space means to hierarchically subdivide

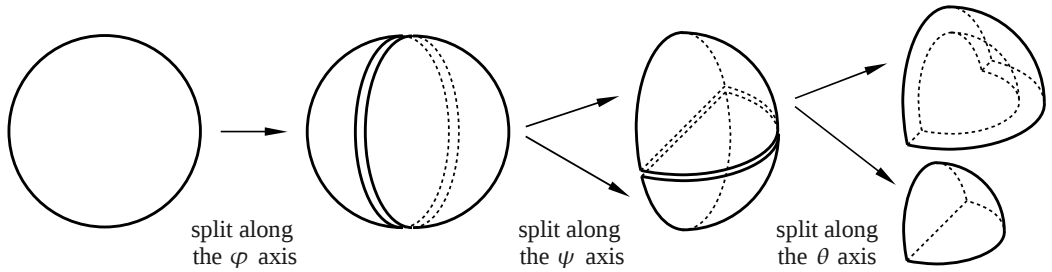


Figure 2.5: Decomposing the rotation space (represented by a solid ball) into spherical boxes of equal volume. In this example, only one spherical box at each splitting step is further decomposed.

$\mathbf{O}(\pi)$ into disjoint spherical boxes of equal volume (see Figure 2.5). Note that the volume of $\mathbf{B}$ is given by

$$\mathrm{vol}_{\mathbf{B}}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta_2) = \int_{\varphi_1}^{\varphi_2} \int_{\psi_1}^{\psi_2} \int_{\theta_1}^{\theta_2} \theta^2 \sin\psi d\theta d\psi d\varphi \tag{2.28}$$

$$= (\varphi_2 - \varphi_1)(\cos\psi_1 - \cos\psi_2)\frac{\theta_2^3 - \theta_1^3}{3}. \tag{2.29}$$

Our aim is to consecutively cut $\mathbf{B}$ along the φ, ψ or θ axis such that the resulting pieces have the same volume. Since $\mathrm{vol}_{\mathbf{B}}$ depends in a different way from each of the ball coordinates φ, ψ and θ we get a different rule for cutting along each axis. We are looking for

$$\varphi \in (\varphi_1, \varphi_2), \quad \psi \in (\psi_1, \psi_2), \quad \theta \in (\theta_1, \theta_2) \tag{2.30}$$

such that

$$\mathrm{vol}_{\mathbf{B}}(\varphi_1, \varphi) = \mathrm{vol}_{\mathbf{B}}(\varphi, \varphi_2), \tag{2.31}$$

$$\mathrm{vol}_{\mathbf{B}}(\psi_1, \psi) = \mathrm{vol}_{\mathbf{B}}(\psi, \psi_2), \tag{2.32}$$

$$\mathrm{vol}_{\mathbf{B}}(\theta_1, \theta) = \mathrm{vol}_{\mathbf{B}}(\theta, \theta_2), \tag{2.33}$$

where, for the sake of clarity, $\mathrm{vol}_{\mathbf{B}}$ is expressed as a function of two variables only, namely, the ones defining the interval which is currently cut. Using (2.29) to solve the equations (2.31)–(2.33) leads to

$$\varphi = \frac{\varphi_1 + \varphi_2}{2}, \quad \psi = \arccos\left(\frac{\cos\psi_1 + \cos\psi_2}{2}\right), \quad \theta = \sqrt[3]{\frac{\theta_1^3 + \theta_2^3}{2}}. \tag{2.34}$$

Thus we fully specified how to hierarchically decompose the space of rotations in disjoint equally sized parts such that a G-BSP tree can be built. Furthermore, the shape of the parts is optimally tailored to our minimization algorithm.

2.4.3 Uniform Sampling from Spherical Boxes

Our method for sampling points uniformly from a spherical box is grounded on the following basic result from Statistics called the inverse probability integral transform. Since it is proved in many textbooks (e.g., in [68]) we state it here without a proof.

Theorem 1. *Let F be a cumulative distribution function (c.d.f.) on $\mathbb{R}$ and let U be a random variable uniformly distributed in $[0,1]$. Then the random variable $X = F(U)^{-1}$ has c.d.f. F.*

Based on this result we perform the uniform sampling from a spherical box $\mathbf{B} = [\varphi_1, \varphi_2) \times [\psi_1, \psi_2) \times [\theta_1, \theta_2)$ in three steps:

1. Sample a φ uniformly from $[\varphi_1, \varphi_2)$.

2. Sample a ψ from $[\psi_1, \psi_2)$ according to a c.d.f. F_2 such that the point in $\mathbb{R}^3$ with spherical coordinates (φ, ψ) is uniformly distributed on the spherical patch $\mathbf{P} = [\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$.

3. Sample a θ from $[\theta_1, \theta_2)$ according to a c.d.f. F_3 such that the point in $\mathbb{R}^3$ with ball coordinates (φ, ψ, θ) is uniformly distributed in the spherical box $\mathbf{B}$.

Step 1 is easy to perform. In step 2, we need to compute the area of a spherical patch (of the unit 2-sphere) as a function of an interval $[\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$:

$$\begin{aligned} \text{area}_{\mathbf{P}}(\varphi_1, \varphi_2, \psi_1, \psi_2) &= \int_{\varphi_1}^{\varphi_2} \int_{\psi_1}^{\psi_2} \sin\psi d\psi d\varphi && (2.35) \\ &= (\varphi_2 - \varphi_1)(\cos\psi_1 - \cos\psi_2). && (2.36) \end{aligned}$$

Thus the c.d.f. we need in step 2 is given by

$$\begin{aligned} F_2(\psi) &= \frac{\text{area}_{\mathbf{P}}(\varphi_1, \varphi_2, \psi_1, \psi)}{\text{area}_{\mathbf{P}}(\varphi_1, \varphi_2, \psi_1, \psi_2)} && (2.37) \\ &= \frac{\cos\psi_1 - \cos\psi}{\cos\psi_1 - \cos\psi_2}, && (2.38) \end{aligned}$$

Analogously, we see that the c.d.f. in step 3 is given by

$$\begin{aligned} F_3(\theta) &= \frac{\text{vol}_{\mathbf{B}}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta)}{\text{vol}_{\mathbf{B}}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta_2)} && (2.39) \\ &= \frac{\theta^3 - \theta_1^3}{\theta_2^3 - \theta_1^3}, && (2.40) \end{aligned}$$

where (2.40) follows from (2.29). Note that both F_2 and F_3 can easily be inverted and we can use Theorem 1 to sample according to F_2 and F_3 and hence uniformly from a spherical box.

2.4.4 Computation of the Search Space and the G-BSP Tree

Now since all details regarding the parametrization and decomposition of $SO(3)$ and the sampling from spherical boxes are given, we define the search space $\mathbf{X}$ and specify how to build the corresponding G-BSP tree. We set

$$\mathbf{X} = \mathbf{R} \times \mathrm{bbox}(\mathbf{M}), \tag{2.41}$$

where $\mathbf{R}$ is, according to (2.26), the domain of the axis-angle based parametrization of $SO(3)$ and $\mathrm{bbox}(\mathbf{M})$ (the bounding box of the model $\mathbf{M}$) represents the translational part of the search space. Since $\mathrm{bbox}(\mathbf{M})$ is a rectangular box of the form $[x_1, x_2] \times [y_1, y_2] \times [z_1, z_2] \subset \mathbb{R}^3$ it can easily be broken up into smaller boxes of the same size by simply bisecting it along the x, y or z axis.

The root η_0^0 of the G-BSP tree represents the whole set $\mathbf{X}$. The child nodes of the root, namely, η_{00}^1 and η_{01}^1, represent the subsets $\mathbf{X}_0$ and $\mathbf{X}_1$, respectively, resulting from cutting the 0th interval of $\mathbf{X}$—which is $[0, 2\pi)$ in (2.26)—using the rule $(2.34)_1$. In general, a node η_s^k (where $k \geq 0$ and s is a binary string of length $k+1$) is at the kth level of the tree, represents a subset $\mathbf{X}_s$ of the 6D search space and has two children, η_{s0}^k and η_{s1}^k. The child nodes represent the sets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$, respectively, which are computed by cutting the (k mod 6)th interval of $\mathbf{X}_s$ according to (2.34) if $0 \leq k \bmod 6 \leq 2$ (rotational part) or by dividing it in the middle if $3 \leq k \bmod 6 \leq 5$ (translational part).

Chapter 3

3D Object Recognition: Many-to-One Rigid Shape Registration

In the context of 3D shape registration, treated in the previous chapter, the input consists of a single model and a single data set which are assumed to (partially) represent one and the same object. In this chapter, we drop this assumption and consider multiple models and a data set, called scene, which is allowed to contain data from several objects plus background clutter. This leads to the problem of 3D object recognition and pose estimation which can loosely be formulated as follows. Given a set $\mathbf{O} = \{\mathbf{M}_1, \ldots, \mathbf{M}_q\}$ of object models and a scene $\mathbf{S}$, the task is to identify the objects present in the scene and to estimate their position and orientation. The output of an object recognition and pose estimation algorithm is a list $\{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_r}, T_r)\}$ of pairs, with $\mathbf{M}_{k_j} \in \mathbf{O}$ being a recognized model instance and T_j being the rigid transform which aligns $\mathbf{M}_{k_j}$ to the scene $\mathbf{S}$. In this sense, 3D object recognition can be seen as many-to-one rigid shape registration. Thus, a shape registration method could be used in a straightforward manner to solve the problem by sequentially matching each model to the scene. This, however scales bad with number of models.

For the sake of simplicity, in the rest of the text, we mean by "object recognition" *both* object identification and pose estimation. Sometimes we also use the term object detection.

One may pose the question why did we study rigid registration in Chapter 2 for it is a special case of object recognition. The reason is that an object recognition method is fed the object models in advance—prior to the actual recognition. The only input which is provided "on the fly" is the 3D scene in which the objects are supposed to be detected. This allows a recognition algorithm to pre-process the models in an offline phase in order to ease the online detection. In contrast, a registration method processes both shapes right away without an extra (computationally more or less expensive) preparation phase.

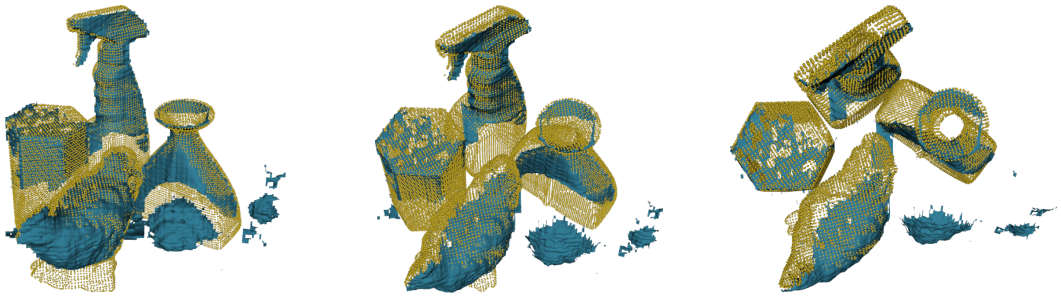


Figure 3.1: Three views of a typical recognition result obtained with our method. The scene is shown as a blue mesh and the four recognized model instances are rendered as yellow point clouds and superimposed over the scene mesh (see Section 5.2 for details).

In this thesis, we discuss a special instance of the object recognition problem given by the following assumptions.

(i) Each model is a finite set of points with corresponding surface normals.

(ii) Each model represents a non-transparent object.

(iii) The scene is a range image.

(iv) Each transform which aligns a recognized model instance to the scene is a rigid transform.

Even under these assumptions the problem remains hard because of several reasons: usually, there are scene parts not belonging to any of the objects of interest, i.e., there is background clutter; the input is typically corrupted by noise and outliers; the objects are only partially visible due to occlusions and scan device limitations. Figure 3.1 shows a 3D scene and the recognized model instances. Note that the objects are highly occluded.

Contributions and Chapter Overview

In this chapter, we present an efficient algorithm for 3D object recognition in the presence of clutter and occlusions in noisy, unsegmented range data. Our approach operates directly on unsegmented point clouds provided by a range scanner. This does not require scene segmentation which may be quite time consuming and error-prone. More specifically, we make the following contributions. (i) A new efficient, localized RANSAC-like sampling strategy is introduced. (ii) We use a hash table for rapid retrieval of pairs of oriented model points which are similar to a sampled pair of oriented scene points. This allows to efficiently generate object and pose hypotheses. (iii) We provide a complexity analysis of our sampling strategy and derive a formula for the number of iterations required to recognize the objects with a predefined success probability.

The rest of the chapter is organized as follows. Related work is reviewed in Section 3.1. In Section 3.2, we establish some notation and explain in more detail two algorithms which are important to our work. Our 3D object recognition approach is introduced in Section 3.3.

3.1 Related Work

Object recognition should not be confused with object classification/shape retrieval. The latter methods only measure the similarity between a given input shape and shapes stored in a model library [69]. Usually, they do not estimate a transform which maps the input to the recognized model. Moreover, the input shape is assumed to be a subset of some of the library shapes. In our case, however, the input contains points originating from multiple objects and scene clutter.

Since 3D object recognition is closely related to rigid 3D shape registration, several methods we already reviewed in Section 2.1 can be modified and employed for object recognition: the voting approaches (generalized Hough transform [10], pose clustering [45], geometric hashing [70] and tensor matching [71]) and the feature-based approaches (spin images [28], local feature histograms [72], 3D/harmonic shape context [73], intrinsic isometry invariant descriptors [74] and manifold harmonic bases [75]).

Another way to tackle the problem is to model an object as an assembly of basic shapes (primitives) and to recover these shapes and their spatial relationships from an input scene. Many types of primitives can be used within this part-based framework: generalized cylinders [76], superquadrics [77], implicit polynomials [78], geometric primitives [79], and parametric shapes [80]. Methods for efficient recovering of superquadrics from range data were introduced in [35, 36, 37]. However, despite their efficiency, the part-based approaches are limited to a certain shape class, namely, the one which can be described by the chosen set of primitives.

In [81], a hashing technique similar to ours was proposed. It is a learning-based method employing a multiple-attribute hash table for efficient 3D object recognition. On the positive side, attribute uncertainties are taken into account and the number of attributes as well as the size of the hash table bins are calculated automatically. However, the system cannot handle free-form objects and in the presented experimental results only objects composed of single-colored surfaces are used. Furthermore, the method relies on a segmentation to identify the planar or cylindrical surface patches the objects are made of.

A further hashing technique was proposed in [82]. Based on a hash table, a fast indexing into a collection of geometry descriptors of *single* model points is performed. In contrast, our hash table stores descriptors of *pairs* of oriented model points (called doublets). This allows to efficiently query the model doublets similar to a sampled scene doublet and it makes it very easy to compute the aligning rigid transform since it is uniquely defined by two corresponding doublets. Moreover, in [82], multiple range images are aligned to each other in order to build a more complete scene representation and a foreground/background segmentation is executed. In contrast, our method operates on a single range image and does not require segmentation.

Furthermore, the test scenes used in [82] contain a single object, whereas we deal with multiple objects per scene.

3.2 Notation and Basic Algorithms

An oriented point $\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)$ consists of a point $\mathbf{p}_u \in \mathbb{R}^3$ and a corresponding surface normal $\mathbf{n}_u \in \mathbb{R}^3$, $\|\mathbf{n}_u\| = 1$. Accordingly, an oriented point pair $(\mathbf{u}, \mathbf{v})$ is a pair of two oriented points $\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)$ and $\mathbf{v} = (\mathbf{p}_v, \mathbf{n}_v)$.

3.2.1 Fast Surface Registration

Assume $\mathbf{S} = \{\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)\}$ is a surface represented by a finite set of oriented points. According to [83], for a pair of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{S} \times \mathbf{S}$, a descriptor $f : \mathbf{S} \times \mathbf{S} \to \mathbb{R}^4$ is computed as

$$f(\mathbf{u}, \mathbf{v}) = \begin{pmatrix} f_1(\mathbf{u}, \mathbf{v}) \\ f_2(\mathbf{u}, \mathbf{v}) \\ f_3(\mathbf{u}, \mathbf{v}) \\ f_4(\mathbf{u}, \mathbf{v}) \end{pmatrix} = \begin{pmatrix} \|\mathbf{p}_u - \mathbf{p}_v\| \\ \angle(\mathbf{n}_u,\, \mathbf{n}_v) \\ \angle(\mathbf{n}_u,\, \mathbf{p}_v - \mathbf{p}_u) \\ \angle(\mathbf{n}_v,\, \mathbf{p}_u - \mathbf{p}_v) \end{pmatrix}, \tag{3.1}$$

where $\angle(\mathbf{a}, \mathbf{b})$ is the angle between the vectors $\mathbf{a}$ and $\mathbf{b}$. In order to register two surfaces $\mathbf{S}_1$ and $\mathbf{S}_2$, each one represented by a set of oriented points, the method proceeds as follows. It samples uniformly oriented point pairs $(\mathbf{u}, \mathbf{v}) \in \mathbf{S}_1 \times \mathbf{S}_1$ and $(\mathbf{w}, \mathbf{x}) \in \mathbf{S}_2 \times \mathbf{S}_2$ and computes and stores their descriptors $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{w}, \mathbf{x})$ in a four-dimensional hash table. This process continues until a collision occurs, i.e., until $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{w}, \mathbf{x})$ end up in the same hash table cell. Computing the rigid transform T which aligns $(\mathbf{u}, \mathbf{v})$ to $(\mathbf{w}, \mathbf{x})$ gives a transform hypothesis which registers $\mathbf{S}_1$ to $\mathbf{S}_2$. Figure 3.2 illustrates the alignment. More formally,

$$T = F_{wx} F_{uv}^{-1} \tag{3.2}$$

is computed based on the pairs' local coordinate systems, each one represented by a 4×4 matrix (homogeneous coordinates) F_{uv} respectively F_{wx}. We have

$$F_{uv} = \begin{pmatrix} \frac{\mathbf{p}_{uv} \times \mathbf{n}_{uv}}{\|\mathbf{p}_{uv} \times \mathbf{n}_{uv}\|} & \frac{\mathbf{p}_{uv}}{\|\mathbf{p}_{uv}\|} & \frac{\mathbf{p}_{uv} \times \mathbf{n}_{uv} \times \mathbf{p}_{uv}}{\|\mathbf{p}_{uv} \times \mathbf{n}_{uv} \times \mathbf{p}_{uv}\|} & \frac{\mathbf{p}_u + \mathbf{p}_v}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

where $\mathbf{p}_{uv} = \mathbf{p}_v - \mathbf{p}_u$ and $\mathbf{n}_{uv} = \mathbf{n}_u + \mathbf{n}_v$. F_{wx} is defined analogously by replacing the indices u and v in (3.3) with w and x, respectively. The transform hypothesis T generated in this way is evaluated by transforming the points of $\mathbf{S}_1$, i.e., $\mathbf{p}'_i = T\mathbf{p}_i$, $\forall \mathbf{p}_i \in \mathbf{S}_1$ and counting those $\mathbf{p}'_i$ which fall within a certain ϵ-band of $\mathbf{S}_2$.

According to [83], this process of generating and evaluating hypotheses is repeated until either of the following stopping criteria is met: (i) a hypothesis is good enough, (ii) a predefined time limit is reached or (iii) all combinations are tested. Unfortunately, non of these criteria is well-grounded: the first two are ad hoc and the third one is computationally infeasible. In

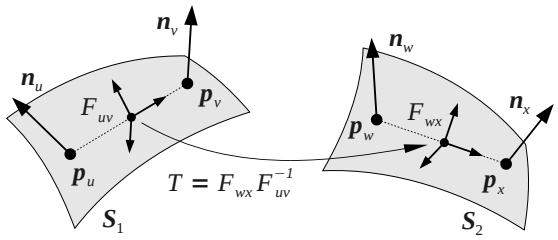


Figure 3.2: Computing the rigid transform T which aligns $\mathbf{S}_1$ to $\mathbf{S}_2$ based on the local coordinate systems F_{uv} and F_{wx} of the oriented point pairs $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{w}, \mathbf{x})$, respectively. See text for details on F_{uv} and F_{wx}.

contrast, we derive the number of iterations required to recognize model instances with a predefined success probability. Furthermore, we modify this technique in a way which allows for the *simultaneous* matching of all object models to the scene.

3.2.2 RANSAC

RANSAC [9] can be seen as a general approach for model recognition. It works by uniformly drawing minimal point sets from the scene and computing a transform which aligns the model with the minimal point set. A minimal point set is the smallest point set which uniquely determines a given type of transform. In the case of rigid transforms, it is a point triple. The score of the aligning transform is the number of transformed model points which lie within an ϵ-band of the scene. After a certain number of trials the model is considered to be recognized at the locations defined by the transforms which achieved a score higher than a predefined threshold.

The probability P_S of recognizing the model in N trials equals the complementary of N consecutive failures [80], i.e.,

$$P_S = 1 - (1 - P_M)^N, \tag{3.4}$$

where P_M is the probability of recognizing the model in a single iteration. Solving for N gives the number of trials needed to recognize the model in the scene:

$$N = \frac{\ln(1 - P_S)}{\ln(1 - P_M)}. \tag{3.5}$$

Note that since $P_S \approx 1$ and $P_M \approx 0$, one can safely assume that $N \geq 1$.

The RANSAC approach is conceptually simple, general and robust against outliers. Unfortunately, its direct application to the 3D object recognition problem is computationally expensive. In order to compute an aligning rigid transform, we need two corresponding point triples—one from the model and one from the scene. Assuming that the model is completely contained in the scene, the probability of drawing two such triples in a single trial is $P_M(n) = \frac{3!}{(n-2)(n-1)n}$, where n is the number of scene points. Since $P_M(n)$ is a small number we can approximate the

denominator in (3.5) by its Taylor series $\ln(1 - P_M(n)) = -P_M(n) + O(P_M(n)^2)$ and obtain the number of trials as a function of the number of scene points:

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = O(n^3). \tag{3.6}$$

In Section 3.3.3, we will show that using oriented point pairs and our localized sampling strategy leads to a reduction of the time complexity from $O(n^3)$ to $O(n)$.

There are many modifications of the classic RANSAC scheme. Some recently proposed methods like ASSC [84] and ASKC [85] significantly improve outlier robustness by using a different score function. However, these variants are not designed to enhance the performance of RANSAC. In [86], an efficient RANSAC-like registration algorithm was proposed. However, it is not advisable to directly apply the method to 3D object recognition since it will require a sequential matching of each model to the scene. In [80], another efficient RANSAC variant for primitive shape detection was introduced. The method is related to ours since the authors also used a localized minimal point set sampling. Their method, however, is limited to the detection of planes, spheres, cylinders, cones and tori.

3.3 Method Description

Our object recognition method consists of two phases. The first one, the model preprocessing, is performed offline. It is executed only once and does not depend on the scenes in which the objects have to be recognized. The online recognition is the second phase. It is executed on the scene using the model representation computed in the offline phase. In the rest of this section, we describe both phases in detail and discuss the time complexity of our algorithm.

3.3.1 Model Preprocessing Phase

We assume that each object model is represented by a finite set $\mathbf{M} = \{\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)\}$ of oriented points. For a given object model $\mathbf{M}$, we sample the pairs of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{M} \times \mathbf{M}$ for which $\mathbf{p}_u$ and $\mathbf{p}_v$ are approximately d units apart from each other. For each such pair, the descriptor $f(\mathbf{u}, \mathbf{v}) = (f_2(\mathbf{u}, \mathbf{v}),\ f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed according to (3.1) and stored in a three-dimensional hash table. Note that f_1 is not part of the descriptor since a fixed distance d is used. In contrast to [83], *not* all pairs of oriented points are considered, but only those with $\|\mathbf{p}_u - \mathbf{p}_v\| \in [d - \delta_d, d + \delta_d]$, for a given tolerance value δ_d. This has several advantages. It reduces the space complexity from $O(m^2)$ to $O(m)$, where m is the number of model points (this empirical measurement is further discussed in [34]). Using a large d results in wide-pairs which allow a more stable computation of the aligning rigid transform than narrow-pairs do [34]. Furthermore, a larger d leads to fewer pairs which means that computing and storing descriptors of wide-pairs results in less populated hash table cells. Thus, we will have to test fewer transform hypotheses in the online recognition phase and will save computation time.

However, the pair width d should not be too large since occlusions in real world scenes would prevent sampling a pair with points from the same object. For a typical value for d, there are still many pairs with similar descriptors which leads to hash table cells with too many entries. We avoid this overpopulation, by removing as many of the most populated cells to keep only a fraction K of the original number of pairs (in our implementation $K = 0.4$). This results in an information loss about the object shape which we take into account in the online phase of the algorithm.

In order to compute the final representation of all models $\mathbf{M}_1, \ldots, \mathbf{M}_q$, each $\mathbf{M}_i$ is processed in the way described above using the *same* hash table. In this way, a simultaneous recognition of all models is possible instead of sequentially matching each one of them to the scene. Furthermore, in order to keep track of which pair belongs to which model, every hash table cell stores the pairs in separate model-specific lists.

3.3.2 Online Recognition Phase

The input to the online recognition algorithm is a set $\mathbf{O} = \{\mathbf{M}_1, \ldots, \mathbf{M}_q\}$ of object models and a scene range image $\mathbf{S}$. The output is a list $\mathbf{T} = \{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_r}, T_r)\}$, where $\mathbf{M}_{k_j} \in \mathbf{O}$ is a recognized model instance and T_j is a rigid transform (an element of the special Euclidean group $SE(3)$) aligning $\mathbf{M}_{k_j}$ to the scene. Before turning to the details, it is advisable to read Algorithm 1 although some of the steps may not be completely clear at this point. In the rest of this section, the lines we are referring to are the lines of Algorithm 1.

Searching for closest points (line 8) and for points lying on a sphere around a given point (line 6) have to be performed very often in the online recognition phase. Thus, a fast execution of these operations is of great importance for the runtime of the algorithm. An efficient way to achieve this is to use an octree [87].

Step 1) Initialization

In step 1 of the algorithm, an octree with a fixed leaf size L (the edge length of a leaf) is constructed for the input scene points. The full octree leaves (the ones containing at least one point) are voxels ordered in a regular axis-aligned 3D grid and have unique integer coordinates. Two full leaves are considered neighbors if their corresponding integer coordinates differ by not more than 1. Next, a down-sampled scene $\mathbf{S}^*$ is created by setting its points to be the centers of mass of the full octree leaves. The center of mass of a full leaf is the average of the points it contains. In this way, a one-to-one correspondence between the points in $\mathbf{S}^*$ and the full octree leaves is established. Two points in $\mathbf{S}^*$ are neighbors if the corresponding leaves are neighbors.

input : a set $\mathbf{O} = \{\mathbf{M}_1, ..., \mathbf{M}_q\}$ of object models;
a scene range image $\mathbf{S}$;
output: a list $\mathbf{T} = \{(\mathbf{M}_{k_1}, T_1), ..., (\mathbf{M}_{k_r}, T_r)\}$, with $\mathbf{M}_{k_j} \in \mathbf{O}$ and $T_j \in SE(3)$;

`// 1) initialization`
compute an octree for the scene $\mathbf{S}$ to produce a modified scene $\mathbf{S}^*$;
$\mathbf{T} \leftarrow \emptyset$; `// an empty solution list`
`// 2) number of iterations`
compute the number N of iterations;
repeat N *times*
  `// 3) sampling`
  sample a point $\mathbf{p}_u$ uniformly from $\mathbf{S}^*$;
  compute $\mathbf{L} = \{\mathbf{x} \in \mathbf{S}^* \ : \ \|\mathbf{x} - \mathbf{p}_u\| \in [d - \delta_d, d + \delta_d]\}$;
  sample a point $\mathbf{p}_v$ uniformly from $\mathbf{L}$;
  `// 4) normal estimation`
  estimate normals $\mathbf{n}_u$ at $\mathbf{p}_u$ and $\mathbf{n}_v$ at $\mathbf{p}_v$;
  $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$;
  `// 5) hash table access`
  $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$; `// see (3.1)`
  access the model hash table cell at $f_{\mathbf{uv}}$ **and**
  get its oriented model point pairs $(\mathbf{u}_j, \mathbf{v}_j)$;
  `// 6) generate and test`
  foreach $(\mathbf{u}_j, \mathbf{v}_j)$ **do**
    get the model $\mathbf{M}$ of $(\mathbf{u}_j, \mathbf{v}_j)$;
    compute the rigid transform T that aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$; `// see (3.2)`
    if μ *accepts* $(\mathbf{M}, T)$ **then**
      $\mathbf{T} \leftarrow \mathbf{T} \cup (\mathbf{M}, T)$;
    end
  end
end
`// 7) conflicting hypotheses removal`
remove conflicting hypotheses from $\mathbf{T}$;

Algorithm 1: Online recognition phase.

Step 2) Number Of Iterations

In this step, the number N of iterations is estimated such that all objects in the scene will be recognized with a certain user-defined probability. This will be explained in detail in Section 3.3.3.

Step 3) Sampling

As in classic RANSAC, we sample minimal sets from the scene. In our case, since we use normals, a minimal set consists of two oriented points. In contrast to RANSAC, they are *not* sampled uniformly and independently of each other. Only the first point, $\mathbf{p}_u$, is drawn uniformly from $\mathbf{S}^*$. The second one, $\mathbf{p}_v$, is drawn uniformly from the scene points in $\mathbf{S}^*$ which are approximately within a distance d from $\mathbf{p}_u$. To achieve this, we first retrieve the set $\mathbf{L}$ of all full leaves intersecting the sphere with center $\mathbf{p}_u$ and radius d, where d is the pair width used in the offline phase (see Section 3.3.1). This can be performed very efficiently due to the hierarchical structure of the octree. Finally, a leaf is drawn uniformly from $\mathbf{L}$ and $\mathbf{p}_v$ is set to be its center of mass.

Step 4) Normal Estimation

We estimate the normals $\mathbf{n}_u$ and $\mathbf{n}_v$ at the points $\mathbf{p}_u$ and $\mathbf{p}_v$ by performing a PCA: $\mathbf{n}_u$ and $\mathbf{n}_v$ are set to be the eigenvectors corresponding to the smallest eigenvalues of the covariance matrix of the points in the neighborhood of $\mathbf{p}_u$ and $\mathbf{p}_v$. The result of this step is the oriented scene point pair $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$.

Step 5) Hash Table Access

In line 10, $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed according to (3.1). Next, in lines 11 and 12, $f_{\mathbf{uv}}$ is used as a key to the model hash table to retrieve all model pairs $(\mathbf{u}_j, \mathbf{v}_j)$ similar to $(\mathbf{u}, \mathbf{v})$.

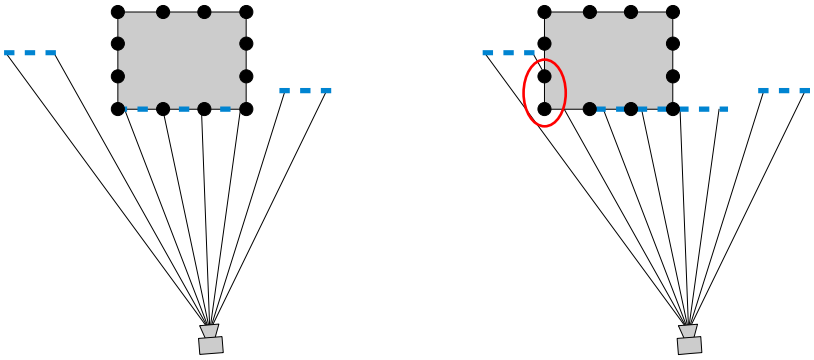
Step 6) Generate and Test

For each $(\mathbf{u}_j, \mathbf{v}_j)$, its model $\mathbf{M}$ is retrieved (line 14) and the rigid transform T which aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$ is computed according to (3.2) (line 15). This results in the hypothesis that the model $\mathbf{M}$ is in the scene at the location defined by T. Finally, the hypothesis is saved in the solution list $\mathbf{T}$ if it is accepted by the acceptance function μ (line 16).

Acceptance Function

μ consists of a visibility term and a penalty term. Similar to RANSAC, the visibility term, μ_V, is proportional to the number m_V of transformed model points which fall within a certain ϵ-band of the scene. More precisely, we set $\mu_V(\mathbf{M}, T) = m_V/m$, where m is the total number of model points. μ_V is an approximation of the visible object surface area expressed as a fraction of the total object surface area. Thus, μ_V can be interpreted as an estimation of the object visibility in the scene.

In contrast to RANSAC, our algorithm contains an additional penalty term, μ_P, which is proportional to the number of transformed model points which occlude the scene. Obviously, a correctly recognized and localized non-transparent object should not occlude any visible scene



(a) $\mu_V = 4/12,\ \mu_P = 0.$ (b) $\mu_V = 3/12,\ \mu_P = 2/12.$

Figure 3.3: A 2D top schematic view of the same scene (blue dashed line) with two different model hypotheses (models are shown as gray boxes). The lines of sight are shown as thin black lines originating from the scanning device. In (a), 4 (out of 12) model points match the scene and no model points are occluding scene points. In (b), 3 model points match the scene and 2 model points (marked by the ellipse) are occluding scene points. The resulting values for μ_V and μ_P are shown below the corresponding figure.

points when the scene is viewed from the viewpoint of the range scanner. In other words, if we view the scene from the perspective of the scanning device, we will not be able to see scene points lying behind the localized model since we cannot see through non-transparent surfaces. The penalty term penalizes hypotheses which violate this condition. It is computed by counting the number m_P of transformed model points which are between the projection center of the range image and a range image pixel and thus are "occluding" reconstructed scene points. We set $\mu_P(\mathbf{M}, T) = m_P/m$, where m is the number of model points.

For $(\mathbf{M}, T)$ to be accepted by μ as a valid hypothesis it has to fulfill

$$\mu_V(\mathbf{M}, T) > V \text{ and } \mu_P(\mathbf{M}, T) < P, \tag{3.7}$$

where $V \in [0, 1]$ is a visibility and $P \in [0, 1]$ a penalty threshold. In Figure 3.3, a simple scene is shown with two different model hypotheses and the corresponding values for μ_V and μ_P.

The visibility threshold is one of the most crucial parameters in the algorithm. In Section 5.2, we experimentally examine how this threshold affects the recognition and the false positives rates of our method. In the case of perfect data, the penalty threshold P should be 0. However, since we are dealing with real range images, we use $P = 0.05$.

Step 7) Conflicting Hypotheses Removal

A hypothesis $(\mathbf{M}, T)$ "explains" a subset $\mathbf{P} \subset \mathbf{S}^*$ if there are points from $T(\mathbf{M})$ lying in the octree leaves corresponding to $\mathbf{P}$. After accepting $(\mathbf{M}, T)$, the points explained by it are *not* removed from $\mathbf{S}^*$ because there could be a better hypothesis, i.e., one which explains a superset

of $\mathbf{P}$. We call hypotheses conflicting if the intersection of the point sets they explain is non-empty. In other words, conflicting hypotheses transform their models such that they intersect in space.

Since the scene points explained by the accepted hypotheses are not removed from $\mathbf{S}^*$, there are many conflicting ones in the solution list $\mathbf{T}$ after the execution of the main loop (lines 4 to 20) of Algorithm 1. To filter the weak hypotheses, we construct a so-called conflict graph. Its nodes are the hypotheses in $\mathbf{T}$ and an edge is connecting two nodes if the hypotheses are conflicting ones.

To produce the final output, the solution list is filtered by performing a non-maximum suppression on the conflict graph. We borrow this technique from image processing. To perform a non-maximum suppression on a gray-scale image, the pixel under observation is set to zero (it is suppressed) if its value is not a maximum in a window placed around that pixel. In this case, the window defines the neighborhood of each pixel. In the case of our conflict graph, the neighborhood is defined by the graph structure. Using the neighborhood of each node, we perform non-maximum suppression essentially in the same way as in image processing: a node η is suppressed if there is a better one in its neighborhood, i.e., a node which explains more scene points than η.

3.3.3 Time Complexity

The dominating factor in the complexity of the proposed method is the number N of iterations needed to recognize all models with a predefined success probability (see the main loop of Algorithm 1, lines 4 to 20). In the following, we discuss the dependency of N on the number of scene points.

Consider a scene $\mathbf{S}^*$ consisting of $|\mathbf{S}^*| = n$ points and a model instance $\mathbf{M}$ therein consisting of $|\mathbf{M}| = m$ points. In Section 3.2.2 on RANSAC, we derived the number $N = \frac{\ln(1-P_S)}{\ln(1-P_M)}$ of iterations required to recognize $\mathbf{M}$ with a predefined success probability P_S, where P_M is the probability of recognizing $\mathbf{M}$ in a single iteration. Again in Section 3.2.2, we obtained $P_M \approx 1/n^3$ which resulted in the cubic time complexity of RANSAC. In the following, we show that our sampling strategy and the use of the model hash table lead to a significant increase of P_M and thus to a reduction of the complexity.

If $P(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M})$ denotes the probability that both points are sampled from $\mathbf{M}$ (lines 5 and 7 of Algorithm 1), then the probability of recognizing $\mathbf{M}$ in a single iteration is

$$P_M = KP(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M}), \tag{3.8}$$

with K being the fraction of oriented point pairs for which the descriptors are stored in the model hash table (see Section 3.3.1). Using conditional probability and the fact that $P(\mathbf{p}_u \in \mathbf{M}) = m/n$ we can rewrite (3.8) to obtain

$$P_M = (m/n)KP(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}). \tag{3.9}$$

$P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M})$ denotes the probability to sample $\mathbf{p}_v$ from $\mathbf{M}$ given that $\mathbf{p}_u \in \mathbf{M}$. Recall from Section 3.3.2 that $\mathbf{p}_v$ depends on $\mathbf{p}_u$ because it is sampled uniformly from the set $\mathbf{L}$ of scene points which are close to the sphere $\mathcal{S}_d(\mathbf{p}_u)$ with center $\mathbf{p}_u$ and radius d, where d is the pair width used in the offline phase. Assuming that the visible object part has an extent larger than $2d$ and that the reconstruction is not too sparse, $\mathbf{L}$ contains points from $\mathbf{M}$. In this case, $P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}) = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ is well-defined and greater than zero.

Let us discuss $C := |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$. It depends on the scene clutter, the number of outliers and the extent and shape of the visible object part. If all scene points originate from known objects (in particular there is no background) and if the objects are well separated then $|\mathbf{L} \cap \mathbf{M}| = |\mathbf{L}|$ since the sphere $\mathcal{S}_d(\mathbf{p}_u)$ intersects scene octree leaves containing only points from the object $\mathbf{p}_u$ belongs to. In this extreme case, we have $C = 1$. On the other hand, occluded scenes with many outliers can be constructed in which $\mathcal{S}_d(\mathbf{p}_u)$ intersects only objects other then the one $\mathbf{p}_u$ belongs to. This leads to $C = 0$ and simply means that the object is too occluded to be recognized.

In our implementation, we estimate C by $1/4$. This accounts for up to 75% outliers and scene clutter. Thus, we obtain for P_M as a function of n (the number of scene points)

$$P_M(n) = (m/n)KC. \tag{3.10}$$

Approximating the denominator $\ln(1-P_M(n))$ in (3.5) by its Taylor series $-P_M(n)+O(P_M(n)^2)$ we obtain for the number of iterations

$$N(n) \approx \frac{-\ln(1-P_S)}{P_M(n)} = \frac{-n\ln(1-P_S)}{mKC} = O(n). \tag{3.11}$$

This shows that the number of iterations depends linearly on the number n of scene points. Furthermore, Eq. (3.11) provides means for computing the number of iterations required to recognize the model instances with the desired success probability P_S.

Chapter 4

A Unified Framework for Shape Modeling and Deformable 3D Shape Registration

In this chapter, we drop the assumption of rigidity we had till now and develop a unified framework for deformation-based 3D shape modeling and deformable 3D shape registration. We start with a short introduction to these two problems.

Deformation-based shape modeling is an active research topic in computer graphics with important applications ranging from automotive design to character animation in film and game production. Loosely speaking, it is the task of warping a shape in a "natural" and physically plausible way such that the warped version shares the distinctive features of the original. Since this is usually done in an interactive, user-guided session, defining the deformation should be intuitive and easy to perform. Pick-and-drag interfaces are particularly well-suited, since they provide a simple way of defining the shape deformation by setting positional constraints on it: the user picks a point (with the mouse) on the shape and either sets it to be fixed or drags it to a new position. The rest of the shape should deform in a realistic way.

For a deformation to look "natural" and physically plausible, it has to fulfil (at least) two criteria: (i) the displacements defined by the user have to be propagated smoothly over the shape and (ii) the deformation has to be detail-preserving, meaning that local shape details should not be unnecessarily distorted, that is, they should move as rigidly as possible instead of being stretched, twisted or sheared.

Furthermore, since the modeling is a user-guided process, a deformation technique has to be efficient enough to run at interactive frame rates and it has to be numerically stable, no matter how severe is the deformation implied by the user constraints.

The shape deformation algorithm developed in this chapter has all the above mentioned properties.

4. A UNIFIED FRAMEWORK FOR SHAPE MODELING AND DEFORMABLE 3D SHAPE REGISTRATION

The second problem we treat within the proposed unified framework is deformable 3D shape registration. It is a fundamental problem in geometry processing with applications in the fields of computer vision, computer graphics and medical image processing, just to name a few (see introduction in Chapter 1). In recent years, 3D geometry acquisition techniques have been developed which allow to capture the surface of deforming objects in real time [88, 41]. In order to analyze the motion of the object it is important to register subsequent scans and/or to register a complete geometric model to the scans. Since the object is undergoing a non-rigid motion, rigid registration algorithms [1, 5, 6, 3] can not be used adequately in this setting.

The problem of deformable shape registration can loosely be defined as follows. Given a shape $\mathcal{M}$, called model, and a shape $\mathcal{D}$, called data set, find a "reasonable" deformation F that brings $\mathcal{M}$ "close" to $\mathcal{D}$. We will introduce the shape representation used in this chapter in Section 4.2. The deformation we are looking for is a mapping $F : \mathcal{M} \to \mathbb{R}^3$. To choose a reasonable one from the space of all mappings, we have to impose some constraints on the deformation. This is called regularization of F. In our approach, F is implicitly regularized since it is obtained by minimizing a deformation energy which is carefully designed to favor smooth, feature-preserving, not too distortive deformations. The closeness of shapes is measured based on closest-point search.

We assume that the input shapes are roughly pre-aligned. This holds in a variety of situations like in the case of scanning a deforming object at high frame rates such that the inter-frame displacements are small.

Contributions and Overview

We propose a unified framework for shape modeling and deformable 3D shape registration. Within this framework, we develop an efficient, numerically stable energy minimization algorithm which solves, with minimal adaptation, both problems.

We model the input shapes as a collection of rigid cells connected to each other with elastic strings. By changing the type of the cells (edges, triangles, tetrahedra, prisms, cubes, etc.), we show that several non-linear shape deformation techniques [89, 90, 91, 92, 93, 94] can be seen as special cases of our general approach.

The rest of the chapter is organized as follows. After reviewing related work in Section 4.1, we introduce our shape representation in Section 4.2. In Section 4.3, we present the unified framework by introducing our energy function and a numerical procedure for minimizing it. Sections 4.4 and 4.5 focus on deformation-based shape modeling and deformable 3D shape registration, respectively.

4.1 Related Work

4.1.1 Deformation-Based 3D Shape Modeling

Interactive deformation-based shape editing is a well-studied and still active research field in computer graphics and there is a large variety of existing approaches. Most of them can be classified as either linear or non-linear techniques.

A good overview over linear deformation methods can be found in [95, 96]. Tensor-product spline surfaces and spline-based freeform deformations [97, 98, 99, 96] are among the most common shape modeling and deformation techniques. The user defines a surface by manipulating control points of some sort of a control lattice (planar rectangular grid, volumetric lattice, wireframe collection, etc.). These techniques provide a great deal of flexibility and freedom to the designer, however, they usually need a lot of user guidance for generating physically plausible surface/solid deformations.

Another class of linear techniques consists of methods minimizing linearized elastic energies. The surface is considered to be a thin physical shell and an energy functional which penalizes stretching and bending is defined [100]. However, since a non-linear minimization of this energy is computationally too expensive to perform at interactive frame rates, the energy is usually simplified and linearized [101, 102]. Further conceptually similar approaches were presented in [103, 104, 105, 106].

A further class of linear methods does not directly operate on spatial coordinates but rather modifies differential surface properties and uses them to reconstruct the desired deformed surface [107, 108, 109, 110]. Some approaches [109, 111] deform the surface by constructing a target gradient field based on the user input and then computing a surface mesh which matches the field in least square sense. Other methods [107, 108] manipulate Laplacians of the mesh vertices instead of using a gradient field. They compute initial Laplacians for the initial (undeformed) state of the surface which are modified using the user constraints and the new mesh is computed based on the modified Laplacians. For more details, refer to the provided references.

All above cited linear deformation techniques produce significant artifacts when the user input implies large shape deformations. Some methods deal well with large translations [106] but have difficulties with large rotations. Others [111, 110] exhibit the opposite behavior. Deformations containing both large rotational and translational parts remain a challenge for linear methods.

This is the reason why researchers started to investigate non-linear techniques. In [90], a prism-based non-linear shape deformation technique, called PriMo, was proposed. First, a thin layer of *rigid* prisms enveloping the input mesh is created by extruding the mesh triangles in both directions along the vertex normals. The prisms which share a face are connected to each other by elastic joints. In the initial, rest state of the shape, the shared prism faces coincide, i.e., the joints have zero thickness and the deformation energy is zero. If, however, the shape is deformed, some of the joints have to be stretched since the prisms are kept rigid. This leads

to an energy increase. Although defined as an integral over the prism faces, the deformation energy simplifies to a quadratic expression in the four face vertices. Thus, the authors [90] solve a quadratic minimization problem under the constraints of prism rigidity.

The PriMo deformation approach was extended in [92]. Instead of dealing with thin prisms the authors propose to embed the input shape in a volumetric grid of cubes with adaptively varying size. The energy formulation is similar to [90] and, again, the cells are kept rigid during minimization. As it will become clear in Section 4.4, we can treat both approaches [90, 92] within our framework by changing the type of cells we use.

In [112], another deformation approach based on shape embedding was proposed. The authors created a so-called deformation graph by distributing its nodes uniformly over the input shape and by connecting each graph node with the k closest ones. An affine transform is then associated to each graph node and an energy function which is a weighted sum of three terms is formulated: one term penalizes differences of the affine transforms from rigid motions, the second one penalizes differences between the affine matrices of neighboring graph nodes and the third one makes sure that the user-defined constraints on the deformation are met. The affine matrices minimizing this energy are smoothly blended over the shape vertices to produce the final shape deformation. Although this technique produces naturally looking deformations, comparable to ours, it is considerably more complicated to implement than our method. Furthermore, the individual energy terms listed above are weighted in an ad-hoc manner to compute the sum.

Another non-linear shape deformation technique, which can be assimilated in our deformation framework, was introduced in [91]. The input mesh is considered as a collection of overlapping cells, each one consisting of the triangles adjacent to a mesh vertex. The energy to be minimized penalizes differences between the form of each cell in the initial, undeformed mesh state and the current, deformed one. The minimization is performed with a two-step algorithm: first, for a given set of fixed cell rotations, optimal translations are computed which are then kept fixed to compute optimal cell rotations in the second step. Both steps take the user constraints on the deformation into account. The authors used a linear deformation technique to compute the initial point for this non-linear optimization procedure.

4.1.2 Deformable 3D Shape Registration

As in the context of rigid 3D shape registration (Chapter 2) and 3D object recognition (Chapter 3), feature-based methods can be used for deformable registration as well. However, since we are dealing with non-rigid transforms, the employed feature descriptors should be invariant to the type of deformation the input data is undergoing [30, 31, 32, 33]. However, as already discussed in the previous two chapters, detecting feature points and establishing the correspondence can be problematic especially in the presence of noise and missing data. Furthermore, many shapes do not have distinctive features which gives rise to many ambiguous correspondences and the matching algorithm degenerates to a brute force search [34].

A different strategy is to transform the shapes to a canonical representation in a suitable space in which the correspondence problem is easier to solve. Several papers [113, 114, 115, 116] proposed to compute isometry-invariant embeddings of the original shapes in a low-dimensional Euclidean space and to establish the correspondence using rigid registration algorithms. These methods, however, tend to be costly and, moreover, fail for incomplete data (caused by surface holes, partial views, etc.).

The methods cited so far solve the correspondence problem even in the presence of significant deformations and without making any assumptions about an initial alignment of the shapes. However, the deformations are restricted to isometries (an exception is [32] which can handle an additional global or local scaling). Furthermore, the actual warp between the shapes has to be computed in an additional step using the established correspondences as constraints [89, 93, 92, 94]. In contrast to this, our method is not restricted to a particular family of transformations and it efficiently computes both a dense correspondence and a warp between the shapes.

There is a variety of registration algorithms specialized to articulated shapes. [117] presented a framework for deformable marker-based fitting of a high-resolution template to 3D scans of different humans in the same pose. In [118] a deformable model was learned that is able to synthesize realistic muscle deformations based on the pose of an articulated human skeleton. Both methods can be used for human shape completion as well. Further shape completion algorithms which use deformable registration modules were presented in [119, 120]. In [121], a fully automatic approach for articulated shape registration was proposed. The problem is converted to a discrete labeling problem and solved via graph cuts. However, this seems to be very costly since the authors report processing times of more than an hour for shapes consisting of around 12,000 points.

A further class of non-rigid registration algorithms consists of iterative solvers. They deform the source shape in an iterative fashion until an "optimal" alignment to the target shape is achieved. Many methods in this class are extensions of the classic ICP algorithm [46, 24]. In [122], a non-rigid registration technique was introduced which decomposes the input scans in a coarse-to-fine hierarchical manner in overlapping rigid pieces which are aligned separately. However, the resulting deformation is not continuous which can lead to artifacts in the overlapping regions. Furthermore, the procedure has a quadratic time complexity in the number of pieces. In [123], the discontinuity issue was resolved by incorporating a global thin-plate splines warp which guarantees the smoothness of the solution. A generalization of this method to the simultaneous matching of multiple scans was proposed in [124].

Instead of assuming a one-to-one correspondence between the shape points, one-to-many relaxations can be used in order to enlarge the basin of convergence and thus to increase robustness to imprecise initializations. Significant contributions along these lines are the softassign and deterministic annealing technique [125] and the coherent point drift algorithm [126]. A statistical registration approach without explicitly establishing point-to-point correspondences was

proposed in [49]. The input point sets are modeled as probability distributions and a distance measure between them is minimized over the transform space. Recently, a Gaussian mixture models-based approach [127] was proposed which can be seen as a generalization of [125, 49, 126]. However, these algorithms compute registration results which are not as precise as ours and are much slower than our method (see the experimental comparisons in Section 5.3.3).

A deformable ICP extension was introduced in [38]. The authors formulated a cost function, similar to the one used in [117], which measures the quality of a given non-rigid alignment between the shapes. The deformation is modeled using one affine 3×4 transformation matrix per shape point. This gives rise to a cost function of $12m$ variables, where m is the number of points in the source shape. In order to solve this highly underdetermined system, a stiffness term (a regularizer) is introduced. It penalizes differences between the transformation matrices of neighboring points.

A similar strategy was proposed in [40]. The authors iteratively minimized an error measure which is based on an elastic convolution between the difference of corresponding points in the shapes. This is the 3D surface patch analog to 2D template matching commonly used in image processing. In [39], the deformation is also modeled using one affine transformation matrix per point. The cost function comprises four energy terms and depends on $15m + 6$ variables. The authors minimized it with the Levenberg-Marquardt algorithm.

Note that the iterative methods cited above model the deformation in a very redundant way: many more degrees of freedom are introduced than needed to describe an arbitrary motion of a system of m points in $\mathbb{R}^3$. This results in high-dimensional and computationally heavy optimization problems which are often solved with general purpose optimizers. In contrast, we use a home-grown, two-step minimization algorithm which avoids to repeatedly solve large linear systems which is usually done during the minimization of non-linear cost functions. Our method is easy to implement and it is very efficient in terms of both computational complexity and memory.

4.2 Shape Representation

Definition 1. *We represent a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ by a pair consisting of a finite vertex set $\mathcal{V}$ and an underlying topology structure $\mathcal{T}$, where*

- $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ *is the set of vertices with initial positions* $\mathbf{x}_1^0, \ldots, \mathbf{x}_m^0 \in \mathbb{R}^3$ *and*
- $\mathcal{T} = \{\alpha_i = (\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_k}) : \mathbf{v}_{i_q} \in \mathcal{V}\}$ *is the set of cells, i.e., edges ($k = 2$), triangles ($k = 3$) and/or polyhedra ($k \geq 4$).*

In the following, the *current* position of a vertex $\mathbf{v}_i$ is denoted by $\mathbf{x}_i \in \mathbb{R}^3$. Figure 4.1(a) shows a simple shape consisting of vertices connected by edges.

Let $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ be a shape and $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ a collection of cells specifically tailored to $\mathcal{S}$ as follows. Each cell $\mathbf{C}_j$ is defined by specifying its type (edge, triangle, tetrahedron, etc.)

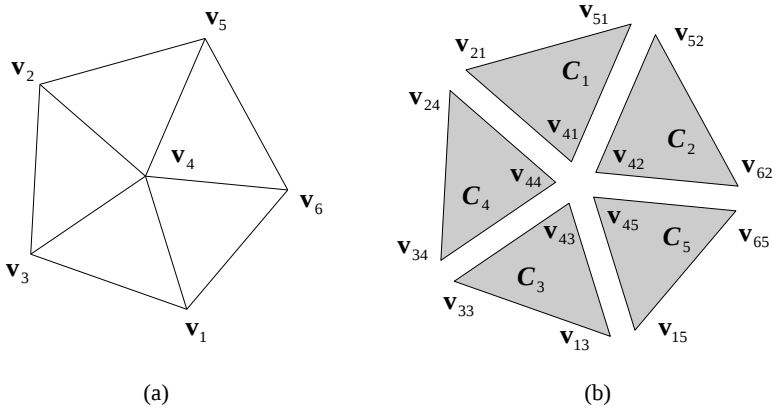


Figure 4.1: (a) A simple shape having six vertices $\mathbf{v}_1, \ldots, \mathbf{v}_6$ and ten edges. (b) A cover consisting of five triangular cells $\mathbf{C}_1, \ldots, \mathbf{C}_5$. For example, $\mathbf{v}_1$ corresponds to $\mathbf{v}_{13}$ and $\mathbf{v}_{15}$ and the neighborhoods of $\mathbf{v}_1$ are $\mathbf{N}_1^s = \{\mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_6\}$ and $\mathbf{N}_1^c = \{\mathbf{v}_{13}, \mathbf{v}_{15}\}$.

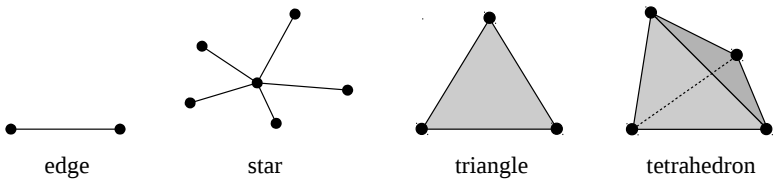


Figure 4.2: Cells of different type.

and the vertices it consists of. Figure 4.2 shows various types of cells. More formally, each $\mathbf{C}_j = (\mathbf{v}_{j_1 j}, \ldots, \mathbf{v}_{j_v j})$ consists of v vertices, where $\mathbf{v}_{j_r j}$ is a doubly indexed vertex positioned at $\mathbf{x}_{j_r j} \in \mathbb{R}^3$. The first index, j_r, is the id of the corresponding vertex in $\mathcal{V}$ and the second one, j, is the cell id. For example, $\mathbf{v}_{25}$ corresponds to $\mathbf{v}_2 \in \mathcal{V}$, denoted by $\mathbf{v}_2 \leftrightarrow \mathbf{v}_{25}$ and it is part of $\mathbf{C}_5$, denoted by $\mathbf{v}_{25} \in \mathbf{C}_5$. Note that $\mathcal{C}$ consists of cells distinct than the ones in $\mathcal{T}$. Furthermore, each cell in $\mathcal{C}$ has its own vertex set which can be modified independently, that is, without altering the other cells or the shape $\mathcal{S}$. Having this concept in mind, we introduce the following

Definition 2. *A cover of a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ is a collection $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ of cells of the same type such that for each shape vertex $\mathbf{v}_i \in \mathcal{V}$ there is at least one $\mathbf{v}_{ij} \in \mathbf{C}_j$ with $\mathbf{v}_i \leftrightarrow \mathbf{v}_{ij}$. In particular, all cells in a cover have the same number of vertices.*

Figure 4.1(b) shows a cover of the shape shown in Figure 4.1(a). We call the vertices in $\mathcal{V}$ shape vertices and the ones belonging to the cells in $\mathcal{C}$ cover vertices. Furthermore, the cells in $\mathcal{T}$ are called shape cells and the ones in $\mathcal{C}$ cover cells.

Definition 3. *Given a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ and a cover $\mathcal{C}$, we define two types of neighborhoods of a vertex $\mathbf{v}_i \in \mathcal{V}$:*

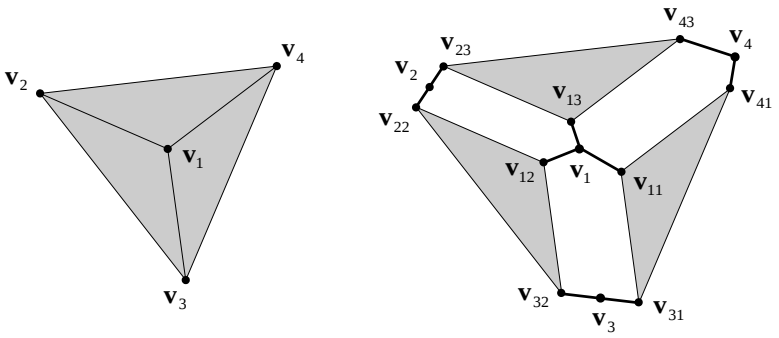


Figure 4.3: (a) A mesh in a rest state. (b) A cover consisting of triangular cells in a non-minimum energy state. The elastic strings connecting the shape vertices $\mathbf{v}_1, \ldots, \mathbf{v}_4$ to their corresponding cover vertices are indicated by the bold lines. The length of the strings is used to measure deformation energy.

- *The* shape *neighborhood* $\mathbf{N}_i^s$ *is the set of* shape *vertices which share a* shape *cell with* $\mathbf{v}_i$*:*

$$\mathbf{N}_i^s = \{\mathbf{v}_j \in \mathcal{V} : \exists \alpha_k \in \mathcal{T} \textit{ such that } \mathbf{v}_i \in \alpha_k \textit{ and } \mathbf{v}_j \in \alpha_k\}. \tag{4.1}$$

- *The* cover *neighborhood* $\mathbf{N}_i^c$ *is the set of* cover *vertices corresponding to* $\mathbf{v}_i$*:*

$$\mathbf{N}_i^c = \{\mathbf{v}_{ij} \in \mathbf{C}_j : \mathbf{C}_j \in \mathcal{C} \textit{ and } \mathbf{v}_i \leftrightarrow \mathbf{v}_{ij}\}. \tag{4.2}$$

To provide an example, both neighborhoods of $\mathbf{v}_1$ are listed in the caption of Figure 4.1.

4.3 Energy Formulation and Minimization

Given a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$, we first compute a cover $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ made of cells of a certain type. We think of the cells as rigid elements connected to each other by elastic strings: each cover vertex $\mathbf{v}_{ij} \in \mathbf{C}_j$ is connected by a string to its corresponding shape vertex $\mathbf{v}_i \in \mathcal{V}$. When the shape is in its initial (rest) state, each string has length zero, since the positions of $\mathbf{v}_{ij}$ and $\mathbf{v}_i$ coincide, and the system is in a minimum energy state. Obviously, the string lengths are preserved if we rigidly transform the whole shape together with the cover cells. If, however, some shape vertices or cells are moved independently then the length of the strings attached to them increases and so does the energy. Figure 4.3 provides an example. The idea to model shape deformations by using rigid cells connected to each other by elastic joints was first proposed in [90].

4.3.1 Problem Formulation

Having the concept of rigid cells connected by elastic strings in mind, we now formulate a shape deformation energy. Recall that the positions of vertices $\mathbf{v}_i$ and $\mathbf{v}_{ij}$ are denoted by $\mathbf{x}_i \in \mathbb{R}^3$ and

$\mathbf{x}_{ij} \in \mathbb{R}^3$. The geometrical configuration of all source shape vertices $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathcal{V}$ is specified by the point $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathbb{R}^{3m}$. Similarly, the geometrical configuration of a cover cell $\mathbf{C}_j = (\mathbf{v}_{j_1 j}, \ldots, \mathbf{v}_{j_v j})$ with v vertices is defined by the point $\mathbf{Y}_j = (\mathbf{x}_{j_1 j}, \ldots, \mathbf{x}_{j_v j}) \in \mathbb{R}^{3v}$ and the geometry of the whole cover $\mathcal{C}$ is given by the point $\mathbf{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_n) = (\ldots, \mathbf{x}_{ij}, \ldots) \in \mathbb{R}^{3nv}$, where n is the number of cells in the cover. (Recall that all cover cells have the same number of vertices.)

Next, we formulate the deformation energies ξ_i and σ_j of a vertex cover neighborhood $\mathbf{N}_i^c$ and a cell $\mathbf{C}_j$, respectively:

$$\xi_i = \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2, \tag{4.3}$$

$$\sigma_j = \sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2, \tag{4.4}$$

Although these expressions look quite similar, the terms involved in the sums are not the same. In (4.3), $\mathbf{N}_i^c$ and thus $\mathbf{v}_i$ are fixed and the sum goes over the $\mathbf{v}_{ij}$'s corresponding to $\mathbf{v}_i$, i.e., j is the running index. In (4.4), the cell $\mathbf{C}_j$ is fixed and we are summing over the pairs $\mathbf{v}_i \leftrightarrow \mathbf{v}_{ij}$ of corresponding vertices which means that the running index is i. If we want to emphasize the dependence of the cell deformation energy on the cover vertex positions $\mathbf{x}_{ij}$, we write $\sigma_j(\mathbf{Y}_j)$.

In order to register a source to a target shape (see Section 4.5), each shape vertex $\mathbf{v}_i$ may have a target position $\mathbf{q}_i \in \mathbb{R}^3$ which attracts it and thus influences the shape deformation. We model this by adding an additional term to the energy defined in (4.3):

$$\eta_i = w_i \|\mathbf{x}_i - \mathbf{q}_i\|^2 + \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2, \tag{4.5}$$

where $w_i > 0$ weights the influence of $\mathbf{q}_i$. We will discuss weighting issues in Section 4.5.1. Thus, the deformation energy of a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ given a cover $\mathcal{C}$ and a set $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ of target positions is

$$E(\mathbf{X}, \mathbf{Y}) = \sum_{\mathbf{v}_i \in \mathcal{V}} \eta_i = \sum_{\mathbf{v}_i \in \mathcal{V}} \left(w_i \|\mathbf{x}_i - \mathbf{q}_i\|^2 + \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2 \right). \tag{4.6}$$

In order to make the role of the cells of the cover more explicit (which will be needed in the next section) we rewrite (4.6) in the following way:

$$E(\mathbf{X}, \mathbf{Y}) = \sum_{\mathbf{v}_i \in \mathcal{V}} \left(w_i \|\mathbf{x}_i - \mathbf{q}_i\|^2 + \xi_i \right) \tag{4.7}$$

$$= \sum_{\mathbf{v}_i \in \mathcal{V}} w_i \|\mathbf{x}_i - \mathbf{q}_i\|^2 + \sum_{\mathbf{v}_i \in \mathcal{V}} \xi_i \tag{4.8}$$

$$= \sum_{\mathbf{v}_i \in \mathcal{V}} w_i \|\mathbf{x}_i - \mathbf{q}_i\|^2 + \sum_{\mathbf{C}_j \in \mathcal{C}} \sigma_j. \tag{4.9}$$

The last equality follows from the fact that $\sum_{\mathbf{v}_i \in \mathcal{V}} \xi_i = \sum_{\mathbf{C}_j \in \mathcal{C}} \sigma_j$ which is easy to see since in both sums each $\|\mathbf{x}_i - \mathbf{x}_{ij}\|^2$ is evaluated exactly once.

The minimization problem we want to solve is

$$\min_{\mathbf{X},\mathbf{Y}\in\mathbb{R}^{3m}\times\mathbb{R}^{3nv}} E(\mathbf{X},\mathbf{Y}) \tag{4.10}$$

$$\text{s.t. each } \mathbf{C}_j \in \mathcal{C} \text{ is rigid (up to scale).} \tag{4.11}$$

It is the second condition, namely, that each cell $\mathbf{C}_j$ is kept rigid, which prevents the shape from degenerating even under strong deformations and makes the numerical computation stable. If we want to include local scale in the deformation/registration process, $\mathbf{C}_j$ should be rigid up to scale, i.e., the motion of the cells is defined by similarity transforms.

4.3.2 Numerical Minimization

Let a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ be given in an initial, rest configuration $\mathbf{X}^0 = (\mathbf{x}_1^0, \ldots, \mathbf{x}_m^0) \in \mathbb{R}^{3m}$ and in a current, deformed state $\widetilde{\mathbf{X}} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathbb{R}^{3m}$. Let $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ be a cover with geometrical configuration $\mathbf{Y}^0 = (\mathbf{Y}_1^0, \ldots, \mathbf{Y}_n^0) = (\ldots, \mathbf{x}_{ij}^0, \ldots) \in \mathbb{R}^{3nv}$ which is identical to the one of the shape $\mathcal{S}$. This means that each cover vertex $\mathbf{v}_{ij} \in \mathbf{C}_j$ has initial position $\mathbf{x}_{ij}^0$ identical to the initial position of its corresponding shape vertex $\mathbf{v}_i$, that is, $\mathbf{x}_{ij}^0 = \mathbf{x}_i^0$.

Furthermore, let $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ be given, where $\mathbf{q}_i \in \mathbb{R}^3$ is the (fixed) target position of the shape vertex $\mathbf{v}_i \in \mathcal{V}$. Using $\widetilde{\mathbf{X}}$ and $\mathbf{Y}^0$ as a starting point, we propose to solve (4.10) subject to (4.11) using a simple and efficient algorithm consisting of the following two steps which are repeated until convergence:

1. (a) Consider all $\mathbf{x}_i$'s as constant and compute for each cell $\mathbf{C}_j$ the rigid/similarity transform T_j which minimizes $\sigma_j(T_j(\mathbf{Y}_j^0))$ in (4.9).
   (b) For each cell $\mathbf{C}_j$ update its vertex positions: $\mathbf{x}_{ij} := T_j(\mathbf{x}_{ij}^0)$.
2. Consider all $\mathbf{x}_{ij}$'s as constant and compute the $\mathbf{x}_i$'s which solve (4.10).

$T_j(\mathbf{Y}_j^0)$ denotes the pointwise application of T_j to $\mathbf{Y}_j^0$, i.e., $T_j(\mathbf{Y}_j^0) = (T_j(\mathbf{x}_{j_1 j}^0), \ldots, T_j(\mathbf{x}_{j_v j}^0))$. The minimization procedure is more rigorously formulated in Algorithm 2. It obviously converges since each step leads to an energy decrease: the first one decreases $\sum \sigma_j$ in (4.9) and the second one decreases each term in the outer sum in (4.6).

Step 1.

The minimization involved in the first step of the proposed numerical procedure (line 7 in Algorithm 2) is called the absolute orientation problem and is often encountered in different fields as part of different computational problems [24, 89, 128]. We seek the similarity transform T_j, defined by a scale factor $s \in \mathbb{R}$, a rotation matrix $R \in SO(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$, which minimizes

$$\sigma_j(T_j(\mathbf{Y}_j^0)) = \sum_{\mathbf{v}_{ij}\in\mathbf{C}_j} \|\mathbf{x}_i - (sR\mathbf{x}_{ij}^0 + \mathbf{t})\|^2, \tag{4.12}$$

input : – shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$; each $\mathbf{v}_i \in \mathcal{V}$ has rest position $\mathbf{x}_i^0$ and current position $\widetilde{\mathbf{x}}_i$;
– cover $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$; each cover vertex $\mathbf{v}_{ij} \in \mathbf{C}_j$ has initial position $\mathbf{x}_{ij}^0$ equal to the initial position of its corresponding shape vertex $\mathbf{v}_i$, i.e., $\mathbf{x}_{ij}^0 = \mathbf{x}_i^0$;
– set $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ of target positions $\mathbf{q}_i \in \mathbb{R}^3$;
output: a pair $(\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{3m} \times \mathbb{R}^{3nv}$ solution of (4.10) s.t. (4.11);

// Initialization
1 $k := 1$;
2 $\mathbf{X}^k := (\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_m)$;
3 $\mathbf{Y}^k := (\mathbf{Y}_1^0, \ldots, \mathbf{Y}_n^0) = (\ldots, \mathbf{x}_{ij}^0, \ldots)$;
// Minimization
4 **repeat**
  // Step 1.
5 set $\mathbf{X}^k$ constant;
6 **for** $\mathbf{C}_j \in \mathcal{C}$ **do**
7 compute a transform T_j which minimizes $\sigma_j\big(T_j(\mathbf{Y}_j^0)\big) = \sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} \|\mathbf{x}_i^k - T_j(\mathbf{x}_{ij}^0)\|^2$;
8 **end**
9 $\mathbf{Y}^{k+1} := (T_1(\mathbf{Y}_1^0), \ldots, T_n(\mathbf{Y}_n^0))$;
  // Step 2.
10 set $\mathbf{Y}^{k+1}$ constant and solve $\mathbf{X}^{k+1} := \underset{\mathbf{X} \in \mathbb{R}^{3v}}{\operatorname{argmin}} E(\mathbf{X}, \mathbf{Y}^{k+1})$ (see (4.10));
  // Prepare for the next iteration
11 $k := k + 1$;
12 **until** $\big|E(\mathbf{X}^k, \mathbf{Y}^k) - E(\mathbf{X}^{k-1}, \mathbf{Y}^{k-1})\big| > \epsilon$;
13 **return** $(\mathbf{X}^k, \mathbf{Y}^k)$;

Algorithm 2: The algorithm solving (4.10) subject to (4.11). The transform T_j in line 7 is either a rigid or a similarity transform. The $\mathbf{X}^k$ returned by the algorithm defines a (local) minimum energy state of the shape.

where $\mathbf{x}_i$ is the current fixed position of the shape vertex $\mathbf{v}_i$ and $\mathbf{x}_{ij}^0$ denotes the initial position of the cover vertex $\mathbf{v}_{ij}$. We adopt the solution presented in [128]. For the sake of clarity we define

$$\mathbf{s}_j = \frac{1}{|\mathbf{C}_j|} \sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} \mathbf{x}_i, \tag{4.13}$$

$$\mathbf{c}_j^0 = \frac{1}{|\mathbf{C}_j|} \sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} \mathbf{x}_{ij}^0, \tag{4.14}$$

with $|\mathbf{C}_j|$ being the number of vertices of $\mathbf{C}_j$. Next, a linear deformation matrix A is computed:

$$A = \sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} (\mathbf{x}_i - \mathbf{s}_j)(\mathbf{x}_{ij}^0 - \mathbf{c}_j^0)^T, \tag{4.15}$$

and the optimal rotation R is extracted from the singular value decomposition $A = U\Sigma V^T$ in the following way:

$$R = UCV^T, \quad C = \mathrm{diag}(1, \ldots, 1, \det(UV^T)), \tag{4.16}$$

where the diagonal matrix C assures that R is a rotation and not a reflection. The scale factor is given by

$$s = \sqrt{\frac{\sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} \|\mathbf{x}_i - \mathbf{s}_j\|^2}{\sum_{\mathbf{v}_{ij} \in \mathbf{C}_j} \|\mathbf{x}_{ij}^0 - \mathbf{c}_j^0\|^2}} \tag{4.17}$$

and the translation vector is computed as

$$\mathbf{t} = \mathbf{s}_j - sR\mathbf{c}_j^0. \tag{4.18}$$

Once we have computed the optimal transform for each cell, the cover vertex positions are updated for the next iteration:

$$\mathbf{x}_{ij} \;:=\; sR\mathbf{x}_{ij}^0 + \mathbf{t}. \tag{4.19}$$

Performing the update with $s = 1$ results in a shape regularizer which is well-suited to model as-rigid-as-possible deformations. Using Eq. (4.17) allows to include local scale.

The rigid transform-based regularizer was first introduced in [89] for the generation of physically plausible animations of deforming objects. Further improvements, again, for animating deformations, were proposed in [93, 94]. Including local scale and using this technique in the context of non-rigid shape registration was first proposed in [2].

Step 2.

The second step (line 10 in Algorithm 2) is easy to perform, since all we have to do is to compute the derivative of E (see (4.6)) with respect to $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, denoted by $\nabla_{\mathbf{X}} E$, and solve the linear system of equations $\nabla_{\mathbf{X}} E = 0$ for $\mathbf{X}$. More precisely, we have to solve

$$\nabla_{\mathbf{X}} E = \begin{pmatrix} \nabla_{\mathbf{x}_1} \eta_1 \\ \nabla_{\mathbf{x}_2} \eta_2 \\ \vdots \\ \nabla_{\mathbf{x}_m} \eta_m \end{pmatrix} = 0. \tag{4.20}$$

Fortunately, since the $\mathbf{v}_i$'s are not directly connected to each other but only over (fixed) cover vertices, each $\nabla_{\mathbf{x}_i} \eta_i$ depends only on $\mathbf{x}_i$. Thus, (4.20) consists of m independent equations stacked on top of each other. The optimal $\mathbf{x}_i$ is the solution of $\nabla_{\mathbf{x}_i} \eta_i = 0$, that is, using (4.5), the solution of

$$w_i(\mathbf{x}_i - \mathbf{q}_i) + \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} (\mathbf{x}_i - \mathbf{x}_{ij}) = 0, \tag{4.21}$$

which is

$$\mathbf{x}_i = \frac{1}{w_i + |\mathbf{N}_i^c|} \left(w_i \mathbf{q}_i + \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} \mathbf{x}_{ij} \right), \tag{4.22}$$

where $|\mathbf{N}_i^c|$ denotes the number of elements in $\mathbf{N}_i^c$.

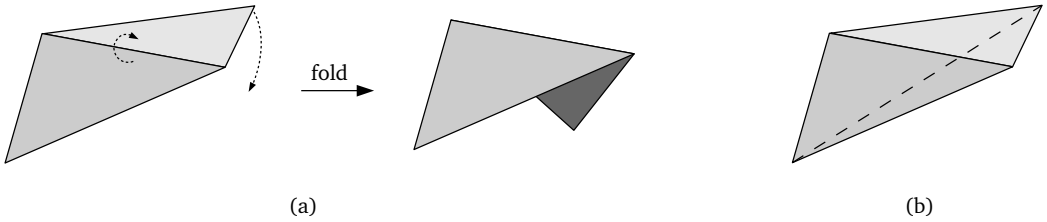


Figure 4.4: (a) The shape can be folded by rotating the upper triangle around the edge without changing edge lengths, i.e., without increasing the deformation energy. (b) Making the shape rigid by inserting an additional edge connecting the vertices opposite to the existing edge.

Remark Recall that at the end of Section 4.1.2 on related work we criticized several deformable registration approaches on the basis that they introduce many more degrees of freedom than necessary to model a general motion of a system of m points in $\mathbb{R}^3$. At the same time, we use the concept of a shape cover which requires to copy shape vertices together with additional topological information depending on the particular cell type. However, this is done very effectively, since just a single copy of the shape is enough because only the initial and the current shape state have to be stored during the minimization. In the case of edge cells, only the initial edge lengths are needed which means that we do not even have to copy the shape vertices.

4.3.3 Shape Covers and Cell Types

In this section, we discuss how to compute a shape cover and, what is more important, what type of cells to use. As we will see in Section 4.4, the particular cell type has a strong effect on the deformation result. Moreover, for certain types, the computation of the optimal transform in line 7 in Algorithm 2 gets much simpler which significantly speeds up the processing. We discuss thoroughly edges and star cells and provide a short discussion on other cell types as well.

Edge Cells

An edge is a pair of vertices connected to each other. Employing edges leads to a minimization method which strives to preserve lengths and essentially behaves like a mass-spring system. Note, however, that we are interested only in the equilibrium state of the system and not in simulating its dynamical behaviour. We compute a cover of edge cells in the following way. If the shape consists of vertices and edges only, we include the edges in the cover. Otherwise, they are extracted from the triangles/polyhedra of the shape. In either case, we end up with a framework in $\mathbb{R}^3$ which we think of as a collection of straight, stiff rods connected by articulated joints at the vertices.

In order for our minimization to make sense, we have to make sure that the framework is rigid, i.e., inflexible. Otherwise it is not guaranteed that a deviation from the undeformed

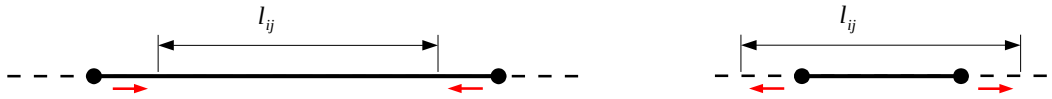


Figure 4.5: The vertices of a stretched (compressed) edge are translated innwards (outwards) until the edge attains its rest length l_{ij}.

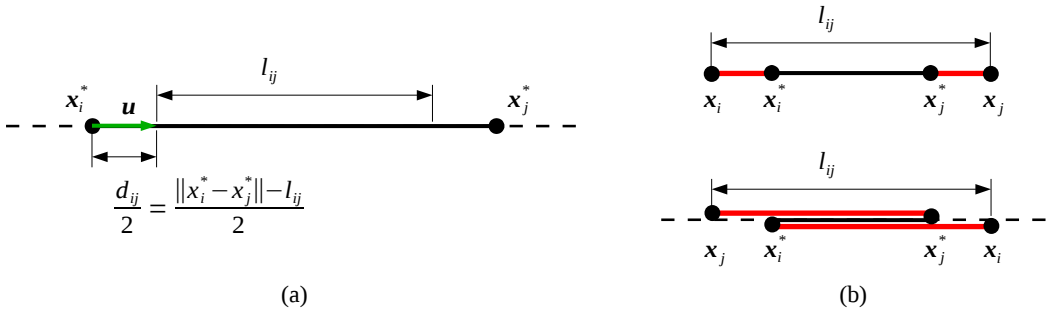


Figure 4.6: (a) The displacement vector $\mathbf{u}$ (shown in green) and the length difference d_{ij} used in Theorem 2. (b) The two configurations $(\mathbf{x}_i, \mathbf{x}_j)$ which are solutions to (4.23) subject to (4.24). In the lower subfigure, the four points are drawn slightly off the straight (dashed) line they are supposed to lie on, in order to see the connectivity between the points. Obviously, the solution shown in the upper subfigure leads to a lower value for f as claimed in Theorem 2.

shape state leads to an energy increase. This is due to the fact that flexible frameworks can deform continuously without changing their edge lengths. Figure 4.4(a) provides an example. In the case of manifold surface meshes, we make the corresponding framework rigid, by inserting an antagonist to each existing (inner) edge by connecting the vertices opposite to it (see Figure 4.4(b)). This guarantees that the mesh cannot be deformed continuously without varying the energy. In the case of manifold tetrahedral meshes, the framework consisting of the edges of the tetrahedra is rigid, since each tetrahedron is a rigid element, rigidly connected (over its faces) to its neighboring elements.

An advantage of an edge cover is that the absolute orientation problem (line 7 in Algorithm 2) is much simpler and faster to solve. Given a deformed (i.e., stretched or compressed) edge with its vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ placed at $\mathbf{x}_i^* \in \mathbb{R}^3$ and $\mathbf{x}_j^* \in \mathbb{R}^3$, our aim is to solve

$$\min_{\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^3} f(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_i^*\|^2 + \|\mathbf{x}_j - \mathbf{x}_j^*\|^2, \tag{4.23}$$

$$\text{s.t. } g(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2 - l_{ij}^2 = 0, \tag{4.24}$$

where l_{ij} is the length of the edge $(\mathbf{v}_i, \mathbf{v}_j)$ in the initial, undeformed shape state. Instead of applying the general method outlined in the first step in Section 4.3.2, we solve the problem by simply translating the vertices along the edge axis until the edge attains its original, rest length. Figure 4.5 illustrates this. We put it formally in the next theorem.

Theorem 2. *The solution of (4.23) subject to (4.24) is given by*

$$\mathbf{x}_i = \mathbf{x}_i^* + \mathbf{u}, \tag{4.25}$$

$$\mathbf{x}_j = \mathbf{x}_j^* - \mathbf{u}, \tag{4.26}$$

where $\mathbf{u}$ *is the displacement vector*

$$\mathbf{u} = \frac{d_{ij}}{2} \frac{\mathbf{x}_j^* - \mathbf{x}_i^*}{\|\mathbf{x}_j^* - \mathbf{x}_i^*\|}, \tag{4.27}$$

with $d_{ij} = \|\mathbf{x}_j^* - \mathbf{x}_i^*\| - l_{ij}$ *being the difference between the current and the initial edge length (see Figure 4.6).*

Proof. Set $\mathbf{M} = \{(\mathbf{x}, \mathbf{y}) \in g^{-1}(0) \subset \mathbb{R}^6 : \nabla g(\mathbf{x}, \mathbf{y}) \neq 0\}$ and note that both f and g are differentiable functions. According to the method of Lagrange multipliers [129, Chapter 2], if f attains a local minimum or maximum on $\mathbf{M}$ at the point $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{M}$, then there exists a $\lambda \in \mathbb{R}$ such that

$$\nabla f(\mathbf{x}_i, \mathbf{x}_j) = \lambda \nabla g(\mathbf{x}_i, \mathbf{x}_j), \tag{4.28}$$

which is equivalent to

$$(\mathbf{x}_i - \mathbf{x}_i^*) = \lambda(\mathbf{x}_i - \mathbf{x}_j) \tag{4.29}$$

$$(\mathbf{x}_j - \mathbf{x}_j^*) = \lambda(\mathbf{x}_j - \mathbf{x}_i). \tag{4.30}$$

These two equations imply that the vectors $\mathbf{x}_i - \mathbf{x}_i^*$, $\mathbf{x}_j - \mathbf{x}_j^*$ and $\mathbf{x}_j - \mathbf{x}_i$ are collinear which means that all solutions to our problem lie on the line through $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$. Furthermore, adding (4.29) to (4.30) yields

$$(\mathbf{x}_i - \mathbf{x}_i^*) = -(\mathbf{x}_j - \mathbf{x}_j^*) \tag{4.31}$$

which implies

$$\|\mathbf{x}_i - \mathbf{x}_i^*\| = \|\mathbf{x}_j - \mathbf{x}_j^*\|. \tag{4.32}$$

To sum up, our solution $(\mathbf{x}_i, \mathbf{x}_j)$ lies on the line through $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ and the distance between $\mathbf{x}_i$ and $\mathbf{x}_i^*$ equals the distance between $\mathbf{x}_j$ and $\mathbf{x}_j^*$. There are only two pairs of points satisfying this and the constraint $g = 0$ at the same time. They are illustrated in Figure 4.5(b) and the one for which f attains a smaller value is given by (4.25) and (4.26), as claimed. □

Despite the simplicity and computational efficiency of the resulting energy minimization algorithm, an edge-based cover leads to many local minima of the energy landscape. This is especially problematic when employing the method in the context of deformation-based shape modeling where the user might define positional constraints on the vertices which lead to strong shape deformations. Thus, it is advisable to use edge-based covers only in situations in which small deformations are expected.

Next, we investigate another cell type which is computationally more expensive but leads to a very well-behaved energy function.

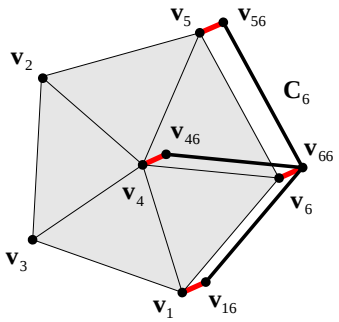


Figure 4.7: An input triangular mesh and a star cell $\mathbf{C}_6$. The cover cell is slightly displaced in order to see the strings (shown in red) connecting the cover vertices with their corresponding shape vertices.

Star Cells

Given a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$, recall from Definition 3 on page 45 how the shape neighborhood $\mathbf{N}_i^s$ of the vertex $\mathbf{v}_i$ is defined. Based on that, the star cells of a cover $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ are defined as

$$\mathbf{C}_j = \{\mathbf{v}_{ij} : \mathbf{v}_i \leftrightarrow \mathbf{v}_{ij} \text{ for all } \mathbf{v}_i \in \{\mathbf{v}_j\} \cup \mathbf{N}_j^s\}. \tag{4.33}$$

This means that the vertices of $\mathbf{C}_j$ are the ones corresponding to $\mathbf{v}_j$ and all its shape neighbors. To put it differently, $\mathbf{C}_j$ is just a copy of $\mathbf{v}_j$ and its shape neighborhood $\mathbf{N}_j^s$. Figure 4.7 illustrates the star cell corresponding to the shape vertex $\mathbf{v}_6$.

In order for a shape cover consisting of star cells to be useful we need the following

Assumption 1. *Each cell $\mathbf{C}_j$ consists of at least three non-collinear vertices.*

If this is not the case, the solution of the absolute orientation problem is not unique, i.e., we are left with some extra degrees of freedom which make it possible to deform the shape without increasing the deformation energy. Fortunately, Assumption 1 holds for many shape representations, like triangular/tetrahedral meshes and voxel grids. For example, in the case of a triangular mesh, each vertex is part of at least one triangle, i.e., it has at least two neighbors which results in cover cells with at least three non-collinear vertices.

If the shape cover consists of star cells, the first step of the energy minimization requires to solve the absolute orientation problem in the general way outlined at the end of Section 4.3.2. This is computationally more expensive then the special solution for the edges presented above. However, if Assumption 1 holds, star cells lead to an energy function which has a unique global minimum. This is due to the fact that each vertex displacement, different than a global rigid transform of all vertices, inevitably leads to a cell distortion and thus to an energy increase.

Other Cell Types

Many other cells types can be used within our deformation framework. In general, for a particular cell type to be of practical use, it has to lead to an energy function which increases as

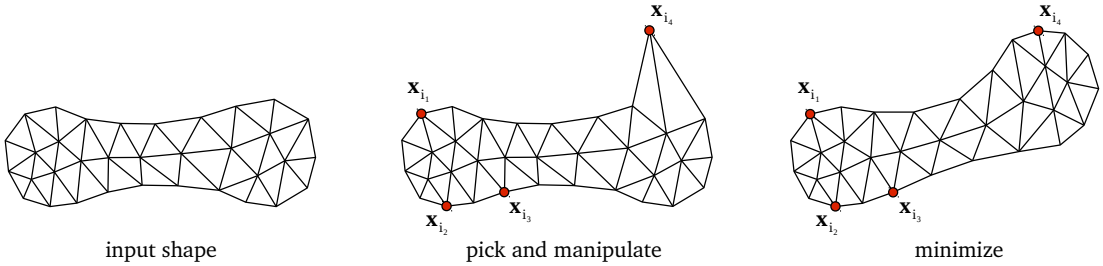


Figure 4.8: Simple example of the shape modeling process. (a) The input shape. (b) Fixing $\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \mathbf{v}_{i_3}$ and moving $\mathbf{v}_{i_4}$ to $\mathbf{x}_{i_4}$. (c) Final shape state after minimizing the deformation energy taking the positional constraints $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_4}$ into account.

soon as a displacement different than a global rigid transform is applied to the shape and/or its cover. In the previous two paragraphs, we saw that under certain conditions this is the case for edges and star cells.

4.4 Deformation-Based Shape Modeling

In this section, we employ our energy minimization approach to create smooth, naturally looking shape deformations. This is done in an interactive, user-guided modeling session. Given a shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$, the first step is to compute a cover $\mathcal{C}$ consisting of cells of a certain type. The cell type has significant influence on the final result as we will see later in the section. Next, the user defines constraints on the deformation by picking vertices and modifying/fixing their positions. Taking these constraints into account, optimal positions for the unconstrained shape vertices are computed by minimizing the deformation energy as described in Section 4.3. Figure 4.8 illustrates the modeling process.

The user input is easily incorporated in our minimization framework by treating the positions $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ of the vertices the user manipulates as constants in Equation (4.6). Furthermore, there are no target positions $\mathbf{q}_1, \ldots, \mathbf{q}_m$ in the context of shape modeling (they are used for shape registration only). Thus, the deformation energy simplifies to

$$E(\mathbf{X}, \mathbf{Y}) = \sum_{\mathbf{v}_i \in \mathcal{V}} \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} \|\mathbf{x}_i - \mathbf{x}_{ij}\|^2 \tag{4.34}$$

and the optimal $\mathbf{x}_i$ is given by (except it is provided by the user)

$$\mathbf{x}_i = \frac{1}{|\mathbf{N}_i^c|} \sum_{\mathbf{v}_{ij} \in \mathbf{N}_i^c} \mathbf{x}_{ij}. \tag{4.35}$$

The first step of the energy minimization (line 7 in Algorithm 2) is unchanged since all $\mathbf{x}_i$'s are anyway treated as constants and there are no $\mathbf{q}_i$'s involved.

Figure 4.9 shows several hand gestures created with our deformation algorithm. On the left side of the figure, the hand is shown in its rest configuration. On the right side, several

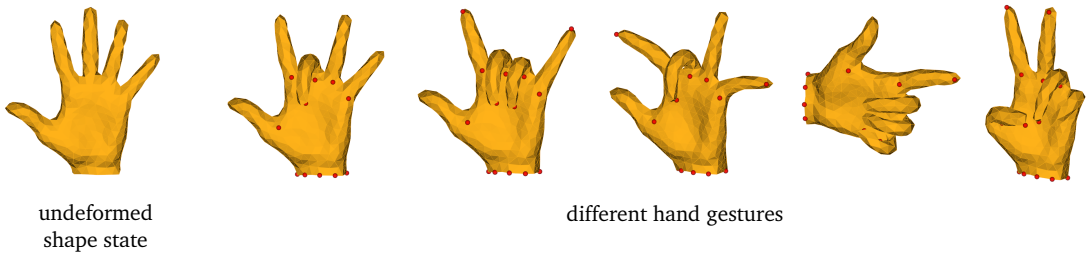


Figure 4.9: Different hand gestures created in an interactive, user-guided modeling session using our deformation-based shape modeling technique with star cells.

Figure 4.10: Comparison between edge-based and star cell-based deformation.

deformation results are shown. The only input provided by the user is the positions of the shape vertices marked by the red dots. Some of them are moved and others are left at their original positions. The rest of the shape is deformed according to our numerical procedure. Note the strong deformation in the examples which, however, is no problem for our method when star cells are used to compute the shape cover.

The situation is different, when edges are used. Figure 4.10 provides a qualitative comparison between deformation results for a star cell and an edge shape cover. Although, an edge-based cover yields good results for small deformations, it fails when the deformation becomes large which is clearly visible on the right side of Figure 4.10.

We end this section with some comments on how other shape modeling approaches can be treated within our framework. If we compute a shape cover consisting of prisms by extruding the triangles of a surface mesh in both directions along the vertex normals, we essentially get the PriMo approach presented in [90]. This results in an energy function with a unique global minimum since each prism is kept rigid and is connected with four strings to each of its neighbors. Thus, a non-rigid motion of the shape leads to an energy decrease.

If we embed the input shape in a cubical grid and consider the grid as the shape and the cubes as the cover cells, we get a shape editing approach similar to [92]. Again, the resulting energy function has a unique global minimum at the initial shape state. Similarly, computing a shape cover made of cells as the ones used in [89, 91, 93, 94] essentially results in the corresponding shape deformation technique.

4.5 Deformable Shape Registration

In this section, we treat the problem of deformable shape registration in the following setting. Given a source shape $\mathcal{S} = (\mathcal{V}, \mathcal{T})$ and a target shape $\mathcal{S}' = (\mathcal{V}', \mathcal{T}')$, our aim to bring the source "close" to the target without distorting the former "too much". This can naturally be treated within our energy minimization framework: (i) setting the $\mathbf{q}_i$'s (see the energy function (4.6)) to be on the target shape makes sure that the source gets attracted to it and (ii) by adjusting the weights w_i we can find a balance between a close shape match and a low shape distortion. Larger weights make $\mathcal{S}$ get closer to $\mathcal{S}'$, however, at the expense of a larger distortion.

4.5.1 Computation of the Target Positions and Their Weights (Correspondence Estimation)

In the following, we define a pointwise correspondence between source and target. As already discussed in Section 4.1.2, there is a substantial amount of work in the field of non-rigid correspondence estimation [32, 33, 31, 30, 130]. These methods solve the problem without making any assumptions about the initial alignment of the shapes but, unfortunately, tend to be costly. Since we assume that the input shapes are roughly pre-aligned, computing the target positions for the source vertices based on closest-point search is fast and yields good results. Define

$$\mathbf{c}_p(\mathbf{x}) = \underset{\mathbf{v}_k \in \mathcal{V}'}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_k\|. \tag{4.36}$$

to be the position of the target vertex closest to $\mathbf{x} \in \mathbb{R}^3$. Using this operator, we set

$$\mathbf{q}_i = \mathbf{c}_p(\mathbf{x}_i), \tag{4.37}$$

where $\mathbf{x}_i$ is the current position of the source vertex $\mathbf{v}_i \in \mathcal{V}$. In this way, each source vertex gets a corresponding target position on $\mathcal{S}'$.

Setting different weights w_i in the energy function (4.6) allows to trust some target positions more than others or even to completely ignore some of them: setting $w_i = 0$ means that $\mathbf{v}_i$ will move only according to its adjacent cover cells and will not be directly influenced by the target shape.

Again, based on the assumption that the input shapes are roughly pre-aligned, it does not make sense to consider vertices as corresponding ones if they are too far apart. Moreover, in the case of registering a complete object model to an incomplete range scan, we want to ignore model vertices not facing the scan. Along this line of thoughts, we compute the weights based on two criteria: (i) how far is a source vertex to its closest target vertex and (ii) how well do the shape normals at corresponding vertices match. Furthermore, we include an additional global weight $w^g \in (0, 1)$ which does not depend on a particular vertex and allows us to globally adjust how "fast" the source is moving to the target. Thus, we end up with three types of weights per target point which are multiplied to give a single scalar

$$w_i = w_i^d w_i^n w^g, \tag{4.38}$$

where d stands for distance, n for normal agreement and g for global. The individual weights are computed as follows.

$$w_i^d = \begin{cases} 1 & \text{if } \|\mathbf{c}_p(\mathbf{x}_i) - \mathbf{x}_i\| \leq d, \\ 0 & \text{otherwise,} \end{cases} \tag{4.39}$$

where $[0, d]$ is the range of influence of the target shape. For w_i^n we have

$$w_i^n = \begin{cases} \mathbf{n}(\mathbf{v}_i) \cdot \mathbf{n}(\mathbf{v}'_j) & \text{if } \angle\left(\mathbf{n}(\mathbf{v}_i), \mathbf{n}(\mathbf{v}'_j)\right) < 90^\circ, \\ 0 & \text{otherwise,} \end{cases} \tag{4.40}$$

where $\mathbf{n}(\mathbf{v})$ denotes the normal at a vertex $\mathbf{v}$ and $\mathbf{v}'_j$ is the target vertex corresponding to the source vertex $\mathbf{v}_i$.

4.5.2 Convergence Issues

For the convergence of the minimization algorithm presented in Section 4.3.2, it is essential that both the target positions and the weights are constant. However, it is never the case that the closest-point search yields the semantically correct correspondences and the right weights at the very first iteration. Rather, it provides an approximation which needs to be refined in the process of minimization. On the other hand, we do not want to sacrifice the provable convergence of the method.

It is easy to see that updating the target positions according to (4.36) at every iteration still leads to a convergent algorithm. Let $\mathbf{q}_i^k$ denote the target position for $\mathbf{v}_i \in \mathcal{V}$ in the k-th iteration. In the next, $(k+1)$-th, iteration the new target position $\mathbf{q}_i^{k+1}$ is computed as $\mathbf{q}_i^{k+1} := \mathbf{c}_p(\mathbf{x}_i^{k+1})$. This leads to

$$w_i \|\mathbf{x}_i^{k+1} - \mathbf{q}_i^{k+1}\|^2 \leq w_i \|\mathbf{x}_i^{k+1} - \mathbf{q}_i^k\|^2. \tag{4.41}$$

since we have either $\mathbf{q}_i^{k+1} = \mathbf{q}_i^k$, i.e., the closest-point search did not yield a closer target position, or the new $\mathbf{q}_i^{k+1}$ is closer to $\mathbf{x}_i^{k+1}$ than the old $\mathbf{q}_i^k$. In either case (4.41) holds which implies that computing the target points at each iteration based on closest-point search does not increase the energy of the system and the algorithm converges.

Unfortunately, the situation is different for the weights. Updating them at every iteration according to (4.38) – (4.40) does not necessary lead to a decreasing sequence of weights which, in turn, could lead to an energy increase/decrease and thus to oscillations. However, we experimentally noticed that good registration results can be achieved even when updating the weights only a fixed, pre-defined number of times. In this way, the energy level can not oscillate and convergence is again guaranteed.

Chapter 5

Experimental Results

In this chapter, we describe the experiments we performed with our methods and explain and analyze the results. There is a separate section devoted to each of the three algorithms presented in the previous three chapters. All tests were performed on a laptop with a 3GHz CPU, 4GB RAM and a Linux operating system. The algorithms are implemented in C++.

5.1 Rigid 3D Shape Registration

In this section, we test our rigid registration method, presented in Chapter 2, on a variety of point sets. The parameter values used in the experiments are given in Table 5.1.

Since our method is a probabilistic one, it computes each time a (slightly) different result. In order to make a statistical meaningful statement about its performance, we run 100 registration trials for each pair of inputs and report the mean performance values. We measure the success rate and the accuracy under varying amount of noise and outliers in the input sets. The success rate gives the percentage of registration trials in which a transform which is close to the global optimal one is found. The accuracy is measured using the RMS error (see [29]). The type of noise added to some of the model and data sets is Gaussian and the outliers are simulated by

	Parameter	Defined in	Value
cost function	d	Eq. (2.15)	1/4(min bbox side($\mathbf{M}$))
	δ	Eq. (2.16)	0.1
cooling schedule	t_{max}	Eq. (2.22)	50.0
	v	Eq. (2.22)	0.00008
stopping rule	δ_v	Sec. 2.3.7	1° rot. and 1% transl.
	δ_f	Sec. 2.3.7	0.1

Table 5.1: The parameter values used in the rigid point set registration experiments. The value of δ_v equals the volume of a spherical box with side one degree times the volume of a box with sides equal to one percent of the sides of the bounding box of the model point set.

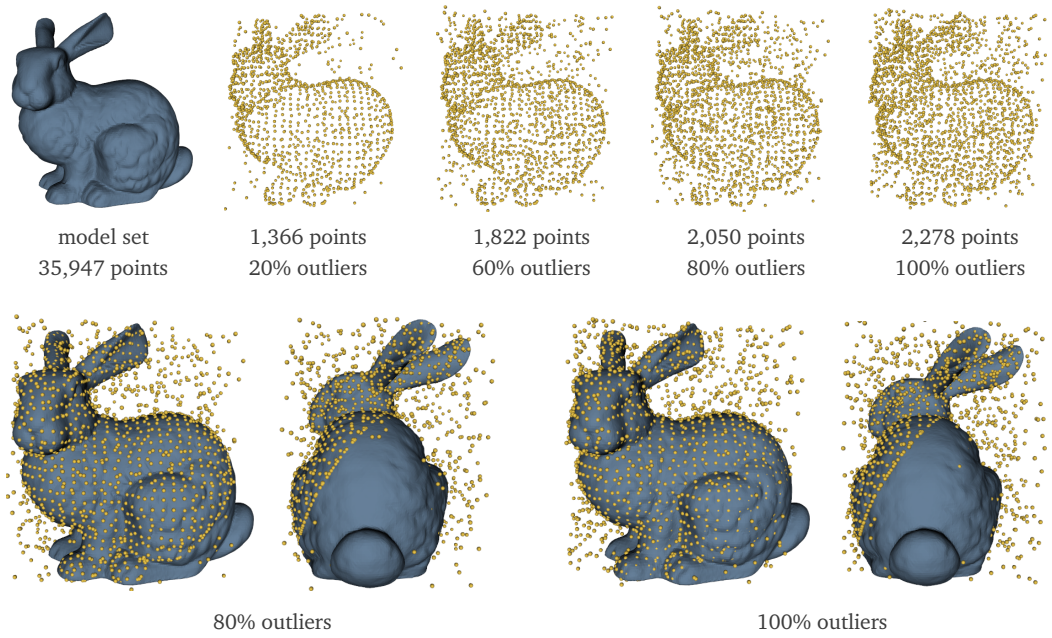


Figure 5.1: (Top row) The model set shown as a blue mesh. The outlier corrupted data sets rendered as yellow point clouds. (Bottom row) Typical registration results obtained with our algorithm based on the inverse distance kernel.

drawing points from a uniform distribution within the bounding box of the corresponding point set. We report the number of outliers as percentage of the original number of points and not as percentage of the points in the corrupted set. For example, 100% means that there are as many outliers as inliers. We did it so because the results in [34], which we use for comparison, are reported in this way.

We also measure the number of cost function evaluations and the computation time for varying cooling speed v (defined in (2.22)). We analyze the robustness of our method using two different kernels in the cost function. Furthermore, we report how two state-of-the-art registration approaches perform on the same point sets and compare the runtime of our algorithm with the one of a deterministic branch-and-bound method. In the following, we describe each test scenario in detail.

5.1.1 Kernel Comparison

First, the success rate and the accuracy of our method are tested with two different kernels, namely, the inverse distance kernel (2.12) used in our cost function and the Huber kernel (2.7) used in [27]. The point sets involved in this test together with some typical registration results are shown in Figure 5.1. Even though the model is rendered as a mesh, only the vertices are used for the registration. Note that the data sets are incomplete and sparser than the model.

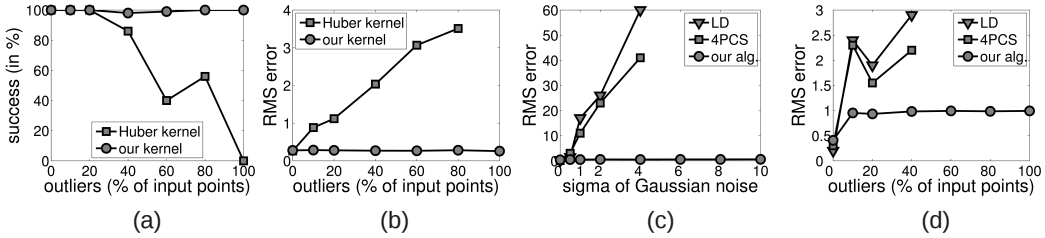


Figure 5.2: (a) The success rate as a function of the percentage of outliers in the data sets shown in Figure 5.1. (b) The RMS error between the ground truth pose and the estimated pose as a function of the percentage of outliers. (c), (d) Comparison of our method with two state-of-the-art rigid registration algorithms.

Furthermore, outliers are added only to the data sets. This case occurs in real world scenarios in which one has a complete (relatively clean) model of an object and wants to align it to a low quality data set which only partially represents the object (due to occlusion and scene clutter). Observe the high quality of the alignment shown in the bottom row in Figure 5.1 even in the presence of a significant amount of outliers. A registration trial took between 9 and 17 seconds depending on the number of points (which are provided in Figure 5.1).

As already mentioned in Section 2.2.1, we expect a registration method which minimizes a cost function based on the unbounded Huber kernel to have difficulties with outlier corrupted data sets. This is confirmed by the results of this test case which are summarized in the Figures 5.2(a) and 5.2(b).

In Figure 5.2(a), the success rate of our registration algorithm is shown when using the inverse distance kernel (2.12) (our kernel) and the Huber kernel (2.7). Note that our kernel leads to an almost constant success rate of 100% even in the presence of a very large amount of outliers whereas at the level of 100% outliers the registration completely fails if the Huber kernel is used.

In Figure 5.2(b), the RMS error as a function of the percentage of outliers is shown for the inverse distance kernel and the Huber kernel. Only the successful trials are used for computing the RMS error. Note that our kernel leads to much more precise registration results which are almost independent of the amount of outliers.

5.1.2 Comparison with State-of-the-Art

Alignment Precision In the second test case, we align two partially overlapping parts of the Coati model under varying conditions. This time, noise and outliers are added to both the model and the data set. This situation occurs in practice when building a complete object model out of multiple partially overlapping scans. We compare our results with the ones reported in [34] which are obtained with the robust 4PCS algorithm and a state-of-the-art local descriptor-based approach (LD). A combination of spin-images and integral invariants are used

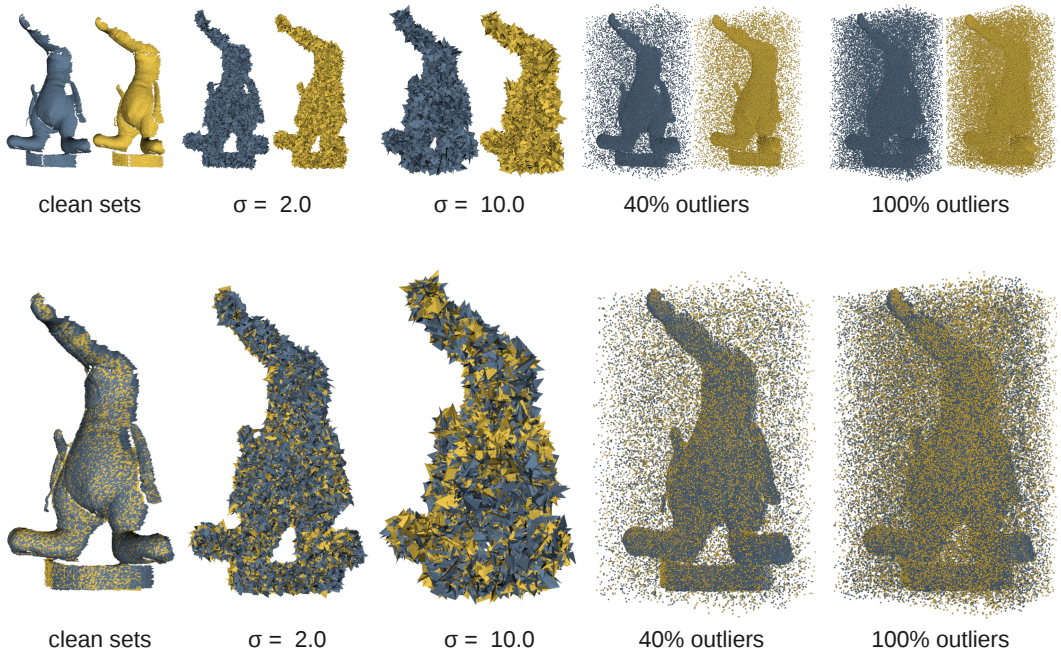


Figure 5.3: Registration of partially overlapping noisy and outlier corrupted point sets. The models are shown in blue whereas the data sets in yellow. (Top row) Partial scans of the Coati model degraded by noise or outliers. (Bottom row) Typical registration results computed with our algorithm.

as local descriptors. We perform the tests on the point sets used in [34]. This allows for a precise comparison without the need of re-implementing either of the two algorithms.

The model and data sets together with typical registration results obtained with our method are shown in Figure 5.3. σ of Gaussian noise or the amount of outliers as percentage of the original number of input points is indicated below each figure. One σ unit equals 1% of the bounding box diagonal length of the corresponding point set. $5,000$ randomly sampled points from each point set are used for the registration. The results are obtained without any noise or outlier removal, ICP refinement [24] or assumptions about the initial pose of the point sets. Each registration trial took about 33 seconds.

In the Figures 5.2(c) and (d), we plot our results together with the ones reported in [34]. Note that the graphs corresponding to 4PCS and LD end by $\sigma = 4.0$ and 40% outliers. This is because the authors of [34] did not test their methods on point sets with more noise or outliers whereas we did. Observe that our algorithm is quite insensitive to noise and outliers and it outperforms both other methods. The alignment is measured using the RMS error between the model and the data after registration. One unit corresponds to 1% of the bounding box diagonal length of the model set.

# points	40	60	80	100	150	200
time (sec) b&b [57]	10.2	48.6	107.5	156.2	722.1	1103.2
time (sec) our alg.	2.0	2.1	2.3	2.4	2.8	3.1

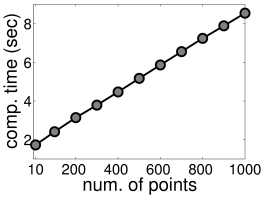


Figure 5.4: (Left) Computation time comparison of our algorithm and the box-and-ball (b&b) registration algorithm of Li and Hartley [57]. (Right) Runtime of our algorithm as a function of the number of input points.

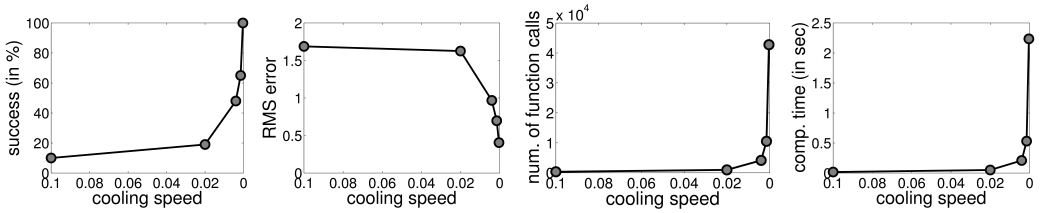


Figure 5.5: From left to right: success rate, RMS error, number of cost function evaluations and computation time of our registration algorithm as a function of the cooling speed v.

Processing Time In the third test scenario, we compare the computation time of our algorithm with the one of the registration method of Li and Hartley [57] which is based on global deterministic Lipschitz optimization theory. We run these tests on a slower PC with a 2.2 GHz CPU in order to make a fair comparison with [57]. The results are summarized on the left in Figure 5.4. For a point set of 200 points, our algorithm outperforms [57] by three orders of magnitude.

The right side in Figure 5.4 shows the dependence of the computation time on the number of input points. The figure clearly indicates a linear time complexity. Model and data used in this test case are down-sampled copies of the outlier-free version of the data set shown in the top row of Figure 5.1. In all tests, our method achieved a success rate of 100%.

5.1.3 Dependence on the Cooling Speed

Next, we measure the performance of our method for varying cooling speed v defined in (2.22). We report the results in Figure 5.5. Model and data used in this test consist of 100 points randomly sampled from the outlier-free version of the data set shown in the top row of Figure 5.1. One RMS error unit equals 1% of the bounding box diagonal length of the point set.

If the cooling speed is too high (> 0.0025), the optimization algorithm often gets stuck in a local minimum of the cost function landscape which explains the low success rate and the high RMS error in the first two graphs in Figure 5.5. Clearly, lower cooling speed means more iterations which is the reason for the increase of the number of cost function evaluations and

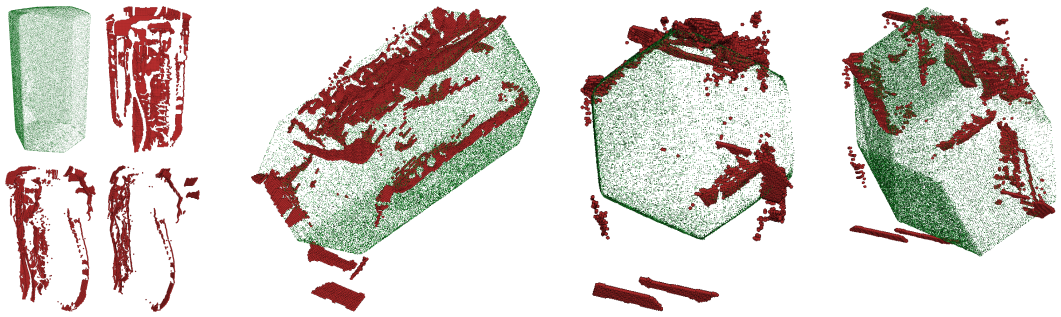


Figure 5.6: (Left) The complete model of a box (shown in green; 236,089 points) and three views of the very low quality data set (shown in red; 9,623 points). (Right) Our method robustly achieved the right alignment in 10 out of 10 trials.

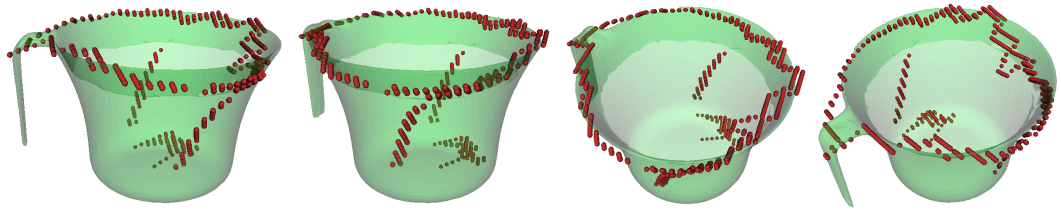


Figure 5.7: Registration result in the case of a noisy and very sparsely reconstructed data set (shown by the red "curve") and a complete noise-free model (transparent green mesh).

processing time (last two graphs in Figure 5.5). Our algorithm achieves a success rate of 100% and an RMS error below 0.5 for less than 2.5 seconds (for point sets consisting of 100 points).

5.1.4 Further Examples

Finally, we demonstrate the ability of our method to deal with partially overlapping and very sparsely sampled point sets corrupted by noise and outliers which are not artificially generated but originate in scan device imprecision.

In Figure 5.6, we show that our method successfully computes the right registration even in the case of an extremely degraded data set which represents only a subset of the model. The data set was obtained with a correlation-based stereo algorithm under poor lighting conditions. 5,000 out of the 9,623 data points were randomly sampled and used for the registration. Each registration trial took about 30 seconds. The high amount of noise and outliers which almost completely destroy the shape of the object makes this a very challenging example.

Figure 5.7 illustrates the stability of our algorithm when dealing with very sparsely sampled data sets. Note that in this case the state-of-the-art integral volume descriptor (used in [29]) will fail since the curve which represents the data set does not enclose a volume in $\mathbb{R}^3$. Local

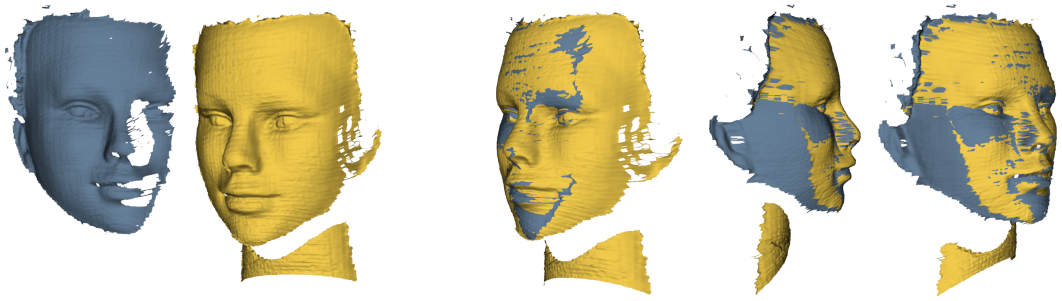


Figure 5.8: Registration of noisy point sets with low overlap.

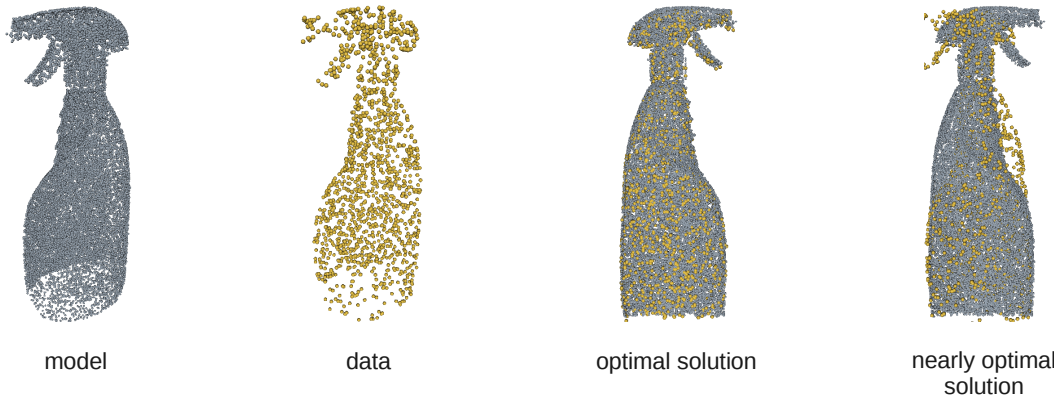
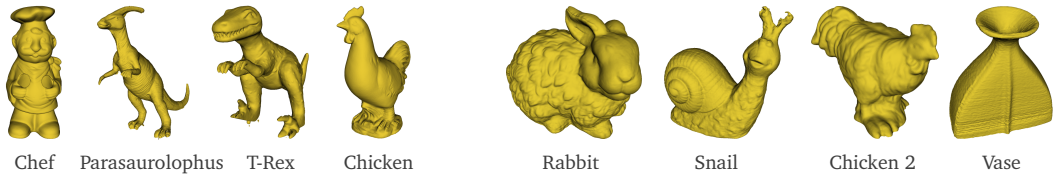


Figure 5.9: Point sets leading to a cost function which has two almost equally low minima.

descriptors which use surface normals like, e.g., spin images [28], will fail as well since in general the normal of a curve which lies on a surface does not match the surface normal.

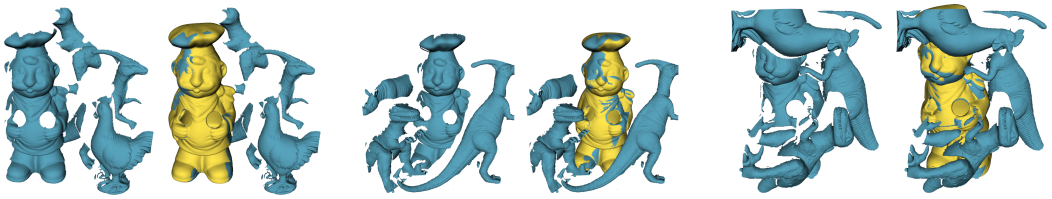
Figure 5.8 shows a typical registration result for partially overlapping points sets. Although rendered as meshes only points are used for the registration. Note that the input scans, shown on the left, represent different parts of the face and the model set (the blue one) contains no parts of the neck.

Note that our registration method can lead to incorrect results for a class of shapes for which several almost equally good alignments exist and the registration ambiguity can be dissolved by small scale features only. An example of such a shape is a large cup with a small handle. In this case, the corresponding point sets lead to a cost function with several local minima which are almost as "good" as the global one. Figure 5.9 provides an example for such a difficult case. The nearly optimal solution differs from the optimal one by a rotation of the data set by 180° about the axis which corresponds to the upright orientation of the bottle.



(a) (b)

Figure 5.10: The models used in the tests of the 3d object recognition algorithm. (a) Models provided by [71]. (b) Our own scans made with a low-cost light intersection-based scanning device.



(a) 62.3% object occlusion (b) 70.4% object occlusion (c) 86.2% object occlusion

Figure 5.11: The test scenes used for the Chef model recognition. The level of occlusion for the Chef is indicated for each scene. On the left of each subfigure, the input scene is shown as a blue mesh, whereas on the right, the recognized Chef model is placed at the location computed by our algorithm and rendered as a yellow mesh.

5.2 3D Object Recognition

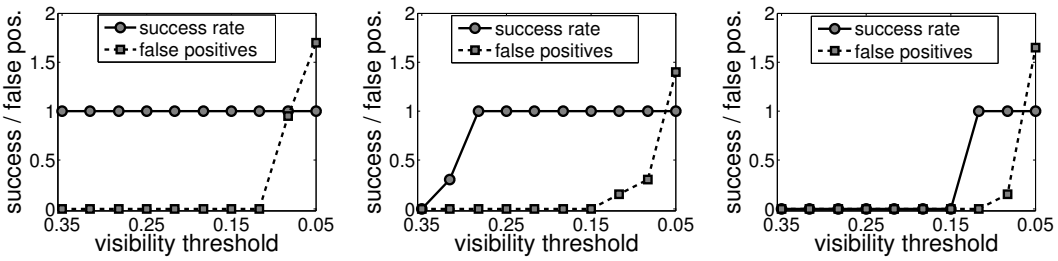
In this section, we experimentally validate the 3D object recognition algorithm proposed in Chapter 3. The object models involved in the tests are shown in Figure 5.10. All scenes used in the test cases were digitized with a Minolta VIVID 910 scanner [131] and provided by [71]. An exception are the scenes shown in the Figures 3.1 and 5.15 which are our own scans made with a low-cost light intersection-based DAVID laser scanner [132]. Examples and more information about the scenes will be given in the following subsections.

In Appendix A, a further experimental validation of the object recognition method within a real-world robotic object grasping scenario is provided.

5.2.1 Recognition of a Single Object in Occluded Scenes

In the first test scenario, we examined how the success rate and the false positives rate of the recognition algorithm depend on the most important parameter, namely, the visibility threshold (introduced in Section 3.3.2) and the actual object occlusion in a scene. According to [28], the occlusion of an object model is given by

$$occlusion = 1 - \frac{\text{visible model surface area}}{\text{total model surface area}}. \tag{5.1}$$



(a) 62.3% object occlusion (b) 70.4% object occlusion (c) 86.2% object occlusion

Figure 5.12: The success rate and the mean number of false positives as functions of the visibility threshold for three different scenes each one containing the Chef model at an occlusion level of (a) 62.3%, (b) 70.4% and (c) 86.2%.

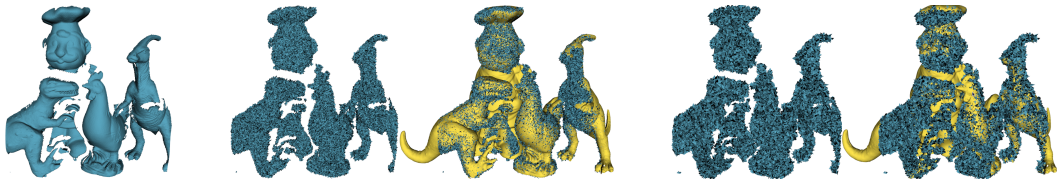
The aim of this test was to establish a value for the visibility threshold which, on the one hand, results in a high success rate even in highly occluded scenes, and on the other hand leads to as few as possible false positives. In this experiment, only the model of the Chef (Figure 5.10(a)) was used for recognition in three different scenes each one containing a total of three or four objects. The Chef was present in each scene at different locations and at different levels of occlusion (self-occlusion as well as occlusion caused by the other objects). The three test scenes together with typical recognition results are shown in Figure 5.11.

Since the recognition algorithm is a probabilistic one, we ran 100 trials on each scene and computed the recognition (success) rate and the mean number of false positives in the following way. We visually inspected the result of each trial. If object **A** (in this case only the Chef) was recognized k times ($0 \leq k \leq 100$), then the recognition rate for **A** is $k/100$. The mean number of false positives is $(k_1 + ... + k_{100})/100$, where k_i is the number of false alarms in the i-th trial.

The results of the test are summarized in Figure 5.12. As expected, the visibility threshold had to fall below a certain value, namely, $1 - occlusion$ in order to achieve a positive recognition rate. More importantly, the plots suggest that the number of false positives practically does not depend on the actual level of occlusion but mainly on the visibility threshold: in all three cases it starts to grow when the visibility threshold falls below 0.15. In summary, it can be said that the method achieved a recognition rate of 1.0 in highly occluded scenes (up to 85% occlusion) at the cost of no false positives. In order to handle more occlusion the visibility threshold had to fall below 0.15 which gave rise to some false positives.

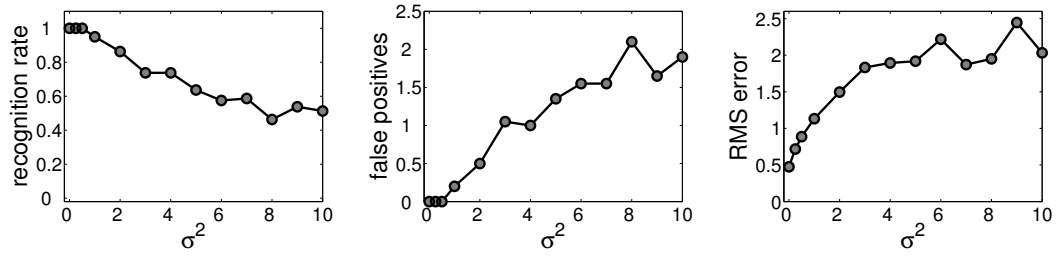
5.2.2 Recognition of Multiple Objects in Noisy Scenes

In this scenario, we tested the algorithm under varying noisy conditions. The four models involved are shown in Figure 5.10(a) and the noise-free scene is shown in Figure 5.13(a). We degraded the noise-free scene with zero-mean Gaussian noise with different variance values σ^2.



(a) Noise-free (b) σ^2=0.25 (c) σ^2=1.0

Figure 5.13: (a) Noise-free scene. (b), (c) Recognition results for data sets degraded by zero-mean Gaussian noise for different variance σ^2 (given as percentage of the bounding box diagonal length of the scene). The left side of each subfigure shows the scene, whereas the right side shows the scene plus the recognized models at the estimated locations.



(a) (b) (c)

Figure 5.14: (a) Recognition rate, (b) mean number of false positives and (c) RMS error as functions of the variance σ^2 of Gaussian noise. Note that the RMS error is computed for the successful trials only. Both σ^2 and the RMS error are given as percentage of the bounding box diagonal length of the scene.

Again, 100 recognition trials on each noisy scene were performed and the recognition rate, the mean number of false positives and the RMS error (RMSe) were computed as functions of σ^2.

For two point sets $\mathbf{P}$ and $\mathbf{Q}$ and a transform T the RMS error measures how close each point $\mathbf{q}_i \in \mathbf{Q}$ comes to its corresponding point $\mathbf{p}_i \in \mathbf{P}$ after transforming $\mathbf{Q}$ by T [29]. The smaller the error the closer the alignment between the point sets. More formally,

$$\mathrm{RMSe}(\mathbf{P}, \mathbf{Q}, T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - T(\mathbf{q}_i)\|^2}, \tag{5.2}$$

with N being the number of points. Since the ground truth location of each model in the test scene is known, the RMS error of the rigid transform computed by our method was easily calculated[1]. Note that we did *not* compute the RMS error for the transformed model and the scene but for the transformed model and the same model placed at the ground truth location. Figure 5.13(b) and (c) exemplary show typical recognition results for two of the twelve noisy scenes. The results of the tests are reported in Figure 5.14.

[1]The ground truth rigid transform for the models is available on `http://www.csse.uwa.edu.au/~ajmal/`

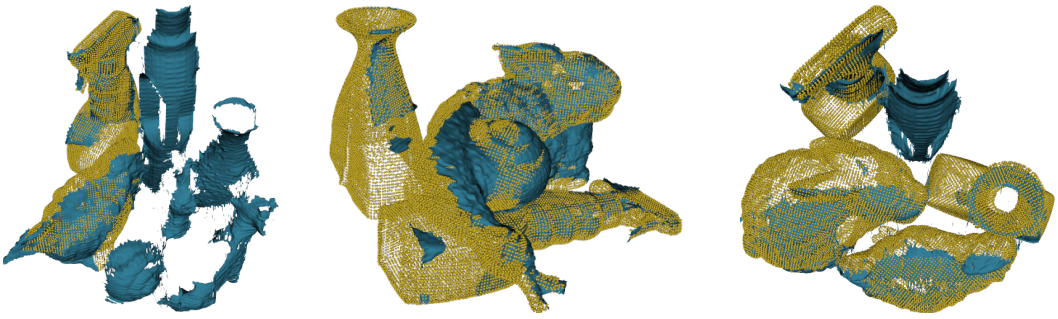


Figure 5.15: Typical recognition results obtained with our method for three test scenes. The scenes are shown as blue meshes and the recognized model instances are rendered as yellow point clouds and superimposed over the meshes. Some of the scenes contain unknown objects (the left and the right one). Note that the scene reconstruction contains only small portions of the objects.

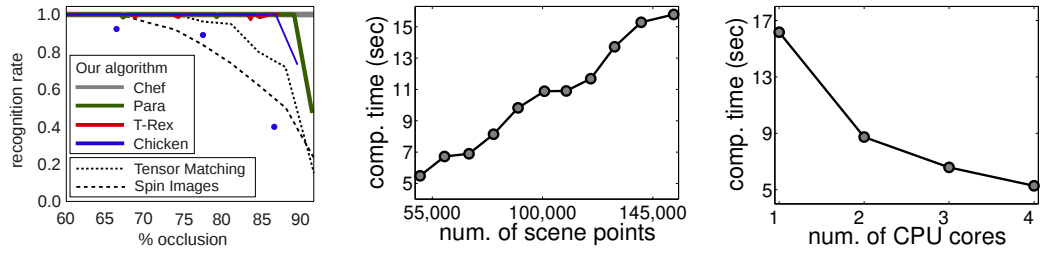
Next, we demonstrate the ability of our method to deal with data sets corrupted by noise which is not artificially generated but originates in scan device imprecision. Note that the scenes used in [28] and [71] are dense and have a relatively good quality. We use a low-cost light section based scanner which gives sparser and noisier data sets. The models used in this test scenario are shown in Figure 5.10(b). Typical recognition results of our method are shown in Figure 3.1 and Figure 5.15.

5.2.3 Comparison

Next, we compared the recognition rate of our algorithm with the spin images [28] and the tensor matching [71] approaches on the same 50 data sets used in [71]. This made a direct comparison possible without the need of re-implementing either of the two algorithms. The models of the four toys involved in the tests are shown in Figure 5.10(a). The toys (not necessarily all four of them) are present in the scenes in different positions and orientations. Since each scene was digitized with a laser range finder from a single viewpoint the back parts of the objects were not visible. Furthermore, the toys were usually placed such that some of them occluded others which made the visible object parts even smaller. Four (out of the 50) test scenes are shown in Figure 5.11 and Figure 5.13(a). Again, we ran 100 recognition trials on each scene and computed the recognition rate for each object in the way described in Section 5.2.1. Since the occlusion of every object in the test scenes was known we report the recognition rate for each object as a function of its occlusion.

The result of the comparison is summarized in Figure 5.16(a). The recognition rate of our algorithm for each object as a function of its occlusion is indicated by the continuous lines. The dashed lines give the recognition rate of the spin images and the tensor matching approaches on the same scenes as reported in [71]. Our algorithm outperforms both other methods. Note

`recognition.html`



(a) (b) (c)

Figure 5.16: (a) Comparison with spin images [28] and tensor matching [71]. (b) Computation time as a function of the number of scene points for the simultaneous recognition of seven models. (c) Runtime as a function of the number of used CPU cores for the recognition of seven models in a scene consisting of around 60,000 points.

that the chef was recognized in all trials, even in the case of occlusion over 91%. The blue dots represent the recognition rate in the three chicken test scenes in which our method performed worse than the other algorithms. This was due to the fact that in these scenes only the chicken's back part was visible which contains strongly varying normals which made it difficult to compute a stable aligning transform.

Our method needed in average about 7.5 seconds for the recognition of the objects in each scene and sampled about 450 oriented point pairs per scene. For a comparison, 250 tensors, respectively, 4000 spin images per scene were used in the experiments performed in [71].

5.2.4 Runtime

We experimentally validated the linear time complexity of our algorithm in the number of scene points. Eleven different data sets were involved in this test case—a subset from the scenes used in the comparison test case (Section 5.2.3). Note that we did *not* take a single data set and down/up-sampled it to get the desired number of points. Instead, we chose eleven *different* scenes with varying scene extent, number of points and number of objects. This suggests that the results will hold for arbitrary scenes. We report the results of this test in Figure 5.16(b). The plot indicates a linear complexity.

Note that the iterations of the main loop of the recognition algorithm (lines 4 to 20 of Algorithm 1) can be executed independently of each other which makes it possible to run them in parallel. This is a very important issue since parallel computing has become the dominant paradigm in computing architectures, mainly in the form of multicore processors [133]. In Figure 5.16(c), we report the processing time as a function of the used CPU cores. Note that the parallel execution on four cores runs more than three times faster than on a single core. This indicates a great potential for further speed-up when more CPU cores become available.

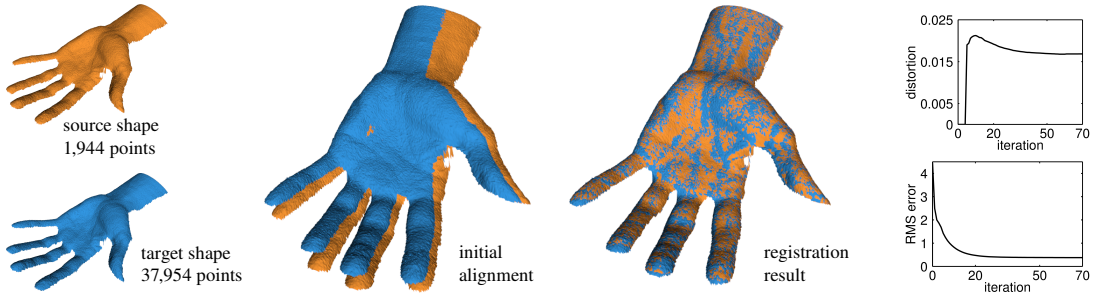


Figure 5.17: Registering two range images representing the front part of the same hand in two different poses. The data sets were obtained with a 3D geometry scanner [88] and are publicly available on the authors webpage.

5.3 Deformable 3D Shape Registration

In this section, the deformable registration algorithm, proposed in Chapter 4, is experimentally validated on a variety of real data sets.

In order to evaluate the algorithm quantitatively, we measure the source shape distortion and the RMS error between source and target and plot them versus the iteration number. First, we define the vertex distortion as

$$\mathrm{dis}(\mathbf{v}_i) = \frac{1}{|\mathbf{N}_i^s|} \sum_{\mathbf{v}_j \in \mathbf{N}_i^s} \frac{|\|\mathbf{x}_i - \mathbf{x}_j\| - l_{ij}|}{l_{ij}}, \tag{5.3}$$

where $\mathbf{N}_i^s$ is the set of shape vertex neighbors of $\mathbf{v}_i$ (see Definition 3 in Section 4.2), $\mathbf{x}_j$ is the current position of vertex $\mathbf{v}_j$ and l_{ij} is the distance between $\mathbf{v}_i$ and $\mathbf{v}_j$ in the initial, undeformed shape state. If we consider the source shape as a mass-spring system in which each two neighboring vertices are connected by a spring, each term in (5.3) gives the strain of the spring $(\mathbf{x}_i, \mathbf{x}_j)$ (see [134]). Using $\mathrm{dis}(\mathbf{v}_i)$, we define the distortion of the source shape $\mathcal{S}$ as

$$\mathrm{dis}(\mathcal{S}) = \frac{1}{m} \sum_{\mathbf{v}_i \in \mathcal{S}} \mathrm{dis}(\mathbf{v}_i), \tag{5.4}$$

with m being the number of source shape vertices. The RMS error between the source $\mathcal{S}$ and the target $\mathcal{S}'$ is given by

$$\mathrm{RMSe}(\mathcal{S}, \mathcal{S}') = \sqrt{\frac{1}{m} \sum_{\mathbf{v}_i \in \mathcal{S}} \|\mathbf{x}_i - \mathbf{c}_p(\mathbf{x}_i)\|^2}, \tag{5.5}$$

where m is the number of source shape vertices and $\mathbf{c}_p(\mathbf{x}_i)$ is the target point closest to $\mathbf{v}_i \in \mathcal{S}$ (see (4.36) in Section 4.5.1).

5.3.1 Range Scan Pairs

First, we run our method on pairs of range scans representing the same object in different poses. Figures 5.17 to 5.21 show the data sets used in the test. The scans are shown as they were

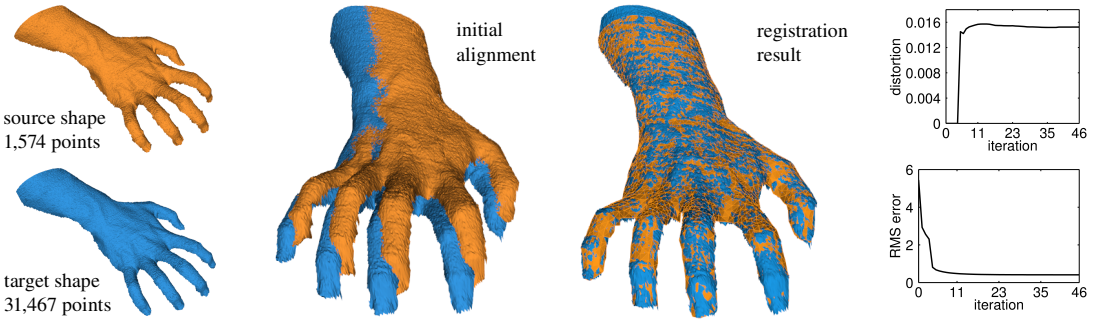


Figure 5.18: Range scans representing the back part of the same hand in two different poses. The poses differ not only by the local bending deformation of the fingers but also by a global translation. The data sets were obtained with a 3D geometry scanner [88] and used in [40].

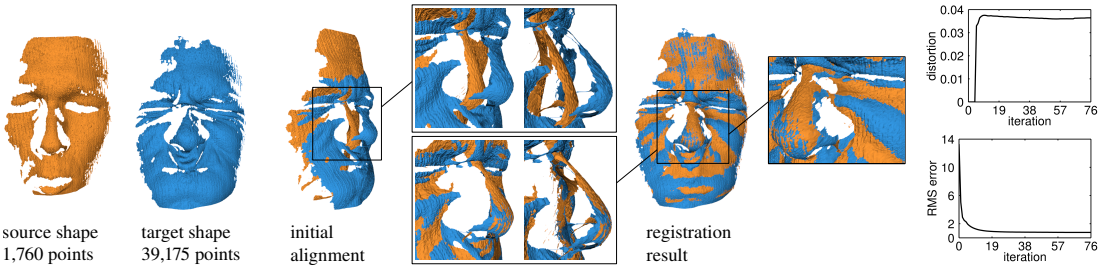


Figure 5.19: Registering two facial expressions. Note that the scans are noisy and incomplete. Our methods correctly aligns the shapes even in areas of low overlap.

captured by the scanning devices without any additional alignment. These configurations are used as starting point for our registration algorithm.

Figure 5.17 shows a range scan pair which is part of a sequence representing a slowly closing hand. The inter-frame displacement is small and mainly caused by the bending finders. The registration computed with our method together with the deformation measure and the RMS error are shown on the right side of the figure.

Figure 5.18 shows a further example of a closing hand. This time, there is a larger bending deformation plus an additional global translation. As is to be expected from the initial configuration of the scans the RMS error at the beginning of the registration is larger. Furthermore, since the fingers bend more than in the last example the amount of deformation required to register the scans is larger. This is confirmed by the plots on the right side of the figure.

Figure 5.19 demonstrates the ability of our algorithm to deal with incomplete data. Note that there are many holes in both scans caused by self-occlusion and scan device imperfection. Our method successfully registers the scans even in areas of low overlap as the magnified parts of the figure show.

Figure 5.20 shows registration of an articulated object, namely, a bending arm. Note that there is a significant deformation between the scans. This is a challenging example for feature-

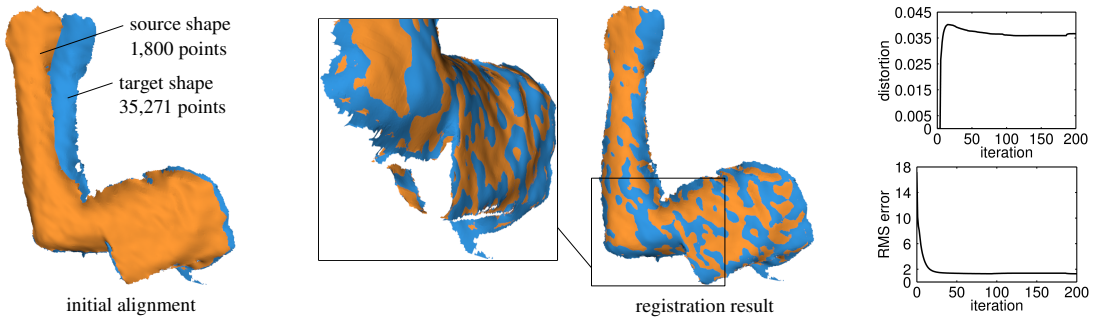


Figure 5.20: Registration of a bending arm.

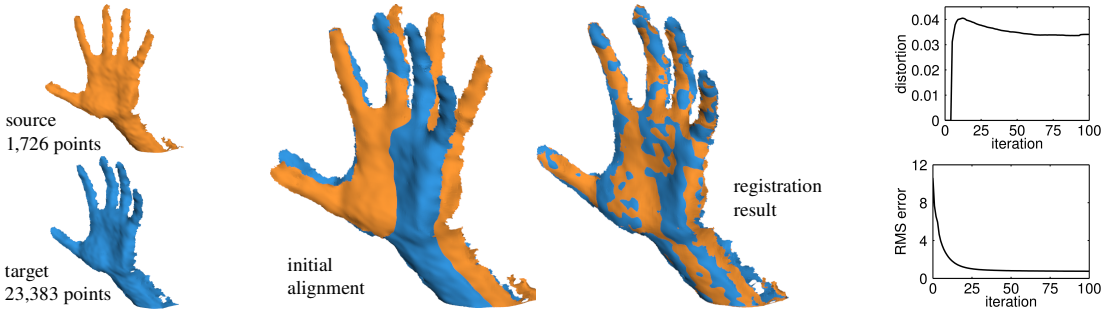


Figure 5.21: Registering a closing and rotating hand.

based methods since the scans are smooth and lack distinctive features. Our method successfully recovers the deformation as depicted in the figure.

In Figure 5.21, a further example of a moving hand is shown. Additionally to the local deformation caused by the closing fingers this example contains a significant global rotation.

5.3.2 Complete Source Model and an Incomplete Target Scan

In the tests performed so far, the source and target represented more or less the same part of the deforming object: the same side of a hand, face or an arm. Although there were some points on the one scan without a counterpart on the other (especially in the face example in Figure 5.19), an almost one-to-one correspondence between the shapes was established.

In this section, we present a couple of more involved test cases in which a complete model of an object (the source) has to be registered to a deformed, incomplete scan (the target). This is challenging since large parts of the model have no corresponding parts on the scan. To get more stable results and less shape distortion, we tetrahedralized the models prior to the registration. This is possible since they are closed, i.e., water-tight meshes. We performed the tetrahedralization with TetGen[1].

[1] TetGen is a quality tetrahedral mesh generator and a 3D Delaunay triangulator which is freely available at `http://tetgen.org`.

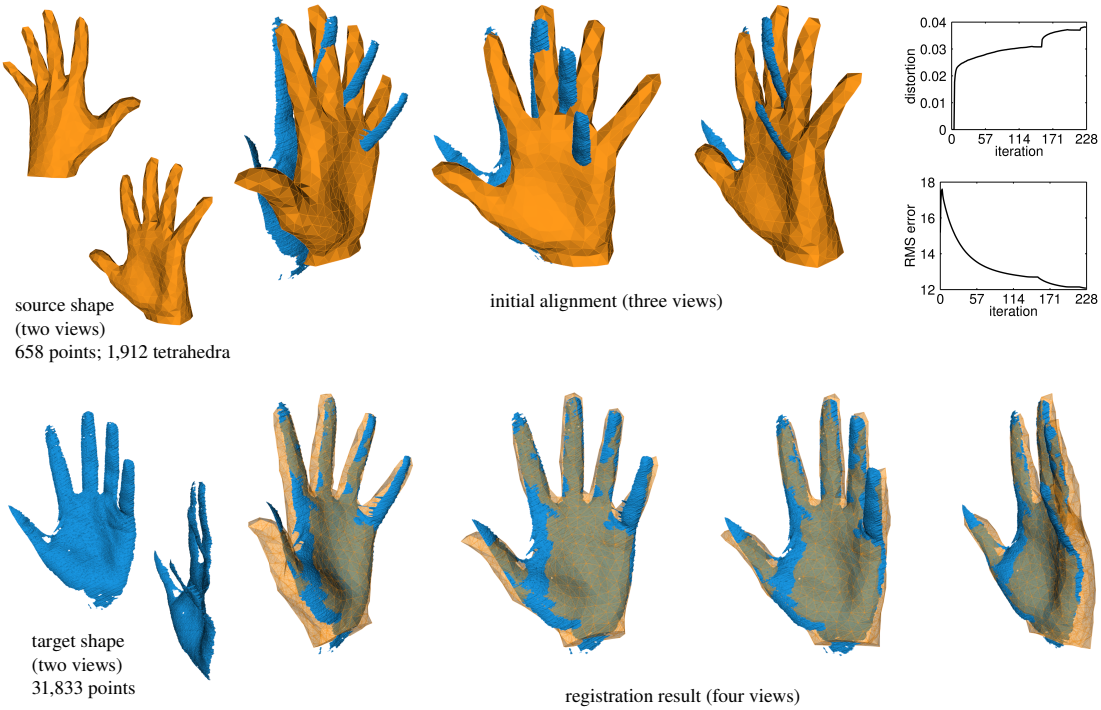


Figure 5.22: Registering a complete geometric model of a hand to a partial range scan. In the last row, the registered model is shown transparent in order to better see the underlying target.

The first example, shown in Figure 5.22, demonstrates the registration of a complete model of a hand to a partial scan of another hand in a different pose. Note that source and target do not stem from the same individual. Furthermore, the front side of the hand is not fully covered by the target scan and the back side is completely missing. This means that more than half of the source points have no counterparts on the target. Still, our algorithm successfully registered the data sets as can be seen in Figure 5.22. Note the two "jumps" in the shape distortion and RMS error plots at iterations 153 and 218 and note that the curves become quite flat just before that. This is due to the fact that the algorithm converged with the current set of weights (see Equation (4.38) on page 57) which resulted in recomputing the weights and continuing the minimization. This usually leads to higher weights for the target positions due to better agreement between the source and target normals (see Equation (4.40) on page 58) which explains the sudden drop of the RMS error and the increase of the shape distortion.

In the next test case, a model of an arm is registered to a partial scan of the same arm in a different pose (see Figure 5.23). Besides providing shape distortion and RMS error plots, we visualize the distortion directly on the registered model by mapping the vertex distortion (see Equation (5.3)) to colors (ranging from blue to red) and shading each mesh triangle according to the colors of its vertices. Note that the front part of the fist in the target scan is almost completely missing and the initial alignment of the shapes is quite imprecise. Nevertheless, our

	data set									
method	**Fig. 5.17**		**Fig. 5.18**		**Fig. 5.19**		**Fig. 5.20**		**Fig. 5.21**	
GMM+TPS	0.04	0.6	0.05	0.8	0.10	1.4	0.08	1.4	0.06	2.1
GMM+GRBF	0.03	0.8	0.03	0.9	0.07	2.4	0.05	2.2	0.06	2.5
SDA+TPS	0.09	0.7	0.09	0.7	0.11	1.2	0.08	1.3	0.08	0.9
CPD+GRBF	0.30	1.4	0.30	1.2	0.20	1.5	0.30	3.6	0.20	1.7
KC+TPS	0.04	0.6	0.05	0.8	0.10	1.3	0.08	1.4	0.06	1.8
KC+GRBF	0.03	0.8	0.03	0.9	0.09	2.2	0.06	2.2	0.07	2.4
our algorithm	0.02	0.4	0.02	0.4	0.04	0.8	0.04	1.3	0.03	0.8
	dis	**RMSe**	**dis**	**RMSe**	**dis**	**RMSe**	**dis**	**RMSe**	**dis**	**RMSe**

Table 5.2: Comparing the quality of the registration computed by our algorithm and six state-of-the-art approaches for the scans shown in Figures 5.17 to 5.21. The shape distortion (dis) is dimensionless and the RMS error (RMSe) is given in millimeters.

method achieves a good registration. As it can be seen from the distortion map in the last row in Figure 5.23, a distortion takes place mainly in those regions of the surface which experience the largest deformation in reality: the elbow and the wrist.

Note that in both examples, in contrast to the ones presented in the previous Section 5.3.1, the RMS error converges to a value significantly larger than 0. This is due to the fact that all source points are involved in the computation of the RMS error and not only those which have corresponding target points (i.e., the ones with positive target weights w_i). Since there are many model points with no counterparts, the RMS error remains large even for a good registration between source and target.

5.3.3 Comparison

In this section, we compare the performance of our method (both registration quality and runtime) with the performance of several state-of-the-art non-rigid registration algorithms: the softassign + deterministic annealing (SDA) approach [125], the kernel correlation-based (KC) method [49], the coherent point drift (CPD) algorithm [126] and the Gaussian mixture models-based (GMM) algorithm [127]. Note that the KC [49] and the GMM [127] methods can perform the registration using two different deformation models, namely, thin plate splines (TPS) and Gaussian radial basis functions (GRBF). This effectively results in six different registration methods which we will denote as follows: SDA+TPS is the abbreviation for [125], CPD+GRBF stands for [126], KC+TPS, KC+GRBF denote [49], and GMM+TPS, GMM+GRBF stand for [127], depending on which deformation model is employed.

We used an implementation of the above mentioned methods which can be downloaded from `http://gmmreg.googlecode.com` and ran them on the same hardware and on the same data sets as our algorithm. Figures 5.17 to 5.21 show the data sets used in the test. The scans are shown as they were captured by the scanning devices without any additional alignment. These configurations were used as starting point for all registration algorithms.

	data set				
method	**Fig. 5.17**	**Fig. 5.18**	**Fig. 5.19**	**Fig. 5.20**	**Fig. 5.21**
GMM+TPS	497	315	273	332	453
GMM+GRBF	361	244	237	269	267
SDA+TPS	1004	621	501	642	1033
CPD+GRBF	4389	2443	1368	2269	5952
KC+TPS	491	314	273	331	451
KC+GRBF	361	243	237	267	267
our algorithm	1.36	1.16	1.1	2.72	2.1

Table 5.3: Computation time (in seconds) taken by our algorithm and six state-of-the-art approaches for the registration of the scans shown in Figures 5.17 to 5.21.

The quality of the registration computed by the algorithms is compared using the source shape distortion measure (5.4) and the RMS error (5.5). Table 5.2 shows the results of the comparison. Our algorithm outperforms the others in all test cases. In all but one test, we achieved *both* a lower RMS error and a lower shape distortion. In only one case (Fig. 5.20), the SDA+TPS method led to an RMS error equal to ours, however, at the cost of a higher distortion.

The results of the runtime comparison are summarized in Table 5.3. Our algorithm clearly outperforms all six methods with the difference in processing time being up to three orders of magnitude.

5.3.4 Deformable Hand Tracking

Finally, we qualitatively tested our method on two range scan sequences representing a hand which undergoes an articulated motion. Although tracking is not the focus of this thesis, we demonstrate that our method can be applied to this challenging problem and that it achieves good results. Refer to [135, 136, 137] for specialized algorithms for articulated hand tracking. The range scans used in this section are provided by the Computer Graphics and Visualization lab at the Technische Universität Dresden[1].

The test scenario is the following. The source shape (the yellow mesh shown in Figure 5.22) is registered sequentially to the frames of the range scan sequence. The result for the current frame is used as initialization for the next one. Both sequences are single-view, meaning that the deforming hand is captured only from one viewpoint.

The first sequence consists of 166 frames. Figure 5.24 exemplary shows some of them and the corresponding registration results. Note that towards the end of the sequence, the scans get noisy and very incomplete. This is due to the fact that only a small portion of the hand is facing the scanning device. Nevertheless, our method successfully tracks the hand.

[1] `http://cgv.inf.tu-dresden.de/3dscanning/datasets.html`

The second test sequence is more challenging and consists of 430 frames. Besides undergoing an articulated motion, the hand rotates several times (by the wrist being twisted). Thus, the forehand which originally faces the scanner gets completely occluded and the backhand becomes visible. Figure 5.25, on pages 80, 81 and 82, exemplary shows some frames and the corresponding registration results. Note that our method successfully tracks the hand while it is rotating in the frames 136–161 and 228–249 even though little information is provided due to severe occlusion. Furthermore, as frames 291 and 303 show, our algorithm can deal with very sparse and noisy scans.

After frame 330, the method loses the fingers while the hand is closing. This has to do with the distance threshold d, used in Equation (4.39) on page 58, which was set to 6mm in this experiment. This implies that the pointwise source-to-target correspondences are rejected if the distance between the points is more than 6mm. Due to scan device limitations the fingers disappear during the hand closing (frames 330–334) and reappear as the hand is already made to a fist (frame 349). However, by that time, all source shape points on the fingers are further away than 6mm from the corresponding target points. Setting the distance threshold to a larger value does not solve the problem. Instead, it leads to strong shape distortions and self-intersections since source shape parts with no counterparts in the scan can be wrongly attracted to distant scan regions. Nevertheless, our method is able to correctly recover the global rotation of the hand in the frames 361–430 even though the scans are noisy and only partially represent the fist.

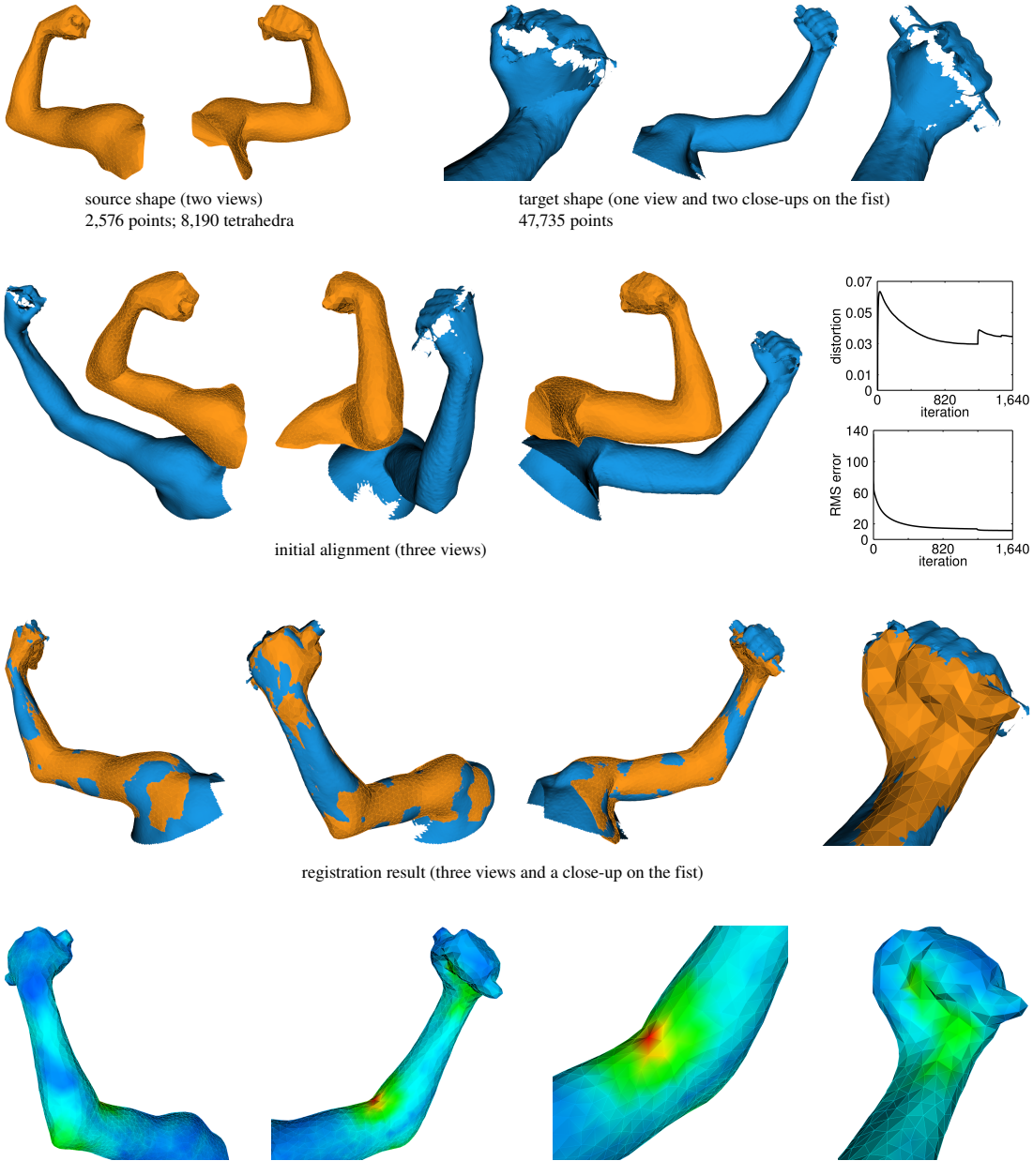


Figure 5.23: Registering a complete geometric model of an arm to a partial scan of the same arm in a significantly different pose.

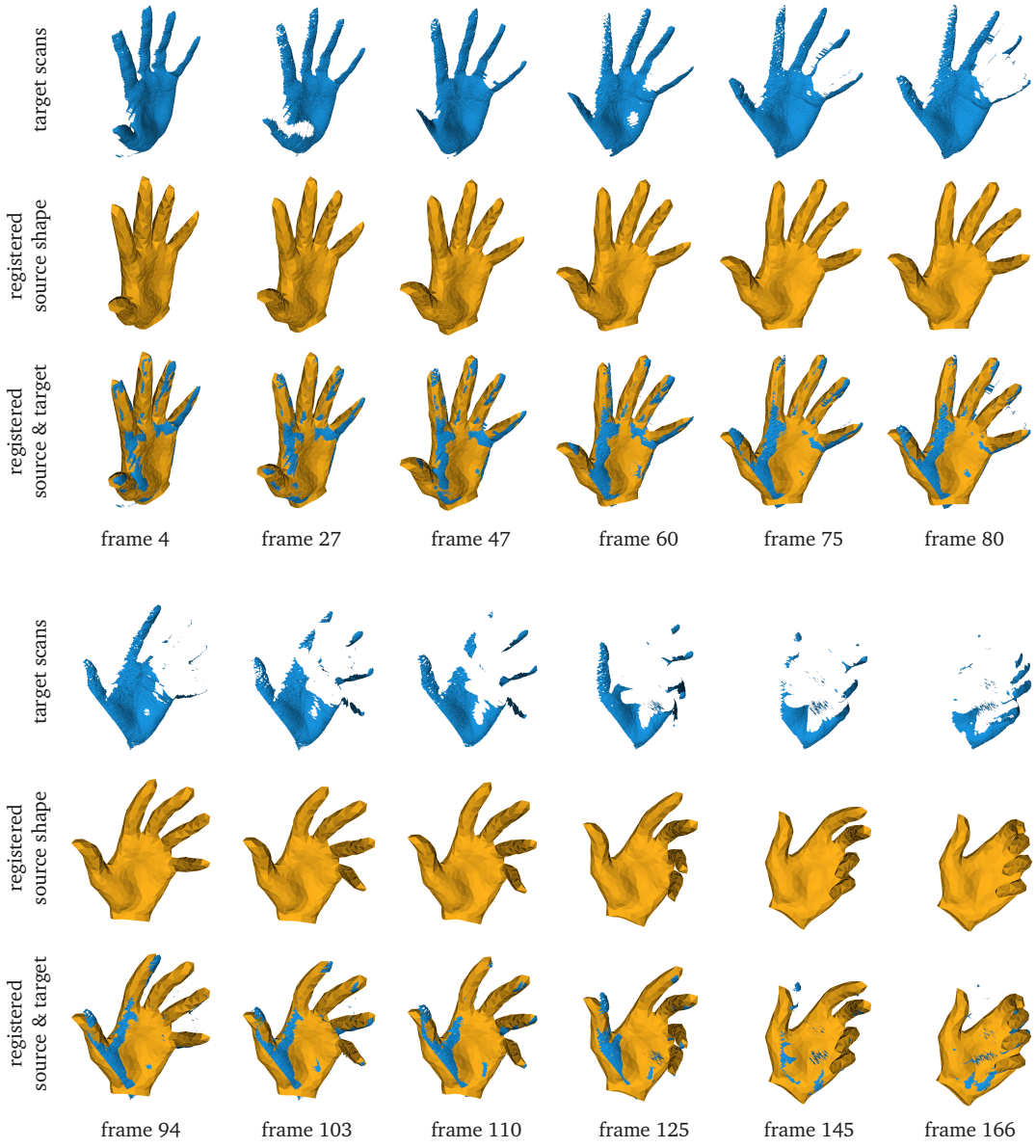


Figure 5.24: Sample frames of the tracking results obtained with our method. The target scans are shown in blue in the 1st and 4th row. In the 2nd and 5th row, the registered source shape is shown in yellow. The 3rd and 6th row show the registered source shape overimposed over the corresponding target scan.

Figure 5.25: Sample frames of the tracking results obtained with our method (continued on the next page). The target scans are shown in blue in the 1st and 4th row. In the 2nd and 5th row, the registered source shape is shown in yellow. The 3rd and 6th row show the registered source shape rendered over the corresponding target scan.

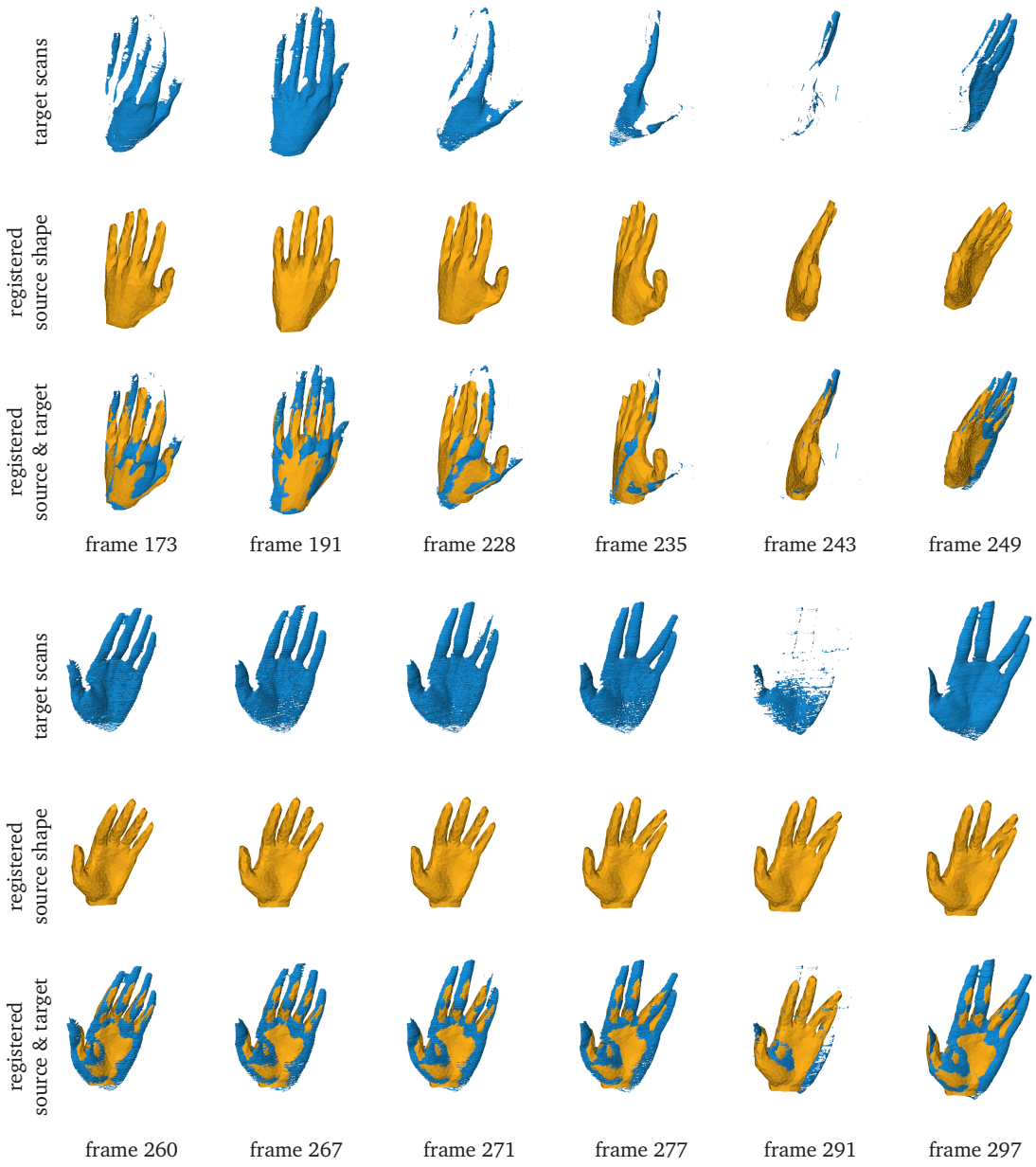


Figure 5.25 (continued): Sample frames of the tracking results obtained with our method (continued on the next page).

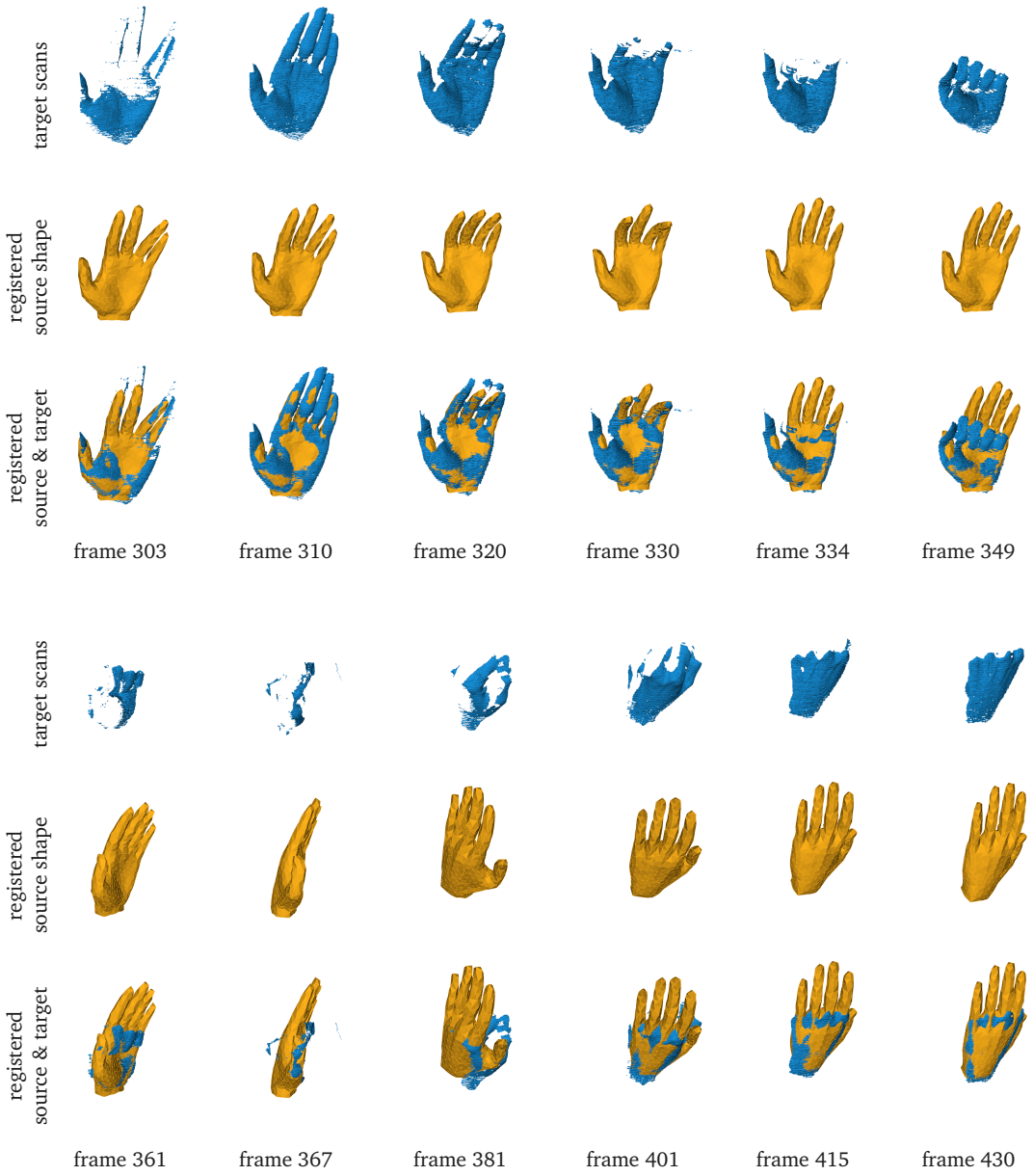


Figure 5.25 (continued): Sample frames of the tracking results obtained with our method.

Chapter 6

Conclusions and Future Work

In this thesis, we presented algorithms for rigid/deformable 3D shape registration and 3D object recognition and pose estimation. We took into account the requirements posed in the introductory Chapter 1, namely, robustness to noise and outliers and the ability to deal with partial views and holes in the data reconstruction. In the next two sections, we draw some conclusions and discuss possible directions for future investigation.

6.1 Conclusions

We introduced a new technique for pairwise rigid registration of point clouds. Our method is based on a noise robust and outlier resistant cost function which itself is based on an inverse distance kernel. One of the main messages is that a registration method which minimizes an objective function based on an unbounded kernel will be sensitive to noise and outliers in the point sets. This was fully validated by comparisons between our kernel and the Huber kernel (see Section 5.1).

A further property of the rigid registration algorithm is that it does not rely on any initial estimation of the globally optimal rigid transform. This was achieved by employing a new stochastic algorithm for global optimization. In order to minimize efficiently over complex shaped search spaces, like the space of rotations, we generalized the BSP trees and introduced a new technique for hierarchical rotation space decomposition. Furthermore, we derived a new procedure for uniform point sampling from spherical boxes.

Tests on a variety of point sets showed that the proposed method is insensitive to noise and outliers and can cope very well with sparsely sampled and incomplete data sets. Comparisons showed that our algorithm is by three orders of magnitude faster than a deterministic branch-and-bound method. Furthermore, it outperformed a recently proposed generate-and-test approach and a state-of-the-art local descriptor-based method in terms of accuracy and robustness.

The presented rigid registration algorithm is based on the assumption that both input point clouds are (partially) representing one and the same object. However, in some situations, like sorting grocery items by a robot (see Appendix A), we are interested in simultaneously registering several object models to a scene which contains data from multiple objects plus background clutter. This led us to the problem of 3D object recognition and pose estimation. We presented a solution able to robustly recognize and localize objects in partially reconstructed and unsegmented scenes.

The algorithm is based on a robust geometric descriptor, a hashing technique and an efficient, localized RANSAC-like sampling strategy. We provided a theoretical complexity analysis and derived a formula for computing the number of iterations required to recognize the objects with a predefined success probability. The result of the complexity analysis, namely a linear time dependency on the number of scene points, was experimentally validated. Tests on real range data confirmed that our method performs well on complex scenes in which only small parts of the objects are visible. In a direct comparison with the spin images [28] and the tensor matching [71] approaches, our method performed better in terms of recognition rate.

In Appendix A, a further experimental validation with the DLR Lightweight-Robot III [138] showed how well this new method can be exploited for grasping in unstructured and cluttered environments. The presented solution is capable of quickly recognizing and robustly grasping known objects from an unsorted pile of different everyday items. This is extremely useful for typical service robotics or industrial co-worker tasks.

Furthermore, we dropped the assumption of rigidity and developed a deformable 3D shape registration method. We introduced a unifying framework which allowed to treat deformation-based shape modeling together with deformable shape registration. We focused on modeling as-rigid-as-possible shape deformations which can be optionally augmented with local scale. In contrast to many recent methods, our approach is not formulated within a general-purpose optimization framework. The minimization of high-dimensional, non-linear cost functions is computationally very demanding since it involves the repeated solution of large linear systems. Instead, we rely on a simple numerical minimization scheme tailored to the problem at hand. The proposed solution proved to be very efficient since an experimental comparison to six state-of-the-art approaches showed that our algorithm outperforms them in terms of both registration quality and processing time. Furthermore, we experimentally validated our method on a variety of real range scans and demonstrated that it performs well on noisy and incomplete data.

Appendix B showed an application of our deformable registration method to the problem of knowledge transfer between geometric models. More precisely, we demonstrated how the proposed method can be used to map stable grasping points from one object model to another. This can be generalized and used for an automatic transfer of many different kinds of data associated with a geometric model, e.g., texture coordinates, friction parameters, pressure values and many more.

6.2 Future Work

This thesis offers several directions for future research. In the following few paragraphs, we propose some possible methods specific improvements.

Stochastic Optimization We plan to further investigate the proposed stochastic optimization method from a theoretical point of view. More specifically, we will prove the asymptotic convergence of the method, i.e., prove that, with probability 1, the sequence

$$\{\mathbf{f}(\mathbf{x}_k)\}_{k\in\mathbb{N}},$$

where $\mathbf{f}$ is the cost function and $\mathbf{x}_k$ is a candidate point in the space of rigid transforms, converges to the minimum of $\mathbf{f}$ for $k \to \infty$ [139]. Besides this, the rate of convergence is also of interest. A further issue is to determine, for a given function $\mathbf{f}$ and a probability p, the parameter set for which the algorithm will converge to a global minimum of $\mathbf{f}$ with a probability equal or higher that p.

3D Object Recognition There is still room for improvement of the proposed 3D object recognition method. First, we would like to further reduce the number of false positives by modifying the acceptance function introduced in Section 3.3.2. In its current version, the visibility term of the function just counts the number of transformed model points which fall within a certain ϵ-band of the scene and divides it by the total number of model points. If the result exceeds the visibility threshold and the transformed object model does not occlude to much of the scene, the hypothesis is accepted. Essentially, this is only a 0th order approximation of the scene by the model since only points are taken into account. A first order approximation can be achieved by taking the normal agreement into account. A second order approximation would require the model and scene to have the same curvature at the contact points. In this way, a more precise alignment will be needed in order for a hypothesis to be accepted which will reduce the number of false positives. On the other hand, curvature estimation is challenging in presence of noise in the input data. This is the reason why we think that a first order contact would be a good tradeoff between shape approximation and robustness to noise.

A second improvement concerns the processing speed of the algorithm. So far, each hypothesis generated by aligning an oriented point pair from the hashtable with one sampled from the scene is tested by transforming the model points and evaluating the acceptance function. This takes around 90% of the whole processing time. Instead, we plan to discretize the space of rigid transform by dividing it in a finite number of bins. Thus, a hypothesis will be evaluated if and only if the bin it ends up in is free. This will avoid evaluating similar hypotheses which should significantly reduce the processing time.

Deformable 3D Shape Registration The proposed deformable registration method is universal in the sense that it can be applied to arbitrary deformable objects, like arms, hands, faces, etc. However, if the application area, e.g., human body registration, is specified in advance it could be beneficial to integrate domain-specific knowledge. This can be done by incorporating a kinematical model which will be used to compensate for large articulated motion while the deformable registration will be used for the fine alignment. We expect this to both enlarge the basin of convergence of the method and reduce shape distortion since the gross motion is accounted for by a model with fewer degrees of freedom.

A further possibility to improve robustness could be to treat the registration problem within an extended Kalman filter framework, where the state prediction is performed by a domain-specific (e.g., physical, biomedical, etc.) model and the state update is done by the registration method proposed in this thesis.

Appendix A

Vision-Based Robotic Grasping of Known Objects

In this appendix, we demonstrate how our 3D object recognition approach presented in Chapter 3 can be used to support a real-world object manipulation task. The recognition is integrated into a robotic system to allow a robotic manipulator to grasp objects in unstructured, dynamically changing scenes. In order to perform such tasks in non-industrial environments, a robot cannot rely on hard-coded knowledge about the scene structure (Figure A.1). Since human actions, in particular, modify the environment in a way which cannot be foreseen, a vision-based object recognition and localization system is very useful for providing the necessary updates of the scene knowledge.

In recent years, advances in 3D geometry acquisition technology have led to a growing interest in object recognition and pose estimation techniques which operate directly on 3D data. Furthermore, as already mentioned in the introductory Chapter 1, the knowledge of the 3D geometric shape and the pose of an object greatly facilitates the execution of a stable grasp. The 2D appearance of an object may not provide reliable information about its pose in space because surface texture elements may be misaligned (as it often happens to labels of household objects). Furthermore, 2D techniques have to deal with changes in viewpoint and illumination.

Contributions and Overview

The efficiency of our 3D object recognition algorithm (see Chapter 3) allows its seamless integration into a grasping framework, where the recognition interleaves with the actual manipulation task without causing noticeable delays in the overall process. The method shows its potential in a complex experimental use-case. We employ the DLR Lightweight Robot III (LWR-III) [138], which is equipped with a Cartesian impedance control method and is able to react to environment disturbances and to process faults caused by unexpected contact forces in real-time. Using the proposed 3D object recognition method, impedance control with reactive recovery

Figure A.1: A robot operating in a household environment.

strategies, and a simple grasp planner the robot quickly and robustly grasps objects from unsorted and cluttered piles. Furthermore, in case of failures, it reacts accordingly and continues the process if possible. This grasping application demonstrates how the integration of computer vision and soft-robotics leads to a robotic system capable of acting in unstructured and occluded environments.

In Section A.1, the robot and its overall control concept for robust and sensitive grasping are shortly described. Section A.2 presents some experimental results. For more details please refer to [3].

A.1 Robotic Object Manipulation

The object recognition is only one link in the overall processing chain. In order to enable the robot to perform the recognition together with the object manipulation in a loop and to recover from faulty grasps, the overall process is controlled by a hybrid state automaton. The scheme is based on the work in [140, 141, 142] and relies on the disturbance observer introduced in [143]. The high-level schematic is shown in Figure A.2 and consists of the following phases:

1. *Go to overview*: The robot moves to an overview pose in order not to occlude the scene.
2. *Recognize objects*: Recognize the objects in the scene (see Chapter 3).
3. *Select a grasp*: Select an object (from the list of recognized ones) together with a suitable grasp.
4. *Grasp object*: The robot performs the grasp on the selected object.
5. *Carry away*: The robot carries the object to the place designated for it.
6. *Place down*: The robot softly and safely puts the object on its place.

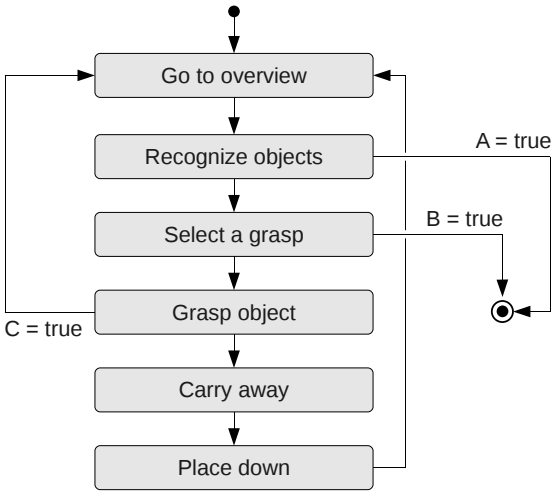


Figure A.2: Hybrid state automaton for controlling the overall object manipulation process. The logical clauses A, B and C defining the transition conditions, are the following: $A = no\ object$, $B = no\ grasp$ and $C = collision \vee grasp\ failure$. A grasp is considered as a failure if the robot gripper completely closes.

Next, phase 3 will be shortly explained. The phases 4, 5 and 6 are out of the scope of this work and will not be discussed any further (refer to [3] for more details).

The first step in the manipulation chain is the selection of an object (from the list of recognized ones returned by the recognition method) together with a suitable grasp. Each object in the database is associated with a finite set of plausible grasps. A grasp G (also called grasp frame) consists of an orientation and a position of the robot end effector relative to the object. The orientation is represented by a 3×3 rotation matrix. Its last column is a vector, called the approach vector $\mathbf{v}_{appr}$, which is aligned with the z-axis of the end effector and points towards the object.

The idea of the grasp selection is to go for the up most object, i.e., the one whose center of mass has the largest z-coordinate. This is measured according to the world coordinate system which has its z-axis, $\mathbf{z}_w$, perpendicular to the table the objects are placed on. More details on the scene setup will be given in Section A.2.2. Next, among the grasp frames associated with the selected object, the one with the lowest

$$\text{cost}(G) = w_1 \text{crit}_1(G) + w_2 \text{crit}_2(G), \tag{A.1}$$

is chosen, where

$$\text{crit}_1(G) = \angle(-\mathbf{v}_{appr}, \mathbf{z}_w) \tag{A.2}$$

is the alignment of the end effector with respect to the z-axis of the world coordinate system and

$$\text{crit}_2(G) = |\varphi_0^{wrist} - \varphi_{req}^{wrist}|, \tag{A.3}$$

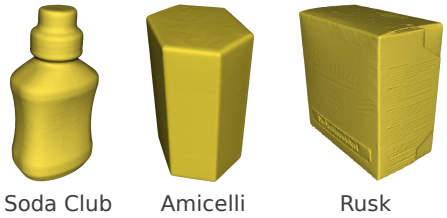


Figure A.3: The models used in the test scenarios (our own scans made with the low-cost DAVID laser scanner [132]).

is the absolute difference between φ_0^{wrist}, a desired (neutral) wrist orientation, and φ_{req}^{wrist}, the one required to perform the grasp. In our implementation, we set the weights in Equation (A.1) to $w_1 = 3$ and $w_2 = 1$ which makes the end effector alignment more important than the wrist orientation. Note that in Equation (A.2) the negative of $\mathbf{v}_{appr}$ is used in order to point in the same direction as $\mathbf{z}_w$. Furthermore, the selected grasp is discarded if it is "too parallel" to the table plane, i.e., if $\text{crit}_1(G) \geq \varphi_z$ (we set $\varphi_z = 30°$). If this is the case, then the next lower object is inspected.

To acquire the grasping motion, the selected grasp frame G is projected 0.1 m along the approach vector.

A.2 Experiments

In this section, we experimentally validate the "blind" impedance controlled grasping strategy (Section A.2.1) and the grasping capabilities of the vision-based impedance controlled system (Section A.2.2). We used the 7-degrees-of-freedom Cartesian impedance controlled DLR Lightweight robot III [138] developed at the German Aerospace Center (DLR). It was mounted on a table and covered an area of approximately 2.5 square meters. The scene was digitized with a Kinect sensor [41]. Since all objects were standing on or above the table, its plane was detected in each range image (using a simple RANSAC procedure) and all points belonging to the plane or lying below were removed. The object models involved in the tests are shown in Figure A.3 and the experiment setup is shown in Figure A.4.

A.2.1 "Blind" Impedance Controlled Grasping

In this section, we conducted a series of grasping experiments with the aim to find a set of impedance parameters that maximizes the grasping success in the presence of simulated object pose errors. Since such errors are inevitable for any object recognition and pose estimation algorithm which deals with real-world data, we find it useful to determine a parameter set which makes the robot as insensitive as possible to these inaccuracies.

We altered the robot's Cartesian translation stiffness in y-direction (denoted by $K_{t,y}$) and its rotation stiffness in x-direction (denoted by $K_{r,x}$). These directions are the lateral compliance

	$K_{t,y}$[N/m]	$K_{r,x}$[Nm/rad]	success [%]
1	200	20	60
2	200	75	80
3	200	200	90
4	750	20	70
5	750	75	80
6	750	200	40
7	2000	20	50
8	2000	75	60
9	2000	200	70

Table A.1: Grasping success with varying stiffness for a translational object pose error of 1.5 cm.

	object		
test scenario	Soda Club	Amicelli	Rusk
single standing objects	100%	95%	95%
object pile	95%	95%	90%

Table A.2: Success rates in the grasping experiments.

along the gripper motion and the rotation perpendicular to this. Due to the inherent structure of the gripper, they are the significant parameters governing the grasping process. The object involved in this test scenario was the soda club bottle shown in Figure A.3. The object pose error was simulated by translating the bottle by 1.5 cm in a random direction parallel to the table in front of the robot. We performed ten grasping trials for each of the following stiffness configurations: "soft", "moderately stiff" and "rigid". The success rate is listed in Table A.1. The optimal values (line 3) correspond to a soft (very compliant) translation and a rigid rotation behavior. A soft translation and a moderately stiff rotation (line 2) as well as a moderate stiffness in both translation and rotation (line 5) led to good success rates too.

A.2.2 Vision-Based Impedance Controlled Grasping

In this section, we experimentally validate the overall vision-based impedance controlled grasping system. The models used in these tests are shown in Figure A.3. We started with grasping single standing objects, moved on to object grasping from an unsorted pile and finished with a more complex task of cleaning up a table.

Figure A.4: The setup of the vision-based grasping experiments. The robot has grasped a green soda club bottle and is about to put it in the further red bin. The Kinect sensor can be seen in the upper right corner. In the lower left corner, the range image and the recognized models are shown (before the bottle has been taken away) from a viewpoint close to the one of the sensor.

A.2.2.1 Single Standing Objects

In the first scenario, multiple grasps were performed on single standing objects (see Figure A.5). We varied the pose of the objects such that all pre-saved grasp poses were executed. A grasp trial was considered successful if the object was correctly recognized, grasped and carried to the right place (table corner for the rusk box or one of the red bins for the Amicelli box/soda club bottle). We ran ten trials for each object pose and recorded the number of successful trials. The results are summarized in the first row of Table A.2. One grasp failed for the Amicelli and the rusk box, respectively. This was due to the fact that the alignment computed by the matching algorithm was too imprecise.

A.2.2.2 Object Pile

Next, the robot performed multiple grasps on a pile consisting of seven objects placed next and on top of each other. Again, we changed the positions of the items such that the robot tried all pre-saved grasp poses for each object. A grasp was considered successful if the robot picked a correctly recognized object and carried it to the right place. After an object had been taken away, we built up a new pile, i.e., the robot had to deal every time with a full pile. This experiment added some additional difficulties to both the geometry matching and the robot control algorithms. Obviously, the risk of recognition failures increased since there were more

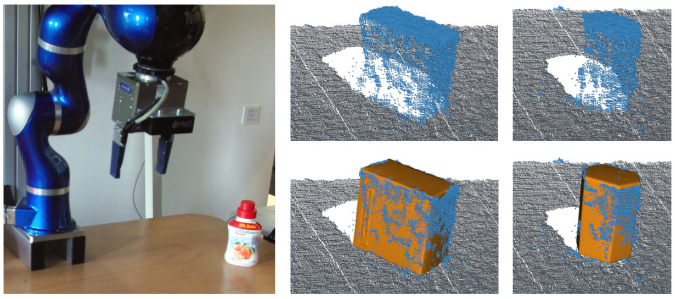


Figure A.5: (Left) The single standing object grasping scenario. (Middle, Right) two input range images (top) and the recognition results (bottom) for the rusk and the Amicelli box, respectively. The points off the plane (used for matching) are shown in light blue.

objects in the scene. Besides that, objects in a pile are in a more unstable configuration (from a physical point of view) compared to single standing ones. This made it more difficult for the impedance-based control to compensate for matching imprecision. We performed ten trials for each grasp pose and recorded the number of successful trials. The results are compiled in the second row of Table A.2. As to be expected, the failure rate increased compared to the first grasping experiment.

A.2.2.3 Table Cleanup

In the last test scenario, we let the robot repeatedly perform a more complex task, namely, cleaning up the table in front of it. Seven objects were randomly placed on a pile which resulted in highly cluttered and occluded scenes. The task to the robot was to pick each object, put it away and halt when it "believes" that the table is empty. The recognition process was restarted each time an object was carried away. Thus, the robot was able to deal with the changing scene and with unforeseen situations which happened during the cleanup like, e.g., an object falling off the pile. The task was accomplished if at the end each object was at the place designated for it. This time we did not consider it a failure when an object slipped out of the gripper as long as it was picked up later on and left in the right place. After each cleanup trial we built up a new pile and let the robot perform the task again. We repeated this 15 times and counted the number of successful trials. The robot achieved a success rate of 80%. Figure A.6 exemplary shows one cleanup process. More examples can be seen on youtube[1,2].

[1] http://www.youtube.com/watch?v=RuZ605o60LQ&feature=relmfu
[2] http://www.youtube.com/watch?v=pFyIPNxRZ8w&feature=relmfu

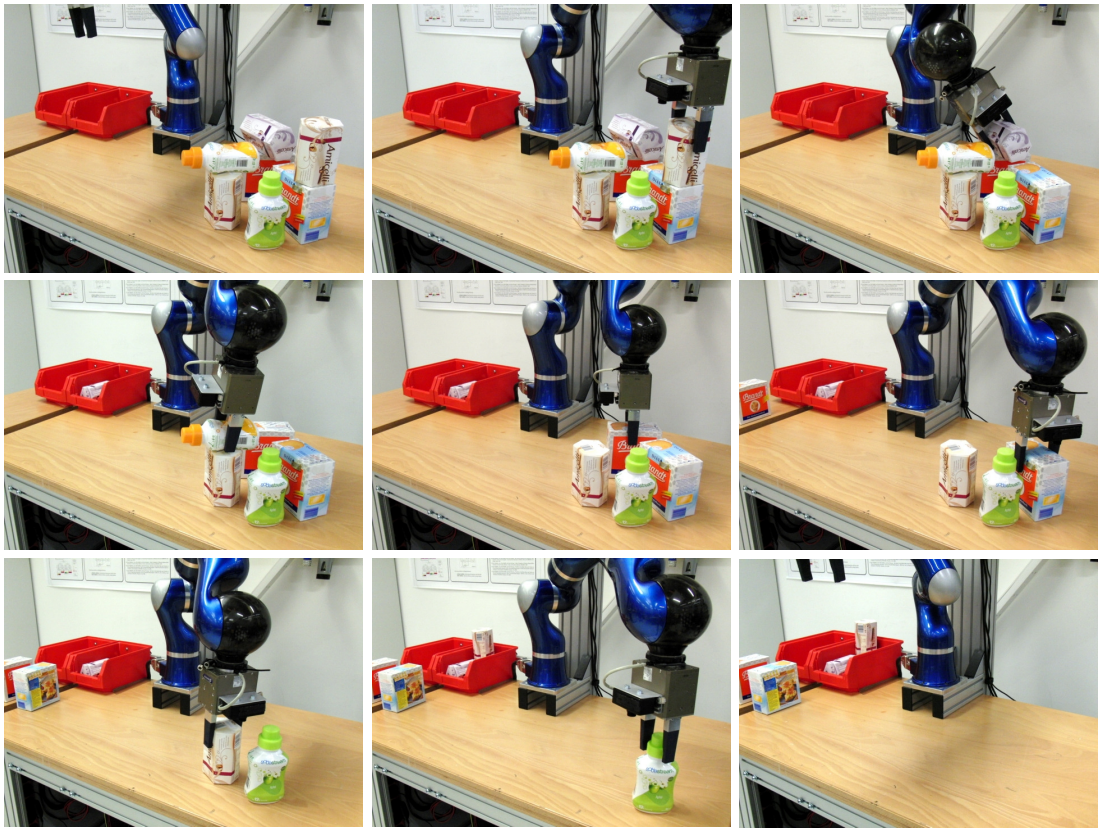


Figure A.6: (Left to right, top to bottom) A sequence of images showing the robot cleaning up the table. The first and the last image show the beginning and the end of the task, respectively. Each in-between shot shows the robot in the moment of grasping an object. Note that the first Amicelli box and all bottles are placed in a lying orientation in the red bins and are not visible from this point of view.

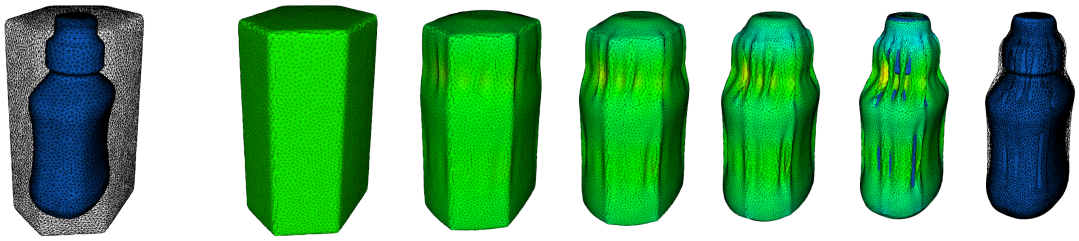
Appendix B

Knowledge Transfer through Deformable Registration

In this appendix, we discuss how our deformable registration method can be used to transfer knowledge between geometric models. What we mean by that is best explained by an example. In the case of object grasping, there is usually additional information associated with an object model, like stable grasp points on the object's surface. These are the points which should be preferred when a robotic gripper is about to establish a physical contact with the object to be grasped. In that case, if a recognition method localizes the model in the scene, the physical object can be grasped according to the saved points. However, we wish to use that information for objects which are similar but not identical to the one we have a model of, i.e., we need to transfer knowledge from one geometric primitive to another. The transfer can be done either online from a model to a part of a scene (i.e., right after the model has been localized) or offline from one model to another.

We provide an example which demonstrates how our deformable registration method (see Chapter 4) can be used for the offline transfer of grasp points from a hexagonal box to a round bottle. First, the models are rigidly aligned using the rigid registration method proposed in Chapter 2. The result is shown in Figure B.1(a). Next, the deformable registration algorithm is applied to compensate for the differences in the geometry and to map the box to the bottle. The process is shown at several time instances in Figure B.1(b). The degree of distortion is visualized on the deforming box surface using colors—blue/red stands for a scale down/up while green means no change in scale.

Note that the regularizer, part of the deformable registration process, strives to minimize stretching which explains why at the end of the registration the box surface is bent so much and looks like a paper bag with the air been extracted. After the registration, the information associated with each vertex in the box model can be transferred to the corresponding (closest) bottle vertex. The degree of distortion computed for each shape vertex can be used as an



(a) Box & bottle after rigid registration

(b) Deformable registration process

Figure B.1: Shape registration for transfer of grasp points from a box to a bottle. (a) The box (shown as a wireframe model) is rigidly registered to the bottle. (b) The deformation process at several time instances. At the end, the box is also shown as a wireframe model over the bottle.

indication for the reliability of the transferred grasp point—the larger the distortion the more unreliable the grasp point. Indeed, as can be seen in Figure B.1(b), the green surface regions at the bottle lid and the middle part of the bottle body are certainly better suited for a stable grasp than the blue/orange ones at the bottle neck and the top part of the lid.

Author's Publications

[1] Chavdar Papazov and Darius Burschka. **Stochastic Optimization for Rigid Point Set Registration**. In *Proceedings of the 5th International Symposium on Visual Computing (ISVC'09)*, 2009.

[2] Chavdar Papazov and Darius Burschka. **Deformable 3D Shape Registration Based on Local Similarity Transforms**. *Computer Graphics Forum (special issue SGP'11)*, **30**, 2011. 50

[3] Chavdar Papazov, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka. **Rigid 3D Geometry Matching for Grasping of Known Objects in Cluttered Scenes**. *International Journal of Robotics Research*, **31**(4), 2012. 89

[4] Chavdar Papazov, Vincent J. Dercksen, Hans Lamecker, and Hans-Christian Hege. **Visualizing Morphogenesis and Growth by Temporal Interpolation of Surface-Based 3D Atlases**. In *Proceedings of the 2008 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2008.

[5] Chavdar Papazov and Darius Burschka. **An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes**. In *Proceedings of the 10th Asian Conference on Computer Vision (ACCV'10)*, 2010.

[6] Chavdar Papazov and Darius Burschka. **Stochastic Global Optimization for Robust Point Set Registration**. *Computer Vision and Image Understanding*, **115**, 2011.

References

[7] Peter M. Roth and Martin Winter. **Survey of Appearance-Based Methods for Object Recognition**. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, 2008. 1, 2

[8] David G. Lowe. **Object Recognition from Local Scale-Invariant Features**. In *Proceedings of the International Conference on Computer Vision (ICCV'99)*, 1999. 1

[9] Martin A. Fischler and Robert C. Bolles. **Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography**. *Commun. ACM*, **24**(6):381–395, 1981. 1, 31

[10] Dana H. Ballard. **Generalizing the Hough Transform to Detect Arbitrary Shapes**. *Pattern Recognition*, **13**(2):111–122, 1981. 1, 29

[11] Martin Winter. *Spatial Relations of Features and Descriptors for Appearance Based Object Recognition*. PhD thesis, Faculty of Computer Science, Graz University of Technology, Austria, 2007. 1

[12] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2002. 2

[13] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001. 2

[14] Daniel D. Lee and H. Sebastian Seung. **Learning the Parts of Objects by Non-Negative Matrix Factorization**. *Nature*, **401**, 1999. 2

[15] Peter M. Roth. *On-line Conservative Learning*. PhD thesis, Faculty of Computer Science, Graz University of Technology, Austria, 2008. 2

[16] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science Department, Technische Universität Müchen, Germany, October 2009. 2, vii

[17] Myron Z. Brown, Darius Burschka, and Gregory D. Hager. **Advances in Computational Stereo**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **25**(8):993–1008, 2003. 3, 2

[18] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. **Bundle Adjustment - A Modern Synthesis**. In *Vision Algorithms: Theory and Practice, International Workshop on Vision Algorithms*, pages 298–372, 1999. 3, 2

[19] Jason J. Corso, Darius Burschka, and Gregory D. Hager. **Direct Plane Tracking in Stereo Images for Mobile Navigation**. In *ICRA*, pages 875–880, 2003. 3, 2

[20] Darius Burschka. *Vision-Based Exploration of Indoor Environments at an Example of a Binocular Stereo System*. PhD thesis, Technische Universität Müchen, Germany, 1999. 3, 2

[21] Dong-Min Woo and Dong-Chul Park. **Stereoscopic Building Reconstruction Using High-Resolution Satellite Image Data**. In *ACIS-ICIS*, pages 194–198, 2011. 3, 2

[22] Philipp Fechteler, Peter Eisert, and Jürgen Rurainsky. **Fast and High Resolution 3D Face Scanning**. In *Proceedings of the International Conference on Image Processing (ICIP'07)*, 2007. 3

[23] P. Fechteler and P. Eisert. **Adaptive Color Classification for Structured Light Systems**. In *Computer Vision and Pattern Recognition Workshops (CVPRW'08). IEEE Computer Society Conference on*, 2008. 3

[24] P. Besl and N. McKay. **A Method for Registration of 3-D Shapes**. *IEEE Trans. PAMI*, **14**(2), 1992. 3, 10, 11, 14, 43, 48, 62, vii, x, 40

[25] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. **Registration of Point Cloud Data from a Geometric Optimization Perspective**. In *Symposium on Geometry Processing*, pages 23–31, 2004. 3, 12, 14

[26] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu. **Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes**. *International Journal of Computer Vision*, **67**(3):277–296, 2006. 3, 12, 14

[27] Andrew W. Fitzgibbon. **Robust registration of 2D and 3D point sets**. *Image Vision Comput.*, **21**(13-14):1145–1153, 2003. 3, 12, 14, 60

[28] A. Johnson and M. Hebert. **Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes**. *IEEE Trans. PAMI*, **21**(5):433–449, 1999. 3, 10, 29, 65, 66, 69, 70, 84, xi, xii, 64, 68

[29] N. Gelfand, N. Mitra, L. Guibas, and H. Pottmann. **Robust Global Registration**. *Eurographics Symposium on Geometry Processing*, pages 197–206, 2005. 3, 10, 59, 64, 68, xi, 40, 67

[30] Stefanie Wuhrer, Zouhour Ben Azouz, and Chang Shu. **Posture Invariant Surface Description and Feature Extraction**. In *CVPR*, 2010. 3, 42, 57, 55

[31] Huai-Yu Wu, Hongbin Zha, Tao Luo, Xu-Lei Wang, and Songde Ma. **Global and Local Isometry-Invariant Descriptor for 3D Shape Comparison and Partial Matching**. In *CVPR*, 2010. 3, 42, 57, 55

[32] M. M. Bronstein and I. Kokkinos. **Scale-Invariant Heat Kernel Signatures for Non-Rigid Shape Recognition**. In *CVPR*, 2010. 3, 42, 43, 57, 55

[33] D. Raviv, M. M. Bronstein, A. M. Bronstein, and R. Kimmel. **Volumetric Heat Kernel Signatures**. In *3DOR*, 2010. 3, 42, 57, 55

[34] D. Aiger, N. Mitra, and D. Cohen-Or. **4-Points Congruent Sets for Robust Pairwise Surface Registration**. *ACM Trans. Graph.*, **27**(3), 2008. 4, 10, 32, 42, 60, 61, 62, x, 11

[35] Franc Solina and Ruzena Bajcsy. **Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations**. *IEEE TPAMI*, **12**(2):131–147, 1990. 4, 29

[36] Sven J. Dickinson, Dimitris N. Metaxas, and Alex Pentland. **The Role of Model-Based Segmentation in the Recovery of Volumetric Parts From Range Data**. *IEEE TPAMI*, **19**(3):259–267, 1997. 4, 29

[37] Georg Biegelbauer, Markus Vincze, and Walter Wohlkinger. **Model-based 3D object detection**. *Mach. Vis. Appl.*, **21**(4):497–516, 2010. 4, 29

[38] Brian Amberg, Sami Romdhani, and Thomas Vetter. **Optimal Step Nonrigid ICP Algorithms for Surface Registration**. In *CVPR*, 2007. 4, 44

[39] Hao Li, Robert W. Sumner, and Mark Pauly. **Global Correspondence Optimization for Non-Rigid Registration of Depth Scans**. *Comput. Graph. Forum*, **27**(5), 2008. 4, 44

[40] R. Sagawa, K. Akasaka, Y.S. Yagi, H. Hamer, and L.J. Van Gool. **Elastic Convolved ICP for the Registration of Deformable Objects**. In *3DIM*, 2009. 4, 44, 72, xii, 71

[41] Kinect for Xbox 360. `http://www.xbox.com/en-US/kinect`, 2011. Accessed: 20/04/2011. 6, 40, 90, vii, 2, 5, 83

[42] Hans-Christian Hege. **Current Applications of Scientific Visualization**. *IT – Information Technology*, **46**(3):142–147, 2004. 7, 4

[43] Y. Hecker and R. Bolle. **On Geometric Hashing and the Generalized Hough Transform**. *IEEE Trans. on Systems, Man, and Cybernetics*, **24**, 1994. 10

[44] H.J. Wolfson and I. Rigoutsos. **Geometric Hashing: an Overview**. *Computational Science & Engineering, IEEE*, **4**:10–21, 1997. 10

[45] G. Stockman. **Object Recognition and Localization via Pose Clustering**. *Computer Vision, Graphics, and Image Processing*, **40**(3):361–387, 1987. 10, 29

[46] Y. Chen and G. Medioni. **Object Modeling by Registration of Multiple Range Images**. *Robotics and Automation, Proceedings., IEEE International Conference on*, **3**:2724–2729, 1991. 11, 43, 40

[47] S. Rusinkiewicz and M. Levoy. **Efficient Variants of the ICP Algorithm**. *3DIM*, pages 145–152, 2001. 11

[48] S. Granger and X. Pennec. **Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration**. *in: ECCV, Proceedings*, pages 418–432, 2002. 11

[49] Y. Tsin and T. Kanade. **A Correlation-Based Approach to Robust Point Set Registration**. *in: ECCV, Proceedings*, pages 558–569, 2004. 11, 44, 75, 74

[50] B. Jian and B. C. Vemuri. **A Robust Algorithm for Point Set Registration Using Mixture of Gaussians**. *in: ICCV, Proceedings*, pages 1246–1251, 2005. 11

[51] Toru Tamaki, Miho Abe, Bisser Raytchev, and Kazufumi Kaneda. **Softassign and EM-ICP on GPU**. *in: 2nd Workshop on Ultra Performance and Dependable Acceleration Systems (UPDAS), Proceedings*, 2010. 11

[52] B. Ma and R. E. Ellis. **Surface-Based Registration with a Particle Filter**. *in: MICCAI, Proceedings*, pages 566–573, 2004. 11

[53] Mehdi Hedjazi Moghari and Purang Abolmaesumi. **Point-Based Rigid-Body Registration Using an Unscented Kalman Filter**. *IEEE Trans. Med. Imaging*, **26**(12):1708–1728, 2007. 11, 12

[54] Romeil Sandhu, Samuel Dambreville, and Allen Tannenbaum. **Point Set Registration via Particle Filtering and Stochastic Dynamics**. *IEEE Trans. PAMI*, **32**(8):1459–1473, 2010. 11, 12

[55] Thomas M. Breuel. **Implementation techniques for geometric branch-and-bound matching methods**. *Computer Vision and Image Understanding*, **90**(3):258–294, 2003. 12

[56] Carl Olsson, Fredrik Kahl, and Magnus Oskarsson. **Branch-and-Bound Methods for Euclidean Registration Problems**. *IEEE Trans. PAMI*, **31**(5):783–794, 2009. 12

[57] Hongdong Li and Richard I. Hartley. **The 3D-3D Registration Problem Revisited**. *in: ICCV, Proceedings*, pages 1–8, 2007. 12, 16, 63, x, 62

[58] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. **Equation of State Calculations by Fast Computing Machines**. *The Journal of Chemical Physics*, **21**(6):1087–1092, 1953. 12, 20, 13

[59] V. Cerny. **Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm**. *Journal of Optimization Theory and Applications*, **45**:41–51, 1985. 12, 13

[60] S. Kirkpatrick, C.D. Gelatt, and M. Vecchi. **Optimization by Simmulated Annealing**. *Science*, **220**(4598):671–680, 1983. 12, 13

[61] P. Pardalos and E. Romeijn, editors. *Handbook of Global Optimization 2*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, 2002. 12, 13, 17

[62] D. Bulger and G. Wood. **Hesitant Adaptive Search for Global Optimisation**. *Math. Program.*, **81**:89–102, 1998. 13

[63] G. Bilbro and W. Snyder. **Optimization of Functions with Many Minima**. *IEEE Trans. on Systems, Man, and Cybernetics*, **21**(4):840–849, 1991. 13, 16, 17

[64] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990. 17

[65] K. Kanatani. *Group-Theoretical Methods in Image Understanding*. Springer Series in Information Sciences. Springer, 1990. 22, 23

[66] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992. 22

[67] Richard I. Hartley and Fredrik Kahl. **Global Optimization through Rotation Space Search**. *International Journal of Computer Vision*, **82**(1):64–79, 2009. 22, 23

[68] N. Madras. *Lectures on Monte Carlo Methods*. American Mathematical Society, 2002. 25

[69] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, J. Hermans, Shun Kawamura, Y. Kurita, Guillaume LavouÃ©, H.V. Nguyen, Ryutarou Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, Martin Reuter, Ivan Sipiran, Dirk Smeets, Paul Suetens, Hedi Tabia, and Dirk Vandermeulen. **SHREC'11 Track:**

Shape Retrieval on Non-Rigid 3D Watertight Meshes. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval (3DOR'11)*, 2011. 29

[70] Y. Lamdan and H.J. Wolfson. **Geometric Hashing: A General and Efficient Model-Based Recognition Scheme**. In *ICCV*, 1988. 29

[71] Ajmal S. Mian, Mohammed Bennamoun, and Robyn A. Owens. **Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes**. *IEEE TPAMI*, **28**(10):1584–1601, 2006. 29, 66, 69, 70, 84, xi, xii, 64, 68

[72] Günter Hetzel, Bastian Leibe, Paul Levi, and Bernt Schiele. **3D Object Recognition from Range Images Using Local Feature Histograms**. In *CVPR*, pages 394–399, 2001. 29

[73] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. **Recognizing Objects in Range Data Using Regional Point Descriptors**. In *ECCV*, pages 224–237, 2004. 29

[74] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. **A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion**. *Comput. Graph. Forum*, **28**(5):1383–1392, 2009. 29

[75] Huai-Yu Wu, Hongbin Zha, Tao Luo, Xulei Wang, and Songde Ma. **Global and Local Isometry-Invariant Descriptor for 3D Shape Comparison and Partial Matching**. In *CVPR*, pages 438–445, 2010. 29

[76] T.O. Binford. **Visual Perception by a Computer**. In *IEEE Conf. on Systems and Control*, 1971. 29

[77] A.H. Barr. **Superquadrics and Angle-Preserving Transformations**. *Computer Graphics and Applications*, **1**(1):11 –23, 1981. 29

[78] Daniel Keren, David B. Cooper, and Jayashree Subrahmonia. **Describing Complicated Objects by Implicit Polynomials**. *IEEE TPAMI*, **16**(1):38–53, 1994. 29

[79] Geoffrey Taylor and Lindsay Kleeman. **Robust Range Data Segmentation using Geometric Primitives for Robotic Applications**. In *SIP*, pages 467–472, 2003. 29

[80] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. **Efficient RANSAC for Point-Cloud Shape Detection**. *Comput. Graph. Forum*, **26**(2):214–226, 2007. 29, 31, 32

[81] Lynne Grewe and Avinash C. Kak. **Interactive Learning of a Multiple-Attribute Hash Table Classifier for Fast Object Recognition**. *Computer Vision and Image Understanding*, **61**(3):387–416, 1995. 29

[82] Bogdan Matei, Ying Shan, Harpreet S. Sawhney, Yi Tan, Rakesh Kumar, Daniel F. Huber, and Martial Hebert. **Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation**. *IEEE TPAMI*, **28**(7):1111–1126, 2006. 29, 30

[83] Simon Winkelbach, Sven Molkenstruck, and Friedrich M. Wahl. **Low-Cost Laser Range Scanner and Fast Surface Registration Approach**. In *Pattern Recognition, 28th DAGM Symposium, Proceedings*, pages 718–728, 2006. 30, 32, 29

[84] Hanzi Wang and David Suter. **Robust Adaptive-Scale Parametric Model Estimation for Computer Vision**. *IEEE TPAMI*, **26**(11):1459–1474, 2004. 32, 31

[85] Hanzi Wang, Daniel Mirota, and Gregory D. Hager. **A Generalized Kernel Consensus-Based Robust Estimator**. *IEEE TPAMI*, **32**(1):178–184, 2010. 32, 31

[86] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng. **RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images**. *IEEE Trans. PAMI*, **21**(11):1229–1234, 1999. 32, 31

[87] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2 edition, 2000. 33, 34

[88] Thibaut Weise, Bastian Leibe, and Luc J. Van Gool. **Fast 3D Scanning with Automatic Motion Compensation**. In *CVPR*, 2007. 40, 71, 72, xii

[89] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus H. Gross. **Meshless Deformations Based on Shape Matching**. *ACM TOG*, **24**(3), 2005. 40, 43, 48, 50, 56

[90] Mario Botsch, Mark Pauly, Markus H. Gross, and Leif Kobbelt. **PriMo: Coupled Prisms for Intuitive Surface Modeling**. In *Symposium on Geometry Processing*, 2006. 40, 41, 42, 46, 56

[91] Olga Sorkine and Marc Alexa. **As-Rigid-As-Possible Surface Modeling**. In *Symposium on Geometry Processing*, pages 109–116, 2007. 40, 42, 56

[92] Mario Botsch, Mark Pauly, Martin Wicke, and Markus H. Gross. **Adaptive Space Deformations Based on Rigid Cells**. *Comput. Graph. Forum*, **26**(3), 2007. 40, 42, 43, 56

[93] Alec R. Rivers and Doug L. James. **FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation**. *ACM TOG*, **26**(3), 2007. 40, 43, 50, 56

[94] D. Steinemann, M. A. Otaduy, and M. Gross. **Fast Adaptive Shape Matching Deformations**. In *SCA*, 2008. 40, 43, 50, 56

[95] Mario Botsch and Olga Sorkine. **On Linear Variational Surface Deformation Methods**. *IEEE Trans. Vis. Comput. Graph.*, **14**(1):213–230, 2008. 41

[96] Olga Sorkine and Mario Botsch. **Tutorial: Interactive Shape Modeling and Deformation**. In *EUROGRAPHICS*, 2009. 41

[97] Thomas W. Sederberg and Scott R. Parry. **Free-Form Deformation of Solid Geometric Models**. In *SIGGRAPH*, pages 151–160, 1986. 41

[98] William M. Hsu, John F. Hughes, and Henry Kaufman. **Direct Manipulation of Free-Form Deformations**. In *SIGGRAPH*, pages 177–184, 1992. 41

[99] Karan Singh and Eugene Fiume. **Wires: A Geometric Deformation Technique**. In *SIGGRAPH*, 1998. 41

[100] Demetri Terzopoulos, John C. Platt, Alan H. Barr, and Kurt W. Fleischer. **Elastically Deformable Models**. In *SIGGRAPH*, pages 205–214, 1987. 41

[101] George Celniker and Dave Gossard. **Deformable Curve and Surface Finite-Elements for Free-Form Shape Design**. In *SIGGRAPH*, pages 257–266, 1991. 41

[102] William Welch and Andrew Witkin. **Variational Surface Modeling**. In *SIGGRAPH*, pages 157–166, 1992. 41

[103] Denis Zorin, Peter Schröder, and Wim Sweldens. **Interactive Multiresolution Mesh Editing**. In *SIGGRAPH*, pages 259–268, 1997. 41

[104] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. **Interactive Multi-Resolution Modeling on Arbitrary Meshes**. In *SIGGRAPH*, pages 105–114, 1998. 41

[105] Igor Guskov, Wim Sweldens, and Peter Schröder. **Multiresolution Signal Processing for Meshes**. In *SIGGRAPH*, pages 325–334, 1999. 41

[106] Mario Botsch and Leif Kobbelt. **An Intuitive Framework for Real-Time Freeform Modeling**. *ACM Trans. Graph.*, **23**(3):630–634, 2004. 41

[107] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. **Differential Coordinates for Interactive Mesh Editing**. In *SMI*, pages 181–190, 2004. 41

[108] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. **Laplacian Surface Editing**. In *Symposium on Geometry Processing*, pages 179–188, 2004. 41

[109] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. **Mesh Editing with Poisson-Based Gradient Field Manipulation**. *ACM Trans. Graph.*, **23**(3):644–651, 2004. 41

[110] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. **Linear Rotation-Invariant Coordinates for Meshes**. *ACM Trans. Graph.*, **24**(3):479–487, 2005. 41

[111] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. **Harmonic Guidance for Surface Deformation**. *Comput. Graph. Forum*, **24**(3):601–609, 2005. 41

[112] Robert W. Sumner, Johannes Schmid, and Mark Pauly. **Embedded Deformation for Shape Manipulation**. *ACM TOG*, **26**(3), 2007. 42

[113] Asi Elad and Ron Kimmel. **On Bending Invariant Signatures for Surfaces**. *IEEE TPAMI*, **25**(10), 2003. 43

[114] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. **Generalized Multidimensional Scaling: A Framework for Isometry-Invariant Partial Surface Matching**. *PNAS*, **103**(5), 2006. 43

[115] Stefanie Wuhrer, Chang Shu, Prosenjit Bose, and Zouhour Ben Azouz. **Posture Invariant Correspondence of Incomplete Triangular Manifolds**. *International Journal of Shape Modeling*, **13**(2), 2007. 43

[116] Stefanie Wuhrer, Chang Shu, and Prosenjit Bose. **Posture Invariant Correspondence Of Triangular Meshes In Shape Space**. In *3DIM*, 2009. 43

[117] Brett Allen, Brian Curless, and Zoran Popovic. **The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans**. *ACM TOG*, **22**(3), 2003. 43, 44

[118] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. **SCAPE: Shape Completion and Animation of People**. *ACM TOG*, **24**(3), 2005. 43

[119] Vladislav Kraevoy and Alla Sheffer. **Template-Based Mesh Completion**. In *Symposium on Geometry Processing*, 2005. 43

[120] Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus H. Gross, and Leonidas J. Guibas. **Example-Based 3D Scan Completion**. In *Symposium on Geometry Processing*, 2005. 43

[121] Will Chang and Matthias Zwicker. **Automatic Registration for Articulated Shapes**. *Comput. Graph. Forum*, **27**(5), 2008. 43

[122] Leslie Ikemoto, Natasha Gelfand, and Marc Levoy. **A Hierarchical Method for Aligning Warped Meshes**. In *3DIM*, 2003. 43

[123] Benedict Brown and Szymon Rusinkiewicz. **Non-Rigid Range-Scan Alignment Using Thin-Plate Splines**. In *3DPVT*, 2004. 43

[124] Benedict Brown and Szymon Rusinkiewicz. **Global Non-Rigid Alignment of 3-D Scans**. *ACM TOG*, **26**(3), 2007. 43

[125] Haili Chui and Anand Rangarajan. **A New Point Matching Algorithm for Non-Rigid Registration**. *CVIU*, **89**(2-3), 2003. 43, 44, 75, 74

[126] Andriy Myronenko and Xubo B. Song. **Point Set Registration: Coherent Point Drift**. *IEEE TPAMI*, **32**(12), 2010. 43, 44, 75, 74

[127] B. Jian and B. Vemuri. **Robust Point Set Registration Using Gaussian Mixture Models**. *IEEE TPAMI*, 2011. 44, 75, 74

[128] Andriy Myronenko and Xubo B. Song. **On the Closed-Form Solution of the Rotation Matrix Arising in Computer Vision Problems**. *CoRR*, **abs/0904.1613**, 2009. 48, 49

[129] C. H. Edwards Jr. *Advanced Calculus of Several Variables*. Dover Books on Mathematics, 1994. 53

[130] Yun Zeng, Chaohui Wang, Yang Wang, Xianfeng Gu, Dimitris Samaras, and Nikos Paragios. **Dense Non-Rigid Surface Registration Using High-Order Graph Matching**. In *CVPR*, 2010. 57, 55

[131] Konica Minolta. **Minolta VIVID 910**. http://www.konicaminolta.com/instruments/products/3d/non-contact/vivid910/index.html, 2011. Accessed: 18/09/2011. 66, 64

[132] DAVID Laser Scanner. **DAVID Vision Systems**. http://www.david-laserscanner.com/, 2012. Accessed: 16/07/2012. 66, 90, xiii, 64, 82

[133] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. **The Landscape of Parallel Computing Research: A View from Berkeley**. Technical report, EECS Department, University of California, Berkeley, 2006. 70, 69

[134] Huamin Wang, James O'Brien, and Ravi Ramamoorthi. **Multi-Resolution Isotropic Strain Limiting**. *ACM TOG*, **29**(6), 2010. 71, 70

[135] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. **Markerless and Efficient 26-DOF Hand Pose Recovery**. In *Proceedings of the Asian Conference on Computer Vision (ACCV'10)*, pages 744–757, 2010. 76

[136] I. Oikonomidis, N. Kyriazis, and A. Argyros. **Efficient Model-Based 3D Tracking of Hand Articulations Using Kinect**. In *Proceedings of the British Machine Vision Conference (BMVC'11)*, 2011. 76

[137] N. Kyriazis, I. Oikonomidis, and A. Argyros. **A GPU-Powered Computational Framework for Efficient 3D Model-Based Vision**. Technical Report TR420, ICS-FORTH, July 2011. 76

[138] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger. **The DLR Lightweight Robot – Lightweight Design and Soft Robotics Control Concepts for Robots in Human Environments**. *Industrial Robot Journal*, **34**(5):376–385, 2007. 84, 87, 90, 79

[139] Zelda B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Kluwer Academic Publishers, 2002. 85

[140] S. Haddadin, M. Suppa, S. Fuchs, Tim Bodenmüller, A. Albu-Schäffer, and G. Hirzinger. **Towards the Robotic Co-Worker**. In *International Symposium on Robotics Research (ISRR2009), Lucerne, Switzerland*, 2009. 88, 80

[141] Sven Parusel, Sami Haddadin, and Alin Albu-Schäffer. **Modular State-Based Behavior Control for Safe Human-Robot Interaction: A Lightweight Control Architecture for a Lightweight Robot**. In *IEEE Int. Conf. on Robotics and Automation (IROS2011), Shanghai, China*, 2011. 88, 80

[142] S. Fuchs, S. Haddadin, S. Parusel, M. Keller, and A. Kolb. **Cooperative Bin-Picking with Time-of-Flight Camera and Impedance Controlled DLR Lightweight Robot III**. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008)*, pages 4862–4867, 2010. 88, 80

[143] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger. **Collision Detection & Reaction: A Contribution to Safe Physical Human-Robot Interaction**. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008), Nice, France*, pages 3356–3363, 2008. 88, 80