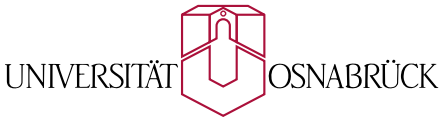


Institute for Computer Science
Knowledge-Based Systems

Master's Thesis

Deforming Scans for Improving the Map Quality using Plane Extraction and Thin Plate Splines

Dorit Borrmann
Jan Elseberg

August 2009

First supervisor: Prof. Dr. Joachim Hertzberg
Second supervisor: Prof. Dr. Werner Brockmann

Abstract

This thesis presents methods to improve the quality of laser scan maps. Most man-made environments consist of a large amount of planar surfaces. The idea behind our approach is to detect those planes in the point clouds produced by laser scans and to use them to decrease the impacts of noise and systematic errors. For plane detection the Hough Transform is applied. A thorough evaluation of different Hough methods and their applicability for plane detection in 3D point clouds is presented in this thesis. Furthermore, different ways to non-rigidly deform laser scans in order to fit them to the planes are explored. The Thin Plate Spline, a non-rigid alignment procedure, is compared to a novel slice-based deformation that is designed to remove the systematic errors produced by rotating 2D laser scanners to achieve 3D point clouds. To better estimate the actual structure of the laser scans the global point positioning of Brown and Rusinkiewicz is evaluated and extended leading to better results in accuracy and speed. Further improvements of the laser scan maps are achieved by introducing classification and clustering algorithms to create a reliable plane model. The methods in this thesis can be combined for the task of improving the map quality of laser scan maps. However, the methods works independently, meaning that they can be substituted by any method achieving a similar task. Additionally, the algorithms here presented can be used separately to solve other problems.

Zusammenfassung

In dieser Arbeit werden Methoden vorgestellt zur Verbesserung der Qualität von Laserscan-Karten. Die meisten künstlich geschaffenen Umgebungen bestehen aus einer großen Anzahl ebener Oberflächen. Die Idee hinter unserem Ansatz besteht darin, diese Ebenen in den Punktwolken, die ein Laserscanner liefert, zu identifizieren und zur Verringerung des Einflusses von Sensorrauschen und systematischen Fehlern zu verwenden. Für die Ebenenerkennung verwenden wir die Hough Transformation. Diese Arbeit liefert eine gründliche Evaluierung verschiedener Hough-Methoden und ihrer Anwendbarkeit für die Ebenenerkennung in 3D-Punktwolken. Des Weiteren werden verschiedene Möglichkeiten untersucht, Laserscans nicht-rigide zu verformen um sie an die Ebenen anzupassen. Thin Plate Splines werden als eine dieser Methoden verglichen mit einem neuartigen schicht-basierten Deformierungs-Verfahren, das dafür bestimmt ist, systematische Fehler zu beseitigen die bei der Rotation eines 2D-Laserscanners entstehen, wenn 3D-Punktwolken erzeugt werden sollen. Um eine bessere Schätzung der tatsächlichen Struktur eines Laserscans zu erzielen, wird die globale Punktpositionierung von Brown und Rusinkiewicz untersucht und erweitert, so dass eine größere Genauigkeit und höhere Geschwindigkeit erreicht werden. Die Verwendung von Klassifizierungs- und Clusteralgorithmen zur Erstellung eines zuverlässigen Ebenenmodells führt zu weiteren Verbesserungen der Laserscan-Karten. Die Methoden in dieser Arbeit können kombiniert werden um die Qualität von Laserscan-Karten zu verbessern. Sie arbeiten jedoch unabhängig voneinander. Folglich können sie durch jede ähnliche Methode ersetzt werden. Außerdem können die hier vorgestellten Algorithmen auch zur Lösung anderer Problemstellungen verwendet werden.

Contents

1 Introduction **3**
- 1.1 3D Robotic Vision . . . 4
- 1.2 Problem Definition . . . 6
- 1.3 Scientific Contribution . . . 7
- 1.4 Outline of this Thesis . . . 7

2 State of the Art **9**

3 Mapping **13**
- 3.1 Dimensionality of Mapping . . . 14
  - 3.1.1 Planar 2D Mapping . . . 15
  - 3.1.2 Planar 3D Mapping . . . 15
  - 3.1.3 Slice-wise 6D SLAM. . . . 16
  - 3.1.4 Full 6D SLAM. . . . 16
- 3.2 Pairwise Scan Matching . . . 16
  - 3.2.1 Point Selection . . . 17
  - 3.2.2 Finding Corresponding Point Pairs . . . 18
  - 3.2.3 Elimination of Outliers . . . 22
  - 3.2.4 Confidence Weighting of Correspondences . . . 22
  - 3.2.5 Minimization of the Error Function . . . 23
- 3.3 Global Mapping . . . 30
  - 3.3.1 Problem Formulation . . . 31
  - 3.3.2 Scan Matching . . . 32
  - 3.3.3 Minimization of the Error Function . . . 32
  - 3.3.4 Global Registration using the Small Angle Approximation . . . 34
  - 3.3.5 Uncertainty-based Global Registration . . . 36
- 3.4 Algorithm Overview . . . 39

4 The Thin Plate Spline **41**
- 4.1 Definition of the Thin Plate Spline . . . 42
- 4.2 Solving the Thin Plate Spline . . . 43
- 4.3 Approximating the Thin Plate Spline . . . 46
  - 4.3.1 Subsampling . . . 47
  - 4.3.2 Basis Function Subset . . . 50

4.3.3 Experimental Evaluation . . . 50
4.4 Variations on the Thin Plate Spline . . . 52
4.4.1 Landmark Errors . . . 52
4.4.2 Orientation Attributes . . . 54

5 Plane Extraction **61**
5.1 Hough Transform . . . 61
5.1.1 Standard Hough Transform . . . 62
5.1.2 Probabilistic Hough Transform . . . 65
5.1.3 Randomized Hough Transform . . . 68
5.1.4 Multiresolution Hough Transform . . . 70
5.1.5 Fast Hough Transform . . . 71
5.2 Structure of the Accumulator . . . 73
5.2.1 Accumulator Array . . . 74
5.2.2 Accumulator Cube . . . 75
5.2.3 Polyhedral Accumulator . . . 77
5.2.4 Accumulator Ball . . . 78
5.2.5 Dynamic Data Structures . . . 80
5.3 Plane Fitting . . . 82
5.3.1 Point Selection . . . 82
5.3.2 Least Squares Best Fitting Plane . . . 83
5.3.3 Point Segmentation/Clustering . . . 84
5.3.4 Jarvis' March Convex Hull . . . 86
5.4 Experimental Evaluation of the Hough Transform . . . 88
5.4.1 Comparison of the Different Accumulator Designs . . . 88
5.4.2 Standard Hough Transform . . . 90
5.4.3 Evaluation of Hough Transform Methods . . . 93

6 Improvement of the Plane Model **99**
6.1 Classification . . . 99
6.1.1 A Prolog Program . . . 100
6.1.2 Graph Coloring Approach . . . 101
6.1.3 Clustering Algorithm . . . 102
6.2 Improvement Step . . . 103
6.3 Experimental Results . . . 107

7 Deforming Scans **111**
7.1 Deformation using Geometrical Information . . . 111
7.1.1 Deformation with TPS . . . 113
7.1.2 Deformation without TPS . . . 114
7.2 Deformation without Geometrical Information . . . 119
7.2.1 Finding Features and their Correspondences . . . 121
7.2.2 Computing the Global Point Positions . . . 122
7.2.3 Mapping to the Global Points . . . 126

7.2.4 Experimental Evaluation 126
7.3 Manual Deformation 132

8 Conclusions and Outlook **135**

Table of Symbols

$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\boldsymbol{m}_i$	Reference point with index i that will be mapped to d_i
$\boldsymbol{d}_i$	Destination point with index i
$O(f)$	Big O Notation, $O(f) := \{g : \mathbb{N} \to \mathbb{N} \ \exists c > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 : g(n) \leq c \cdot f(n)\}$
$\boldsymbol{a}_i = \begin{pmatrix} a_{i,x} \\ a_{i,y} \\ a_{i,z} \end{pmatrix}$	3-dimensional vector with index i
$\|\boldsymbol{a}\|$	Norm of vector $\boldsymbol{a}$
$\boldsymbol{A}^T$	The transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	The inverse of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{[a:b,c:d]}$	Submatrix of $\boldsymbol{A}$ comprised of rows a to b and columns c to d
$\boldsymbol{a}_{[b:c]}$	Subvector of $\boldsymbol{a}$ comprised of coordinates b to c
$\boldsymbol{R}_{\theta_x,\theta_y,\theta_z}$	A 3×3 rotation matrix as defined by the Euler angles θ_x, θ_y and θ_z
$\boldsymbol{I}$	The Identity matrix
$\boldsymbol{I}_d$	The $d \times d$ Identity matrix
$\boldsymbol{A} = (\boldsymbol{A}_{i,j})$	The matrix $\boldsymbol{A}$ is given by the submatrices $\boldsymbol{A}_{i,j}$, where $\boldsymbol{A}_{i,j}$ may be any expression dependent upon i and j.
$N_\varphi, N_\theta, N_\rho$	The Number of cells in the direction of $\varphi, \theta and \rho$ in the accumulator for the Hough Transform
φ', θ', ρ'	size of a cell in the accumulator for the Hough Transform
$\mathbb{S}^2$	2-sphere, i.e., the set of vectors in $\mathbb{R}^3$ with unit length

Chapter 1

Introduction

One of the main research topics in mobile robotics attends to creating maps of the robot's environment. The acquisition of the data needed to create a map as well as the application of the maps are manifold. First, for an autonomous mobile robot to enter an unknown environment and start performing work is a difficult task. A map eases localization and path planning and thereby enables the robot to perform more specific tasks. Second, the ability of a robot to build its own map autonomously also entails the possibility to let a robot explore an unknown environment and map it for later inspection by humans. Typical applications are mapping of disaster areas to allow for fast inspection to find victims without unnecessarily jeopardizing the life of rescue workers. Further possibilities are automatically generated 3D models of expedition areas and fairgrounds that allow visitors to explore the sites beforehand to make plans, or afterwards to recapitulate what they have seen.

Mapping is often achieved by matching separately collected partial views, for example laser scans, of the environment. A laser scan is a set of distance values measured from a robot pose. Each laser scan, taken by a robot at different positions, represents a small part of the environment, the part that is observable from the current robot position. The process of joining these partial views into one map of a larger environment is referred to as *scan matching*. The main challenge in scan matching is the estimation of the correct poses for all laser scans. For human applicability however, the quality of the map plays an important role. This thesis focuses on the improvement of the quality of a map created by an autonomous mobile robot.

Scan matching is subject to different types of errors. Next to matching errors that are caused by the inevitable imprecision of pose estimates achieved by different scan matching algorithms, each laser scan itself is erroneous. Laser scanners on mobile robots are easily shattered. High-precision/High-end laser scanners are costly and fragile. Therefore the equipment that is mounted on mobile robots is usually cheap and more error-prone. Next to sensor noise that causes inaccuracy in the laser data systematic errors often occur. Fig. 1.1 shows an example of systematic errors in laser scans. The data was acquired by a 2D-laser scanner that was extended with a mount and a small servomotor. Imprecise vertical movements cause the waves that become obvious when examining flat planes in the laser data. These structures are not only unaesthetic but make it also harder to work with the resulting map. The structure of indoor environments usually comprises a large amount of planar surfaces. This knowledge can be used

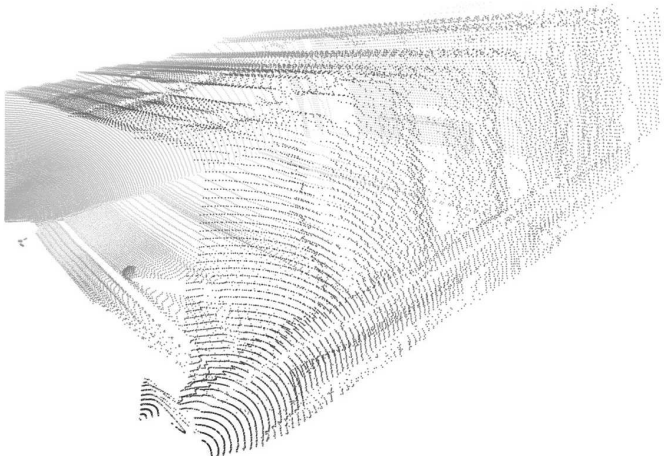


Fig. 1.1: Laser scan of an empty room. On the ceiling the wave-like systematic errors become visible.

to help reconstruct the real structure of the environment by diminishing systematic errors. In this thesis planar surfaces are extracted from point clouds collected by laser scanners and the entire laser scans are fitted to the extracted planes.

1.1 3D Robotic Vision

For most human beings vision is the sensory perception they consider as most important. For a robot to gather data that is later used by humans, vision is accordingly one of the most demanded requirements. To achieve a realistic view and depict all the given information 3D images are necessary. Two different types of sensors are commonly used to achieve this task, cameras and laser range scanners.

Currently there are basically two different ways in mobile robotics to perceive 3-dimensional images [68]. First, a depth estimate of two corresponding features is achieved by triangulation. For this method the baseline has to be known. Second, the time-of-flight principle is used to determine the depth of objects in images. This principle is based on measuring the time needed between sending out a signal and receiving the returning signal. For visual signals the speed of light is used to determine the distance of the object that reflects the signal. In a special time-of-flight method, the phase-shift measurement, waves of different frequency are sent out and the phase shift between the reflected signals and the currently sent signals is analyzed to calculate the distances. For measurements via triangulation transmitter and receiver are attached at a certain distance from each other. Based on the deviation at which the signal is detected using a pinhole camera, the distance of the reflecting object can be determined.

The two technologies mostly used for 3D image perception, cameras and range scanner have different characteristics. Using a camera, the image to be recorded needs to be brought into focus first. Afterwards the sensor captures light using a defined timing. The light intensities are transformed into electrical signals that are quantified by a converter and transmitted to a computer. Cameras produce detailed and colored images of the environment. The drawback is

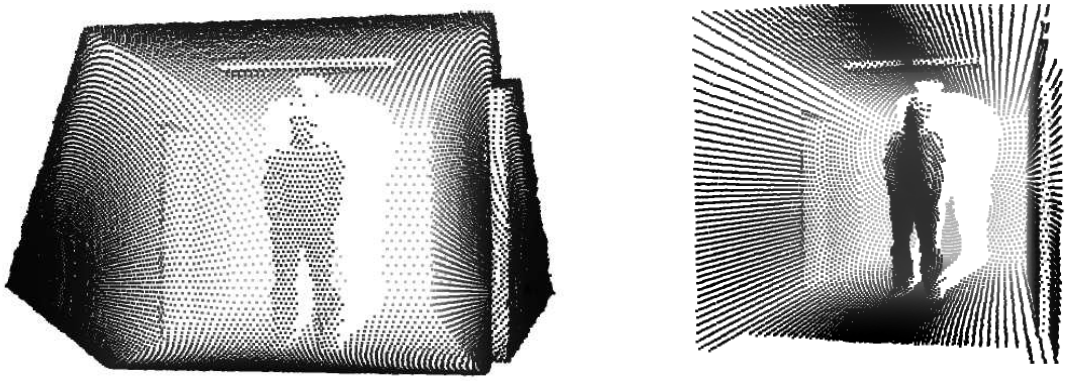


Fig. 1.2: Left: 3D point cloud scanned with a pitching scanner. Right: 3D point cloud scanned with a yawing scanner. Figures courtesy of Oliver Wulf, Leibniz Universität Hannover, Germany.

that they transform 3D environments into 2D images. Using several images, the third dimension can be reconstructed. However, the projection to 2D is not invariant to rotation and translation of the camera. Therefore, it is often not easy to find correspondences between images due to different angles, different sizes and different lighting conditions. Stereo cameras are specially designed pairs of cameras that are set up in a known relation to each other which makes it easier to find correspondences. The remaining problem is creating a larger model out of several images perceived with a stereo camera.

A laser scanner uses laser light to measure distances. Laser beams are sent out and the distances at which they are reflected by an object is determined. For a 2D laser scanner the laser beam is sent out into different directions. Rather than rotating the laser, most systems use a rotating mirror to scan a line by sending out laser beams at different angles. The apex angle, that usually varies between 90 °, 180 °, 270 ° or 360 °, is discretized with different resolutions. The scanners commonly used in robotics are safety scanners that have a large beam divergence, i.e., all objects within the apex angle are detected. The value measured can thus be the distance to the closest object, the furthest or any object in between. Due to this, the measurements are often subject to erroneous values. For 3D images there exist 3D laser scanners that are the state of the art tool in geodesic applications. In this context they are called LiDAR (Light Detection and Ranging) or LADAR (Laser Detection and Ranging). Their functionality is based on rotating a 2D scanning unit, which consists again of a rotating laser beam. Accurate 3D laser scanners are mostly very heavy and cannot be mounted on a mobile robot. Commonly used in robotics are 2D laser scanners that are mounted on a rotating unit to achieve the third dimension. Table 1.1 shows different configurations that have been used together with a list of their advantages and disadvantages. Fig. 1.2 shows two point clouds acquired using different configurations of the scanner. It becomes clear that the points are differently aligned. The pitching scanner that produced the point cloud on the left is similar to the hardware used for this thesis. The point clouds in Fig. 1.2 are nicely aligned. The movement caused by mobile robots puts enormous stress on the equipment. Over time the equipment looses its precision

Table 1.1: 3D scanner configurations using a SICK scanner. Figures courtesy of Oliver Wulf, Leibniz Universität Hannover, Germany. [66]

Mode	Symbol	continuously rotating	tilting	Advantages and Disadvantages
Yaw				+ complete 360°scans − high point densities on top and bottom + good point arrangements using a scanner with 90°apex angle
Yaw-Top				+ fast scan time, since half a rotation is sufficient + high point density in viewing direction of the scanner − only a half-space scannable
Roll				+ fast scan time, since half a rotation is sufficient + high point density in viewing direction of the scanner − only a half-space scannable
Pitch				− high point density at the sides − only small fractions of the space scannable

leading to noisy sensor data. The need to take laser scans very quickly adds to noise due to movement caused by rotating the scanner to adjust to the next scanning position.

A recent trend is the development of 3D cameras. These cameras determine based on the phase shift method the time-of-flight to detect the distance of objects. Additionally the color values of that specific point are recorded as with a normal camera. The result is a colored point cloud. At this point, the technology of 3D cameras still leaves plenty of room for improvement. The low resolution of affordable cameras make them impractical for real applications up to this point.

1.2 Problem Definition

The most precise method to create 3-dimensional maps from data acquired by a robot is scan matching. Even though laser scanners acquire more precise numerical information about the position of objects than cameras, they are still subject to several types of errors. Errors in pose estimates are as inevitable as noise in the actual laser data. Since equipment on mobile robots is easily shattered, cheap and therefore more error-prone laser scanners are typically used on

robots. This causes a high rate of noise. Systematic errors often occur as well as a result from improvised 3D systems due to the lack of affordable and light-weight 3D laser scanners. The noisy input data increases the risk of matching errors in the pose estimation algorithms. Besides that, point clouds are hard to interpret as they are. The noise further decreases the visual usability of the maps.

1.3 Scientific Contribution

Laser scan maps are a state of the art methods in robotic mapping. The quality of the resulting map is highly dependent on the quality of each laser scan. While scan matching procedures improve and the challenges of large amounts of data for 3-dimensional models are about to be overcome by technological advances, the quality of the laser scans that are produced by laser scanners that can be mounted on a mobile robot still pose a problem to many groups.

The contributions of this thesis are manifold. We evaluate existing methods for their applicability in improving the quality of laser scan maps. Thin Plates Splines have been used to smoothen noisy data in the field of geostatistics. In this thesis we apply them to deform scans of 3D environments. The Hough Transform, a common method to detect shapes in 2-dimensional data has so far attained only little attention for 3D problems. We use it to detect planes in 3D point clouds and evaluate several variations with respect to runtime and quality of the results. We pay further attention to a reasonable discretization of the 3D Hough Space. A last contribution is the combination of those methods and further considerations to improve the quality of laser scan maps.

1.4 Outline of this Thesis

Chapter 1: The first chapter gives an introduction into the subject of laser scan mapping and the arising problems.

Chapter 2: A summary on the state of the art for the problems treated in this thesis is given in this chapter. This includes rigid and non-rigid scan alignment and plane extraction.

Chapter 3: A description of pairwise and global scan matching methods is presented here.

Chapter 4: This chapter deals with Thin Plate Splines (TPS), their definition and application.

Chapter 5: The Hough Transform as a method to extract planes from 3D point clouds is evaluated in this chapter. The evaluation includes different Hough methods thoughts and considerations on the structure of the accumulator.

Chapter 6: Methods to improve the planes detected by the Hough Transform are described in this chapter.

Chapter 7: This chapter rounds up the different parts of this thesis. The plane detection, TPS, considerations on the plane model and further methods are combined into

algorithms to improve the quality of laser scan maps.

Chapter 8: Concluding statements and considerations on further work are presented in this last chapter, as well as remarks on the applicability of the evaluated methods.

Chapter 2

State of the Art

Most shape matching techniques are developed on the basis of rigid transformations. The area of non-rigid registration is largely unexplored in anything but the medical imaging community. Rigid alignment usually models the underlying problem very well so that good results are achieved for many problems. Chapter 3 gives a comprehensive overview over rigid mapping. The problem of non-rigid registration is twofold. At first, a set of correspondences between two or more data sets needs to be determined. Based on these correspondences a non-rigid deformation is computed that best aligns the data sets. However, using non-rigid transformations introduces a new problem. No matter the accuracy of the estimated correspondences, the non-rigid deformation is able to perfectly align them. Even with only a little error this may happen at the cost of deforming the data sets beyond recognition. The interplay of correspondence search and alignment estimation that algorithms such as the Iterative Closest Point rely on is lost in non-rigid registration algorithms.

Non-rigid shape matching is widespread in medical imaging. It is used for fusing multimodal images [58], i.e., patient data acquired by several types of imaging devices, comparing images of different patients [4] and reconstructing a 3D model from successive 2D images [24]. All these problems involve 2D images with only minimal warp between them. The correspondences are usually established by searching for significant features in a local search window. In case of the multimodal image fusion user input is sometimes also used to increase the registration quality. Most of these solutions are naturally highly specialized for the area of application. Allen et al. [1], [2] require an underlying model, describing a human body to register human body shapes. Models that explicitly represent underlying structures and the possible non-rigid deformation thereof are also used in other areas. Xiaoguang et al. [98] model non-rigid facial movements, e.g., expressions, to improve upon image based face recognition. Registration is done by first establishing correspondences between significant facial features such as eyes and the tip of the nose. Then the range scan of the head is registered using a smooth non-rigid deformation called the Thin Plate Spline. A non-rigid deformation algorithm is outlined in [64], where a 3D point cloud is gradually aligned to a predefined destination shape. The alignment is rigid at first but gets progressively less rigid in later stages of the algorithm. In addition to the underlying model the user controls the amount of deformation. Chui and Rangarajan [23] propose a promising point matching algorithm that is capable of aligning 2D as well as 3D point

clouds with each other. They employ a variant of the Thin Plate Spline to smoothly deform one point cloud, too. Similar to [64] registration goes from rigid to non-rigid. The central idea of the algorithm is to compute the non-rigid mapping by soft assigned correspondences. Unfortunately, it is still assumed that there is a strict one-to-one correspondence for each point in the data set. This and the fact that soft assign only allows for a small number of points makes the algorithm infeasible for our purposes.

Outside of medical imaging non-rigid alignment is very rarely investigated. Haehnel et al. [40] propose a sequential non-rigid registration algorithm. Their goal is to better cope with dynamic structures, i.e., moving objects such as people in an otherwise static environment. They modify their probabilistic ICP algorithm to also estimate the position of each measured point. As this is only possible when greatly reducing the number of points the results appear noisy. In addition, by simply randomly subsampling the scans, the occurrence of incorrect correspondences increases. This is especially true as the correspondence detection is done by simple closest point selection as in the unmodified ICP algorithm.

Williams et al. [95] describe a global registration algorithm that is similarly capable of estimating point measurements. The algorithm was developed within the background of rigid registration. It includes point estimation to compensate for noisy sensor data. However, it can be seen as a global non-rigid registration algorithm designed for low scale high frequency deformations as well.

In 2007 a dynamic scan matching algorithm was also presented by Mitra et al. [62]. Aiming to correctly register moving and deforming objects in front of a static time-of-flight camera they exploit the high frequency of scans. They assume that the object moves so slow, or conversely that the used 3D scanner is so fast that differences between sequential scans are small. Therefore, no point correspondences need to be computed. Instead they estimate the deformation of a single measured point by the local space-time neighborhood. Consequently, higher point density is not a difficulty but actually a benefit for the algorithm. On the other hand the algorithm is inapplicable to problems with large deformations and movements between individual scans.

The most prominent contributors to non-rigid alignment outside the field of medical imaging is the Stanford Graphics Laboratory in the field of cultural heritage. Their goal is to reconstruct statues from a set of high density laser scans to a very high degree of accuracy. As these types of scanners often exhibit a low frequency warp non-rigid registration algorithms were developed. The solution of Ikemoto et al. [47] to this problem is to dice each laser scan into a set of overlapping regions, which are allowed to translate and rotate relative to each other. Dicing is done after a standard global registration process has successfully aligned the scans. The new dices are then again rigidly aligned and the whole process is iterated until convergence. The rigidly transformed set of regions clearly constitutes a piecewise approximation of the non-rigid deformation. Extreme care has to be taken when and how to dice each scan so as not to destroy the internal consistency of a scan.

Brown and Rusinkiewicz developed a global non-rigid registration process for the Michelangelo Project [18], [19]. Similar to Haehnel [40] they also allow for the misplacement of points by representing each point's position relative to all other points. However, they also introduced a novel ICP variant in [19] that finds corresponding points very accurately. The proposed alignment algorithm is very time intensive as well, so that random subsampling of the points is necessary. Since they use the Thin Plate Spline to generalize the deformation of the subsampled

points onto the complete scan this is less of an issue.

Although the computation time of this algorithm is reported to be several days for some data sets, we believe this to be the most promising non-rigid alignment process. We will therefore examine this algorithm more closely and propose an improvement on it in this thesis. We also propose other non-rigid alignment algorithms that are specifically designed for structured office environments. A significant part of these algorithms is the extraction of planes from the range scans.

Compared to non-rigid registration, plane extraction is a more well researched field of study. Plane extraction, or plane fitting is the problem of modeling a given 3D point cloud as a set of planes that ideally explain every data point. The two standard solutions to this problem are the Hough Transform and the Random Sample Consensus (RANSAC) algorithm.

The RANSAC algorithm is a general algorithm that is capable of finding an accurate model for observed data that contains a large number of outliers [31]. It is a non-deterministic algorithm that iteratively generates hypothetical models of the observations until it is reasonably certain the correct model has been found. There is however a general problem with the RANSAC algorithm that also appears when extracting planes from 3D scan data. First, the RANSAC algorithm is not guaranteed to find the optimal model in a finite amount of time. In fact, it may take many iterations to find a good-enough solution. This problem can be overcome by fine-tuning the algorithm for a specific application and data set. Schnabel et al. [79] have adapted RANSAC for plane extraction and found that the algorithm performs precise and fast plane extraction, but only if the parameters have properly been fine-tuned.

The Hough Transform [45] is a method developed in 1962 to extract patterns from any type of dense data. It is mostly used in computer vision to extract lines from images. Most research on the Hough Transform has been done for the 2-dimensional case. While the algorithm is easily extendable to 3D, no exhaustive comparison has been done on the many variants of the Hough Transform and the problem of discretizing the Hough Space for plane detection has not thoroughly been examined. The Hough Transform is a transformation of points from Cartesian coordinates into the Hough Space. In the case of plane extraction, each point in the Hough Space represents one plane. The transformation of a 3D point into Hough Space yields a curve of all the planes this point lies on. In the course of this thesis several distinct variants were implemented and compared. Chapter 5 will deal more closely with the Hough Transform and detail the results in depth. We chose the Hough Transform over the RANSAC algorithm since in the general case of automatic plane extraction after scan matching no fine-tuning of the parameters is possible. Out of the various variants of the Hough Transform a few appear promising for our application. The parameters of the Hough Transform can be set dependent on the laser scanner and the size of the map and some of the variants allow for short runtimes with good results.

The so-called radon transform is interesting to point out in this context. It is closely related to the Fourier transform and is used in medical imaging to reconstruct 2-dimensional images from the raw sensor data of a CT scanner [69]. As it is a very generally stated mathematical transformation it can also be used to detect planes. Similar to the Hough Transform the data cloud is transformed into the radon domain. Then the planes associated with the highest densities are selected. This procedure was implemented by Bauer et al. [8]. The speed of the algorithm is comparable to that of the Hough Transform, however the accuracy of the detected planes is significantly worse.

There are several other plane detection algorithms. These are usually highly specialized for a specific application and are not in widespread use for miscellaneous reasons. One such algorithm was proposed by Lakämper and Latecki [55]. They use an Expectation Maximization (EM) algorithm to fit planes that are initially randomly generated. In the iterative approach the planes are split and merged until the current set of planes is determined to be accurate. Unfortunately, the algorithm is in $O(nm)$ where n is the number of points and m the number of planes in the data set.

Another algorithm is given by Wulf et al. [97]. It relies on the specific properties of a sweeping laser scanner. Each point measurement is labeled as floor, wall or ceiling according to the angles to its preceding points. While this algorithm does not compute planes, the point labeling can be an important first step in a plane detection. The point labeling is often incorrect and the algorithm depends on the assumption that the laser scanner sweeps in a preordained fashion.

Yu et al. [101] developed a clustering approach to the problem. Based on a similarity measure points are locally assigned to regions. Regions are then clustered hierarchically and refined afterwards. On a standard data set the optimization requires several hours. Additionally the algorithm tends to overfit, i.e., it finds too many planes.

Chapter 3

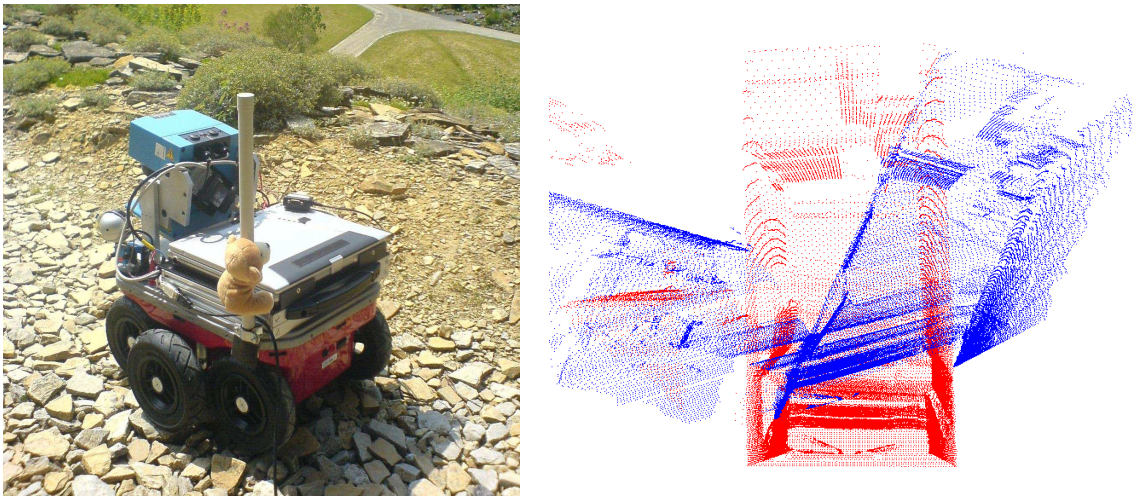
Mapping

An autonomous mobile robot whose task it is to explore an unknown environment needs to be able to map its environment. This is especially important in scenarios where rescue robots are sent out to search for victims in disaster areas or for mapping of large-scale environments. The mobile robot used in this thesis is Kurt3D Outdoor (Fig. 3.1(a)). It has a size of 45 cm (l) × 33 cm (w) × 29 cm (h) and a weight of 22.6 kg. Two 90 W motors are used to power the 6 skid-steered wheels, where the front and rear wheels have no tread pattern to enhance rotating. The core of the robot is a Pentium-Centrino-1400 with 768 MB RAM [17].

A major field in robotics research deals with *Simultaneous Localization and Mapping* (SLAM). The idea behind SLAM is that an autonomous mobile robot creates its own map while exploring an unknown environment and uses this map to localize itself. Maps can be distinguished as either topological or metrical. Topological maps are not precise as far as scale, distance and direction are concerned. Their main purpose is the representation of relationships between objects. Metrical maps on the other hand give an exact model of the environment by using quantitative measurements. The map can either be a 2-dimensional planar projection or a 3-dimensional model of the environment. The state of the art procedure for creating a map is by collecting range scan images at different locations and combining them using scan matching [76]. The general idea behind scan matching is to combine a set of scans together in a way that they yield a correct and connected model of the environment.

A laser range scanner functions by sending out laser beams in different directions. Once a laser beam hits an obstacle it is reflected. Depending on the time needed before the laser beams return to the scanner the distance of the object is determined to a relatively exact extent. Taking into account the angle of the laser beam the position of the object in the local coordinate system of the laser scanner is calculated. The acquired distance data yields a point cloud that is an image of the environment of the robot (cf. Fig. 3.1(b)). In the following a set of distance values taken from one robot pose will be referred to as laser scan or laser range image.

Commonly used in research are SICK laser scanners. Used for this thesis is the SICK-LMS200. As for many other laser scanners, the laser beam is manipulable by a rotating mirror. Thus an opening angle of 180° with a resolution between 1.0° and 0.25° is achieved. The systematic error is denoted with 15 mm and the statistical error with 5 mm [81]. For collecting 3D laser scans the laser scanner is fixed on a mount that can be rotated around the horizontal



(a) Kurt3D (b) Laser Scans

Fig. 3.1: (a) The mobile robot Kurt3D. (b) Point clouds of two 3D laser scans that have a large overlap but are badly misaligned.

axis with a servo motor to achieve a vertical opening angle close to 90 °.

The structure of the 3D point clouds generated from a laser scanner resembles the structure of the mapped environment. Scan matching aims at recognizing similar structures in partly overlapping scans and aligning these scans according to the structures. Starting from an initial estimate most algorithms iteratively try to find a local minimum. One of the best-known scan matching algorithms is the ICP algorithm (*iterative closest point*) [11]. Point pairs are detected in two overlapping scans and the distances between the corresponding pairs are minimized. Other algorithms minimize the distances between previously extracted lines or planes.

3.1 Dimensionality of Mapping

One way to categorize maps is by their dimensionality [16]. For robotic mapping the dimensionality of the data, which can be 2D or 3D as well as the degrees of freedom (DoF) of the robot pose need to be considered. A 3D pose estimate consists of the robot coordinates (x, y) and the orientation angle α. 6D poses contain the position in x, y, z-coordinates and the combined rotation of the robot as roll, pitch and yaw angle.

2D maps have for a long time been the state of the art in robotic mapping due to the lower hardware costs and the lower computation times. 3D maps can again be divided into three major groups. The 3D methods differ in dimensionality of the sensor data as well as the localization. Full 6D mapping is essential in unstructured outdoor environments. In this scenario a map can only be created by means of 3D sensor data and precise 6D pose estimates. Furthermore, in some cases planar localization methods are combined with a 3D sensor or 6D localization is combined with 2D sensor data. The different types of mapping are listed in Table 3.1. 2D methods are marked in gray whereas 3D methods are marked in black.

Table 3.1: Overview of the dimensionality of SLAM approaches. Grey: 2D maps. Black: 3D maps. (taken from [16])

		Dimensionality of pose representation	
		3D	6D
Sensor data	2D	**Planar 2D mapping (3.1.1)** 2D mapping of planar sonar and laser scans.	**Slice-wise 6D SLAM (3.1.3)** 3D mapping using a precise localization, considering the x,y,z-position and the roll yaw and pitch angle.
	3D	**Planar 3D mapping (3.1.2)** 3D mapping using a planar localization method and, e.g., an upward looking laser scanner or 3D scanner.	**Full 6D SLAM (3.1.4)** 3D mapping using 3D laser scanners or (stereo) cameras with pose estimates calculated from the sensor data.

3.1.1 Planar 2D Mapping

2D mapping methods have been subject to thorough research in the past. Due to the lack of precise and affordable 3D sensors 2D metric maps were the de facto standard in SLAM. State of the art are probabilistic methods. Probabilistic robot motion and uncertain perception models are integrated with a Bayes filter to localize the robot. By extending this estimation problem mapping can be done in a similar manner. Improvements are achieved by including landmarks in the mapping procedure or by detection of closed loops. The information that a visited area has been seen before helps to modify the already mapped areas in a way that a topologically consistent map is achieved. In [86] Thrun presents an overview on SLAM methods. Frese uses maximum likelihood estimation [34], in [32] and [87] expectation maximization is used. The extended Kalman filter is presented in [27] and the (sparse extended) information filter in [89]. For FastSLAM [88] the robot pose is considered as posterior probabilities and approximated by particles. Lu and Milios present a global approach to scan matching that is based on the ICP algorithm [57].

3.1.2 Planar 3D Mapping

As mentioned before the availability of precise and affordable hardware to acquire 3D sensor data is limited. For this reason some groups have devised a workaround to build 3D volumetric models of environments. Instead of using 3D laser range finders, which yield consistent 3D scans they use transformed 2D scanners and build representations of the environment by combining the data.

Thrun et al. [88], Früh et al. [36] and Zhao et al. [103] use one horizontally mounted and one vertically mounted 2D laser scanner for acquiring 3D data. The vertical scan line acquired by the vertical scanner is transformed into 3D points based on the current 3D robot pose. The pose is calculated using the data from the horizontal scanner. It is obvious, that the precision of the 3D scan is highly dependent on the precision of the pose estimate. For better results Zhao et al. use two additional laser scanners. They are mounted in a 45 ° angle to the vertical scanner

in order to scan sides of objects which will be occluded in the data from the vertical scanner.

Recent approaches tend towards 3D laser scans with higher resolution acquired from a single laser scanner. For this purpose the laser scanner is mounted on a rotatable unit [54, 85, 97].

3.1.3 Slice-wise 6D SLAM.

A fast method for building 3D maps for mobile robot navigation is to use 6D pose estimates and several translated 2D laser scanners. The resulting maps lack a complete 3D model but suffice for navigation as demonstrated by the Stanford racing team [90]. They used this technique for high speed terrain classification when competing in the grand challenge.

3.1.4 Full 6D SLAM.

Recently highly accurate 3D laser scanners have been applied for mapping purposes. Since the 3D laser scanners are rather immobile only few groups use them for robotic mapping [3,39,80]. In the RESOLV project robots equipped with a RIEGL laser range finder are used [80]. The scans are combined via the ICP algorithm [11]. Goal of the project is to model interiors for virtual reality and tele-presence. Urban environments are mapped in the AVENUE project [3]. The developed robot is equipped with a CYRAX scanner and uses a feature-based scan matching approach for registering the 3D scans. However, in their recent work the laser data is not used for localization [39]. The Zoeller+Fröhlich 3D laser range scanner is used in M. Hebert's group [41]. They aim at reconstructing environments without odometry information as initial pose estimates. A scan matching method without the ICP algorithm was proposed by Magnusson and Duckett [59]. They create 3D models with the normal distribution transform.

The probabilistic methods proposed for planar 2D mapping (see Section 3.1.1) can be extended to be used for full 6D SLAM. However, the extension from 2D points to 3D points and from 3D pose estimates to 6D pose estimates bring along higher computational costs. Since the costs grow exponentially with the DoF the methods need to be adapted to handle the larger amounts of data.

Several groups have applied probabilistic methods to 3D scan matching, including a probabilistic notion of ICP [51], extended Kalman filter [25,94] and treemap SLAM [33]. Scan matching methods can be separated into pairwise and global mapping. In this thesis scan matching is a combination of the pairwise ICP algorithm [11] and the global minimization method proposed by Lu and Milios [57]. The interaction of these methods in the 3D case has been presented in [16]. In the following sections the two parts of the algorithm will be described in detail.

3.2 Pairwise Scan Matching

In 1992 Besl and McKay proposed the Iterative Closest Point algorithm. Since then it has become one of the standard methods for registration of 3D laser scans. The idea behind ICP is to find for each point from a set the closest point from a corresponding structure. This way a transformation is computed that moves the second figure such as needed to minimize the distance between the corresponding point pairs. This iterative algorithm converges quickly to a local minimum and does not require preprocessing of the data. A small number of iterations

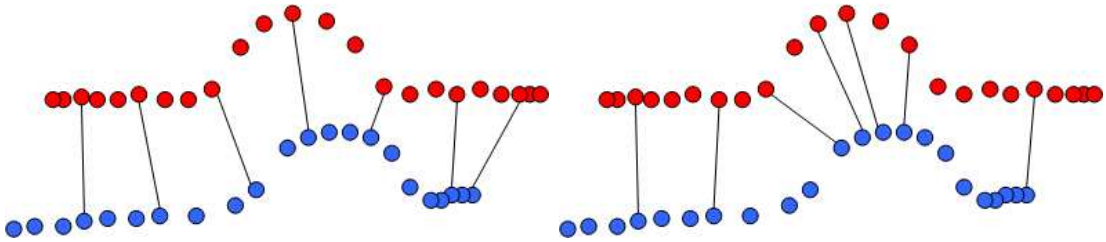


Fig. 3.2: Two ways of matching points. Left: Uniform sampling of points from the red (top) data set. Right: Selection of points from the distinctive areas of the data set. Selecting points from the bump minimizes the risk that the ICP does not converge.

already leads to good matching results. The handled data can be either point sets, polylines, curves or planes. No prior feature extraction, or calculation of derivatives is necessary because the algorithm works directly on the points of the objects. Planes, polylines and curves are usually sampled through points. A remaining problem is that the algorithm often converges only to a local minimum. This may lead to erroneous registration.

In the following the ICP is explained with respect to processing of laser range data since this is the scope of this thesis. The fundamental procedure consists of six steps:

1. Choose n points from a laser range image.
2. Search for the closest corresponding point in an overlapping scan.
3. Weight the correspondences with respect to their confidence.
4. Ignore all point pairs whose distance strongly differs from the mean distance.
5. Calculate the error function for all point pairs.
6. Minimize the error function.

There exist numerous variants of the ICP algorithm that aim at optimizing one of these steps for decreasing the computation time, increasing the tolerance towards sensor noise or outliers or for allowing worse estimates of the starting pose [76].

3.2.1 Point Selection

A laser scan consists of a huge amount of points. Consideration of all these points for scan matching leads to an enormous computational effort. To quicken the time needed for scan matching the number of points in the set needs to be reduced. Among the possible approaches are those that take a random or uniform sample of points from each scan. However, improper selection leads to undesirable effects.

Fig. 3.2 shows two point sets and two ways of selecting points from these sets for scan matching. The left hand side shows uniform sampling for the red (top) points, e.g., every third point is selected. The corresponding blue points are chosen with respect to the distance. The most significant structure in these point sets is the dent in the middle. From this part only

a single point is chosen by uniform sampling. This selection procedure might result in slow convergence or faulty matching of the scan pair. It seems more reasonable to omit a few points from the straight line in favor of more points from the dent as shown on the right hand side. This is especially useful if the characteristic parts of a laser scan are a long distance away from the origin of the laser scanner since with growing distance the point density diminishes. Elaborate selection of points may also help in case of erroneous pose estimates or noisy scan data.

A method to quicken the convergence by smart selection of points was proposed by Gelfand et al. [38]. If too many points are selected from unstable regions without significant properties ICP tends to slide the two scans around without reaching the correct pose or even without convergence. The reason for this are the continuous low error values in these regions. The solution presented by Gelfand et al. is to try to select as many points as possible from stable regions. For this purpose they determine the covariances of a small subset of points from the input data set. On the basis of the covariances the stability of a region is determined aiming at getting stable correspondences that consider all degrees of freedom.

3.2.2 Finding Corresponding Point Pairs

The most important step of the ICP algorithm is finding corresponding point pairs. Due to the large amount of points this is also the computationally most expensive part. The straightforward approach starts with one point from the first scan. Under consideration of the pose estimate the distance to each point from the second scan is calculated and the point with the shortest distance is chosen.

The computation time for the intuitive method is in $O(n^2)$ for n scan points. This effort can be reduced by use of special data structures. Each point is an entry in the data structure defined by its keys, e.g., the x-, y- and z-coordinates. This management helps to make the search for point pairs more efficient.

Dividing the key space into small equal sized cells a spiraliform search for the next cell with an entry yields the closest point. This search touches less points but has still high computational requirements concerning time and space. This is especially the case when dealing with key sets of high dimension. Cluster techniques can be used to further diminish the number of regarded entries. A second possibility is to organize the keys in linear lists and to search only through those points whose keys lie within an interval around the keys of the given point. The computation time for n points depends on the strategy for searching through the list. Friedman et al. state that it is proportional to $3n^{-\frac{2}{3}}$ for 3-dimensional points [35].

Dynamic Data Structures for Storing Points

To allow fast access of points they can be stored in various forms of dynamic data structures. Typical dynamic data structures hold the design of a tree since balanced trees drastically reduce the computational times for searching in the data set. So instead of storing all points in a linear list points are ordered according to their coordinates. The dynamic tree in Fig. 3.3 was presented by Xu et al. in [99]. It stores 2-dimensional points but can easily be extended to 3D. Each node of the tree corresponds to a point (a_1, a_2) and stores the coordinates of the point. The nodes are arranged according to the coordinates a_1 and a_2. The root node is the one with

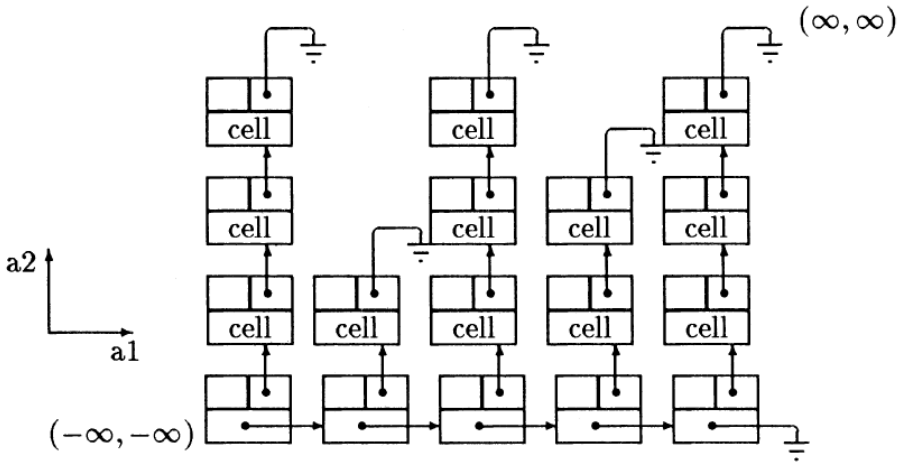


Fig. 3.3: An example of the 2-dimensional dynamic data structure used in [99]. Each cell stores a point $p = (a_1, a_2)$ in cartesian coordinates. The cells are arranged such as they are ordered with respect to the values of a_1 and a_2.

the smallest values for both a_1 and a_2, respectively. In Fig. 3.3 the root is located in the lower left corner. To the right the a_1 values of the nodes become larger. The bottom row contains cells that do not store a point but are only markers for the a_1 values. Towards the top the a_2 values increase. This means all nodes that are depicted in one column have the same value for a_1 while the values for a_1 increase in ascending order. Adding a further dimension a_3 to the data structure is achieved by adding branches to the according cells. All cells depicted in the image so far become marker cells and points are added as children of the cells with the according a_1 and a_2 values. a_3 then determines their position in the branch. If a point is added to the tree its position in the tree is looked up, first depending on a_1, then depending on a_2 and a new node is inserted at the appropriate position. For representing laser scan points this data structure has one major drawback. The points in the first dimension (a_1) need to be discretized in order to achieve an advantage in the data structure. Data structures that deal with this problem are quadtrees or octrees [35]. In these data structures the parameter space is divided into regular cells to simplify storage and search for points.

A quadtree is a data structure that allows for insertion in logarithmic time [30]. 2-dimensional data is stored in a tree in which each node has up to four sons. Each node contains a single point. The root node divides the parameter space into four quadrants by lines parallel to the coordinate axes and through the point that is stored in the node. The subspaces are numbered counter-clockwise starting in the upper left corner. All points in the first quadrant are stored in the first subtree of the root node, and the other quadrants in the following subtrees, respectively. Points that lie directly on the separating lines are assigned to quadrants two and four only. Each node divides its quadrant further into four new spaces. For inserting a point into the tree, the tree is descended starting at the root. At each node, the path is chosen that corresponds to the quadrant the point lies in. If the quadrant does not contain any points, e.g., the current node has no son at this position, a new node is created containing the point to be inserted and is

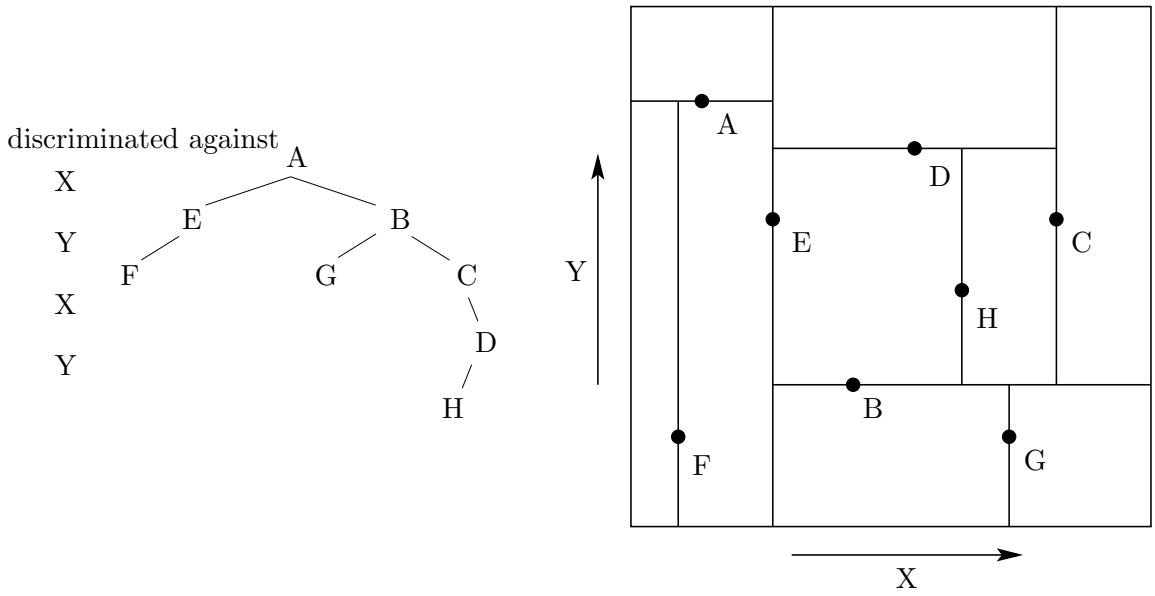


Fig. 3.4: Example of a 2D-tree. Right: The points in a 2D space. Left: The tree structure corresponding to the points on the right.

added at the position determined by the descent. Adaptions of the insertion rule can help to attain a better balancing of the tree. Searching in a quadtree can be divided into two classes, searching for a single point $\boldsymbol{p}$, i.e., the point that is closest to another point $\boldsymbol{q}$ or searching for several points within an area. The search for a single point in a quadtree T is performed much like inserting into the quadtree. When descending into the tree all quadrants have to be searched whose minimum distance to $\boldsymbol{p}$ is smaller than the maximal distance of the quadrant closest to the query point $\boldsymbol{p}$. The remaining points have to be compared with respect to their distance to $\boldsymbol{q}$. In a region search all subtrees of T are recursively searched whose bounds overlap with a circle of a predefined radius around $\boldsymbol{p}$ and the resulting points are returned.

For 3-dimensional data the quadtree is extended into an octree. Instead of letting every point divide the space into quadrants the division in an octree is in octants. Several variants exist. As for the quadtree, every inserted point can serve as a divider. Another typical method is to split the space into eight regular cubes. Both methods lead to a different efficiency depending on the alignment of the points.

Based on quad- and octrees Bentley et al. designed a k-dimensional tree, the kd-tree [10]. While each node in a quadtree has up to four children nodes and an octree has up to eight children nodes, respectively, a kd-tree is a binary tree. It serves for storage of objects with k different keys. Each node stores one object. The tree is structured in a way that each level alternately discriminates against one of the k keys. In the left subtree of a node $\boldsymbol{p}$ in level i only those objects are stored that are smaller than $\boldsymbol{p}$ with respect to the characteristic $i \mod k$. In the right subtree are those objects, that are larger than $\boldsymbol{p}$, respectively.

Fig. 3.4 depicts an example for a 2D-tree. The kd-tree on the left corresponds to the points arranged in a 2-dimensional plane on the right. The search for a corresponding point to a point

$\boldsymbol{p}$ can be accelerated by performing a recursive tree-search [35] as outlined in algorithm 1.

Algorithm 1 Search for closest point in a kd-tree

Input: tree b, point $\boldsymbol{p}$
Output: closest point $\boldsymbol{n}$

```
if b is leaf node then
  return point in b
end if
if p belongs into the left subtree then
  point q1 = search( leftSon, p )
  if distance( p, q1 ) < distance( p, rightSon ) then
    point q2 = search( rightSubtree, p )
  end if
  if distance ( p1, q1 ) < distance( p, q2 ) then
    return q1
  else
    return q2
  end if
else
  point q1 = search( rightSon, p )
  if distance( p, q1 ) < distance( p, leftSon ) then
    point q2 = search( leftSubtree, p )
  end if
  if distance ( p1, q1) < distance( p, q2 ) then
    return q1
  else
    return q2
  end if
end if
```

Each recursion steps one level deeper into the tree and reduces the set of possible points. Along the way the search interval is also made smaller. The active node is regarded and it is decided whether the point in question would be in the left or the right subtree. The value of the discriminating key bounds the new interval. This way the search space is continuously downsized. If a leave node is reached the point $\boldsymbol{q}$ in this entry is close to $\boldsymbol{p}$. To decide whether $\boldsymbol{q}$ is the wanted point the algorithm ascends in the tree. At each level it is checked whether a ball with the distance of $\boldsymbol{p}$ and $\boldsymbol{q}$ as its radius overlaps with the boundaries of the interval represented by the subtree in which $\boldsymbol{q}$ does not lie. If this is the case a recursive search in the relevant tree is performed. Otherwise the ascent continues until the root node is reached. The computation time for the search algorithm depends on the number of nodes, the dimension of the points and on the structure of the kd-tree.

The more balanced the tree is and the more evenly the discrimination is the faster a search through the tree can be performed. Friedman et al. describe an optimization of the kd-tree search that is performed in $O(\log n)$ [35]. When inserting a point into the kd-tree the discrim-

inator is chosen depending on the distribution of the key values. For each node the key with the widest spread of values is the discriminator. The median of the key values becomes the boundary. This strategy reduces the depth of the tree and the probability that when trying to find the closest point the actually closest point lies in a different subtree. Aside from the computational time for the search the construction of the tree, which is in $O(kn \log n)$, also adds to the performance of finding point pairs.

Not always is the closest point also the best point. Several groups evaluate point pairs according to compatibility. Next to distance, color, direction of normal vectors or features like curvature and derivatives are taken into account [76]. Since this requires the existence of features it is not applicable for laser scans without pre-processing.

3.2.3 Elimination of Outliers

Sampling of points from the laser scan might lead to loss of robustness of scan matching due to the growing weight of selected outliers. Accordingly, for improving the quality of the ICP algorithm the elimination of outliers is essential. Chetverikov et al. proposed for this purpose the Trimmed ICP (TrICP) [22]. Based on the Least Trimmed Squares Procedure (LTS) [75] outliers are detected and eliminated. Opposite to the classical distance calculation the LTS procedure still works with a large number of outliers as they often appear when using laser scans taken at different positions. This is due to a tolerance threshold that is independent of outliers.

To transform point set D according to the corresponding point set M for each point from D the closest corresponding point in M is determined. The distance between a point $\boldsymbol{d}_i$ in D and its corresponding point in M is defined as

$$\mathrm{dist}_i(\boldsymbol{R}, \boldsymbol{t}) = \left\| \arg\min_{\boldsymbol{m} \in M} \|\boldsymbol{m} - \boldsymbol{R} \cdot \boldsymbol{d}_i + t\| - \boldsymbol{R} \cdot \boldsymbol{d}_i + t \right\|$$

with the rotation $\boldsymbol{R}$ and the translation $\boldsymbol{t}$ that are necessary to transform D onto M.

Starting from a transformation $(\boldsymbol{R}, \boldsymbol{t})$ the distances dist_i are squared and sorted in descending order. Then the N_{do} pairs with the shortest distances are chosen, with N_{do} the number of points from the area where M and D overlap. N_{do} is assumed to be known but can be determined using the TrICP algorithm.

If no termination rule applies S_{LTS}, which was initially set to a very high threshold, is set to the sum S'_{LTS} of the N_{po} selected distances dist_i^2. The optimal transformation $(\boldsymbol{R}, \boldsymbol{t})$ that minimizes S_{LTS} is computed and applied to D. This procedure is repeated until the maximal number of iterations is reached or until S'_{LTS}/N_{do} is sufficiently small or stable.

In an experimental evaluation Chetverikov et al. show that their approach leads to improvements even for bad initial pose estimates, e.g., relative rotations of up to 30 °. The reason for the improvement is that points outside of the overlapping region do not influence the computation of the transformation.

3.2.4 Confidence Weighting of Correspondences

When handling point clouds with small overlap or low density sampling and elimination of outliers easily results in subsets that are too small to work with. An alternative to completely

ignoring points is to weight point correspondences with respect to their confidence. Each point pair $(\boldsymbol{d}_i, \boldsymbol{m}_j)$ gets assigned a weight $w_{i,j}$ in the error function.

In the unweighted case $w_{i,j}$ is set to 1 if $\boldsymbol{m}_j$ from M is a corresponding point to $\boldsymbol{d}_i$ from D, otherwise $w_{i,j}$ is set to 0. There are different ways to weight point pairs according to their distance.

$$w_{i,j} = 1 - \frac{\text{dist}(\boldsymbol{d}_i, \boldsymbol{m}_j)}{\text{dist}_{\max}}$$

assigns lower weights to point pairs with larger distances. The variance from the mean of all distances is weighted with

$$w_{i,j} = 1 - \frac{|\text{dist}(\boldsymbol{d}_i, \boldsymbol{m}_j) - \text{dist}_{\text{median}}|}{\text{dist}_{\max}}.$$

Given normal vectors for each point confidence can be calculated dependent on the compatibility of normals:

$$w_{i,j} = n_{\boldsymbol{d}_i} \cdot n_{\boldsymbol{m}_j}.$$

3.2.5 Minimization of the Error Function

The elementary step of the ICP algorithm is the minimization of the error function. The two mostly used error functions for the ICP are the point-to-point error function [68] and the point-to-plane error function as implemented by Chen and Medioni [21]. Since the former is used in the experiments described in this thesis it is explained in more detail. Wanted is the rotation $\boldsymbol{R}$ and the translation $\boldsymbol{t}$ that applied to the point set D minimizes the error function

$$E(\boldsymbol{R}, \boldsymbol{t}) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t})\|^2 . \tag{3.1}$$

Error function (3.1) sums up the distances between all corresponding point pairs $p_i = (m_i, d_i)$ with $d_i \in D$ and $m_i \in M$ after applying transformation $(\boldsymbol{R}, \boldsymbol{t})$ to D.

Closed Form Solutions for Point-to-Point Error Metric

There are currently four algorithms known to solve the ICP error function in closed form [67]. The error function consists of the rotation and the translation needed to transform a point set D onto a second point set M. Since for the rotation matrix orthonormality has to be ensured three of the algorithms compute the rotation matrix $\boldsymbol{R}$ and the translation $\boldsymbol{t}$ separately.

For calculating the rotation the mean of the point sets is first determined by

$$\boldsymbol{c}_m = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{m}_i, \qquad \boldsymbol{c}_d = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{d}_i. \tag{3.2}$$

Subtracting the mean from all points leads to two new sets that are arranged around the origin of the coordinate system

$$M' = \{\boldsymbol{m}'_i = \boldsymbol{m}_i - \boldsymbol{c}_m\}_{1,\ldots,N}, \qquad D' = \{\boldsymbol{d}'_i \ = \boldsymbol{d}_i \ - \boldsymbol{c}_d\}_{1,\ldots,N}. \tag{3.3}$$

Inserting replacement (3.2) and (3.3) into the error function (3.1) yields

$$E(\boldsymbol{R},\boldsymbol{t}) = \frac{1}{N}\sum_{i=1}^{N}||\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i - \underbrace{(\boldsymbol{t} - \boldsymbol{c}_m + \boldsymbol{R}\boldsymbol{c}_d)}_{=\tilde{\boldsymbol{t}}}||^2 \tag{3.4}$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left|\left|\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i\right|\right|^2 - \frac{2}{N}\tilde{\boldsymbol{t}}\cdot\sum_{i=1}^{N}\left(\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i\right) + \frac{1}{N}\sum_{i=1}^{N}\left|\left|\tilde{\boldsymbol{t}}\right|\right|^2.$$

The error function can be reduced to the term consisting of the points and the rotation

$$E(\boldsymbol{R},\boldsymbol{t}) \propto \sum_{i=1}^{N}\left|\left|\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i\right|\right|^2. \tag{3.5}$$

This is due to the fact that all sums in Eq. (3.4) have to be minimized to minimize the entire function. In the second term all values refer to the centroid, thus it adds up to zero. The last term can be put down to rotation of the centroids

$$\boldsymbol{t} = \boldsymbol{c}_m - \boldsymbol{R}\boldsymbol{c}_d$$

or has a minimum for $\tilde{\boldsymbol{t}} = \boldsymbol{0}$. Consequently Eq. (3.5) remains to be minimized with one of the following methods.

Single Value Decomposition Arun, Huang and Blostein developed a method using single value decomposition [5]. The optimal orthonormal 3×3 rotation matrix $\boldsymbol{R}$ is calculated as $\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T$ derived from

$$\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T = \boldsymbol{H} = \sum_{i=1}^{N}\boldsymbol{m}_i'^T\boldsymbol{d}'_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}, \tag{3.6}$$

with $S_{xx} = \sum_{i=1}^{N} m'_{x,i}d'_{x,i}$, $S_{xy} = \sum_{i=1}^{N} m'_{x,i}d'_{y,i}$, ...

Polar Decomposition Developed by Horn, Hilden and Negahdaripour was a similar method based on polar decomposition [44]. Polar decomposition requires definition of the square root of a matrix as the optimal rotation is given by

$$\boldsymbol{R} = \boldsymbol{P} = \boldsymbol{H}\left(\frac{1}{\sqrt{\lambda_1}}\boldsymbol{u}_1\boldsymbol{u}_1^T + \frac{1}{\sqrt{\lambda_2}}\boldsymbol{u}_2\boldsymbol{u}_2^T + \frac{1}{\sqrt{\lambda_3}}\boldsymbol{u}_3\boldsymbol{u}_3^T\right),$$

calculated from the polar decomposition defined as $\boldsymbol{H} = \boldsymbol{P}\boldsymbol{S}$ where $\boldsymbol{S} = (\boldsymbol{H}^T\boldsymbol{H})^{1/2}$. $\boldsymbol{H}$ is the same as in Eq. (3.6), $\{\lambda_i\}$ are the eigenvalues and $\{\boldsymbol{u}_i\}$ the corresponding eigenvectors of the matrix $\boldsymbol{H}^T\boldsymbol{H}$ [44].

Quaternions Another way to define rotations is by means of quaternions [43]. Horn invented rotations using unit quaternions $\dot{\boldsymbol{q}}$. The largest eigenvalue of cross covariance matrix $\boldsymbol{N}$

$$\begin{pmatrix} (S_{xx}+S_{yy}+S_{zz}) & (S_{yz}+S_{zy}) & (S_{zx}+S_{xz}) & (S_{xy}+S_{yx}) \\ (S_{yz}+S_{zy}) & (S_{xx}-S_{yy}-S_{zz}) & (S_{xy}+S_{yx}) & (S_{zx}+S_{xz}) \\ (S_{zx}+S_{xz}) & (S_{xy}+S_{yx}) & (-S_{xx}+S_{yy}-S_{zz}) & (S_{yz}+S_{zy}) \\ (S_{xy}+S_{yx}) & (S_{yz}+S_{zy}) & (S_{zx}+S_{xz}) & (-S_{xx}-S_{yy}+S_{zz}) \end{pmatrix}$$

corresponds to the unit quaternion that represents the rotation that minimizes the point-to-point error function.

Dual Quaternions Walker, Shao and Volz minimize Eq. (3.1) [92] using dual quaternions. The optimal transformation is the solution of the eigenvalue problem of a 4×4 matrix function that is built from corresponding point pairs. Translation and rotation are calculated simultaneously in a single step without previously translating the centroids onto the origin.

Linearized Solutions

Helix Transform Under the assumption that the transformation $(\boldsymbol{R}, \boldsymbol{t})$ calculated by the ICP algorithm is small Hofer and Pottmann approximate the solution by applying instantaneous kinematics [42,71]. The so-called helical motion is used to compute the displacement of a 3D point by an affine transformation. To map the point set D a small displacement $\boldsymbol{v}(\boldsymbol{d}_i) = \bar{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{d}_i$ has to be added to all points in D leading to the adapted error function

$$E(\bar{\boldsymbol{c}}, \boldsymbol{c}) = \sum_{i=1}^{N} ||\boldsymbol{m}_i - (\boldsymbol{d}_i + \boldsymbol{v}(\boldsymbol{d}_i))||^2 = \sum_{i=1}^{N} ||\boldsymbol{m}_i - (\boldsymbol{d}_i + \bar{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{d}_i)||^2 . \tag{3.7}$$

Minimizing $|E(\bar{\boldsymbol{c}}, \boldsymbol{c})|$ requires finding the optimal displacement given by the unknown variables $\bar{\boldsymbol{c}}, \boldsymbol{c}$. Since Eq. (3.7) is a quadratic function the solution corresponds to the solution of the linear equation system

$$\sum_{i=1}^{N} \boldsymbol{D}_i^T \boldsymbol{D}_i \boldsymbol{u} = \sum_{i=1}^{N} \boldsymbol{D}_i^T (\boldsymbol{m}_i - \boldsymbol{d}_i) \tag{3.8}$$

using the approximation $\bar{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{d}_i = \boldsymbol{D}_i \cdot \boldsymbol{u}$ with $\boldsymbol{u} = (\boldsymbol{c}^T, \bar{\boldsymbol{c}}^T)^T$ and

$$\boldsymbol{D}_i = \begin{pmatrix} 0 & d_{z,i} & -d_{y,i} & 1 & 0 & 0 \\ -d_{z,i} & 0 & d_{x,i} & 0 & 1 & 0 \\ d_{y,i} & -d_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix} .$$

The optimal displacement calculated in Eq. (3.8) corresponds to an affine movement as displayed in Fig. 3.5. The rigid transformation $(\boldsymbol{R}, \boldsymbol{t})$ is computed from $(\bar{\boldsymbol{c}}, \boldsymbol{c})$ in a post processing step. If $\boldsymbol{c} = \boldsymbol{0}$ $\boldsymbol{t} = \bar{\boldsymbol{c}}$ holds since no rotation is present. The Plücker coordinates,

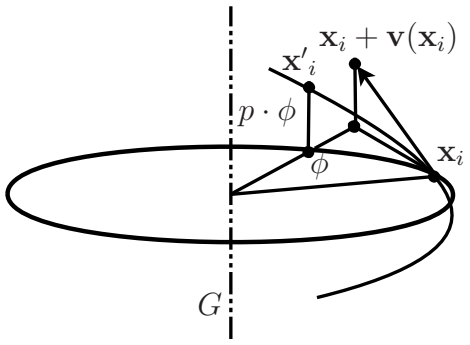


Fig. 3.5: The affine position of a 3D point $\boldsymbol{x}_i + \boldsymbol{v}(\boldsymbol{x}_i)$ is different from the rigid transformation that results in point $\boldsymbol{x}'_i$. Based on [42].

i.e., the direction vector ($\boldsymbol{g}$) and the momentum vector ($\bar{\boldsymbol{g}}$) of an rotation axis G can be computed:

$$\boldsymbol{g} = \frac{\boldsymbol{c}}{||\boldsymbol{c}||}, \quad \bar{\boldsymbol{g}} = \frac{\boldsymbol{c} - p\boldsymbol{c}}{||\boldsymbol{c}||}, \quad p = \frac{\boldsymbol{c}^T \cdot \bar{\boldsymbol{c}}}{\boldsymbol{c}^2}.$$

The transformed points d'_i are computed as follows:

$$\boldsymbol{d}'_i = \boldsymbol{R}(\boldsymbol{d}_i - \boldsymbol{p}) + (p\ \phi)\boldsymbol{g} + \boldsymbol{p}$$

The rotation matrix

$$\boldsymbol{R} = \frac{1}{b_0^2 + b_1^2 + b_2^2 + b_3^2} \begin{pmatrix} b_0^2 + b_1^2 - b_2^2 - b_3^2 & 2(b_1 b_2 + b_0 b_3) & 2(b_1 b_3 - b_0 b_2) \\ 2(b_1 b_2 - b_0 b_3) & b_0^2 - b_1^2 + b_2^2 - b_3^2 & 2(b_2 b_3 + b_0 b_1) \\ 2(b_1 b_3 + b_0 b_2) & 2(b_2 b_3 - b_0 b_1) & b_0^2 - b_1^2 - b_2^2 + b_3^2 \end{pmatrix},$$

is constructed with $b_0 = \cos(\phi/2)$, $b_1 = g_x \sin(\phi/2)$, $b_2 = g_y \sin(\phi/2)$, and $b_3 = g_z \sin(\phi/2)$, where g_x, g_y, g_z are the coordinates of the direction vector of axis G. $\boldsymbol{p}$ is an arbitrary point on the axis G (cf. [42]).

Small Angle Approximation Rotating a point in 3D with the Euler angles θ_x around the x-axis, θ_y around the y-axis and θ_z around the z-axis is achieved by multiplying matrix

$$\boldsymbol{R} = \begin{pmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ \sin\theta_x \sin\theta_z - \cos\theta_x \cos\theta_z \sin\theta_y & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{pmatrix}$$

with the point. For small angles R can be approximated by use of the Taylor series [67] with

$$\sin\theta \approx \theta - \frac{\theta^3}{3} + \frac{\theta^5}{5} - \cdots$$
$$\cos\theta \approx 1 - \frac{\theta^2}{2} + \frac{\theta^4}{4} - \cdots$$

resulting in

$$\boldsymbol{R} \approx \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_x\theta_y + \theta_z & 1 - \theta_x\theta_y\theta_z & -\theta_x \\ \theta_x\theta_z - \theta_y & \theta_x + \theta_y\theta_z & 1 \end{pmatrix}. \tag{3.9}$$

Under assumption that multiplication of small angles results in even smaller angles that can be neglected Eq. (3.9) further simplifies to

$$\boldsymbol{R} \approx \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{pmatrix} = \boldsymbol{I}_{3\times 3} + \begin{pmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{pmatrix}.$$

This matrix corresponds to a non-orthonormal version of the Rodrigues formula. After insertion of the approximation the ICP error function can be rearranged such as separating the unknown values into a vector:

$$\begin{aligned} \boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t}) &\approx \boldsymbol{m}_i - \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{pmatrix} \boldsymbol{d}_i - \boldsymbol{t} \\ &= \boldsymbol{m}_i - \boldsymbol{d}_i - \begin{pmatrix} 0 & d_{z,i} & -d_{y,i} & 1 & 0 & 0 \\ -d_{z,i} & 0 & d_{x,i} & 0 & 1 & 0 \\ d_{y,i} & -d_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_z \\ \theta_y \\ \theta_x \\ t_x \\ t_y \\ t_z \end{pmatrix}. \end{aligned} \tag{3.10}$$

The resulting equation has to be interpreted as a so-called helical motion as introduced by Hofer and Pottmann [42, 71] with the top part of the vector being $\boldsymbol{c}$ and the bottom part $\bar{\boldsymbol{c}}$, respectively. The helical motion rotates around a previously calculated rotation axis while Euler angles represent a rotation around the origin. For applying the small angle approximation a separation of rotation and translation using the centroids of the point clouds is once again necessary:

$$\boldsymbol{m}'_i - \boldsymbol{R}\boldsymbol{d}'_i \approx \boldsymbol{m}'_i - \boldsymbol{d}'_i - \begin{pmatrix} 0 & d'_{z,i} & -d'_{y,i} \\ -d'_{z,i} & 0 & d'_{x,i} \\ d'_{y,i} & -d'_{x,i} & 0 \end{pmatrix} \begin{pmatrix} \theta_z \\ \theta_y \\ \theta_x \end{pmatrix}.$$

The optimal displacement for the ICP error function

$$\boldsymbol{D}_i = \begin{pmatrix} 0 & d'_{z,i} & -d'_{y,i} \\ -d'_{z,i} & 0 & d'_{x,i} \\ d'_{y,i} & -d'_{x,i} & 0 \end{pmatrix}$$

is the solution of the linear equation system

$$\sum_{i=1}^{N} \boldsymbol{D}_i^T \boldsymbol{D}_i \boldsymbol{u} = \sum_{i=1}^{N} \boldsymbol{D}_i (\boldsymbol{m}_i' - \boldsymbol{d}_i')$$

with $\boldsymbol{u} = (\theta_z, \theta_y, \theta_x)$ and

$$\boldsymbol{D}_i = \begin{pmatrix} 0 & d'_{z,i} & -d'_{y,i} \\ -d'_{z,i} & 0 & d'_{x,i} \\ d'_{y,i} & -d'_{x,i} & 0 \end{pmatrix}.$$

The translation is determined according to Eq. (3.4).

Uncertainty-based Registration The probabilistic approach by Lu and Milios [57] introduces the notion of the uncertainty of the mapping algorithms for applications where this is of advantage. The initial algorithm was restricted to 3 DoF. A detailed view on the extension to 6 DoF is given in [16] and [17]. Required are the pose $\boldsymbol{X}$, the pose estimate $\bar{\boldsymbol{X}}$ and the pose error $\Delta\boldsymbol{X}$.

The error of a pose $\boldsymbol{X}$ is given by

$$E = \sum_{i=1}^{m} \|\boldsymbol{X} \oplus \boldsymbol{d}_i - \boldsymbol{m}_i\|^2 = \sum_{i=1}^{m} \|\boldsymbol{Z}_i(\boldsymbol{X})\|^2$$

The compounding operation that transforms a point $\boldsymbol{d}_i$ into the global coordinate system is denoted with $\oplus$. For small $\Delta\boldsymbol{X}$ the Taylor expansion is applied to approximate the error by linearizing the error function. The pose can then be separated from it via matrix decomposition $\boldsymbol{M}_i\boldsymbol{H}$ of $\nabla\boldsymbol{Z}_i(\bar{\boldsymbol{X}})$. Using Euler angles pose, pose estimate and pose error are of form,

$$\boldsymbol{X} = \begin{pmatrix} t_x \\ t_y \\ t_z \\ \theta_x \\ \theta_y \\ \theta_z \end{pmatrix}, \bar{\boldsymbol{X}} = \begin{pmatrix} \bar{t}_x \\ \bar{t}_y \\ \bar{t}_z \\ \bar{\theta}_x \\ \bar{\theta}_y \\ \bar{\theta}_z \end{pmatrix}, \Delta\boldsymbol{X} = \begin{pmatrix} \Delta t_x \\ \Delta t_y \\ \Delta t_z \\ \Delta\theta_x \\ \Delta\theta_y \\ \Delta\theta_z \end{pmatrix},$$

and the corresponding matrix decomposition consists of

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & \bar{t}_z\cos(\bar{\theta}_x) + \bar{t}_y\sin(\bar{\theta}_x) & \bar{t}_y\cos(\bar{\theta}_x)\cos(\bar{\theta}_y) - \bar{t}_z\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) \\ 0 & 1 & 0 & -\bar{t}_z & -\bar{t}_x\sin(\bar{\theta}_x) & -\bar{t}_x\cos(\bar{\theta}_x)\cos(\bar{\theta}_y) - \bar{t}_z\sin(\bar{\theta}_y) \\ 0 & 0 & 1 & \bar{t}_y & -\bar{t}_x\cos(\bar{\theta}_x) & \bar{t}_x\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) + \bar{t}_y\sin(\bar{\theta}_y) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_y) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_x) & \cos(\bar{\theta}_x)\cos(\bar{\theta}_y) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_x) & -\cos(\bar{\theta}_y)\sin(\bar{\theta}_x) \end{pmatrix}$$

and

$$\boldsymbol{M}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & -d_{y,i} & -d_{z,i} \\ 0 & 1 & 0 & d_{z,i} & d_{x,i} & 0 \\ 0 & 0 & 1 & -d_{y,i} & 0 & d_{x,i} \end{pmatrix}.$$

The respective formulation for quaternions is given in [67]. $\boldsymbol{M}_i$ comprises the point information whereas $\boldsymbol{H}$ represents the pose. Accordingly, the squared distance between the concatenation $\boldsymbol{Z}$ of all pose dependent terms $\boldsymbol{Z}_i(\bar{\boldsymbol{X}})$ and the concatenation $\boldsymbol{M}$ of all pose independent terms $\boldsymbol{M}_i$ approximates the positional error

$$E \approx (\boldsymbol{Z} - \boldsymbol{M}\boldsymbol{H}\Delta\boldsymbol{X})^T(\boldsymbol{Z} - \boldsymbol{M}\boldsymbol{H}\Delta\boldsymbol{X}),$$

that can be minimized by

$$\bar{\boldsymbol{E}} = (\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\boldsymbol{Z}$$

with the covariance

$$\boldsymbol{C} = s^2(\boldsymbol{M}^T\boldsymbol{M}).$$

s^2 is the unbiased estimate of the covariance of the identically, independently distributed errors of $\boldsymbol{Z}_i$:

$$s^2 = (\boldsymbol{Z} - \boldsymbol{M}\bar{\boldsymbol{E}})^T(\boldsymbol{Z} - \boldsymbol{M}\bar{\boldsymbol{E}})/(2m - 3).$$

To obtain the optimal pose by use of the minimum $\bar{\boldsymbol{E}}$ for the linearized pose $\boldsymbol{H}\Delta\boldsymbol{X}$ the transformation

$$\begin{aligned} \boldsymbol{X} &= \bar{\boldsymbol{X}} - \boldsymbol{H}^{-1}\bar{\boldsymbol{E}}, \\ \boldsymbol{C} &= (\boldsymbol{H}^{-1})\boldsymbol{C}(\boldsymbol{H}^{-1})^T. \end{aligned}$$

has to be performed.

Point-to-Plane Error Metric

The point-to-plane error metric by Chen And Medioni [21] does not consider distances between the two points from a pair. Instead the distances between the points from D and a plane from the point set M influence the error function. The advantage of this approach comes into effect if D is slid along a plane. The point-to-plane error function prevents increase of the error if the scan points D come to lie on a plane in a wrong position and are moved parallel to this plane. Especially if the mapped environment is dominated by planar surfaces this impact plays an important role.

The algorithm slightly differs from the previously described version. For each point $\boldsymbol{d}_i \in D$ the face normal $\boldsymbol{n}_{\boldsymbol{d}_i}$ has to be computed first. Afterwards the intersection $\boldsymbol{m}_i$ of the line defined

by $\boldsymbol{d}_i$ and $\boldsymbol{n}_{\boldsymbol{d}_i}$ and the point set M is calculated as well as the tangent plane of M through $\boldsymbol{m}_i$. The error function to be minimized is

$$E = \sum_{i=1}^{n} ((\boldsymbol{R}\boldsymbol{d}_i + \boldsymbol{t} - \boldsymbol{m}_i) \cdot \boldsymbol{n}_{\boldsymbol{m}_i})^2. \tag{3.11}$$

There exists no closed form solution to Eq. (3.11). Rusinkiewicz and Levoy [76, 77] propose a linearization under assumption of small rotation angles and approximation of $\sin\theta$ by θ and $\cos\theta$ by 0. The resulting linear equation system

$$E \approx \sum_{i=1}^{n} ((\boldsymbol{m}_i - \boldsymbol{d}_i) \cdot \boldsymbol{n}_{\boldsymbol{d}_i} + \boldsymbol{r} \cdot (\boldsymbol{m}_i \times \boldsymbol{d}_i) + \boldsymbol{t} \cdot \boldsymbol{n}_{\boldsymbol{d}_i})^2$$

can be solved for the transformation

$$\boldsymbol{r} = (\theta_x, \theta_y, \theta_z)^T$$
$$\boldsymbol{t} = (x, y, z)^T.$$

3.3 Global Mapping

The methods described in the previous section work with two scans. There exist several other procedures for pairwise scan matching. However, in practical applications one is usually faced with a significantly larger number of scans. It is of course possible to apply the same procedure several times and to sequentially match all scans by applying the transformation of scan i to all successive scans. This might lead to undesired results. Scan matching is never perfect. It only tries to minimize the error between possibly erroneous correspondences. When applying the transformation of scan i to its successors the errors are also passed on and accumulate as the sequential procedure proceeds. If a scan overlaps with more than one other scan the accumulation of errors might cause the scan to be matched nicely to its predecessor but to show inconsistencies with respect to other overlapping scans.

Globally consistent scan matching tries to match a set of laser scans in a way that each scan is correctly positioned with respect to all other scans. Different methods exist that yield, depending on the problem, results of different quality [76].

For scans that feature large overlaps independent of the order in which they are taken the use of an anchor scan is suggestive. Given that one scan exists that overlaps with all other scans in the set this scan is chosen as anchor scan and all scans are matched to it. This procedure is however mostly used for modeling of objects where a cylindrical low resolutional scan is taken as anchor scan and scans of higher resolution precisely model the object. For robot mapping this is difficult to achieve.

As an alternative a scan can be matched to the union of already registered scans. All registered scans are combined into a so-called metascan and the new scan is matched with this metascan. This leads to an enormous increase in computational time because the point pairs have to be selected from a significantly larger set. Besides, inconsistencies cannot be avoided if scans are added to different parts of the metascan without any change of the already registered scans.

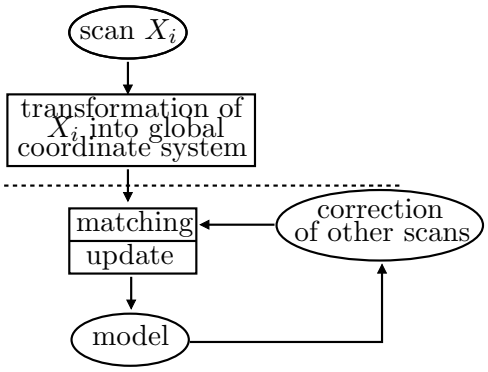


Fig. 3.6: Structure of an incremental globally consistent scan matching algorithm.

The previous thoughts make it obvious that it is necessary to adapt the poses of previously registered scans to achieve consistency. A possible strategy is outlined in Fig. 3.6. Considering the map as a global model consisting of laser scans and their poses, several steps are necessary when adding a new scan X_i to the model. At first scan X_i is transformed into the global coordinate system and matched to the model. Then the already registered scans poses are adapted according to the newly acquired information.

Robotic mapping that consists of exploration of an unknown environment underlies certain characteristics. The scans have only small overlapping regions with a small number of predecessors and successors because the scans are taken along a robot path. Overlaps with scans with a time-wise large distance are rather seldom. Therefore consideration of all scans leads to unnecessary expenses. Instead a search for so-called loops appears more reasonable. A loop is a robot path where the robot starts at one position and returns to the same position, or a region close by later on. Nüchter et al. [68, 84] developed a procedure for the special case of one large loop. When detecting a loop the computed error between the first and the last scan is uniformly distributed over all scans in the loop.

Lu and Milios presented a GraphSLAM method for globally consistent scan matching [57]. This method was initially designed for 2D data, the extension to 3D is described in [16]. The algorithm manages a graph of laser scans. In addition to the point clouds the robot poses and the spatial relations between the scans acquired by pairwise scan matching are represented in the graph. By means of an equation system that is constricted by the spatial relations between the scans the robot poses are adapted such that the relations are preserved as good as possible.

3.3.1 Problem Formulation

Consider the case of a robot exploring an environment (cf. [15, 67]). The robot travels along a path and stops at $n + 1$ poses $V_0, \ldots, V_n$. Each time it stops a laser scan of the robot's environment is taken. By pairwise mapping scans from two different poses a set of relations between the poses is acquired. The robot path is represented as a graph. Each node represents a pose and the edges correspond to the relations acquired through scan matching.

Given such a graph with $n + 1$ nodes $X_0, \ldots, X_n$ and the directed edges $D_{i,j}$. Each node

is a d-dimensional pose vector of unknown value. The $D_{i,j}$ correspond to the measured pose differences between X_i and X_j and represent a function of the two nodes they connect. The goal is to build a consistent map of the environment by optimally estimating all poses. To simplify the measurement we assume linear pose differences

$$D_{i,j} = X_i - X_j.$$

The true pose difference is estimated by an observation that is modeled as $\bar{D}_{i,j} = D_{i,j} + \Delta D_{i,j}$ with the Gaussian distributed random error $\Delta D_{i,j}$ with zero mean and a known covariance matrix $C_{i,j}$. Given a set of pose observations $\bar{D}_{i,j}$ and covariance matrices $C_{i,j}$ we want to compute the optimal poses X_i with all the provided information. This is achieved by an approximation using maximum likelihood estimation. Assuming independently distributed Gaussian errors in the observations the problem of maximizing the probability of all $D_{i,j}$ given their observation $\bar{D}_{i,j}$, e.g., $P(D_{i,j}|\bar{D}_{i,j})$ is equivalent to minimizing the Mahalanobis distance:

$$\mathbf{W} = \sum_{(i,j)} (D_{i,j} - \bar{D}_{i,j})^T C_{i,j}^{-1} (D_{i,j} - \bar{D}_{i,j}). \tag{3.12}$$

3.3.2 Scan Matching

Like the pairwise registration the global registration is based on finding correspondences between laser scans and minimizing the distances between these correspondences. The major difference is that the distances between several scans have to be considered instead of using just one pair. Consequently the minimization yields new poses for all scans but one fixed anchor scan. The basic structure of the algorithm can be outlined as:

1. Determine point correspondences.
2. For each edge in the graph determine pose differences and covariances.
3. Create an equation system from pose differences and covariance matrices according to Eq. (3.12).
4. Linearize and solve the equation system.
5. Revise poses and covariances.

The first step of the algorithm is equal to the ICP algorithm. Point correspondences have to be established. For this purpose the same considerations as for the ICP hold true. The point selection process, finding of corresponding point pairs as well as elimination of outliers can help to improve the matching process as described in the pairwise scan matching Section 3.2. The following section covers the creation and minimization of the error function.

3.3.3 Minimization of the Error Function

Under the assumption that we have a network of overlapping scans the error minimization methods that were presented for the ICP algorithm are now adapted for the n-scan registration case [67]. In all methods we reduce the optimization problem to a linear equation system that needs to be solved.

Helix Transform

The error function for the helix transform is extended to include all poses $\boldsymbol{X}_j$:

$$\begin{aligned} E &= \sum_{j\to k}\sum_{i}(\boldsymbol{m}_i - \boldsymbol{d}_i + (\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i) - (\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_i))^2 \\ &= \sum_{j\to k}\Bigg(\sum_{i}((\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i) - (\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_i))^2 \\ &\qquad + \sum_{i} 2((\boldsymbol{m}_i - \boldsymbol{d}_i)\cdot(\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i) + (\boldsymbol{d}_i - \boldsymbol{m}_i)\cdot(\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_i)) \\ &\qquad + \sum_{i}(\boldsymbol{m}_i - \boldsymbol{d}_i)^2\Bigg). \end{aligned}$$

Reformulating E as

$$E = \boldsymbol{X}^T \cdot \boldsymbol{B} \cdot \boldsymbol{X} + 2\boldsymbol{A}\cdot\boldsymbol{X} + \sum_i (\boldsymbol{m}_i - \boldsymbol{d}_i)^2,$$

allows us to solve for the optimal poses $\boldsymbol{X}$ in the linear equation system:

$$\boldsymbol{B}\boldsymbol{X} = \boldsymbol{A}.$$

$\boldsymbol{A}$ is attained as follows:

$$\begin{aligned} 2\boldsymbol{A}\boldsymbol{X} &= \sum_{j\to k}\sum_{i} 2((\boldsymbol{m}_i - \boldsymbol{d}_i)(\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i) + (\boldsymbol{d}_i - \boldsymbol{m}_i)(\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_i)) \\ \boldsymbol{A}_j &= \sum_{j\to k}\sum_{i}\begin{pmatrix} \boldsymbol{m}_i \times (\boldsymbol{m}_i - \boldsymbol{d}_i) \\ \boldsymbol{m}_i - \boldsymbol{d}_i \end{pmatrix} + \sum_{k\to j}\sum_{i}\begin{pmatrix} \boldsymbol{m}_i \times (\boldsymbol{d}_i - \boldsymbol{m}_i) \\ \boldsymbol{d}_i - \boldsymbol{m}_i \end{pmatrix}. \end{aligned}$$

$\boldsymbol{B}$ is defined by

$$\boldsymbol{X}^T \cdot \boldsymbol{B} \cdot \boldsymbol{X} = \sum_{j\to k}\sum_{i}((\bar{\boldsymbol{c}}_j + \boldsymbol{c}_j \times \boldsymbol{m}_i) - (\bar{\boldsymbol{c}}_k + \boldsymbol{c}_k \times \boldsymbol{m}_i))^2,$$

and given as

$$\begin{aligned} \boldsymbol{B}_{j,j} &= \sum_{\substack{j\to k \\ k\to j}}\sum_{i}\boldsymbol{M}_i \\ \boldsymbol{B}_{j,k} &= \sum_{j\to k}\sum_{i} -\boldsymbol{M}_i \end{aligned}$$

where

$$\boldsymbol{M}_i = \begin{pmatrix} m_{y,i}^2 + m_{z,i}^2 & -m_{x,i}m_{y,i} & -m_{x,i}m_{z,i} & 0 & -m_{z,i} & m_{y,i} \\ -m_{x,i}m_{y,i} & m_{x,i}^2 + m_{z,i}^2 & -m_{y,i}m_{z,i} & m_{z,i} & 0 & -m_{x,i} \\ -m_{x,i}m_{z,i} & -m_{y,i}m_{z,i} & m_{x,i}^2 + m_{y,i}^2 & -m_{y,i} & m_{x,i} & 0 \\ 0 & m_{z,i} & -m_{y,i} & 1 & 0 & 0 \\ -m_{z,i} & 0 & m_{x,i} & 0 & 1 & 0 \\ m_{y,i} & -m_{x,i} & 0 & 0 & 0 & 1 \end{pmatrix}.$$

3.3.4 Global Registration using the Small Angle Approximation

For the small angle approximation the error metric is extended to include all poses $\boldsymbol{X}_j$ to:

$$E = \sum_{j \to k} \sum_i \left| \boldsymbol{R}_j \boldsymbol{m}_i + \boldsymbol{t}_j - (\boldsymbol{R}_k \boldsymbol{d}_i + \boldsymbol{t}_k) \right|^2 .$$

Using the centroids $\boldsymbol{c}_d$ and $\boldsymbol{c}_m$, where $\boldsymbol{m}'_i = \boldsymbol{m}_i - \boldsymbol{c}_m$ and $\boldsymbol{d}'_i = \boldsymbol{d}_i - \boldsymbol{c}_d$, to arrange the point sets around the origin E is restated as:

$$\begin{aligned} E &= \sum_{j \to k} \sum_i \left| \boldsymbol{R}_j \boldsymbol{m}'_i + \boldsymbol{R}_j \boldsymbol{c}_m + \boldsymbol{t}_j - (\boldsymbol{R}_k \boldsymbol{d}'_i + \boldsymbol{R}_k \boldsymbol{c}_d + \boldsymbol{t}_k) \right|^2 \\ &= \sum_{j \to k} \sum_i \left| \boldsymbol{R}_j \boldsymbol{m}'_i - \boldsymbol{R}_k \boldsymbol{d}'_i - (\boldsymbol{t}_k - \boldsymbol{t}_j + \boldsymbol{R}_k \boldsymbol{c}_d - \boldsymbol{R}_j \boldsymbol{c}_m) \right|^2 \\ &= \sum_{j \to k} \Bigg(\sum_i \left| \boldsymbol{R}_j \boldsymbol{m}'_i - \boldsymbol{R}_k \boldsymbol{d}'_i \right|^2 \\ &\qquad - 2 \sum_i (\boldsymbol{t}_k - \boldsymbol{t}_j + \boldsymbol{R}_k \boldsymbol{c}_d - \boldsymbol{R}_j \boldsymbol{c}_m)(\boldsymbol{R}_j \boldsymbol{m}'_i - \boldsymbol{R}_k \boldsymbol{d}'_i) \\ &\qquad + \sum_i \left| \boldsymbol{t}_k - \boldsymbol{t}_j + \boldsymbol{R}_k \boldsymbol{c}_d - \boldsymbol{R}_j \boldsymbol{c}_m \right|^2 \Bigg) . \end{aligned} \tag{3.13}$$

The term $-2 \sum_i (\boldsymbol{t}_k - \boldsymbol{t}_j + \boldsymbol{R}_k \boldsymbol{c}_d - \boldsymbol{R}_j \boldsymbol{c}_m)(\boldsymbol{R}_j \boldsymbol{m}'_i - \boldsymbol{R}_k \boldsymbol{d}'_i)$ can be eliminated because all values refer to the centroids and therefore equate to zero. This enables us to separate rotation and translation of all poses.

Computing the Rotation

The optimal rotation of all poses is calculated by minimizing the first term of the error metric as stated in Eq. (3.13). We use the small angle approximation introduced in Eq. (3.10) to handle the nonlinear rotation:

$$\begin{aligned} \boldsymbol{R}_j \boldsymbol{m}_i &= \begin{pmatrix} 0 & m_{z,i} & -m_{y,i} \\ -m_{z,i} & 0 & m_{x,i} \\ m_{y,i} & -m_{x,i} & 0 \end{pmatrix} \cdot \boldsymbol{X}_j + \boldsymbol{m}_i \\ &= \boldsymbol{M}_i \cdot \boldsymbol{X}_j + \boldsymbol{m}_i . \end{aligned}$$

The rotational error to be minimized is written as:

$$\begin{aligned} E_R &= \sum_{j\to k}\sum_{i} \left(\boldsymbol{M}_i \cdot \boldsymbol{X}_j - \boldsymbol{D}_i \cdot \boldsymbol{X}_k - (\boldsymbol{m}_i - \boldsymbol{d}_i)\right)^2 \\ &= \sum_{j\to k}\sum_{i} \left(\boldsymbol{M}_i \cdot \boldsymbol{X}_j - \boldsymbol{D}_i \cdot \boldsymbol{X}_k\right)^2 + (\boldsymbol{m}_i - \boldsymbol{d}_i)^2 \\ &\qquad - 2\left(\boldsymbol{M}_i \cdot \boldsymbol{X}_j - \boldsymbol{D}_i \cdot \boldsymbol{X}_i\right) \cdot (\boldsymbol{m}_i - \boldsymbol{d}_i), \end{aligned}$$

where $\boldsymbol{X}_j = (\theta_{z,j}, \theta_{y,j}, \theta_{x,j})^T$ represents the rotation of the jth pose. The error term is rewritten

$$E_R = \boldsymbol{X}\boldsymbol{B}\boldsymbol{X} + 2\boldsymbol{A}\boldsymbol{X} + (\boldsymbol{m}_k - \boldsymbol{d}_k)^2$$

in order to solve the linear equation system

$$\boldsymbol{B}\boldsymbol{X} + \boldsymbol{A} = 0.$$

$\boldsymbol{B}$ is given by:

$$\boldsymbol{B}_{j,j} = \sum_{j\to k}\sum_{i} \boldsymbol{D}_i^T \cdot \boldsymbol{D}_i + \sum_{(k,j)}\sum_{i} \boldsymbol{M}_i^T \cdot \boldsymbol{M}_i$$

case $j < k$:

$$\boldsymbol{B}_{j,k} = -\sum_{j\to k}\sum_{i} \boldsymbol{M}_i^T \cdot \boldsymbol{D}_i$$

case $j > k$:

$$\boldsymbol{B}_{j,k} = -\sum_{j\to k}\sum_{i} \boldsymbol{D}_i^T \cdot \boldsymbol{M}_i,$$

and $\boldsymbol{A}$ by:

$$\begin{aligned} \boldsymbol{A}_j = &\sum_{k\to j}\sum_{i} \begin{pmatrix} (m_{z,i} - d_{z,i}) \cdot d_{y,i} - (m_{y,i} - d_{y,i}) \cdot d_{z,i} \\ (m_{x,i} - d_{x,i}) \cdot d_{z,i} - (m_{z,i} - d_{z,i}) \cdot d_{x,i} \\ (m_{y,i} - d_{y,i}) \cdot d_{x,i} - (m_{x,i} - d_{x,i}) \cdot d_{y,i} \end{pmatrix} \\ &- \sum_{j\to k}\sum_{i} \begin{pmatrix} (m_{z,i} - d_{z,i}) \cdot m_{y,i} - (m_{y,i} - d_{y,i}) \cdot m_{z,i} \\ (m_{x,i} - d_{x,i}) \cdot m_{z,i} - (m_{z,i} - d_{z,i}) \cdot m_{x,i} \\ (m_{y,i} - d_{y,i}) \cdot m_{x,i} - (m_{x,i} - d_{x,i}) \cdot m_{y,i} \end{pmatrix}. \end{aligned}$$

Computing the Translation

After calculating the optimal rotations, minimizing

$$E_T = \sum_{j\to k}\sum_{i} \left(\boldsymbol{t}_k - \boldsymbol{t}_j + \boldsymbol{R}_k\boldsymbol{c}_d - \boldsymbol{R}_j\boldsymbol{c}_m\right)^2$$

yields the optimal translations.

$\boldsymbol{R}_k\boldsymbol{c}_d - \boldsymbol{R}_j\boldsymbol{c}_m$ is abbreviated with $\boldsymbol{R}_{j,k}$:

$$E_T = \sum_{j\to k}\sum_i (\boldsymbol{t}_k - \boldsymbol{t}_j)^2 - 2(\boldsymbol{t}_k - \boldsymbol{t}_j)\boldsymbol{R}_{j,k} + \boldsymbol{R}_{j,k}^2.$$

Written in matrix notation:

$$E_T = \boldsymbol{T}^T\boldsymbol{B}\boldsymbol{T} + 2\boldsymbol{A}\boldsymbol{T} + \sum_{j\to k}\sum_i \boldsymbol{R}_{j,k}^2$$

it becomes clear that the optimal translation is the solution of the minimal equation system:

$$\boldsymbol{B}\boldsymbol{T} + \boldsymbol{A} = 0$$

with $\boldsymbol{B}$ composed of:

$$\boldsymbol{B}_{j,j} = \sum_{\substack{j\to k\\ k\to j}} \boldsymbol{I}$$

case $j < k$:

$$\boldsymbol{B}_{j,k} = -\sum_{j\to k} \boldsymbol{I}$$

case $j > k$:

$$\boldsymbol{B}_{j,k} = -\sum_{j\to k} \boldsymbol{I},$$

and $\boldsymbol{A}$ of:

$$\boldsymbol{A}_j = \sum_{j\to k} \boldsymbol{R}_j\boldsymbol{c}_m - \boldsymbol{R}_k\boldsymbol{c}_d - \sum_{k\to j} \boldsymbol{R}_j\boldsymbol{c}_m - \boldsymbol{R}_k\boldsymbol{c}_d.$$

3.3.5 Uncertainty-based Global Registration

The uncertainty-based global registration can also be expanded for the n-scan case. The positional error of two poses $\boldsymbol{X}_j$ and $\boldsymbol{X}_k$ is described by:

$$E_{j,k} = \sum_{i=1}^{m} \|\boldsymbol{X}_j \oplus \boldsymbol{d}_i - \boldsymbol{X}_k \oplus \boldsymbol{m}_i\|^2$$

Analogous to the simple 2-scan case, the pose differences are linearized and approximated by applying Taylor expansion as

$$\begin{aligned}\boldsymbol{E}'_{j,k} &= (\boldsymbol{H}_j \Delta \boldsymbol{X}_j - \boldsymbol{H}_k \Delta \boldsymbol{X}_k) \\ &= (\boldsymbol{X}'_j - \boldsymbol{X}'_k).\end{aligned}$$

$\boldsymbol{E}'_{j,k}$ is linear in the quantities $\boldsymbol{X}'_j$ that will be estimated by the algorithm. Again, the minimum of $\boldsymbol{E}'_{j,k}$ and the corresponding covariance are given by

$$\begin{aligned}\bar{\boldsymbol{E}}_{j,k} &= (\boldsymbol{M}^T \boldsymbol{M})^{-1} \boldsymbol{M}^T \boldsymbol{Z} \\ \boldsymbol{C}_{j,k} &= s^2 (\boldsymbol{M}^T \boldsymbol{M}).\end{aligned}$$

$\boldsymbol{Z}$ is the concatenated vector consisting of all $\boldsymbol{Z}_i = \bar{\boldsymbol{X}}_j \oplus \boldsymbol{d}_i - \bar{\boldsymbol{X}}_k \oplus \boldsymbol{m}_i$.

To determine the global error of all poses the linearized error metric $\boldsymbol{E}'_{j,k}$ and the Gaussian distribution $(\bar{\boldsymbol{E}}_{j,k}, \boldsymbol{C}_{j,k})$ are included into the Mahalanobis distance:

$$\begin{aligned}\boldsymbol{W} &= \sum_{j \to k} (\bar{\boldsymbol{E}}_{j,k} - \boldsymbol{E}'_{j,k})^T \boldsymbol{C}^{-1}_{j,k} (\bar{\boldsymbol{E}}'_{j,k} - \boldsymbol{E}'_{j,k}) \\ &= \sum_{j \to k} (\bar{\boldsymbol{E}}_{j,k} - (\boldsymbol{X}'_j - \boldsymbol{X}'_k)) \boldsymbol{C}^{-1}_{j,k} (\bar{\boldsymbol{E}}'_{j,k} - (\boldsymbol{X}'_j - \boldsymbol{X}'_k)),\end{aligned}$$

or written in matrix notation, $\boldsymbol{W}$:

$$\boldsymbol{W} = (\bar{\boldsymbol{E}} - \boldsymbol{H}\boldsymbol{X})^T \boldsymbol{C}^{-1} (\bar{\boldsymbol{E}} - \boldsymbol{H}\boldsymbol{X}).$$

Here $\boldsymbol{H}$ is the signed incidence matrix of the pose graph, $\bar{\boldsymbol{E}}$ is the concatenated vector consisting of all $\bar{\boldsymbol{E}}'_{j,k}$ and $\boldsymbol{C}$ is a block-diagonal matrix comprised of $\boldsymbol{C}^{-1}_{j,k}$ as submatrices. Minimizing this function yields new optimal pose estimates. Solving the following linear equation system minimizes $\boldsymbol{W}$:

$$\begin{aligned}(\boldsymbol{H}^T \boldsymbol{C}^{-1} \boldsymbol{H}) \boldsymbol{X} &= \boldsymbol{H}^T \boldsymbol{C}^{-1} \bar{\boldsymbol{E}} \\ \boldsymbol{B}\boldsymbol{X} &= \boldsymbol{A}.\end{aligned}$$

The matrix $\boldsymbol{B}$ consists of the submatrices

$$\boldsymbol{B}_{j,k} = \begin{cases} \sum_{k=0}^{n} \boldsymbol{C}^{-1}_{j,k} & (j = k) \\ \boldsymbol{C}^{-1}_{j,k} & (j \neq k). \end{cases}$$

The entries of $\boldsymbol{A}$ are given by:

$$A_j = \sum_{\substack{k=0 \\ k \neq j}}^{n} \boldsymbol{C}^{-1}_{j,k} \bar{\boldsymbol{E}}_{j,k}.$$

In addition to $\boldsymbol{X}$, the associated covariance of $\boldsymbol{C}_X$ is computed as follows:

$$\boldsymbol{C}_X = \boldsymbol{B}^{-1}$$

Just like in the simple 2-scan case the results again have to be transformed in order to obtain the optimal pose estimates:

$$\begin{aligned}\boldsymbol{X}_j &= \bar{\boldsymbol{X}}_j - \boldsymbol{H}_j^{-1}\boldsymbol{X}'_j,\\ \boldsymbol{C}_j &= (\boldsymbol{H}_j^{-1})\boldsymbol{C}_j^X(\boldsymbol{H}_j^{-1})^T.\end{aligned}$$

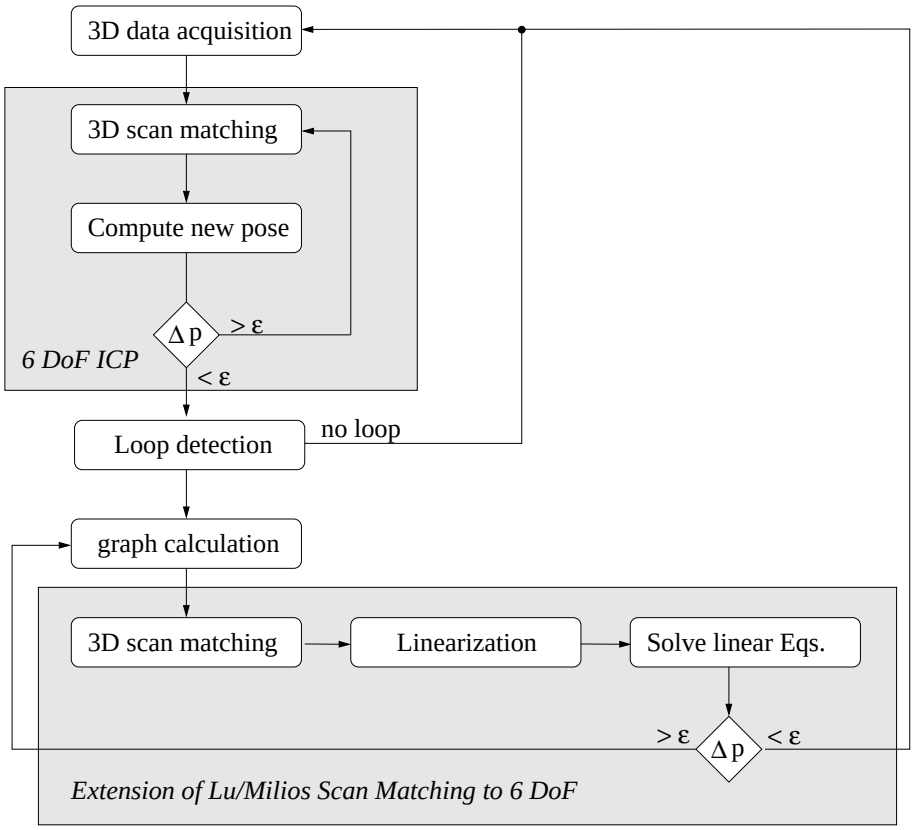


Fig. 3.7: The interaction of ICP and LUM in the overall algorithm. The threshold ϵ used to determine if a scan changed its pose is the same in both algorithm parts. [16]

3.4 Algorithm Overview

This Section outlines the complete global relaxation algorithm as depicted in Fig. 3.7. The algorithm is a combination of the two algorithms described in the previous sections, the local ICP algorithm and the global optimization according to Lu and Milios (abbreviated as LUM).

All acquired data is first sequentially matched using the ICP algorithm. Starting from the first scan, each successive scan is registered against its predecessor until convergence. Based on the final pose of this scan a loop detection algorithm is used to check whether the estimated pose of this scan is near a previously registered pose. If this is not the case, the procedure moves on to the next scan. Once a loop is detected, the graph for LUM is dynamically calculated. Each pose of the already registered scans becomes a node in the graph. Edges are added between consecutive scans and between scans that possess a certain number of corresponding points. Pose differences between all scans are calculated to create the equation system that is then linearized and solved. This is again an iterative procedure that is performed until a threshold is met. If the error function gets below this threshold new scans are added to the model via the ICP algorithm. Otherwise the graph is newly calculated and the global relaxation performed again.

Chapter 4

The Thin Plate Spline

The Thin Plate Spline (TPS) was first formulated by J. Duchon in 1976 [28] as a way to model the deformation of a thin metal sheet. In the model, an infinitesimally thin metal sheet that extends to infinity on all sides is deflected at arbitrarily many points. The deflection is in the z-direction, i.e., orthogonal to the metal sheet. Thus, the TPS in its original form is a function that maps points on the plane to their respective amount of deformation (see Fig. 4.1). However, since the TPS was designed as a high-dimensional extension of regular splines, i.e., of B-splines and similar splines that are used in computer graphics [78], the TPS is able to model any deformation of arbitrary dimension.

Due to its high degree of flexibility the TPS is used in a wide variety of fields. Bookstein [13] used the TPS to model and identify the gradual changes in the bone structure of animals that occur during their evolution. He was the first to formulate the notion of principal warps in a TPS function and to give a closed form solution for the calculation of an interpolating TPS. The latter problem, i.e., calculating a TPS functional that interpolates a set of supporting points was also solved by Wahba in 1990 [91], who gave a solution using QR-decomposition. He also introduced the smoothing parameter λ that controls the smoothness of the TPS by sacrificing exact interpolation.

Other uses of the TPS include shape matching in 2D [23], deformable 3D scan matching [19] or compensating for errors made by document scanners [12]. Similar applications are finger print matching in forensic science [9] or the estimation of large disparity motion in images [96]. In the field of geostatistics TPS are widely used as a kriging technique to smooth and interpolate data [46]. Because the TPS can be seen as a general function approximator it also has applications in the field of machine learning [83].

The properties of the TPS make it an ideal candidate for consideration in the application of deformable scan matching. The TPS is able to represent any perceivable deformation of a scan, while guaranteeing the maximum amount of smoothness. It is important to choose a deformation representation that is robust to errors in the correspondences and also measurement errors that are likely to appear in practice. This ability is provided in the form of a smoothing parameter that lets the TPS smooth over noisy data.

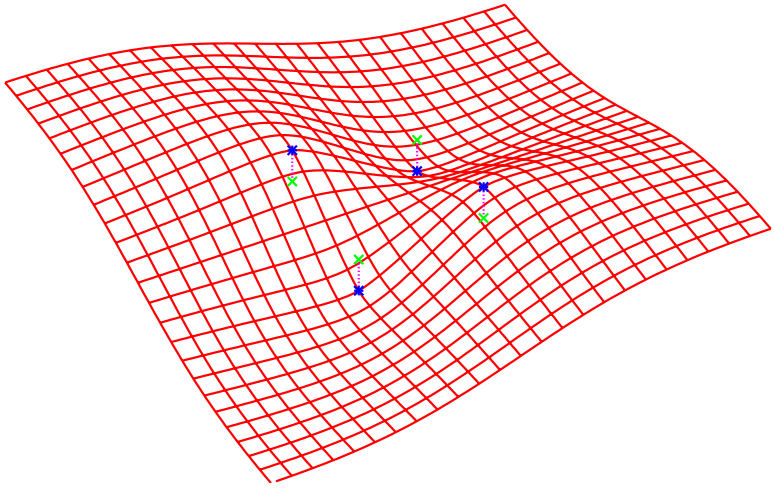


Fig. 4.1: A thin metal sheet that has been deformed at four points near the center. Green points: The resting position of the undeformed metal sheet. Blue points: The new positions of the green points after they have been deflected perpendicular to the sheet.

4.1 Definition of the Thin Plate Spline

Given two corresponding point sets or landmarks $\boldsymbol{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n)^T$ and $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n)^T$ with $\boldsymbol{m}_i \in \mathbb{R}^e$ and the homogeneous coordinates $\boldsymbol{d}_i \in \mathbb{R}^d$ the TPS function $\boldsymbol{T}\colon \mathbb{R}^d \to \mathbb{R}^e$ fulfills

$$\boldsymbol{T}(\boldsymbol{d}_i) = \boldsymbol{m}_i, \tag{4.1}$$

while minimizing the bending energy J:

$$J = \int \left(\sum_{i,j} \boldsymbol{T}^2_{\boldsymbol{d}_i, \boldsymbol{d}_j} \right) \mathrm{d}\boldsymbol{d}_1, \ldots, \mathrm{d}\boldsymbol{d}_n.$$

In other words $\boldsymbol{T}$ maps each $\boldsymbol{d}_i$ to its corresponding point $\boldsymbol{m}_i$, while exhibiting the least amount of non-affine warping possible. Duchon [28] has shown that such a function $\boldsymbol{T}$ always exists and that it is unique, i.e., there is exactly one such function.

It is necessary to express the landmarks $\boldsymbol{d}_i$ as homogeneous coordinates to describe the affine transformation in a linear fashion. The principle behind homogeneous coordinates is simple; a 1 is appended to the regular coordinates. After the homogeneous vector is multiplied with an affine matrix the result can easily be transformed into the regular coordinate system by dividing it by its last coordinate.

The TPS function is therefore separable into an affine part $\boldsymbol{A} \in \mathbb{R}^{e \times d}$ and a non-affine part, the warping parameters $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)^T$ with $\boldsymbol{W} \in \mathbb{R}^{e \times n}$. $\boldsymbol{T}$ is given by:

$$\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \sum_{i=1}^{n} \boldsymbol{w}_i U(|\boldsymbol{d}_i - \boldsymbol{x}|).$$

The function U is the so-called kernel function. Any Green's function that satisfies the differential equation

$$\Delta^2 U = 0$$

is applicable. For 2D applications the following kernel is commonly used.

$$U(|\boldsymbol{d} - \boldsymbol{m}|) = U(r) = r^2 \log r^2.$$

For 3D applications a kernel of the type

$$U(|\boldsymbol{d} - \boldsymbol{m}|) = U(r) = r^{2n+1}$$

for any $n \in \mathbb{N}$, is often employed. The most common variants are $U(r) = r$ and $U(r) = r^3$. Note however, that all kernels may be used for deformations of any dimension. The kernels $r^2 \log r^2$ and r^3 produce very similar smooth output, while r often produces non-differentiable deformations. See Fig. 4.2 for an overview of the effects of different kernels.

In case the exact interpolation of the given points $\boldsymbol{G}$ is undesirable the minimization problem can be restated in more general terms. First, the smoothing parameter λ that will control the degree of approximation for the Thin Plate Spline is introduced. Instead of minimizing J under the interpolation constraint (4.1) the following energy function is minimized:

$$E_\lambda = \frac{1}{n} \sum_{i=1}^{n} |\boldsymbol{m}_i - \boldsymbol{T}(\boldsymbol{d}_i)|^2 + \lambda J. \tag{4.2}$$

If $\lambda = 0$ this reduces to the special case of interpolation as already discussed. In the case of $\lambda > 0$ the smoothness of the resulting deformation is increased at the cost of greater approximation errors. As λ tends to positive infinity the resulting TPS functional tends to an affine transformation that best maps $\boldsymbol{D}$ to $\boldsymbol{M}$. For an example see Fig. 4.3.

4.2 Solving the Thin Plate Spline

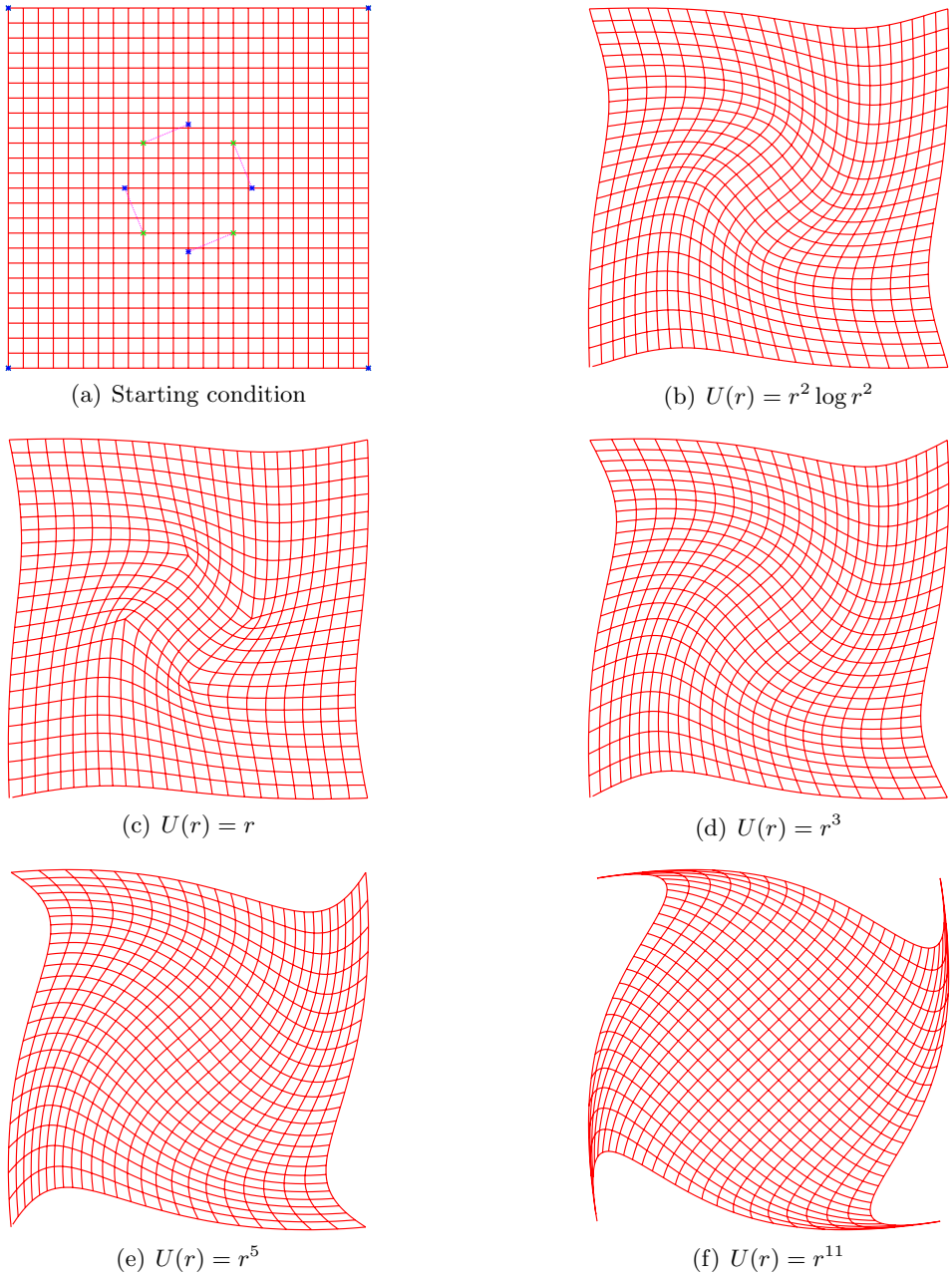
With the TPS function T defined in the previous section, the means of computing the affine transformation $\boldsymbol{A}$ and the non-affine warping parameters $\boldsymbol{W}$ will be given in this section. We follow the solution as given in [91]. Furthermore, we evaluate the complexity of the computation of the TPS.

Minimizing the energy functional (4.2) is done via the QR-decomposition of $\boldsymbol{D}$. Keep in mind that $\boldsymbol{M}$ contains homogeneous coordinates, i.e., the last column contains all ones. The decomposition is given by:

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} \\ 0 \end{pmatrix},$$

with the orthonormal matrices $\boldsymbol{Q}_1 \in \mathbb{R}^{n \times e}$ and $\boldsymbol{Q}_2 \in \mathbb{R}^{n \times (n-e)}$ and the upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{e \times e}$. Substituting the QR-decomposition into the energy functional gives:

$$E_\lambda = \left|\boldsymbol{Q}_2^T \boldsymbol{M} - \boldsymbol{Q}_2^T \boldsymbol{K} \boldsymbol{Q}_2 \boldsymbol{\Gamma}\right|^2 + \left|\boldsymbol{Q}_1^T \boldsymbol{M} - \boldsymbol{R}\boldsymbol{A} - \boldsymbol{Q}_1^T \boldsymbol{K} \boldsymbol{Q}_2 \boldsymbol{\Gamma}\right|^2 + \lambda \boldsymbol{\Gamma}^T \boldsymbol{Q}_2^T \boldsymbol{K} \boldsymbol{Q}_2 \boldsymbol{\Gamma}.$$



(a) Starting condition

(b) $U(r) = r^2 \log r^2$

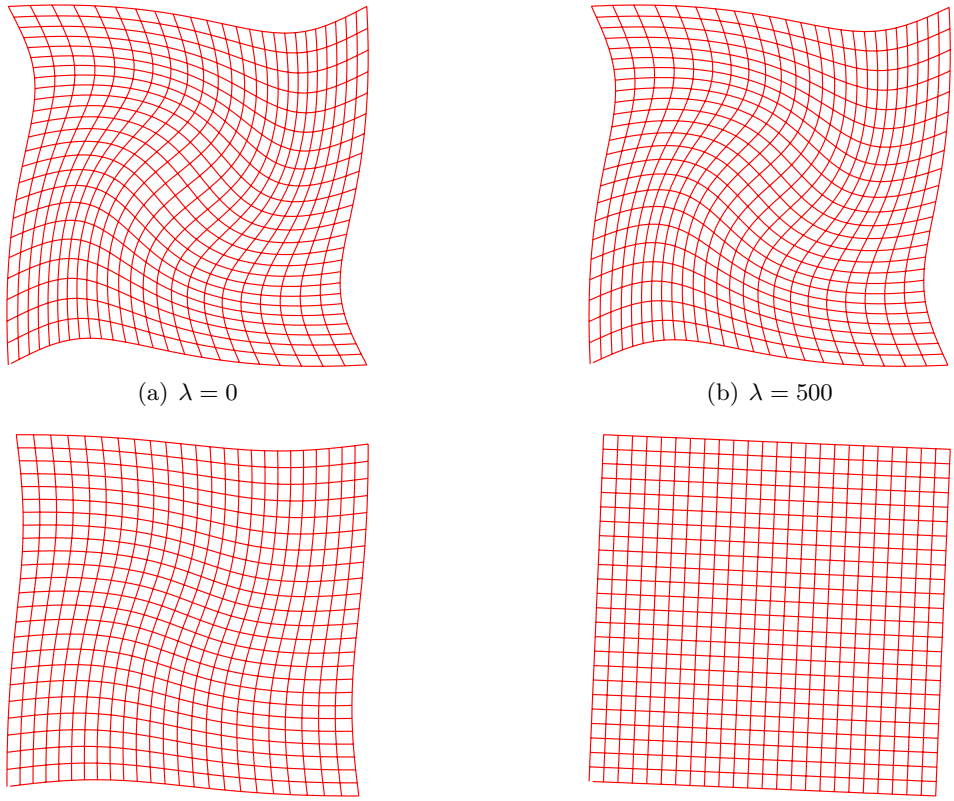
(c) $U(r) = r$

(d) $U(r) = r^3$

(e) $U(r) = r^5$

(f) $U(r) = r^{11}$

Fig. 4.2: Effects of different kernels on the deformation of a plane using the same landmarks. (a) The starting condition of the deformation. The four corners (green crosses) of the square near the center of the plane are rotated 90° clockwise (blue stars). Note: The four corners of the plane are held in place in order to create a deformation instead of a rigid transformation. (b) A commonly used kernel in 2D scenarios. (c) The linear kernel often creates a non-differentiable deformation. This is visible in the abrupt changes in the gridlines near the central four points. (d)-(f) The effects of the deformation get increasingly stronger and more local the higher the exponent in the kernel function $U(r) = r^{2n+1}$ is. The points near the border are further apart from each other than the points near the center. The relative influence on the deformation of the outer points decreases with higher exponents.



(a) $\lambda = 0$ (b) $\lambda = 500$

(c) $\lambda = 10000$ (d) $\lambda = 1000000$

Fig. 4.3: Effects of using different settings for the smoothing parameter λ, using the same set of parameters as in Fig. 4.2. The used kernel is $U(r) = r^3$. (a) Exact interpolation of the corresponding points with $\lambda = 0$. (b) and (c) With increasing λ the resulting deformation gets more and more rigid, (d) until it starts to converge to the most rigid transformation possible at high values.

The $n \times n$ matrix $\boldsymbol{K}$ is of central importance to the solution of the TPS and is given by:

$$\boldsymbol{K}_{i,j} = U(|\boldsymbol{m}_i - \boldsymbol{m}_j|).$$

$\boldsymbol{\Gamma} \in \mathbb{R}^{(n-e)\times e}$ is a matrix defined by $\boldsymbol{W} = \boldsymbol{Q}_2\boldsymbol{\Gamma}$. Minimizing E_λ first with respect to $\boldsymbol{\Gamma}$ and then with respect to $\boldsymbol{A}$ yields:

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{Q}_2 \left(\boldsymbol{Q}_2^T \boldsymbol{K} \boldsymbol{Q}_2 + \lambda \boldsymbol{I}_{n-e}\right)^{-1} \boldsymbol{Q}_2^T \boldsymbol{M} \\ \boldsymbol{A} &= \boldsymbol{R}^{-1} \boldsymbol{Q}_1^T \left(\boldsymbol{D} - \boldsymbol{M}\boldsymbol{W}\right). \end{aligned} \tag{4.3}$$

The required parameters for the TPS can therefore be computed by decomposing $\boldsymbol{D}$, constructing $\boldsymbol{K}$ and then multiplying the matrices as in Eq. (4.3).

Alternatively, for both the interpolating and approximating case the following linear equation

system will yield the required parameters:

$$\begin{aligned} \boldsymbol{A}\boldsymbol{D} + \boldsymbol{W}(\boldsymbol{K} + n\lambda\boldsymbol{I}) &= \boldsymbol{M} \\ \boldsymbol{W}\boldsymbol{D}^T &= 0. \end{aligned}$$

Combining the matrices gives the simple equation:

$$\boldsymbol{G}\boldsymbol{X} = \boldsymbol{B} \tag{4.4}$$

with:

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{K} & \boldsymbol{D} \\ \boldsymbol{D}^T & 0 \end{pmatrix} = \begin{pmatrix} 0 & U(|\boldsymbol{d}_1 - \boldsymbol{d}_2|) & \cdots & U(|\boldsymbol{d}_{n-1} - \boldsymbol{d}_n|) & d_{1,1} & \cdots & d_{1,d-1} & 1 \\ U(|\boldsymbol{d}_1 - \boldsymbol{d}_2|) & 0 & \cdots & U(|\boldsymbol{d}_{n-2} - \boldsymbol{d}_n|) & d_{2,1} & \cdots & d_{2,d-1} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots \\ U(|\boldsymbol{d}_{n-1} - \boldsymbol{d}_n|) & U(|\boldsymbol{d}_{n-2} - \boldsymbol{d}_n|) & \cdots & 0 & d_{n,1} & \cdots & d_{n,d-1} & 1 \\ d_{1,1} & d_{2,1} & \cdots & d_{n,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{1,d-1} & d_{2,d-1} & \cdots & d_{n,d-1} & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{W} \\ \boldsymbol{A} \end{pmatrix}$$

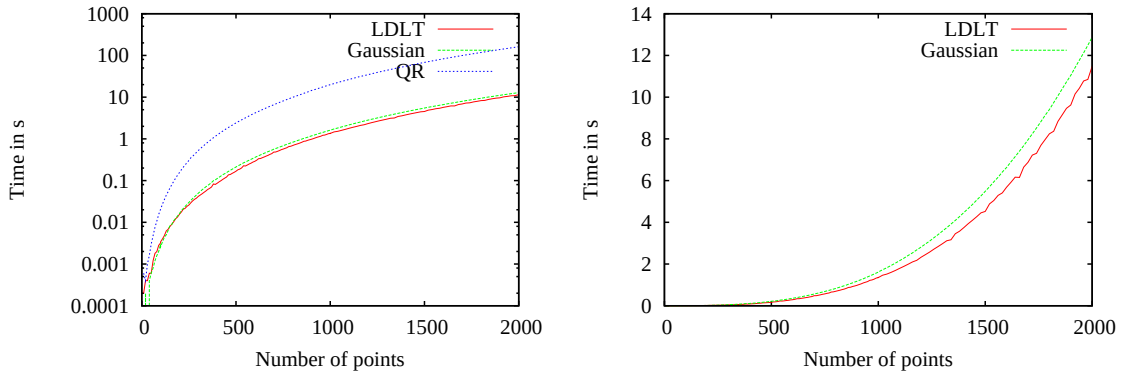
$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{M} \\ 0 \end{pmatrix}.$$

The problem of calculating the TPS functional is therefore reduced to the inversion of the matrix $\boldsymbol{G}$. As this matrix is of size $(n+e) \times (n+e)$ the worst case complexity for solving the equation system is in the order of $O\left((n+e)^3\right)$. Similarly, computing the TPS coefficients using the QR decomposition involves the inversion of $\boldsymbol{Q}_2^T\boldsymbol{K}\boldsymbol{Q}_2$, which is in the order of $O\left((n-e)^3\right)$. As the dimension e is usually much smaller than the number of points n, there might be a small or non-obvious difference in the actual computation time of the algorithms. Therefore the technique for solving the TPS problem has to be chosen carefully. Since $\boldsymbol{G}$ is neither sparse nor positive definite advanced matrix inversion algorithms such as the sparse Cholesky decomposition are not applicable. However the LDL^T decomposition [93] may be used to exploit the symmetric properties of $\boldsymbol{G}$.

The different algorithms for computing the TPS were implemented and are compared in Fig. 4.4. With an increasing number of points the computation time of the QR decomposition explodes. This is due to the 7 additional multiplications of large matrices necessary to compute $\boldsymbol{W}$ and $\boldsymbol{A}$. Some of those multiplications are in the order of $O(n^3)$. The LDL^T decomposition requires no further multiplication and is therefore the fastest approach of all three.

4.3 Approximating the Thin Plate Spline

As seen in the previous section computing the TPS is a time consuming task. This section will discuss two approximation techniques that have been proposed for 2-D shape matching in [37] and for machine learning in [83].



(a) Logarithmic scale (b) Linear scale

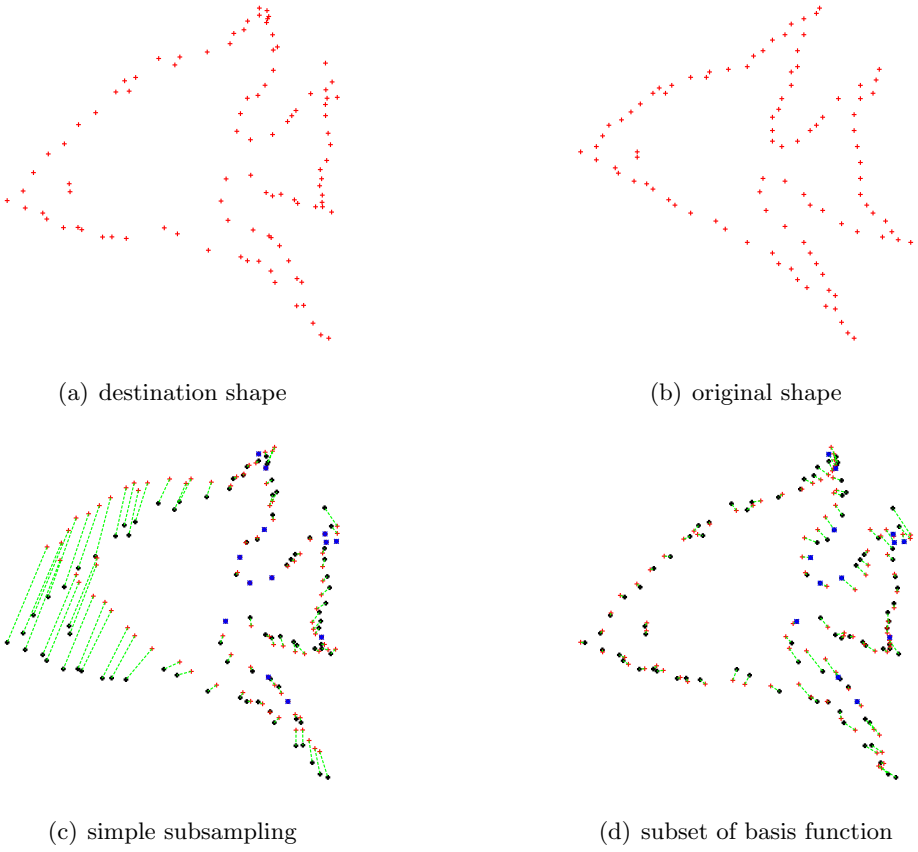
Fig. 4.4: Comparison of three approaches to the computation of the TPS. With increasing number of points the computation time of all algorithms increases exponentially. The QR decomposition approach is by far the slowest of all algorithms. The LDL^T algorithm yields a small decrease in computation time. At about 1000 points the gain in efficiency is significant enough to make the LDL^T a viable alternative.

4.3.1 Subsampling

The most obvious method to reduce computation time is to reduce the number of points by only using a randomly, or otherwise, selected subsample of all corresponding points. Assume the TPS functional that is supposed to be approximated is given by the n landmarks $\boldsymbol{D}$ and $\boldsymbol{M}$. Further assume, without loss of generality, that points in $\boldsymbol{D}$ and $\boldsymbol{M}$ are random, so that the first p entries constitute a randomly selected subsample. Approximating the TPS functional amounts to using the first p points in place of the $\boldsymbol{D}$,$\boldsymbol{M}$ as well as $\boldsymbol{K}_1$ instead of $\boldsymbol{K}$ in Eq. (4.4). $\boldsymbol{K}_1$ is a $p \times p$ submatrix of $\boldsymbol{K}$ given by the following partition:

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_1 & \boldsymbol{K}_2 \\ \boldsymbol{K}_2^T & \boldsymbol{K}_3 \end{pmatrix}, \tag{4.5}$$

with $\boldsymbol{K}_2 \in \mathbb{R}^{p\times(n-p)}$ and $\boldsymbol{K}_3 \in \mathbb{R}^{(n-p)\times(n-p)}$. The TPS coefficients computed in this manner are then used to extrapolate the deformation to the remaining points. Results of this approximation can be seen in Fig. 4.5 for a 2-dimensional example and in Fig. 4.6 for an example in 3 dimensions.



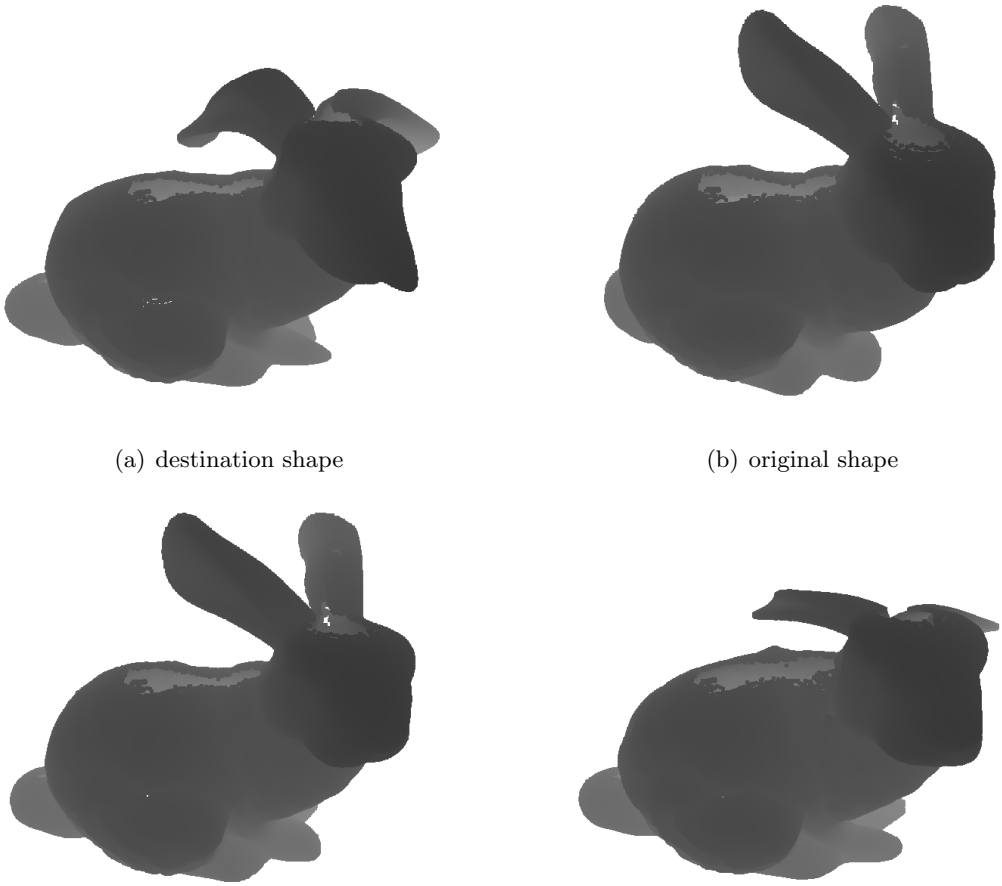
(a) destination shape

(b) original shape

(c) simple subsampling

(d) subset of basis function

Fig. 4.5: A 2-dimensional example of the TPS approximations. (a) shows the desired shape that is to be achieved by deforming (b). Only a subset of all points is used to estimate the whole deformation. These points are marked in blue. (c) The error that occurs in simple subsampling when the samples are not evenly distributed over the whole area. In areas without sufficient samples, the deformation is very far off from the correct shape. Note how the front of the fish in the estimation is actually further away from the desired shape than in the starting point cloud. (d) The approximated deformation using the subset of basis function approach. The approximation was computed using the same subsample as in (a) for the basis functions. The overall error as well as the error in the front of the fish is greatly reduced by considering all points in the approximation. This approximation technique does not necessarily ensure that the subsampled points are exactly interpolated. In this case the error is actually highest in the region nearest to the subsampled points. This may be counterintuitive, but consider that the subsampled points only decide where the basis functions are "located", while all points influence the deformation itself equally. This is a consequence of the new bending energy, that may decide to sacrifice exact interpolation of the subsample in favor of reducing the overall error. The desired points are drawn black, the estimated points in red and the errors and correspondences are marked with green lines.



(a) destination shape

(b) original shape

(c) simple subsampling

(d) subset of basis function

Fig. 4.6: A 3-dimensional example of the TPS approximations using the Stanford Bunny [73]. (a) The desired shape that is to be achieved by deforming the Stanford Bunny (b). As in the 2-dimensional example only a small subset of all points is used to estimate the whole deformation. (c) The estimation using simple sampling. As the samples mostly lie on the body of the bunny, the deformation of the ears and head are not present in the approximation. (d) shows the approximated deformation using the subset of basis function approach with the same sampling as in (a) for the basis functions. The ears as well as the rest of the bunny are deformed in the same vein as the destination shape. Unlike the experiment in 2D (Fig. 4.5) the ears do not match the desired shape as much as the body does. This is due to the fact that the point density near the ears is much less than it is on the body. Therefore the body of the bunny contribute much more to the bending energy than the ears and head.

4.3.2 Basis Function Subset

The approximation can be improved upon by estimating only a subset of the TPS coefficients $\boldsymbol{W}' \in \mathbb{R}^{e \times p}$ while still considering all of the corresponding points $\boldsymbol{D}$ and $\boldsymbol{M}$. The idea is to formulate the solution of the bending energy as a TPS with only p coefficients, while still evaluating the error over each landmark. The bending energy to be minimized is reformulated as:

$$E_\lambda = \frac{1}{2}\left|\boldsymbol{M} - \boldsymbol{K}'\boldsymbol{W}' - \boldsymbol{D}\boldsymbol{A}\right|^2 + \frac{\lambda}{2}\boldsymbol{W}'^T\boldsymbol{A}\boldsymbol{W}'.$$

The minimum is given by the solution of the following linear equation system:

$$\begin{pmatrix} \boldsymbol{K}'^T\boldsymbol{K}' + \lambda\boldsymbol{A} & \boldsymbol{K}'^T\boldsymbol{D} \\ \boldsymbol{D}^T\boldsymbol{K}' & \boldsymbol{D}^T\boldsymbol{D} \end{pmatrix} \begin{pmatrix} \boldsymbol{W}' \\ \boldsymbol{A} \end{pmatrix} = \begin{pmatrix} \boldsymbol{K}'^T\boldsymbol{M} \\ \boldsymbol{D}^T\boldsymbol{M} \end{pmatrix}.$$

Here $\boldsymbol{K}' = \begin{pmatrix} \boldsymbol{K}_1 \boldsymbol{K}_2 \end{pmatrix}$. The results of this approximation can be seen in Fig. 4.5 as well as in Fig. 4.6.

4.3.3 Experimental Evaluation

This section discusses the accuracy as well as the performance of the previously presented approximation approaches. Two experiments were conducted to evaluate both aspect of the two algorithms.

Performance evaluation

The performance of the simple subsampling approximation does not need to be evaluated in a separate experiment. As subsampling to a number of p points is equivalent to computing a TPS with p points the evaluation of the performance of the different TPS computation techniques is sufficient. See Fig. 4.4 for the results.

The performance of the second approximation technique, i.e., the basis function subset approximation is more interesting. In the course of computing this approximation several large matrices are multiplied. The performance of the basis function subset algorithm was evaluated for different percentages P. $P = \frac{p}{n}$ is the ratio of p points that contribute directly to the basis functions to the total number n of points used in the computation. For some representative values of P the computing time required by the approximation algorithm is compared to the exact computation using the LDL^T algorithm and all points. The results are plotted in Fig. 4.7. Note that the x-axis shows the total number of points for all test runs and not the number of points used as a subset for the approximation.

As suspected, the computation time increases as P increases if the total number of points remain constant. The computation time also increases cubically with the number of landmarks. This is not surprising, since the dominating operation in the calculation of the approximation is the inversion of a $(p+d)\times(p+d)$ matrix. Comparing the computation times of the approximation directly with the time of the exact algorithm reveals a much more interesting fact. Using about

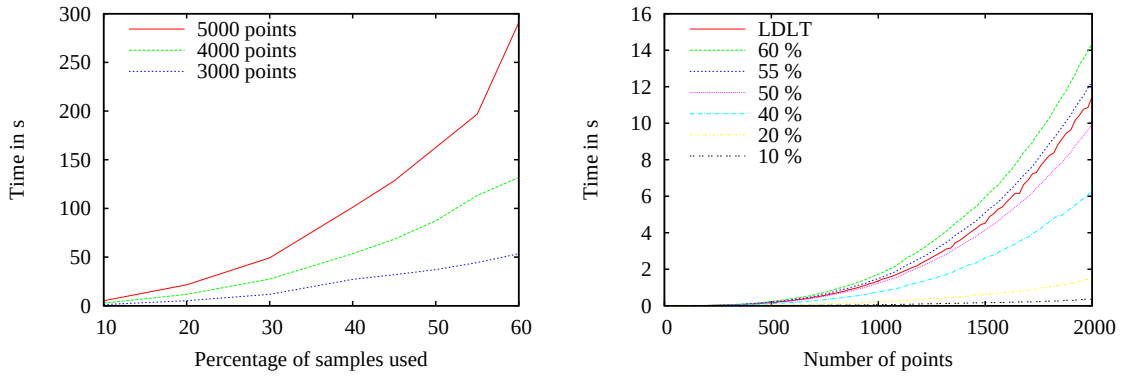


Fig. 4.7: Computing time of the subset of basis function approximation using different amounts of points and basis functions. The increase in computation time with increasing percentages and constant n can be estimated to be almost quadratic. The increase of computation time with static P and increasing n is cubic in Pn. This is expected since the matrix inversion is in the order of $O((Pn+d)^3)$. Unfortunately, the overhead involved in computing the subset of basis function approximation is very large. Even when using only half the points the computation time is as high as computing the exact deformation. Comparing this to the performance of the subsampling approach reveals that the speed benefit is not as high as the first impression may lead to believe.

50 % of the total number of points needs as much time as calculating the exact solution using LDL^T. This holds true with any number of total points. Any $P \geq 50\,\%$ is therefore useless for approximation as the alternative, i.e., exact calculation, is both more accurate and faster. This also means that there is no P for which the approximation is faster than the subsampling that uses $p = Pn$ points. In other words, the subsampling approach will always be faster than the subset of basis function approach, in any fair comparison.

Accuracy evaluation

In order to evaluate the accuracy of the two approximation techniques experiments with a randomly generated deformation were performed. The experiments involve a 3-dimensional Cartesian grid, i.e., a regular grid with unit distances between the individual grid points. All experiments were conducted using a Cartesian grid with side length 15 as in Fig. 4.8. Therefore there were a total number of $n = 2275$ points involved in the computations. The grid was deformed using a randomly generated TPS deformation in the following manner.

1. Randomly choose 100 points $\boldsymbol{D}$ from the Cartesian grid.
2. Apply additional noise to each point $\boldsymbol{d}_i$, i.e., $\boldsymbol{m}_i = \boldsymbol{d}_i + \varepsilon$, where the error ε is uniformly distributed in $[-3, 3]$.
3. Compute the TPS functional T using $\boldsymbol{D}$ and $\boldsymbol{M}$.
4. Apply T to the Cartesian grid.

Both approximation algorithms will attempt to recover the randomly generated deformation T with a varying number of randomly selected landmarks. The landmarks are chosen as points from the Cartesian grid. By construction the corresponding destination point in the deformed grid is well-known. Both algorithms receive the same landmarks. The basis function approximation will of course utilize all 2275 points, however only the subsampled points are used for the computation of matrix $\boldsymbol{K}_1$ (see Eq. (4.5)). The attempts were repeated 10 times for every setting of P with a new random selection of points.

As the correctly deformed grid is known, the error of the approximated TPS functional T' is computed by the mean square error of the approximated grid coordinates and the correctly deformed grid coordinates:

$$E(T') = \frac{1}{n} \sum_{i=1}^{n} \left| T(\boldsymbol{g}_i) - T'(\boldsymbol{g}_i) \right|^2,$$

where $\boldsymbol{g}_i$ is the ith grid coordinate. The results are presented in Fig. 4.9. Unsurprisingly, the error of both approximation techniques increases when less samples are used. The subset of basis function approximation performs best when only a small percentage of points is used. When the percentage of points is larger than approximately 45 the subsampling approach actually outperforms the smarter subset of basis function approach. This, confounded with the findings in the performance evaluation leads to the conclusion that the basis function algorithm is infeasible when $P \geq 45\,\%$. The ideal percentage of used points is about $10\,\%$ to $20\,\%$. In this area the performance gain is major, while the error is still relatively small. In case the total number of points is small enough to comfortably compute the TPS with a subsample of $40\,\%$ or more the simple subsampling is preferred to the subset of basis function approximation.

4.4 Variations on the Thin Plate Spline

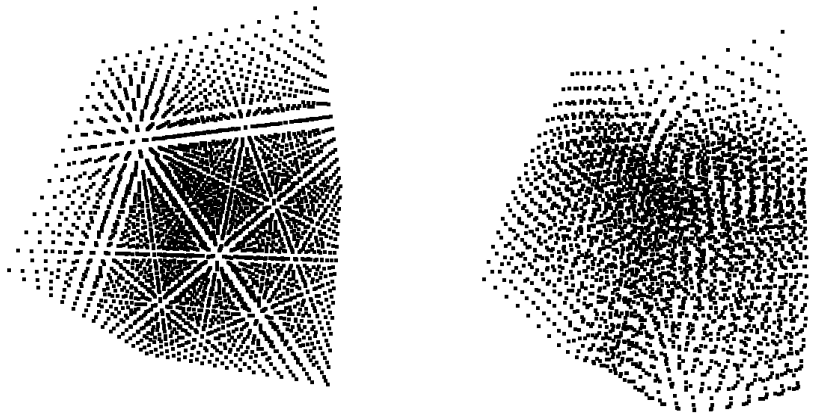
The Thin Plate Spline can be modified to include additional information about the mapping at the corresponding points. For example, if the correspondence of one point pair is known to be more likely to be correct than the correspondence of another point pair an approximating TPS mapping may be more weighted towards interpolation at this landmark. Another example is derivative information, i.e., orientations at certain points, that can be used to constrain the local orientation of the resulting deformation.

4.4.1 Landmark Errors

In order to include statistical information about the uncertainty of landmark localizations, the bending energy (4.2) is restated as:

$$E_\lambda = \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{m}_i - \boldsymbol{T}(\boldsymbol{d}_i) \right)^T \boldsymbol{\Sigma}_i^{-1} \left(\boldsymbol{m}_i - \boldsymbol{T}(\boldsymbol{d}_i) \right) + \lambda J. \tag{4.6}$$

Here the matrices $\boldsymbol{\Sigma}_i$ represent the landmark errors at each landmark i. The bending energy E_λ again contains two terms. The first term measures the error made by the approximating



(a) Regular grid (b) Deformed grid

Fig. 4.8: Left: A regular 3-dimensional Cartesian grid. The Cartesian grid is used to evaluate the accuracy of the presented approximation techniques. The grid in question is regular with unit distances between grid points. It extends 15 units in all 3 directions, i.e., there are $15 \cdot 15 \cdot 15 = 2275$ points. Right: An exemplary deformation that was applied to the Cartesian grid.

TPS, only now each corresponding point pair i is weighted with the covariance matrix Σ_i. The second term remains unchanged and still represents the smoothness of the deformation weighted with the smoothness parameter λ. The bending energy E_λ is minimized by the solution to the following linear equation system:

$$\begin{aligned} \boldsymbol{A'D'} + \boldsymbol{W'}(\boldsymbol{K'} + n\lambda\boldsymbol{\Sigma}) &= \boldsymbol{M'} \\ \boldsymbol{W'}\boldsymbol{D'}^T &= 0, \end{aligned}$$

where $\boldsymbol{\Sigma}$ is a block-diagonal $\big(n \cdot (d-1)\big) \times \big(n \cdot (d-1)\big)$ matrix that contains all landmark errors $\boldsymbol{\Sigma}_i$, i.e., $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_n)$. As a necessary consequence of the $d-1 \times d-1$ matrices $\boldsymbol{\Sigma}_i$ the equation system (4.4) increases in size by a factor of $d-1$. $\boldsymbol{A'} = (a_{1,1}, \cdots, a_{1,d}, \cdots, a_{e,d})^T$ and $\boldsymbol{W'} = (\boldsymbol{w}_1^T, \cdots, \boldsymbol{w}_n^T)^T$ are now ed and dn-dimensional vectors, respectively. Consequently, $\boldsymbol{K'} \in \mathbb{R}^{dn \times dn}$,$\boldsymbol{D'} \in \mathbb{R}^{nd \times dd}$ and $\boldsymbol{M'} \in \mathbb{R}^{en}$ are given by:

$$\begin{aligned} \boldsymbol{K'} &= (K_{i,j}\boldsymbol{I}_{d-1}) \\ \boldsymbol{D'} &= (D_{i,j}\boldsymbol{I}_{d-1}) \\ \boldsymbol{M'} &= (\boldsymbol{m}_1^T, \cdots, \boldsymbol{m}_n^T)^T. \end{aligned}$$

This means there are a total number of $nd+ed$ equations, i.e., a matrix of size $(nd+ed) \times (nd+ed)$ needs to be inverted to calculate the TPS coefficients. Additionally the incorporation of the covariance matrices will only have an effect on the TPS if the smoothing parameter λ does not

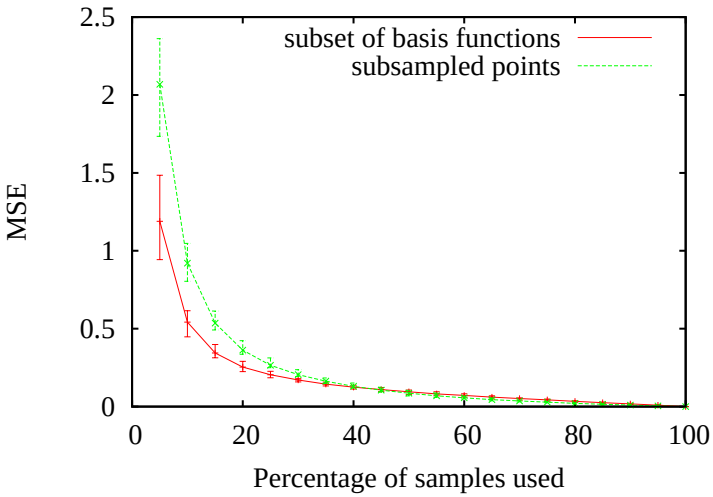


Fig. 4.9: Comparison of the accuracy of the approximated TPS functionals. The accuracy is measured as the mean square error of the approximated grid as compared to the correctly deformed grid. The error is dimensionless, i.e., a mean square error of 1 is the same distance a grid point has to its nearest neighbors in the regular Cartesian grid. On the whole, the accuracy of the subset of basis function approximation is significantly better then that of the simple subsampling approach. At $P = 45\,\%$ and greater the subsampling approach performs better. Even though this is surprising, the effective difference in this area is relatively small.

equal 0. The heavy computational burden of the increased matrix dimension can be minimized in case all $\boldsymbol{\Sigma}_i$ are diagonal matrices that are trivially formed by a single value on the diagonal, i.e., $\boldsymbol{\Sigma}_i = \sigma_i \boldsymbol{I}_{d-1}$. In this case the equation system is:

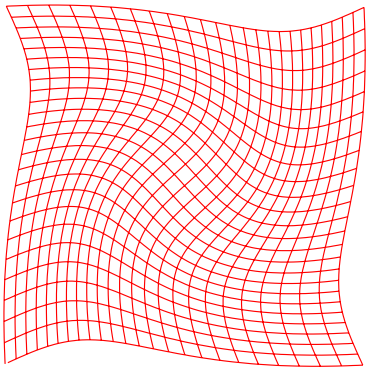
$$\begin{aligned} \boldsymbol{A}\boldsymbol{D} + \boldsymbol{W}(\boldsymbol{K} + n\lambda\boldsymbol{\Sigma}) &= \boldsymbol{M} \\ \boldsymbol{W}\boldsymbol{D}^T &= 0, \end{aligned}$$

with $\boldsymbol{\Sigma}' = \mathrm{diag}(\sigma_1, \cdots, \sigma_n)$ and the familiar values for $\boldsymbol{K}$,$\boldsymbol{A}$, $\boldsymbol{W}$,$\boldsymbol{D}$ and $\boldsymbol{M}$. See Fig. 4.10 for an example of landmark errors.

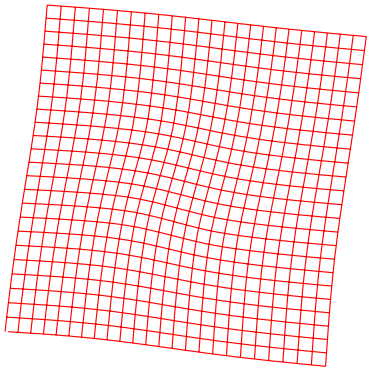
4.4.2 Orientation Attributes

Based on the works of Mardia et al. [60] and Bookstein and Green [14] a novel way to incorporate orientation information into the TPS functional was formulated by Rohr et al. [74]. In contrast to previous work on orientations within TPS functions this formulation is independent of the length of the orientation vector and is applicable to both the interpolation and approximation TPS.

Assume that there are orientations to be matched at a set of corresponding points. These n_θ additional orientation landmarks are denoted by the matrices $\boldsymbol{D}_\theta = (\boldsymbol{d}_{\theta 1}, \cdots, \boldsymbol{d}_{\theta n_\theta})^T$ and $\boldsymbol{M}_\theta = (\boldsymbol{m}_{\theta 1}, \cdots, \boldsymbol{m}_{\theta n_\theta})^T$. Each pair $(\boldsymbol{d}_{\theta i}, \boldsymbol{m}_{\theta i})$ of corresponding points is associated with two direction vectors $\boldsymbol{p}_i \in \mathbb{R}^{d-1}$ and $\boldsymbol{q}_i \in \mathbb{R}^{d-1}$, where the direction $\boldsymbol{p}_i$ is mapped to the direction



(a) $\lambda = 500$, $\sigma_1, \cdots, \sigma_8 = 1$



(b) $\lambda = 500$, $\sigma_1, \cdots, \sigma_4 = 5$,$\sigma_5, \cdots, \sigma_8 = 1$

Fig. 4.10: Left: The same deformed grid example as in Fig. 4.3. λ is set to 500, while all landmark errors are uniformly set to 1. Right: The same deformation as on the left side, with the exception that the landmark errors for the outer 4 points are set to 5.

$\boldsymbol{q}_i$. Note that due to the inherent nature of the orientation formulation the dimension d of the domain and e of the codomain of the TPS function must be related by $e = d - 1$. Since d includes the homogeneous coordinate the actual dimensions of input space and output space are the same. See Fig. 4.11 for a simple example of the orientation constraints that may be used in the newly formulated TPS.

In order to account for the matching criterion of the orientations in the bending energy functional the expression $\boldsymbol{p}_i^T \nabla \boldsymbol{T}(\boldsymbol{d}_{\theta i})$ is required. This term represents the derivative of the TPS functional $\boldsymbol{T}$ at the point $\boldsymbol{d}_{\theta i}$ in the direction of $\boldsymbol{p}_i^T$. In addition, $d-2$ vectors $\boldsymbol{q}_{i,k}^{\perp}$ are necessary for each orientation $\boldsymbol{q}_i$. Each $\boldsymbol{q}_{i,k}^{\perp}$ is orthogonal to $\boldsymbol{q}_i$ and every other $\boldsymbol{q}_{i,k}^{\perp}$. Now the derivative $\boldsymbol{p}_i^T \nabla \boldsymbol{T}(\boldsymbol{d}_{\theta i})$ is required to be perpendicular to the vectors $\boldsymbol{q}_{i,k}^{\perp}$. This is achieved by letting the scalar product of both vectors contribute to the bending energy E_λ. Since the scalar product is zero if both vectors are perpendicular irrespectively of their length, the orientations are valid even if they are not unit vectors. This length independence property only holds in the interpolation case. In an approximate TPS the orientations may not match exactly, therefore the penalty brought on by the scalar product is dependent upon the length of the orientations.

The new bending energy that incorporates orientations as well as landmark errors is given by:

$$E_\lambda = \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{m}_i - \boldsymbol{T}(\boldsymbol{d}_i)\right)^T \boldsymbol{\Sigma}_i^{-1} \left(\boldsymbol{m}_i - \boldsymbol{T}(\boldsymbol{d}_i)\right) + \frac{c}{n_\theta (d-2)} \sum_{i=1}^{n_\theta} \sum_{k=1}^{d-2} \left(\boldsymbol{p}_i^T \nabla T^T(\boldsymbol{d}_{\theta i}) \boldsymbol{q}_{i,k}^{\perp}\right)^2 + \lambda J.$$

This functional is the bending energy as given in Eq. (4.6) with an additional term that accounts for the orientation constraints. The parameter c weights the orientation term with respect to the landmark error term as well as the smoothness constraint. It is an additional smoothness

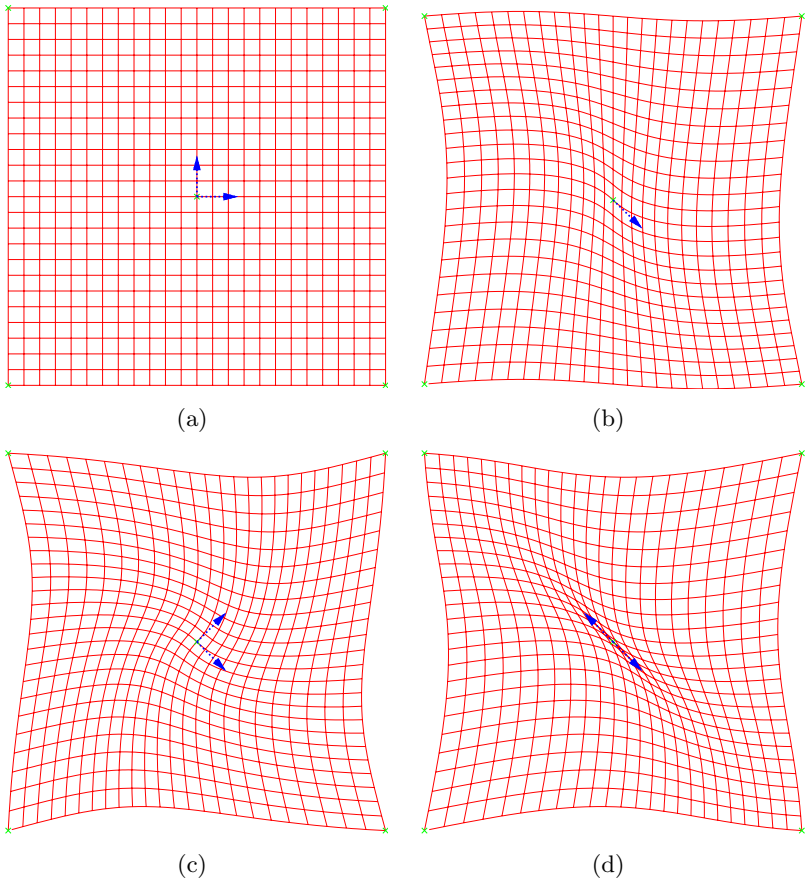
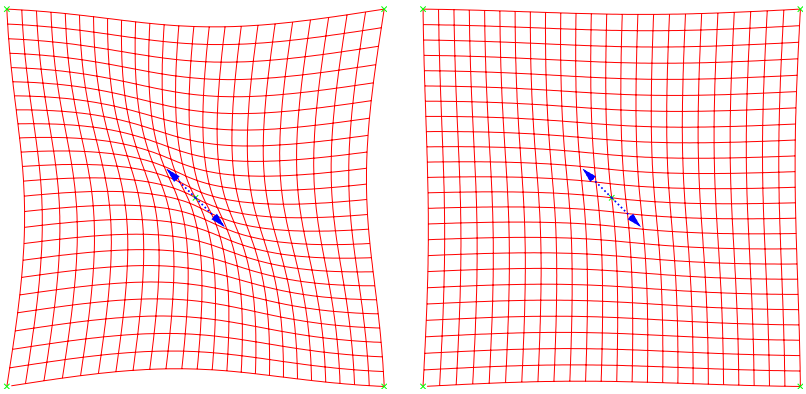


Fig. 4.11: Demonstration of orientation attributes in the TPS functional. The orientation of the central point is mapped according to the arrows indicated in blue. (a) The starting grid and orientations. (b) Only a single orientation (pointing right) is used in this mapping. The other orientation is left unaffected. (c) Both orientations are rotated by 45 °. Because the corners of the grid are held in place, the change in the orientation is local to the central point. (d) Orientations of a single landmark may be mapped to seemingly inconsistent destination orientations. Here perpendicular orientations are constrained to lie on a line.



(a) $\lambda = 1$, $c = 0.01$ (b) $\lambda = 1$, $c = 0.001$

Fig. 4.12: The same setup as in Fig. 4.11. These images demonstrate the effects of the smoothing parameter c. The parameter *lambda* is set to one. Aside from enabling effects of c this has no further effect itself because there is no positional mapping that can be approximated. The higher c, the more the orientations are interpolated, i.e., in case $c = \infty$ the orientations are fully interpolated. As c approaches zero the orientations are less and less important for the bending energy.

parameter that also determines whether the TPS approximates or interpolates the orientations. See Fig. 4.12 for the effects of c on the resulting TPS.

In addition to the modified bending energy, the TPS function $\boldsymbol{T}$ itself has to be modified to incorporate orientations. $\boldsymbol{T}$ is formulated as:

$$\begin{aligned}\boldsymbol{T}(\boldsymbol{x}) =& \boldsymbol{A}\boldsymbol{x} + \sum_{i=1}^{n} \boldsymbol{w}_i U(|\boldsymbol{d}_i - \boldsymbol{x}|) \\ & - \sum_{i=1}^{n_\theta} \sum_{k=1}^{d-2} w'_{i,k} \boldsymbol{p}_i^T \nabla U(|\boldsymbol{x} - \boldsymbol{d}_{\theta i}|) \boldsymbol{q}_{i,k}^{\perp},\end{aligned}$$

where $\nabla U(|\boldsymbol{x} - \boldsymbol{d}_{\theta i}|)$ is the derivative of U with respect to $\boldsymbol{x} - \boldsymbol{d}_{\theta i}$. Furthermore, the new TPS has n_θ additional coefficients $\boldsymbol{w}_i'^T = (w'_{i,1}, \cdots, w'_{i,d-2})$. As with the TPS that incorporates landmark errors the coefficients are collected in a vector $\boldsymbol{W'} = (\boldsymbol{w}_1^T, \cdots, \boldsymbol{w}_n^T, \boldsymbol{w}_1'^T, \cdots, \boldsymbol{w}_{n_\theta}'^T)^T$ and the vector $\boldsymbol{A'} = (\boldsymbol{a}_1^T, \cdots, \boldsymbol{a}_d^T)^T$. These coefficients are computed by solving the linear system of equations:

$$\begin{aligned}\boldsymbol{A'D'} + \boldsymbol{W'K'} &= \boldsymbol{M'} \\ \boldsymbol{W'D'}^T &= 0,\end{aligned}$$

with the $(ne + n_2) \times ed$ matrix $\boldsymbol{D'}$ given by:

$$\begin{aligned}
\boldsymbol{D'} &= \begin{pmatrix} \boldsymbol{D'}_1 \\ \boldsymbol{D'}_2 \end{pmatrix} \\
\boldsymbol{D'}_1 &= (\boldsymbol{D}_{i,j}\boldsymbol{I}_{d-1}) \\
\boldsymbol{D'}_2 &= (\boldsymbol{D'}_{2,1}, \cdots, \boldsymbol{D'}_{2,n_\theta})^T \\
\boldsymbol{D'}_{2,i} &= \left(\boldsymbol{0}, p_{i,1} \cdot \begin{pmatrix} \boldsymbol{q}^\perp_{i,1} & \cdots & \boldsymbol{q}^\perp_{i,d-2} \end{pmatrix}^T, \cdots, p_{i,d-1} \cdot \begin{pmatrix} \boldsymbol{q}^\perp_{i,1} & \cdots & \boldsymbol{q}^\perp_{i,d-2} \end{pmatrix}^T\right)^T.
\end{aligned}$$

$\boldsymbol{M'} = \left(\boldsymbol{m}_1^T, \cdots, \boldsymbol{m}_n^T, 0, \cdots, 0\right)^T$ is an $(n(d-1) + n_2)$-dimensional vector with n_2 zeros at the end. The formulation of matrix $\boldsymbol{K'}$ is somewhat more involved:

$$\boldsymbol{K'} = \begin{pmatrix} \boldsymbol{K'}_1 + n\lambda\boldsymbol{\Sigma} & \boldsymbol{K'}_2 \\ \boldsymbol{K'}_2^T & \boldsymbol{K'}_3 + n_2\frac{\lambda}{c}\boldsymbol{I}_{n_2} \end{pmatrix},$$

where $n_2 = n_\theta(d-2)$ is introduced to simplify the notations and $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_n)$ is the same concatenation of the covariances for the landmark errors. The submatrices $\boldsymbol{K'}_1$,$\boldsymbol{K'}_2$ and $\boldsymbol{K'}_3$ are given by:

$$\begin{aligned}
\boldsymbol{K'}_1 &= (K_{i,j}\boldsymbol{I}_{d-1}) = (U(\boldsymbol{d}_i, \boldsymbol{d}_j)\boldsymbol{I}_{d-1}) \\
\boldsymbol{K'}_2 &= \left(-\boldsymbol{p}_j^T \nabla U(|\boldsymbol{d}_i - \boldsymbol{d}_{\theta j}|) \cdot \begin{pmatrix} \boldsymbol{q}^\perp_{j,1} & \cdots & \boldsymbol{q}^\perp_{j,d-2} \end{pmatrix}\right) \\
\boldsymbol{K'}_3 &= \left(-\boldsymbol{p}_j^T \nabla \boldsymbol{p}_i^T \nabla U(|\boldsymbol{d}_{\theta i} - \boldsymbol{d}_{\theta j}|) \cdot \boldsymbol{Q}_{i,j}\right) \\
\boldsymbol{Q}_{i,j} &= \begin{pmatrix} \boldsymbol{q}^{\perp T}_{i,1}\boldsymbol{q}^\perp_{j,1} & \cdots & \boldsymbol{q}^{\perp T}_{i,1}\boldsymbol{q}^\perp_{j,d-2} \\ \vdots & \ddots & \vdots \\ \boldsymbol{q}^{\perp T}_{i,d-2}\boldsymbol{q}^\perp_{j,1} & \cdots & \boldsymbol{q}^{\perp T}_{i,d-2}\boldsymbol{q}^\perp_{j,d-2} \end{pmatrix}.
\end{aligned}$$

The term $\boldsymbol{p}_j^T \nabla \boldsymbol{p}_i^T \nabla U(|\boldsymbol{d}_{\theta i} - \boldsymbol{d}_{\theta j}|)$ is the derivative of $\nabla U(|\boldsymbol{d}_{\theta i} - \boldsymbol{d}_{\theta j}|)$ with respect to $\boldsymbol{d}_{\theta i} - \boldsymbol{d}_{\theta j}$ in the direction of $\boldsymbol{p}_j^T$, i.e., the derivative multiplied by $\boldsymbol{p}_j^T$. It is evident from these equations that the computation of a TPS functional that incorporates orientations is computationally expensive. As with the TPS that only includes landmark errors, the number of equations to solve is increased by a factor of $d-1$. Unlike before however, the orientation sensitive TPS cannot be simplified in order to reduce the number of equations. As matrix inversion is in $O(n^3)$, incorporating orientations in a deformation of a 3-dimensional shape would increase the computation time by about a factor of 27.

To give the reader a better overview on the calculations involved, the matrices for the simple 2D case are given in a more extensive fashion. The 3 and more dimensional cases should then be easier to generalize from the above equations. The corresponding points and orientations $\boldsymbol{D'}$,

$\boldsymbol{M}'$ in the 2D case are given by:

$$\boldsymbol{D}' = \begin{pmatrix} 1 & 0 & d_{1,x} & 0 & d_{1,y} & 0 \\ 0 & 1 & 0 & d_{1,x} & 0 & d_{1,y} \\ & & & \vdots & & \\ 1 & 0 & d_{n,x} & 0 & d_{n,y} & 0 \\ 0 & 1 & 0 & d_{n,x} & 0 & d_{n,y} \\ 0 & 0 & p_{1,x}q_{1,x}^{\perp} & p_{1,x}q_{1,y}^{\perp} & p_{1,y}q_{1,x}^{\perp} & p_{1,y}q_{1,y}^{\perp} \\ & & & \vdots & & \\ 0 & 0 & p_{n_\theta,x}q_{n_\theta,x}^{\perp} & p_{n_\theta,x}q_{n_\theta,y}^{\perp} & p_{n_\theta,y}q_{n_\theta,x}^{\perp} & p_{n_\theta,y}q_{n_\theta,y}^{\perp} \end{pmatrix}$$

$$\boldsymbol{M}' = \Big(m_{1,x}, m_{1,y}, \cdots, m_{n,x}, m_{n,y}, 0, \cdots, 0\Big)^T.$$

Assuming we wish to calculate the TPS using the kernel $U(r) = |r|^3$ the essential matrices of the equation system are given by:

$$\boldsymbol{K}_1' = \begin{pmatrix} 0 & 0 & U(|\boldsymbol{d}_1 - \boldsymbol{d}_2|) & 0 & & U(|\boldsymbol{d}_1 - \boldsymbol{d}_n|) & 0 \\ 0 & 0 & 0 & U(|\boldsymbol{d}_1 - \boldsymbol{d}_2|) & \cdots & 0 & U(|\boldsymbol{d}_1 - \boldsymbol{d}_n|) \\ U(|\boldsymbol{d}_1 - \boldsymbol{d}_2|) & 0 & 0 & 0 & \cdots & U(|\boldsymbol{d}_2 - \boldsymbol{d}_n|) & 0 \\ 0 & U(|\boldsymbol{d}_1 - \boldsymbol{d}_2|) & 0 & 0 & & 0 & U(|\boldsymbol{d}_2 - \boldsymbol{d}_n|) \\ & \vdots & \vdots & & \ddots & \vdots & \vdots \\ U(|\boldsymbol{d}_1 - \boldsymbol{d}_n|) & 0 & U(|\boldsymbol{d}_2 - \boldsymbol{d}_n|) & 0 & \cdots & 0 & 0 \\ 0 & U(|\boldsymbol{d}_1 - \boldsymbol{d}_n|) & 0 & U(|\boldsymbol{d}_2 - \boldsymbol{d}_n|) & \cdots & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{K}_2' = \begin{pmatrix} -\boldsymbol{p}_1^T \nabla U(|\boldsymbol{d}_1 - \boldsymbol{d}_{\theta 1}|) \cdot \boldsymbol{q}_{1,1}^{\perp} & \cdots & -\boldsymbol{p}_{n_\theta}^T \nabla U(|\boldsymbol{d}_1 - \boldsymbol{d}_{\theta n_\theta}|) \cdot \boldsymbol{q}_{n_\theta,1}^{\perp} \\ \vdots & \ddots & \vdots \\ -\boldsymbol{p}_1^T \nabla U(|\boldsymbol{d}_n - \boldsymbol{d}_{\theta 1}|) \cdot \boldsymbol{q}_{1,1}^{\perp} & \cdots & -\boldsymbol{p}_{n_\theta}^T \nabla U(|\boldsymbol{d}_n - \boldsymbol{d}_{\theta n_\theta}|) \cdot \boldsymbol{q}_{n_\theta,1}^{\perp} \end{pmatrix}$$

$$\boldsymbol{K}_3' = \begin{pmatrix} -\boldsymbol{p}_1^T \nabla \boldsymbol{p}_1^T \nabla U(|\boldsymbol{d}_{\theta 1} - \boldsymbol{d}_{\theta 1}|) \cdot \boldsymbol{q}_{1,1}^{\perp T} \boldsymbol{q}_{1,1}^{\perp} & \cdots & -\boldsymbol{p}_{n_\theta}^T \nabla \boldsymbol{p}_1^T \nabla U(|\boldsymbol{d}_{\theta 1} - \boldsymbol{d}_{\theta n_\theta}|) \cdot \boldsymbol{q}_{1,1}^{\perp T} \boldsymbol{q}_{n_\theta,1}^{\perp} \\ \vdots & \ddots & \vdots \\ -\boldsymbol{p}_1^T \nabla \boldsymbol{p}_{n_\theta}^T \nabla U(|\boldsymbol{d}_{\theta n_\theta} - \boldsymbol{d}_{\theta 1}|) \cdot \boldsymbol{q}_{n_\theta,1}^{\perp T} \boldsymbol{q}_{1,1}^{\perp} & \cdots & -\boldsymbol{p}_{n_\theta}^T \nabla \boldsymbol{p}_{n_\theta}^T \nabla U(|\boldsymbol{d}_{\theta n_\theta} - \boldsymbol{d}_{\theta n_\theta}|) \cdot \boldsymbol{q}_{n_\theta,1}^{\perp T} \boldsymbol{q}_{n_\theta,1}^{\perp} \end{pmatrix}$$

The entries in $\boldsymbol{K}_2'$ and $\boldsymbol{K}_3'$ are explained by the first and second derivative of the kernel U:

$$U(\boldsymbol{r}) = |\boldsymbol{r}|^3$$

$$\nabla U(\boldsymbol{r}) = \begin{pmatrix} 3r_x |\boldsymbol{r}| \\ 3r_y |\boldsymbol{r}| \end{pmatrix}$$

$$\nabla \boldsymbol{p}^T \nabla U(\boldsymbol{r}) = \begin{pmatrix} \frac{3p_y r_x r_y + 3p_x\left(2r_x^2 + r_y^2\right)}{|\boldsymbol{r}|} \\ \frac{3p_x r_x r_y + 3p_y\left(2r_x^2 + r_y^2\right)}{|\boldsymbol{r}|} \end{pmatrix}.$$

Chapter 5

Plane Extraction

Most man-made environments feature a large number of planar surfaces. To reduce the systematic errors in laser scans taken in these environments it is useful to take advantage of this structural characteristic. This chapter deals with the application of the Hough Transform for plane extraction. The Hough Transform is chosen over other plane detection methods as it reliably terminates in a reasonable time without fine-tuning of parameters and has lead to good results in other applications. In the following we give a description of the general Hough Transform. Afterwards a few implementations of the Hough Transform are presented. Out of these several algorithms were implemented and evaluated for this thesis. The evaluation in Section 5.4 aims at finding the best procedure to detect an arbitrary number of planes in a point cloud.

5.1 Hough Transform

The Hough Transform [45] is a method for the detection of parametrized objects. Typically used for lines and circles, in this thesis we will focus on the detection of planes in 3D point clouds.

Planes are commonly represented by the signed distance ρ to the origin of the coordinate system and the slope m_x in direction of the x-axis and m_y in the direction of the y-axis, respectively:

$$z = m_x x + m_y y + \rho.$$

To avoid the arising problems due to infinite slopes when trying to represent vertical planes another usual definition uses normal vectors. A plane is thereby given by a point $\boldsymbol{p}$ on the plane, the normal vector $\boldsymbol{n}$ that is perpendicular to the plane and the distance ρ to the origin

$$\boldsymbol{p} \cdot \boldsymbol{n} = \rho. \tag{5.1}$$

Expanding Eq. (5.1) yields

$$p_x n_x + p_y n_y + p_z n_z = \rho.$$

Taking the angles between the normal vector and the coordinate system into consideration the coordinates of $\boldsymbol{n}$ can be factorized to

$$p_x \cdot \cos\theta \cdot \sin\theta + p_y \cdot \sin\varphi \cdot \sin\theta + p_z \cdot \cos\varphi = \rho, \tag{5.2}$$

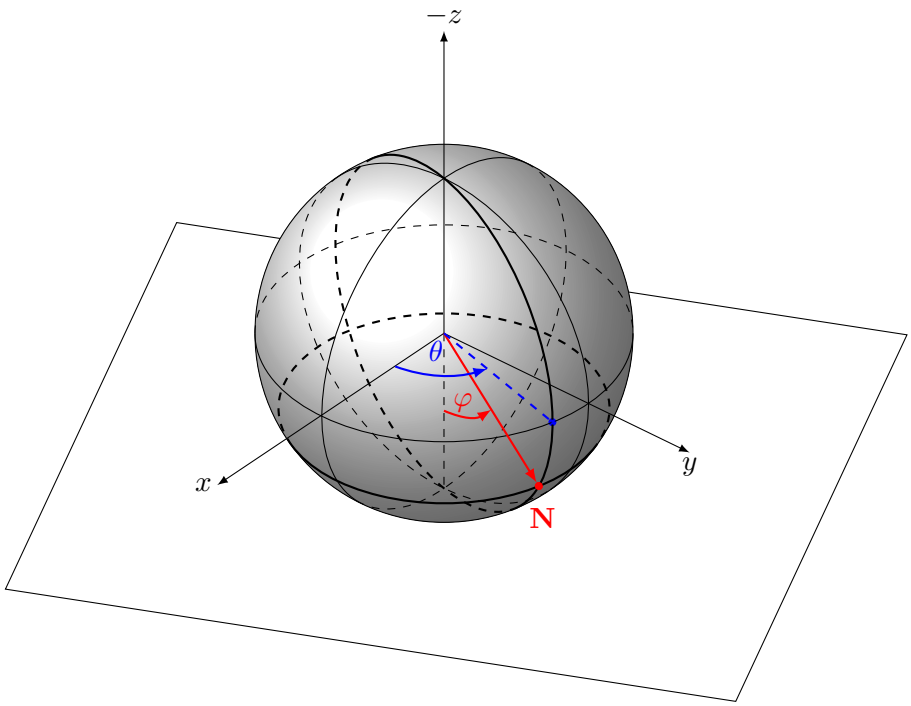


Fig. 5.1: Normal vector described by polar coordinates.

with θ the angle of the normal vector on the xy-plane and φ the angle between the xy-plane and the normal vector in z direction as depicted in Fig. 5.1.

φ, θ and ρ define the 3-dimensional Hough space (θ, φ, ρ space) such that each point in the Hough space corresponds to one plane in $\mathbb{R}^3$.

To find planes in a set of points we need to calculate the Hough Transform for each point. Given a point $\boldsymbol{p}$ in Cartesian coordinates, we have to find all the planes the point lies on, i. e. find all the θ, ϕ and ρ that satisfy Eq. (5.2).

Marking these points in Hough Space leads to a sinusoid curve as depicted in Fig. 5.2 for $\boldsymbol{p}_1 = (0, 0, 1)$, $\boldsymbol{p}_2 = (0, 1, 0)$ and $\boldsymbol{p}_3 = (1, 0, 0)$. Each colored curve represents the Hough Transform for one of these points. The intersections of two curves in Hough space denote the planes that are rotated around the line built by the two points. Consequently, the intersection of three curves in Hough space corresponds to the polar coordinates defining the plane spanned by the three points. In Fig. 5.2 the intersection is marked in black.

Given a set of points P in Cartesian coordinates, we have to transform all points $\boldsymbol{p}_i \in P$ into Hough Space. The more curves intersect in $\boldsymbol{h}_j \in (\theta, \varphi, \rho)$, the more points lie on the plane represented by $\boldsymbol{h}_j$ and the higher is the probability that $\boldsymbol{h}_j$ can actually be extracted from P.

5.1.1 Standard Hough Transform

For practical applications we now have to transform each point into Hough Space. For this purpose, the Hough Space is discretized with ρ', φ' and θ' denoting the extend of each cell in the

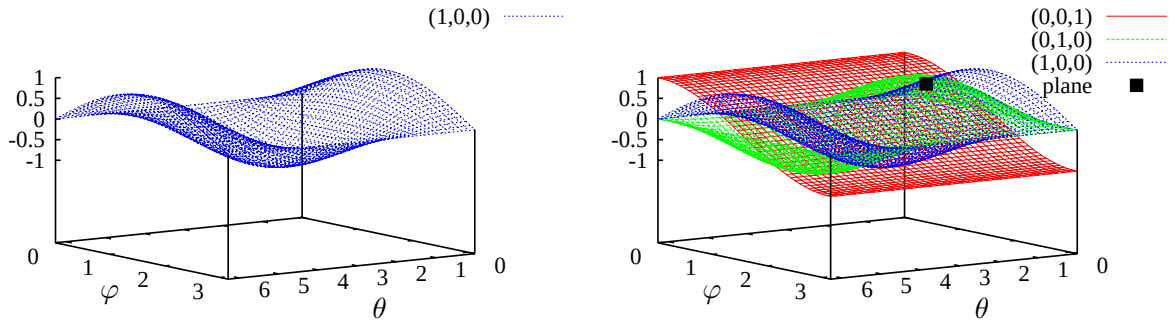


Fig. 5.2: Transformation of three points from $\mathbb{R}$ to Hough space (θ, φ, ρ). The intersection of the curves (marked in black) depicts the plane spanned by the three points.

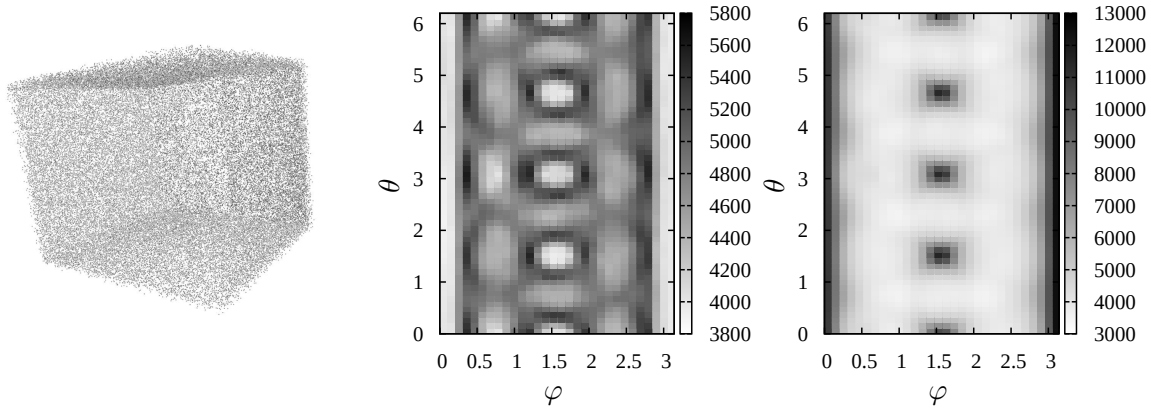


Fig. 5.3: Example of the Standard Hough Transform. The Hough Space is discretized with $\rho' = 20\,\text{cm}$, $\varphi' = 6\,^\circ$ and $\theta' = 6\,^\circ$. Left: Point cloud used for the example, a cube in the origin that is aligned parallel to the axes of the coordinate system. Each side has a length of 400 cm. Middle: Hough Transform of all the points. Depicted is the slice of the Hough Space with $120\,\text{cm} \leq \rho < 140\,\text{cm}$. The color corresponds to the accumulation value, i.e., dark colors mean a large number of points. Right: Hough Transform of all the points. Depicted is the slice of the Hough Space with $180\,\text{cm} \leq \rho < 200\,\text{cm}$. This distance matches the distances of the cube to the origin. You can clearly see 6 peaks in the image.

according direction in Hough Space [29]. A data structure is needed to store all these discretized cells and a score parameter for every single cell. This data structure, called the accumulator, is described in more detail in Section 5.2. For each point p_i we accumulate all the cells, that are touched by its Hough Transform. The incrementation process is often referred to as voting, i.e., each point votes for all sets of parameters (ρ, φ, θ) that define a plane on which it can lie. The cells with the highest values represent the most prominent planes, the plane that covers the most points of the point cloud.

Once all points have voted, the winning planes have to be determined. Due to the discretization of the Hough Space and the noise in the input data it is expedient to search not only for one cell with a high score but for the maximum sum in a small region of the accumulator. A standard practice that considers this problem is the sliding window procedure for peak-detection [52]. In this technique a small 3-dimensional window is defined that is designed to cover the full peak spread. The most prominent plane is the plane that corresponds to the center point of a cube in Hough Space with a maximum sum of accumulation values. The steps of the algorithm are outlined in Algorithm 2.

Algorithm 2 Standard Hough Transform

```
1: for all points p_i in point set P do
2:   for all cells (ρ, φ, θ) in accumulator A do
3:     if point p_i lies on the plane defined by (ρ, φ, θ) then
4:       accumulate cell A(ρ, φ, θ)
5:     end if
6:   end for
7: end for
8: Search for the most prominent cells in the accumulator, that define the detected planes in
   P
```

Fig. 5.3 illustrates an example of the Standard Hough Transform (SHT). Each point of the point cloud on the left is transformed into Hough Space. The 60000 points are arranged as a cube in the origin of the coordinate system with a side length of 400 cm. The images in the middle and on the right show exemplary to slices of the resulting discretized Hough Space. Darker colors mean higher number of accumulation, i.e., the corresponding plane is represented by more points. The depicted slices represent the distances $120\,\text{cm} \leq \rho < 140\,\text{cm}$ (middle) and $180\,\text{cm} \leq \rho < 200\,\text{cm}$ (right) to the origin. The peaks in the right image correspond to the faces of the cube. The planes in the middle image have a smaller distance to the origin. The most points at this distance lie on the planes, that are not parallel to the coordinate axes. Comparing both images demonstrates that the planes corresponding to the faces of the cube have higher peaks (up to 13000) points than the peaks in the other slice.

Due to its high computation time (cf. Section 5.4) the Standard Hough Transform is rather impractical, especially for real-time applications. Therefore numerous variations of the Hough Transform have been devised. Illingworth et al. give a survey on the early development and applications of the Hough Transform [48]. Kälviäinen et al. compare modified versions of the Hough Transform that were proposed to make the algorithm more practical [50]. Some of those procedures are described in the following subsections. Section 5.4 provides an evaluation of a

selection of these methods with the goal to find the optimal variation for the task of detecting a previously unknown number of planes from a 3D point cloud.

5.1.2 Probabilistic Hough Transform

The Standard Hough Transform is performed in two stages. In the first stage all points p_i from the point set P are transformed into Hough Space and the according cells in the accumulator are incremented. The second stage is a search for the highest peaks in the accumulator. The required runtime consists of $O(|P| \cdot N_\varphi \cdot N_\theta)$ operations for transforming the points into Hough Space. N_φ is the number of cells in direction of φ, N_θ the number of cells in direction of θ and N_ρ in direction of ρ, respectively. The search phase needs $O(N_\rho \cdot N_\varphi \cdot N_\theta)$ operations. Since the size of the point cloud $|P|$ is usually much larger than the number $N_\rho \cdot N_\varphi \cdot N_\theta$ of cells in the accumulator array, major improvements concerning computational expenses can be made by reducing the number of points rather than by adjusting the discretization of the Hough Space. For this reason Kiryati et al. propose a probabilistic method for selecting a subset of points from the original point set [52]. They suggest that similar results can be achieved if performing the Hough Transform with a small subset compared to using the entire point cloud.

For the Probabilistic Hough Transform (PHT) m points ($m < |P|$) are randomly selected from the point cloud P. These points are transformed into Hough Space and vote for all the cells that correspond to a plane the points may lie on. The dominant part of the runtime is proportional to $m \cdot N_\varphi \cdot N_\theta$. By reducing m, the runtime can be reduced drastically. However, to obtain the same good results as with the Standard Hough Transform it is important that a feature that is detected using all points is still detectable with a high probability when only using a subset of m points. The adaption of the Standard Hough Transform to the PHT is outlined in Algorithm 3.

Algorithm 3 Probabilistic Hough Transform

```
1: determine m and T
2: randomly select m points to create P^m ⊂ P
3: for all points p_i^m in point set P^m do
4:   for all cells (ρ, φ, θ) in accumulator A do
5:     if point p_i lies on the plane defined by (ρ, φ, θ) then
6:       accumulate cell A(ρ, φ, θ)
7:     end if
8:     Search for the most prominent cells in the accumulator, that define the detected planes
       in P
9:   end for
10: end for
```

The optimal choice of the size m of the subset and the threshold T are dependent on the problem. Several factors account for the quality of the result. First, sensor noise leads to planes that appear thicker than they are in reality. Discretization of the Hough Space in combination with the sliding-window approach help to take care of this problem. A second factor is the number of planes to be detected. The more planes are present in a point cloud the less prominent

are peaks in the accumulator. The same effect appears, when objects are present in the point cloud that do not consist of planes. Depending on these factors the optimal number m might vary between data sets with different characteristics.

Adaptive Probabilistic Hough Transform

The Probabilistic Hough Transform can be used to drastically reduce the computation time for detecting geometrical objects in a point cloud. However, the choice for the optimal subset of points to achieve good results is highly problem dependent. To achieve good results the size of the used subset is usually chosen much larger than needed to minimize the risk of errors. For this reason methods have been developed to determine a reasonable number of selected points. The idea of the Adaptive Probabilistic Hough Transform [100] (APHT) is to monitor the accumulator. The structure of the accumulator changes dynamically during the voting phase. As soon as stable structures emerge and turn into significant peaks voting can be terminated.

Monitoring the entire accumulator array is rather time consuming. The voting of one point only touches a small fraction of the accumulator cells. For saving time after each voting process only those cells need to be monitored that were touched. The maximal cell of those is identified and considered for plane extraction. If several cells have the same high score one of them is chosen. A list of potential maximum cells is updated. Comparison of consecutive peak lists allows for checking the consistency of the peak rankings in the lists. The procedure is outlined in Algorithm 4.

Algorithm 4 Adaptive Probabilistic Hough Transform

while stability order of S_k is smaller than threshold t_k and maximum stability order is smaller than threshold t_{max} **do**
  randomly select a small subset $P^m \subset P$ of size n
  for all points $\boldsymbol{p}_i^m$ in P^m **do**
    vote for the cells in the accumulator
    choose maximum cell from the incremented cells and add it to active list of maxima
  end for
  merge active list of peaks with previous list of peaks
  determine stability order
end while

To speed up the process the list update is only performed after a small batch of points has voted. The list of peaks is incrementally ordered by value of the cells. When updating the list with a peak the coordinates of the peak are taken into consideration. If a peak is close to a higher peak in the list, it is disregarded. In case it is close to lower peaks in the list it is added and the lower peaks in the spatial neighborhood are removed from the list. The size of the list is limited. Consequently lower peaks are deleted when too many higher peaks have been detected. In the early stages of the algorithm changes in the order of peaks are frequent. As updating proceeds the structure of the accumulator will become clearer. The stopping rule for the algorithm is determined by the stability of the most dominant peaks.

A set S_k of k peaks in the list is called a stable set if the set contains all the largest peaks

before and after one update phase. The order within the set is insignificant for the stability. The number m_k of consecutive lists in which S_k is stable is called the stability order of S_k. The maximum stability count is the cardinality k of the set S_k with the highest stability order. In case there are two sets with the same stability order the one with the higher cardinality is preferred. The stability order of the set S_k that has the maximum stability count is referred to as maximum stability order.

Depending on the task the stopping rule has to be modified. If only one object has to be detected, the program has to stop if the stability order of S_1 exceeds a predetermined threshold. For detecting k objects the stability order of S_k has to exceed a predetermined object. Detecting an arbitrary number of planes using several iterations with a fixed k may lead to very long runtimes. Therefore it is recommendable to let the program run until the maximum stability order reaches a threshold. The maximum stability count is then the number of found objects.

Progressive Probabilistic Hough Transform

Algorithm 5 Progressive Probabilistic Hough Transform

while still enough points in P **do**
  select a point $\boldsymbol{p}_i$ from point set P
  for all cells (ρ, φ, θ) in accumulator A **do**
    if point $\boldsymbol{p}_i$ lies on the plane defined by (ρ, φ, θ) **then**
      accumulate cell $A(\rho, \varphi, \theta)$
    end if
  end for
  remove point $\boldsymbol{p}_i$ from P and add it to P_{voted}
  if highest accumulated cell is higher than threshold t **then**
    select all points from P and P_{voted} that are close to the plane defined by the highest peak and add them to P_{plane}
    search for the largest segment $P_{segment}$ in P_{plane}
    remove from P all points that are in $P_{segment}$
    for all points $\boldsymbol{p}_j$ that are in P_{voted} and $P_{segment}$ **do**
      Unvote $\boldsymbol{p}_j$ from the accumulator
      Remove $\boldsymbol{p}_j$ from P_{voted}
    end for
    if the area spanned by $P_{segment}$ is larger than a threshold **then**
      add $P_{segment}$ to the output list.
    end if
  end if
end while

The Progressive Probabilistic Hough Transform (PPHT) [61] calculates stopping times for random selection of points dependent on the number of votes in one accumulator cell and the total number of votes. The background of this approach is to filter those accumulation results that are due to random noise. The algorithm stops whenever a cell count exceeds a threshold

of s points that could be caused by noise in the input. The threshold is calculated anew each time a point has voted. It is predicated on the percentage of votes for one cell from all points that have voted. Once a geometrical object is detected, the votes from all points supporting that object are retracted.

The stopping rule is very flexible. Stopping at any time leads to useful results. Features are detected as soon as the contents of the accumulator array allow a decision. If the algorithm is not stopped it runs until no points are left in the input set. This happens when all points have either voted or have been found to lie on an already detected plane. Even if the algorithm is allowed to run until the end it does not mean that all points must have voted. Depending on the structure of the input many points may have been deleted before voting because they belonged to a plane that was detected. The PPHT is outlined in Algorithm 5.

5.1.3 Randomized Hough Transform

The Standard Hough Transform has a major drawback. For each point all cells are accumulated that represent a curve the point may lie on. This touches many cells representing curves that are not actually present in the image and makes it harder to detect maxima. For detecting all curves in a wide range with a high precision it is necessary to have a large amount of cells with small cell sizes leading to even higher computational requirements.

The Randomized Hough Transform (RHT) [99] diminishes the number of cells touched by making use of the fact that a curve with n parameters is defined by n points. Taking the example of planes, three points from the input space can be mapped onto one point in the Hough Space. This point is the one corresponding to the plane spanned by the three points. In each step of the procedure three points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are randomly picked from the point cloud. The plane spanned by the three points is calculated as

$$\boldsymbol{n} = (\boldsymbol{p}_3 - \boldsymbol{p}_2) \times (\boldsymbol{p}_1 - \boldsymbol{p}_2)$$

and

$$\rho = \boldsymbol{n} \cdot \boldsymbol{p}_1.$$

φ and θ are calculated as explained in Section 5.1 and the corresponding cell $A(\rho, \varphi, \theta)$ is accumulated. If the point cloud consists of a plane with ρ, φ, θ, after a certain number of iterations there will be a high score at $A(\rho, \varphi, \theta)$.

By randomly picking three points it is not guaranteed that they lie on a plane that is actually present in the data set. There will be votes for planes that do not exist. However, when a plane is represented by a large number of points, it is more likely that three points from this plane are selected. Eventually the cells corresponding to actual planes receive more votes and are distinguishable from the other cells. A few considerations help to decrease the number of wrong votes. If points are close together sensor noise will have a strong effect on the direction of the plane spanned by these points. If points are very far apart, they most likely do not belong to one plane. Therefore a point distance criterion is introduced, i.e., $\text{dist}_{\min} \leq \text{dist}_{\min}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ and $\text{dist}_{\max}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3) \leq \text{dist}_{\max}$. $\text{dist}_{\min}$ is the minimum distance allowed between two points, $\text{dist}_{\min}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ calculates the textrmminimum point-to-point textrmdistance between $\boldsymbol{p}_1$, $\boldsymbol{p}_2$

and $\boldsymbol{p}_3$ and $\text{dist}_{\max}$ and $\text{dist}_{\max}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ the maximum distance, respectively. The basic algorithm is structured as described in Algorithm 6.

Algorithm 6 Randomized Hough Transform

```
1: while still enough points in point set P do
2:   Randomly pick three points p_1, p_2, p_3 from the set of points P
3:   if p_1, p_2 and p_3 fulfill the distance criterion then
4:     Calculate plane (ρ, φ, θ) spanned by p_1, p_2 and p_3
5:     Accumulate the cell A(ρ, φ, θ) in the accumulator space.
6:     if |A(ρ, φ, θ)| equals threshold t then
7:       (ρ, φ, θ) parametrize the detected plane
8:       Delete all points close to (ρ, φ, θ) from P
9:       Reset the accumulator
10:    end if
11:  else
12:    continue
13:  end if
14: end while
```

The Randomized Hough Transform has several main advantages over the Standard Hough Transform. The Hough Transform does not have to be applied to all points and for those points considered no complete Hough Transform is necessary. Instead, the intersection of three Hough Transform curves is marked in the accumulator. It is possible to detect the curves one by one. Once there are three points whose plane leads to an accumulation value above a certain threshold t, all points lying on that plane can be removed from the input and hereby the detection efficiency be increased.

Since the algorithm does not calculate the complete Hough Transform for all points, it is probable that not the entire Hough Space needs to be touched. There will be many planes on which no input point lies. This allows for space saving storage procedures that only store the cells actually touched by the Hough Transform. Section 5.2.5 gives an overview on structures for the data storage of the accumulator.

Connective Randomized Hough Transform

The Connective Randomized Hough Transform (CRHT) [49] originates from Computer Vision and was intended to find curves in 2D images. It extends the Randomized Hough Transform by using local information. Like in the basic version one point is randomly chosen from the input data set. For 3D data this point marks the center of a $w \times h \times d$ sized window. Within this window, the CRHT algorithm performs a connective component search, starting at the center point.

Once a connected set within the window is determined and the size of the set is large enough a plane is fitted through the point set as explained in Section 5.3. The plane parameters ρ, θ and ϕ that are calculated in this way are used to accumulate the corresponding cell. Algorithm 7 outlines the Connective Randomized Hough Transform.

Algorithm 7 Connective Randomized Hough Transform

1: **repeat**
2: Randomly pick a point $\boldsymbol{p}_i$ from the set of points P
3: Create a window of size $w \times h \times d$ around $\boldsymbol{p}_i$ and perform the sectored connective component search over all points within the window
4: **if** the number of connective points in the window exceeds a threshold **then**
5: fit a plane to the connective points and calculate ρ, φ, θ
6: **if** the fitting error is below a threshold **then**
7: Accumulate the cell $A(\rho, \varphi, \theta)$ in the accumulator space
8: **end if**
9: **end if**
10: **until** Value of cell $A(\rho, \varphi, \theta)$ exceeds threshold t

In the original 2D version only those points are chosen that are connected to the starting point with an 8-path. Camera images bring the advantage of nicely structured data points. All pixels are arranged in a grid-like structure. Additionally the search can be performed as a sectored search. If the search has started in one direction, the following connected points may only lie in the same direction, or the direction next to it. Considering the grid as a compass, if the search has started in the northern direction, consecutive pixels may lie in north-western, northern and north-eastern direction. The connectivity search for 3D laser range images however asks for a slightly different approach. Possible is the introduction of a distance criterion. A point is added to the connected set if the distance to one point from the set is below a threshold.

5.1.4 Multiresolution Hough Transform

Atiquzzaman proposes a Multiresolution Hough Transform (MHT) [6] and its pipelined implementation on a pyramid multiprocessor [7]. The MHT aims at minimizing the computation time of the Hough Transform by using different resolutions. The intention is to minimize the number of calculations needed for the voting phase of the HT. Coarse analysis is used to limit the parameter space that needs to be more finely analyzed. The MHT is an iterative implementation of the Hough Transform that uses different resolution for the accumulator array and the input data in each iteration step. A set of reduced-resolution images is generated from the original image to decrease the amount of data used in the HT. The Hough Transform is first applied to the smallest image, e.g., the image with the lowest resolution. Additionally the accumulator used is also of low resolution. In each iteration step the resolution of the image and the accumulator increase. Depending on the votes in the previous iteration the range of the parameter space is reduced. This way a trade-off between high resolution and low amounts of data is achieved. The general algorithm is outlined in Algorithm 8.

In the original approach the MHT is designed to detect lines in images. For downscaling images some well-known methods exit. The difference of Gaussian operator can be used to efficiently construct reduced-resolution images from a grayscale image. For 3D point clouds however a different strategy is needed. A possibility is to use an occupancy grid-like structure. An occupancy grid is used in robotic mapping to model free and occupied areas (cf. [86]). The

Algorithm 8 Multiresolution Hough Transform

1: **for all** resolutions **do**
2: generate a data set with reduced resolution from the original data
3: **end for**
4: **for all** resolutions **do**
5: apply the Hough Transform to the reduced-resolution data using a reduced-resolution accumulator array
6: detect the peaks in the accumulator array
7: reduce the parameter range
8: **end for**

environment is modeled as a fine-grained grid. Cells of the grid are either marked as free or occupied. For the Multiresolution Hough Transform the input data would have to be converted into occupancy grids of different resolution for each iteration. Cells are marked as occupied, i.e., are a point in the current resolution, if a point of the original data set lies in them. For a better approximation a threshold can be implemented determining the number of points needed in a cell to make it occupied. Another approach would be to weight the cells with the number of points.

The idea of the MHT is to decrease the computational requirements of the Hough Transform by narrowing down the parameter space. The voting phase of the SHT needs $O(|P| \cdot N_\varphi \cdot N_\theta)$ operations. Instead of decreasing the number of points the MHT intends to decrease the number of cells that need to be considered. After applying the Hough Transform with a low resolution the peak is detected. The peak is to become the new center of the accumulator. In each iteration everything outside of a window with decreasing size around the peak is cut off. This way the size of the parameter space decreases as well. Consequently, the number of cells for the next iteration does not grow as fast as the resolution which causes a decrease of the runtime as compared to the SHT. Rather than performing the Hough Transform on a large number of cells once, the Hough Transform is performed several times on a small number of cells. The algorithm zooms into the accumulator at the location of the highest peak and uses a higher resolution for a decreasing number of cells around the peak to achieve a high accuracy.

The MHT is designed to detect only one pattern in a data set. To detect an arbitrary number of planes in a point cloud a variation of the algorithm is necessary. One possibility is to leave several windows in the parameter space. Otherwise the MHT would have to be repeated to detect planes iteratively. Both these variations would reverse the time improvements which makes the MHT rather impractical for the task needed in this thesis.

5.1.5 Fast Hough Transform

Li et al. presented a possibility to reduce the computational complexity and the storage requirement of the Hough Transform simultaneously by introducing the Fast Hough Transform (FHT) [56]. They use a hyperplane formulation in their definition. The algorithm proceeds by recursively dividing the parameter space into hypercubes leading to higher resolution in each recursion step. Hypercubes are only further divided and voted on if their value exceeds a

threshold T. Votes are assigned to a hypercube if a hyperplane intersects this hypercube. The hyperplane formulation of the problem and the k-tree representation of the parameter space enable incremental testing for intersection without multiplication. This further contributes to efficiency. The algorithm is outlined as follows.

Algorithm 9 Fast Hough Transform

```
1: Compute normalized hypercube radius
2: for all hyperplanes p_i in set P do
3:   if normalized distance from hyperplane p_i to the root node is below a threshold T then
4:     accumulate root node
5:   end if
6: end for
7: if vote of root node ≥ T then
8:   enqueue root on parent_list
9: end if
10: while parent_list is not empty do
11:   dequeue current_node from parent_list
12:   if current_node is on desired level then
13:     add current_node to solution list
14:   else
15:     for all sons node_j of current_node do
16:       for all hyperplanes p_i that voted for current_node do
17:         if normalized distance from the hyperplane p_i to node_j is below threshold T then
18:           accumulate node_j
19:         end if
20:       end for
21:       if vote of node_j ≥ T then
22:         enqueue node_j on parent_list
23:       end if
24:     end for
25:   end if
26: end while
```

The algorithm is demonstrated using plane detection as an example. Given is a 3-dimensional point cloud, the problem is to find one or more planes that best fit these feature points. Using an adapted normal vector representation of the plane and the hyperplane formulation an image-space plane can be represented as

$$x = n_y y + n_z z + b_x$$
$$y = n_x x + n_z z + b_y$$
$$z = n_x x + n_y y + b_z$$

with the normalized normal vector (n_x, n_y, n_z) of the plane and the intercepts b_x, b_y and b_z of the coordinate axes. Choosing (n_y, n_z, b_x), (n_x, n_z, b_y), and (n_x, n_z, b_z) as three sets of parameters we

can transform each feature point (x_j, y_j, z_j) into three plane equations in the three independent parameter spaces

$$\frac{-x_j}{\sqrt{y_j^2+z_j^2+1}} + n_y \cdot \frac{y_j}{\sqrt{y_j^2+z_j^2+1}} + n_z \cdot \frac{z_j}{\sqrt{y_j^2+z_j^2+1}} + \frac{1}{\sqrt{y_j^2+z_j^2+1}} \cdot b_x = 0$$
$$\frac{-y_j}{\sqrt{x_j^2+z_j^2+1}} + n_x \cdot \frac{x_j}{\sqrt{x_j^2+z_j^2+1}} + n_z \cdot \frac{z_j}{\sqrt{x_j^2+z_j^2+1}} + \frac{1}{\sqrt{x_j^2+z_j^2+1}} \cdot b_y = 0$$
$$\frac{-z_j}{\sqrt{x_j^2+y_j^2+1}} + n_x \cdot \frac{x_j}{\sqrt{x_j^2+y_j^2+1}} + n_y \cdot \frac{y_j}{\sqrt{x_j^2+y_j^2+1}} + \frac{1}{\sqrt{x_j^2+y_j^2+1}} \cdot b_z = 0.$$

The parameter space is mapped as a octree with the root at $[\frac{S_0}{2}, \frac{S_0}{2}, \frac{S_0}{2}]$, where S_0 is a predetermined variable restricting the parameter space. The norm distance of the root node is calculated as

$$\frac{1}{S_0} \cdot \frac{-x_j}{\sqrt{y_j^2+z_j^2+1}} + \frac{1}{2} \cdot \frac{y_j}{\sqrt{y_j^2+z_j^2+1}} + \frac{1}{2} \cdot \frac{z_j}{\sqrt{y_j^2+z_j^2+1}} + \frac{1}{2 \cdot \sqrt{y_j^2+z_j^2+1}}$$

for the first parameter space and for the other parameter spaces, respectively. When splitting a node the new distances are calculated depending on the median of all hyperplanes that voted for the parent cube. The FHT aims at limiting the number of points that need to be considered to vote for each cell and limiting the number of cells that need to be evaluated. Like in the MHT the resolution of the accumulator grows. Instead of decreasing the resolution of the data the number of points that vote for a certain area are restricted based on the votes in the previous step with lower resolution of the accumulator. The parameter space is decreased if cells do not receive enough votes. In this case they do not need to be evaluated with higher resolution. Since all parts of the accumulator are evaluated the FHT allows for detection of several planes.

5.2 Structure of the Accumulator

Without prior knowledge of the point cloud it is almost impossible to define proper accumulator arrays. An inappropriate accumulator, however, might lead to the following problems:

1. failures to detect some specific planes,
2. difficulties in finding local maxima,
3. low accuracy,
4. large storage, and
5. low speed.

A trade-off has to be found between a coarse discretization that accurately detects planes and a small number of cells in the accumulator to decrease the time needed for the Hough Transform. Choosing a cell size that is too small might also lead to a harder detection of planes in noisy laser data. In this section we describe different designs of accumulators for the 3D Hough Transform.

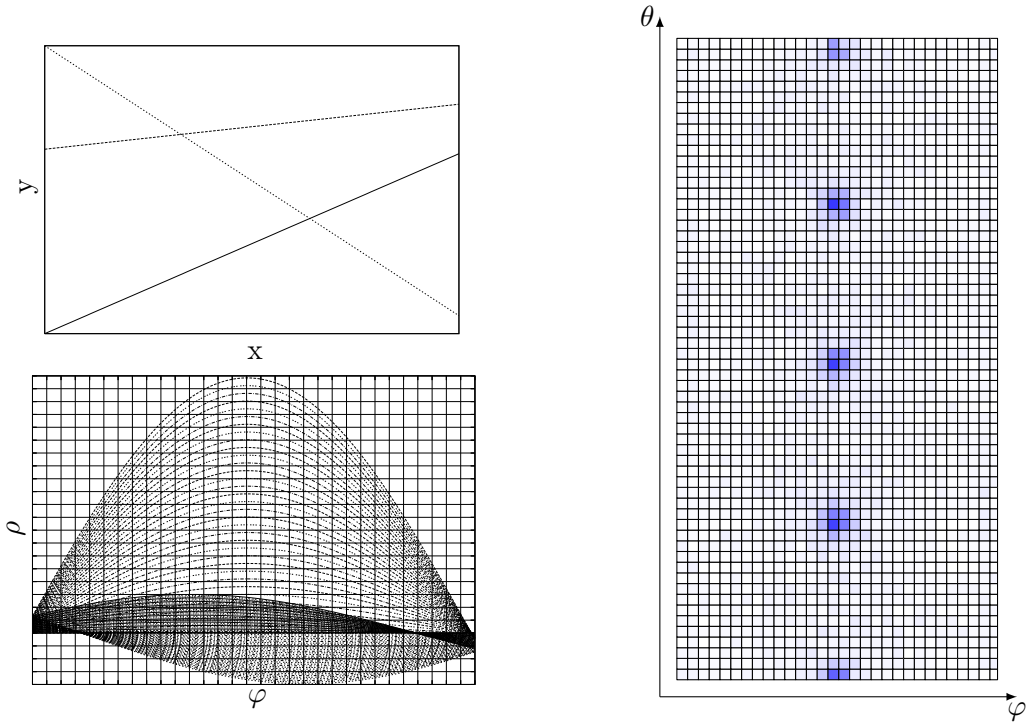


Fig. 5.4: Accumulator Arrays. Left: 2D Hough Transform. Top: Three lines. Bottom: The discretized Hough Space with the votes for the three lines from the top image. Right: Accumulator array for the 3D Hough Space. After performing 1000000 steps of the RHT on the cube data set. Depicted are only three dimension. The color indicates the sum of the values in ρ direction.

5.2.1 Accumulator Array

For the standard implementation of the 2-dimensional Hough Transform [29] the Hough Space is divided into $N_\rho \times N_\varphi$ rectangular cells (cf. Fig. 5.4). The size of the cells is variable and can be chosen problem dependent. Using the same subdivision for the 3-dimensional Hough Space by dividing it into cuboid cells causes some major drawbacks. An exemplary accumulator array is shown in Fig. 5.5. For better clarity only two dimensions, the two angles φ and θ are illustrated. The values in direction of ρ, the distance of the plane to the origin, are summed up, that means the color indicates the summed values of all cells in the direction given by φ and θ. The Hough Space is discretized with $\varphi' = 3°$ and $\theta' = 3°$. Shown are the result after performing 1000000 steps of the Randomized Hough Transform without resetting the accumulator. The incrementation value is depicted by the color of the cell. Blue means high and white means low values. The input point cloud is the data set depicted in Fig. 5.3, a cube with a side length of 400 cm and each side consisting of 10000 points.

The image clearly shows four peaks in the accumulator array. Due to the circular definition of θ one peak is split between the left and the right border. This means that two faces will not be detected in the cube using this accumulator array. A look at the structure of the accumulator when mapped to a ball explains the reason for this behavior. The ball represents all possible normal vectors that can define all different planes. Discretizing the surface of the ball with

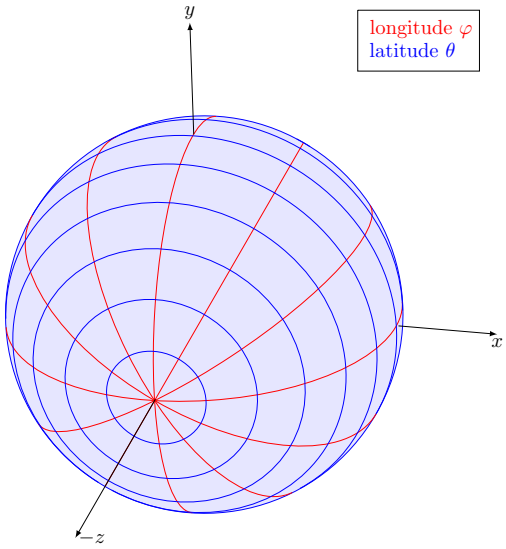


Fig. 5.5: Mapping of the accumulator array designs onto the unit sphere. Accumulator array with regular grid-like discretization of θ and φ with $\varphi' = 3°$ and $\theta' = 3°$. φ is the longitude of the sphere and θ the latitude, respectively.

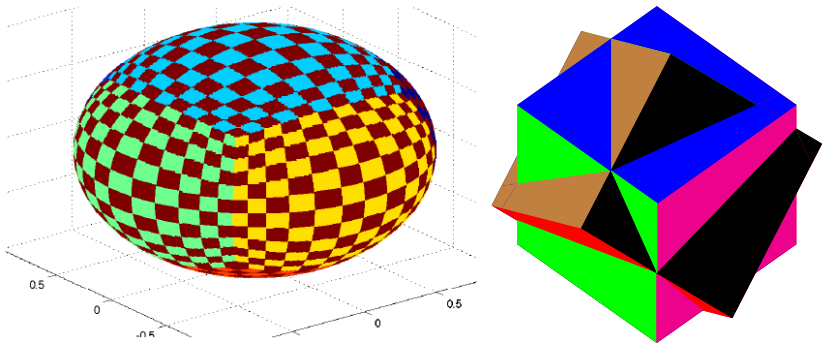
respect to φ and θ leads to different sized pieces. The blue colored circles displaying the latitude circles have the same distance, but the longitude circles meet at the poles. Therefore the cells closer to the poles are smaller and comprise less normal vectors. This means voting favors the larger equatorial cells. In plain terms this means that the points lying on a plane parallel to the xy-plane will vote for a larger number of small cells close to the poles while points lying on a plane parallel to the xz-plane vote for a small number of larger cells. In the first case many cells will have an intermediate high value and are not as likely to be detected than a plane in the second case where a few cells have very high values.

5.2.2 Accumulator Cube

Censi et al. propose a design for an accumulator that is a trade-off between efficiency and ease of implementation [20]. Their intention is to define correspondences between cells in the accumulator and small patches on the unit sphere with the requirement that the difference of size between the patches on the unit sphere is negligible. Their solution is to project the unit sphere $\mathbb{S}^2$ onto the smallest cube that contains the sphere using the diffeomorphism

$$\begin{aligned} \varphi \colon \mathbb{S}^2 &\to \text{cube} \\ s &\mapsto s / \left\| s \right\|_\infty . \end{aligned}$$

Each face of the cube is divided into a regular grid. Fig. 5.6 shows the resulting patches on the sphere. Given the normal vector of a plane the cell to be accumulated is calculated as follows. The side a_i of the cube is determined as the direction of the dominant coordinate n_d of the face normal. Scaling the normal vector with $1/n_d$ results in a projection onto the cube face, where



(a) Accumulator cube (b) Rotated cube

Fig. 5.6: Mapping of the cuboid accumulator design to the unit sphere. (a) The resulting patches on a sphere for the cuboid design of the accumulator. The unit sphere is mapped onto the smallest cube that contains it. Each cube face is divided into a regular grid. (b) Demonstration of irregularity within the cell sizes. Given an environment with rectangularly arranged surfaces, if the surfaces are aligned with the coordinate system they are all equally likely to be detected. Otherwise their normals belong to differently sized cells. This is demonstrated with the two rotated cubes. Consider the blue-green-magenta cube as the accumulator and the other cube as the environment. The normal of the brown face points directly into one corner of the accumulator cube, the area with the smallest cells. The normals of the red and black faces point closer to the middle of the edges where the cells are bigger. This way the brown plane is less likely to be detected than the other planes.

the non-dominant coordinates of the normal vector transform into the cube coordinates

$$c_x = \frac{n_1}{n_d} \qquad\qquad c_y = \frac{n_2}{n_d}.$$

n_1 and n_2 denote the two non-dominant coordinates of the normal vector. c_x and c_y range between -1 and 1. Given the cube coordinates, the cell indices are then calculated as

$$a_x = \begin{cases} 1 & \frac{c_x+1}{2} = 1 \\ 1 + \text{nr\_cells} \cdot \frac{c_x+1}{2} & \text{else.} \end{cases} \qquad a_y = \begin{cases} 1 & \frac{c_y+1}{2} = 1 \\ 1 + \text{nr\_cells} \cdot \frac{c_y+1}{2} & \text{else.} \end{cases}.$$

This short insight into the mathematics shows that the transformation from \$ into accumulator indices and back into \$ is quite simple. The question remains whether the implementation is also efficient in terms of regularity of the cell sizes. Fig. 5.6(a) shows the patches on the sphere. The regularity between all six cube faces is obvious. This means that in an environment that is composed of rectangularly arranged planes that are aligned with the coordinate system all planes will be detected with the same probability due to the same sized accumulator cells. However, towards the edges of the cube faces the cells in the projection become smaller. Fig. 5.6(b) hints to the effects of this irregularity. Consider the blue-green-magenta colored cube as the accumulator and the brown-red-black colored cube as the environment in which planes are to be detected. The normal vector of the brown cube face points directly into the corner of the accumulator cube where the cells are the smallest. The other faces do not point directly into the corners. The cells of the accumulator that will be incremented by points from those faces are smaller. This means

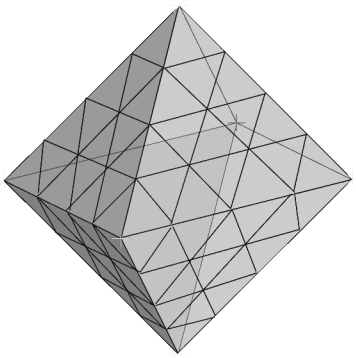


Fig. 5.7: Design of an accumulator as an octahedron.

that the same amount of sensor noise will lead to a different number of cells that will mark these faces. Small cells, compared to larger cells mean more cells onto which the votes will be divided. Therefore the brown plane is less likely to be detected than the other faces. This irregularity will be the case dependent on the orientation of the environment compared to the coordinate system of the accumulator. Due to the uneven matching of 8 corners to 6 faces the irregularities will always be of importance when considering rectangularly arranged environments as well as less structured environments. Experimental evaluation of this irregularity in the design of the accumulator follows in Section 5.4.

5.2.3 Polyhedral Accumulator

Zaharia et al. use the 3D Hough Transform for shape-based similarity retrieval [102]. In this application geometric invariances play an important role. For each object a coordinate system is generated based on Principal Component Analysis (PCA). The objects gravity center is selected as origin of the coordinate system. The spatial alignment is achieved by labeling the principal axes in ascending or descending order of the eigenvalues. The possibilities of labeling and the two different orientations of the coordinate axes lead to 48 possible coordinate systems. The results from applying the Hough Transform to objects aligned this way are compared to detect identical objects.

For really detecting similarity all 48 Hough Transforms need to be considered. Due to circular shift and mirror reflection properties of the Hough Transform this can be narrowed down to 3 cases. Provided that the accumulator has the same decomposition in the direction of all coordinate axes there exists a one-to-one mapping between the three remaining generating configurations of the hough transform. This is achieved when partitioning the Hough Space by projecting the vertices of any regular polyhedron onto the unit space. The level of granularity can be varied by recursively subdividing each of the polyhedral faces. An example is given in Fig. 5.7. Each triangular face of the octahedron is divided into four triangles each of which can again be divided.

It becomes obvious, that each top of the octahedron has the same partitioning, i.e., the structure is invariant against rotation of 90 ° around any of the coordinate axes. This design brings along many advantages for the task of comparing objects that are aligned with respect

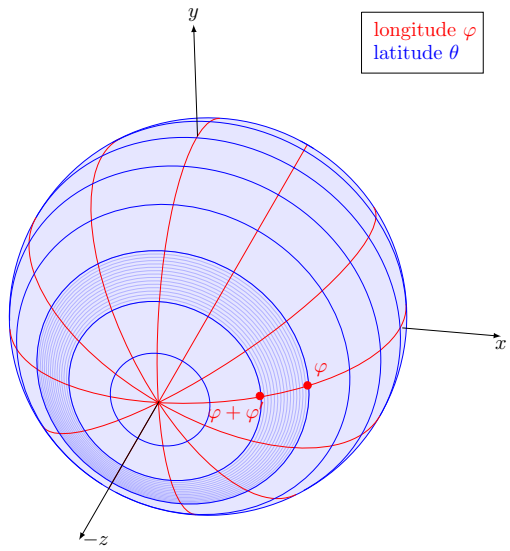


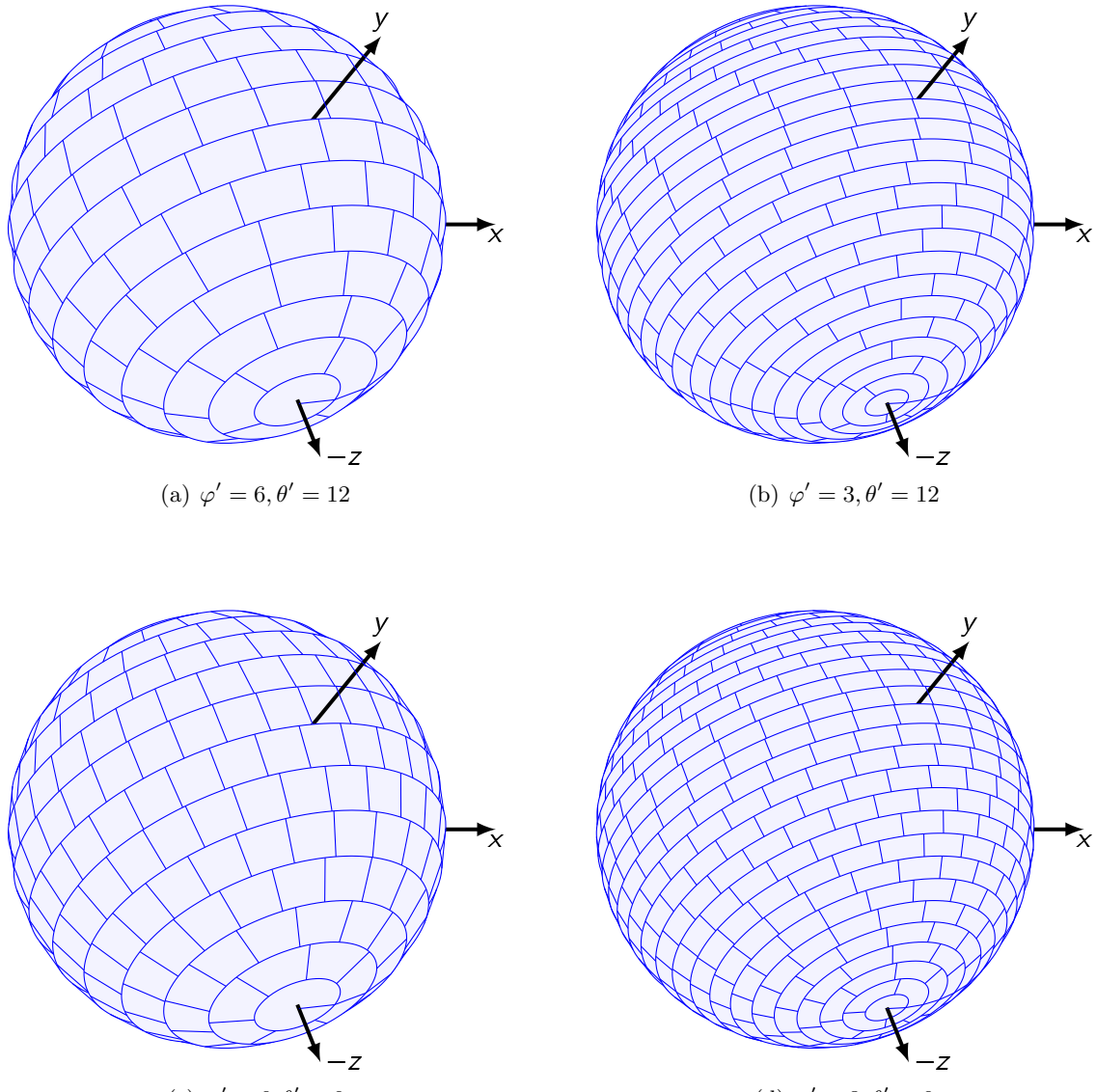
Fig. 5.8: Calculation of the length of a longitude circle at φ_i. The segment in question is the darker colored one.

to their principal axes. For detection of planes however it shows the same drawbacks as the cuboid design described in the previous section. When mapping the partitions back onto the unit sphere the patches will appear to be to be unequally sized.

5.2.4 Accumulator Ball

The previous sections presented different designs for representing the accumulator for the 3D Hough Transform. The three design have one drawback in common, the irregularity between the patches on the unit sphere. While the simple array structure suffers from enormous differences between the patch sizes, the cuboid and octahedral designs reduce these differences drastically. A further benefit is their invariance against rotation of 90 ° around any of the coordinate axes. Problems remain with smaller rotations. If the planes to be detected do not align with the coordinate system the position of the plane decides about its likeliness to be detected. Still, the calculation of the cells to be accumulated is slightly more complicated.

In this section we present a design for the accumulator with the intention of having the same patch size for each cell. To achieve this, the resolution has to be varied dependent on the position on the sphere. For this purpose the sphere is divided into slices. See Fig. 5.8 for an illustration of the idea. The resolution of the longitude φ can be kept as for the accumulator array. φ' determines the distance between the latitude circles on the sphere, e.g, the thickness of the slices. Depending on the longitude of each of the latitude circles the discretization has to be adapted. The discretization can be computed differently. One way is to calculate the step width θ' based on the size of the latitude circle at φ_i. The largest possible circle is the equator located at $\varphi = 0$. For the unit sphere it has the length $\max_l = 2 * \pi$. The length of the latitude circle in the middle of the segment located above φ_i is given by $\text{length}_i = 2 \cdot \pi \cdot (\varphi_i + \varphi')$. The



(a) $\varphi' = 6, \theta' = 12$ (b) $\varphi' = 3, \theta' = 12$

(c) $\varphi' = 6, \theta' = 6$ (d) $\varphi' = 3, \theta' = 6$

Fig. 5.9: Resulting patches on the unit sphere when applying the differentiated discretization for different φ' and θ'.

step width in θ direction for each slice is now computed as

$$\theta'_{\varphi_i} = \frac{360\,^\circ \cdot \max_l}{\text{length}_i \cdot N_\theta}.$$

The resulting design is illustrated in Fig. 5.9. The image clearly shows that all accumulator cells are of the same size. Compared to the previously explained accumulator designs, the accumulator cube and the polyhedral accumulator, a possible drawback becomes obvious when looking at the projections on the unit sphere. The proposed design lacks invariance against rotations of multiples of $90\,^\circ$. Planes that are parallel to the coordinate system possess different likeliness of being detected because their points vote for a different number of cells. Depending on the fineness of the discretization the normal vectors at the poles, e.g., the planes parallel to the xy-plane, often lie directly in the middle of just one cell, while the other coordinate axes lie in most cases in the intersection of longitude and latitude circles. This leads to a distribution of the votes to several cells. In practice however this problem seems negligible since in most cases the planes to be detected do not perfectly align with the coordinate system. Experimental evaluation of the different accumulator designs in Section 5.4 covers this problem.

5.2.5 Dynamic Data Structures

Tree

The computation time of the Hough Transform is a combination of the time needed for the accumulation step and the search for a maximum. The search for a maximum can be accelerated by use of dynamic data structures. In Section 3.2.2 the binary tree is described that Xu et al. [99] propose for accelerating the 2-dimensional Randomized Hough Transform.

Instead of storing all possible accumulator cells only those that have received a vote are stored. In practice, this is performed as follows. Besides storing the point $p = (a_1, a_2)$, each cell stores the score of the cell. The cells are arranged according to their Hough coordinates a_1 and a_2. When points are randomly chosen for the RHT the cell they vote for is determined. If the cell exists in the dynamic data structure, the score is increased. Otherwise, a new cell is created with the coordinates in the Hough Space and a score of one. This cell is inserted into the data structure at the designated position dependent on the coordinates. When searching for the cell with the maximum score, not all cells have to be searched but only those that actually have been voted for. This is especially advantageous concerning the discretization in direction of ρ. If the extension of the input data is not known in advance, it is difficult to strike a balance between a fine discretization and a time and memory minimizing storage. Using this dynamic data structure it is not necessary to limit the extent of the plane distances in advance.

Dynamically Quantized Spaces

O'Rourke [70] introduces dynamically quantized (DQ) spaces for storing the votes in the accumulator array. The intention is to cope with the problem of large space requirements for storage of uniformly quantization schemes. He states that especially for high dimensional parameter spaces even coarse quantization leads to immense storage requirements. When storing accumulator votes in a DQ space, the accumulator initially consists of only a single cell. During the

process of inserting points the cells split or merge in response to the local characteristics of the data.

A cell represents a k-dimensional rectangular boxes of the parameter space. Each cell has a counter that is increased when performing the Hough Transform. The approach intends to have an equal portion of the overall counts in each cell. Furthermore the hits in each cell should be uniformly distributed. To achieve this, a split and a merge rule is implemented. If the count in one cell becomes too large relative to the total count or if the internal imbalance of the cell becomes too large a cell is split along one of the axes. The new count for each of the split cells becomes one half of the original cell. If one cell and one of its neighbors have counts whose sum is not too large and the resulting merged cells remain balanced, the two cells are merged. Splitting and merging are complementary operations. To realize the rules each cell needs certain information about its votes. Therefore an approximation of the gradient is stored in a k-dimensional imbalance vector for each cell. The vector associated with a cell is updated every time a vote hits the cell.

In the implementation by O'Rourke the data structure is represented as a tree. Each cell can be reached via the path starting from the root. After updating the counts of a cell the split and merge rules are applied if necessary and in case of a split or merge the neighboring cells are checked, as well. This way the DQ space recursively and dynamically reorganizes itself.

Advantages of the DQ space representation of the accumulator are easy resolution control and dimensionality reduction. The resolution of the accumulator space adapts to the resolution needed for the specific problem and can be controlled by parameters that limit the number of cells and the imbalance. In case the data is uniformly distributed along one of the dimensions the DQ space will not split in this direction and thus the dimension of the space is reduced. Using DQ Spaces for the Hough Transform reduces the number of cells of the accumulator array. The accumulator adapts to the distribution of the represented data. When using the Hough Transform for motion detection or matching of structures the structure of the input data is nicely represented in the structure of the DQ space. However, when trying to detect most dominant geometric objects, e.g., planes, the search for maxima is more complicated. Instead of searching for a single cell with the highest count, the most discriminated area has to be found and within this region the cell with the highest count has to be detected to determine the wanted parameters. Next to the complex insertion procedures the more complicated maxima search slows down the Hough Transform which leads to the conclusion that the proposed data structure does not lead to major improvements for the task of plane detection.

Dynamically Quantized Pyramids

Dynamically Quantized Pyramids (DQP) [82] are similar to Dynamically Quantized Spaces. Merging two cells in DQ Spaces is a complex procedure. It leads to a reorganization of the data structure causing change in numbers and connectivity of cells. The approach by Sloan [82] tries to overcome these changes by fixing the number of cells and the connectivity, e.g., the connections between fathers and sons. Resulting from the fixed connections the resource allocation is also fixed and the data structure may be reduced to a hardware implementation.

In the DQP the data is organized in a pyramid data structure. A pyramid is a fully balanced tree in which each internal node has exactly 2^n sons. In the case of the Hough Transform n

equals the dimensionality of the parameter space. As for the DQ Spaces each node covers an n-dimensional rectangular region of the parameter space. The pyramid consist of 2^n sub-regions that are divided by n-dimensional cross-hairs, typically fixed in the center of the region. Each cell also counts its votes. A DQP is different to a fixed-boundary pyramid in the way that the boundaries of each cell are continuously changed by means of a hierarchical warping process as each cell tries to track the mean position of its data points in its part of space. The cross-hairs are stored as a vector of percentages. Starting from a fixed-boundary pyramid where all cross-hairs are located at $(50, 50, \ldots, 50)$ the boundaries are gradually adapted to the data. The root node covers the entire space. The boundaries of each son are calculated by applying the percentages associated with its father node. When a new value is added to the pyramid the cross-hair associated with the corresponding node is slightly shifted towards the new value. This can be done by calculating the weighted average of the cross-hair position and the newly inserted value. By changing one cross-hair, the change is recursively forwarded to the entire subtree rooted by this node.

There are two important effects of adjusting the pyramid. First, when inserting a value into the pyramid, the sizes of the cells change in a way that an even distribution of votes in each cell is achieved. This means that the cross-hairs move towards the newly inserted value and decrease the corresponding cell. Second, the order of insertion influences the structure of the DQP as the most recent changes have the most impact on the structure. This is advantageous if recent changes in the world should be represented in the data structure. For static environments however this is undesired, especially if the data points are handled in a regular grid-like structure as it is the case for laser scans.

5.3 Plane Fitting

The Hough Transform presents us with parameters that represent a plane that corresponds to the input data. Regardless of the accumulator used, due to the discretization and the probabilistic aspects of the different Hough strategies the plane will never be perfectly fitted to the data. Besides that, the parameters define an unlimited plane. To build a planar model of an environment we need bounded planes, i.e., we need to create a polygon that forms a convex bound of all points that lie on the plane. The basic structure of the algorithm used to create a plane model from a laser scan is given in Algorithm 10. Some of the essential steps are described in more detail in the following.

5.3.1 Point Selection

In most cases a 3D model consists of more than just one plane. Only few Hough strategies support finding several planes. Therefore, to create a complete model it is necessary to do several iterations of the Hough and detect the most dominant plane in each iteration. After each iteration the points that voted for the detected plane need to be eliminated from the point cloud. But even if all planes are detected at once, to determine the bounds of the polygon it is necessary to identify the points that belong to one specific plane. The selection is performed using a simple distance function. Given the normal vector $\boldsymbol{n} = (n_x, n_y, n_z)$ of a plane, the

Algorithm 10 Plane detection algorithm

1: Choose a point cloud
2: **while** termination rule does not apply **do**
3: Apply the Hough Transform to detect the most dominant plane in the point cloud
4: Determine all points that belong to this plane
5: Fit a plane to the points
6: Detect all points that are close to the fitted plane
7: Determine the largest connected area from these points
8: Delete all points within the convex hull of the largest connected area
9: **end while**

distance dist_p of a point $\boldsymbol{p} = (p_x, p_y, p_z)$ to the plane is given as

$$\text{dist}_p = p_x n_x + p_y n_y + p_z n_z. \tag{5.3}$$

A point $\boldsymbol{p}$ is considered to belong to the plane given by $\boldsymbol{n}$ if $|\text{dist}_p|$ is below a threshold ε. The threshold is supposed to compensate for the noise in the input data and the discretization errors.

5.3.2 Least Squares Best Fitting Plane

Given a cloud of laser scan points and the parameters φ, θ and ρ that define a plane that is spanned by a large number of points from the cloud we want to determine the largest bounded plane that can be found in the point cloud and delete all points that lie within this bounded plane.

The plane is determined using the least squares best fitting method (cf. [63]). Given a point cloud P we calculate the centroid of the point cloud

$$\begin{aligned} c_x &= \sum_{i=0}^{n} \frac{x_i}{n}, \\ c_y &= \sum_{i=0}^{n} \frac{y_i}{n}, \\ c_z &= \sum_{i=0}^{n} \frac{z_i}{n} \end{aligned}$$

and the matrix

$$\boldsymbol{A} = \sum_{\boldsymbol{p}_i \in P} \begin{pmatrix} x_i^c x_i^c & x_i^c y_i^c & x_i^c z_i^c \\ x_i^c y_i^c & y_i^c y_i^c & y_i^c z_i^c \\ x_i^c z_i^c & y_i^c z_i^c & z_i^c z_i^c \end{pmatrix},$$

Fig. 5.10: Left: 3D model of the fifth floor of a university building in Osnabrück, Germany. On the right is the hall to which two computer labs are connected on the left. Depicted is the plane that is spanned by one of the walls from the computer lab. Left: 2D slice of the points spanning the plane in the left image. It becomes obvious that there exist points that lie on the plane but are not wanted for the model, e.g., the points that bound the hallway.

where

$$x_i^c = x_i - c_x,$$
$$y_i^c = y_i - c_y,$$
$$z_i^c = z_i - c_z.$$

are the coordinates of point p_i after the center of the point cloud is moved to the origin. The normal vector n of the least squares best fitting plane is the eigenvector corresponding to the minimum eigenvalue of $\boldsymbol{A}$. The distance of the plane to the origin is calculated using the centroids of the point cloud

$$\rho = n_x c_x + n_y c_y + n_z c_z.$$

5.3.3 Point Segmentation/Clustering

Whenever we perform the Hough Transform points exist that lie on the plane but do not belong to the bounded plane we are looking for. Fig. 5.10 depicts this problem. To eliminate these points we use clustering. We use two different clustering strategies, a simple closest distance clustering procedure and an improvement using quadtrees. For both techniques we assume that all points lie on a plane. Given a point cloud and a plane fitted through that point cloud as described in the previous section, we assume that all deviations from the plane are due to noise. To eliminate the noise we move all points onto the plane by shifting them as far in the direction of the normal vector as their distance to the plane is according to Eq. (5.3):

$$\boldsymbol{p}' = \boldsymbol{p} - \mathrm{dist}_p \cdot \boldsymbol{n}.$$

From all the points that are on a plane we now need to find the largest connected region. The basic idea is to find the largest subset R in the point set S so that for each point $\boldsymbol{p}_i$ in R there exists at least one point $\boldsymbol{p}_j$ in R so that the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is below a threshold.

Each pair of points $\boldsymbol{p}, \boldsymbol{q}$ in a cluster must be connected either directly or indirectly. They are connected directly when their distance $\mathrm{d}(\boldsymbol{p}, \boldsymbol{q})$ is smaller than the maximal distance threshold T. $\boldsymbol{p}$ and $\boldsymbol{q}$ are connected indirectly when there is an arbitrary number of points in the cluster that build a chain of direct connections. Irrespective of the dimension of the point set and the distance metric d, finding these clusters is in $O(n^2)$ where n is the number of points. In the straightforward procedure we start with only one cluster R_1 consisting of one point from the point set. We then go through all points $\boldsymbol{p}_i$ in S and check whether the distance between $\boldsymbol{p}_i$ and one point in any of the existing clusters R_j is below threshold T. Is this the case, $\boldsymbol{p}_i$ is added to R_j, otherwise a new cluster R_k is started consisting of $\boldsymbol{p}_i$. This clustering technique does not create perfect clusters. There are cases where one connected region is divided into several clusters due to the order in which the points are processed. However, the clusters we are looking for, those that represent a flat surface in the laser scans, should be densely sampled by points. In this case the probability that clusters are not recognized due to the order of processing is smaller. Besides that, our strategy makes it less likely that two highly connected regions that are only connected by one chain of points are classified as one cluster. Lastly, if two large regions are split apart the plane is likely to be recognized again in one of the next iterations of the Hough Transform. This increases the runtime for the algorithm but has no major impact on the quality of the final result. However, in Chapter 6 we describe ways to deal with the problems of minor quality clustering.

The straightforward clustering procedure leads only to basic results. Additionally its runtime is in $O(n^2)$ and adds greatly to the total runtime of the plane extraction. This can be improved by employing a spatial data structure that stores the point set. As the dimension of the point set in our application is always 2 we use a quadtree. The concept of quadtrees was explained in Section 3.2.2. In order to cluster more efficiently, the quadtree has to be modified slightly. In addition to storing the containing points or nodes, each of the nodes also stores 2 additional sets of points. Each of these sets contain the points that could potentially be directly connected to the bordering siblings. Therefore, they represent 2 of the 4 borders of the node. Which border is used depends on the type of quadrant the node represents. A node in the north-west would store the borders on its south and east side, and so on. This information can easily be obtained and stored in the usual course of constructing the quadtree without interfering with its efficiency.

Clustering is now done recursively in an inverse breadth-first search. Each leaf clusters its content in the naive way, by doing a pairwise comparison of all of its points. Although this process is in $O(n^2)$, the quadtree is always constructed in such a way that in each leaf n is very small. The overall efficiency of the clustering is therefore not impacted. Nodes join their children's clusters by comparing all point pairs that could potentially be close enough for a direct connection. These point pairs are easily chosen from the borders of child nodes that neighbor each other. By construction of the border regions, no other point pair could possibly be a direct link. The clusters of the children are then merged according to the additional connections established in the border comparison.

Due to the spatial partition of the initial point set the number of necessary point comparisons is greatly reduced. Therefore the complexity of the clustering algorithm is reduced to $O(n \log n)$

In order to cluster a set of 3-dimensional points such that two points $\boldsymbol{p}, \boldsymbol{q}$ are directly connected when $\measuredangle(\boldsymbol{p}, \boldsymbol{q})$ is smaller than some threshold, the point set is transformed into polar coordinates. The distance ρ can be discarded to obtain 2D points. The distance metric used in

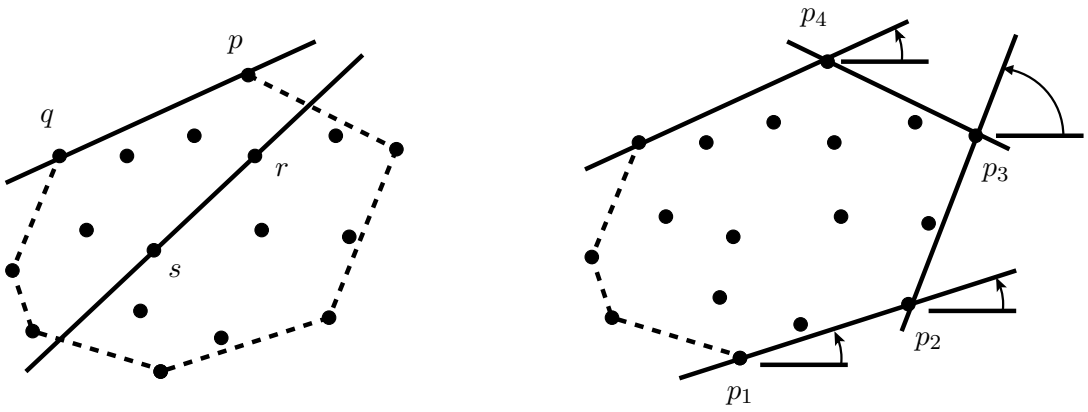


Fig. 5.11: Illustration of the convex hull of a point set. Left: The line connecting $\boldsymbol{p}$ and $\boldsymbol{q}$ is a hull edge because all points of the set lie on one side of it. The line connecting $\boldsymbol{r}$ and $\boldsymbol{s}$ is not a hull edge because points lie on both sides of it. A hull cannot separate the set. Right: Starting at point $\boldsymbol{p}_1$ the Jarvis' march algorithm finds successive hull vertices by turning angles. Discovering a new vertex is done in $O(N)$.

the above clustering algorithm is then the classical great-circle distance on the sphere, i.e., "as the crow flies":

$$\mathrm{d}(\boldsymbol{p}, \boldsymbol{q}) = \arccos\left(\sin\left(\theta_p\right)\sin\left(\theta_q\right) + \cos\left(\theta_p\right)\cos\left(\theta_q\right)\cos\left(\phi_q - \phi_p\right)\right)$$

In this scenario the quadtree represents the coordinates of a sphere as partitioned by latitude and longitude.

After clustering we select the largest cluster. These points represent the largest region on the plane that is present in the point cloud.

5.3.4 Jarvis' March Convex Hull

If we have a number of points that lie on a plane we can assume that a surface in the environment exists that contains all these points. The convex hull of a point set S is the smallest convex set containing S, i.e., a polygon P consisting of the minimum number of points from S where all points from S are within the bounds of P [72]. Speaking figuratively, if a rubber-band is spanned around all points from the set on a plane it will assume the shape of the convex hull of these points.

A convex hull is either defined by its vertices or by the edges connecting the vertices. The Jarvis' march algorithm for creating a convex hull of a point set is based on identifying whether an edge connecting two points from the point set is an edge of the convex hull. Its average complexity is in $O(hN)$ with h being the actual number of vertices and N being the size of point set S.

A line segment l defined by two points of a point set S is an edge of the convex hull of S if and only if all points of S lie on l or on the same side of it. See Fig. 5.11 for an example. The dashed line indicates the convex hull of the point set. $\overline{\boldsymbol{pq}}$ is part of the convex hull because all

points from the set lie on the same side of it. $\overline{\boldsymbol{rs}}$ separates the point set. There are points on both sides of it. Therefore it cannot be part of the convex hull. Considering this circumstance it is straightforward to decide whether a line connecting any pair of points from a point set is part of the convex hull of that set. If N is the number of points in S, there is a total of $\frac{N^2}{2}$ lines connecting every pair of points. For each possible line we have to examine $N-2$ points on their position relative to the line. Accordingly all hull edges can be found in $O(N^3)$. Arranging them into a list of consecutive vertices yields the polygon that is the convex hull.

Jarvis proposed to improve this algorithm by starting with one hull edge $\overline{\boldsymbol{pq}}$. If $\overline{\boldsymbol{pq}}$ is an edge of the convex hull there has to be an edge connecting $\boldsymbol{q}$ and another point that is also an hull edge. The algorithm marches around the convex hull and finds extreme points in order, one at a time. See Fig. 5.11 for an illustration of the approach. The lexicographically lowest point $\boldsymbol{p}_1$, i. e., the lowest coordinate in one of the directions, is certainly a hull point. Starting from that point the goal is to find the next consecutive vertex point $\boldsymbol{p}_2$ on the convex hull. $\boldsymbol{p}_2$ can be found by comparison of polar angles. Two consecutive points have the least positive polar angle with respect to each other. This way each successive point can be found in linear time until the lexicographically highest point is reached ($\boldsymbol{p}_4$ in Fig. 5.11). The other half of the convex hull is detected symmetrically, with reverse direction of the two axes and the least polar angles with respect to the negative x-axis. As the Jarvis' march algorithm finds the next point on the edge by simple comparison of polar angles, the input data set does not have to be sorted in any way.

The time complexity for the Jarvis' march algorithm depends on the structure of the point cloud. In the worst case, if all points of the set compose the convex hull the running time is $O(N^2)$ because of the linear time needed to find each of the N vertices. However, in most scenarios the convex hull consists only of a small number of vertices which leads to an average runtime of $O(hN)$ with h being the number of vertices in the convex hull and N being the number of points in the point set. Considering that the h is typically a small number this is a major improvement compared to $O(N^3)$.

Since we are looking for a 2-dimensional convex hull we need to eliminate one of the coordinates of the scan points. In the previous sections we describe how to detect planes and how to decide if a point belongs to the plane we are looking for. The largest region found through clustering is the point set for which we want to find the convex hull. For this reason we project those points onto the planes of the coordinate system depending on the slope of the plane. If x is the most dominant coordinate of the normal vector of the plane the points are projected onto the yz-plane, if y is the most dominant coordinate the points are projected onto the xz-plane and if z is the most dominant coordinate onto the yz-plane, respectively. After calculating the convex hull, the edge points are projected back onto the detected plane.

5.4 Experimental Evaluation of the Hough Transform

In this section we present an experimental evaluation on the Hough Transform. The evaluation covers the different Hough methods as well as the different designs for the accumulator. The experiments are performed on an Intel QuadCore 2.66 GHz processor with 4 GB RAM. In this section all experiments will be performed on a simulated data set for easier evaluation. Results on plane detection in real laser scans are shown in Chapter 6.

5.4.1 Comparison of the Different Accumulator Designs

In Section 5.2 we described different designs for storing the votes of the Hough Transform. The dynamic data structures greatly improve the storage space needed for the data cells. However, as pointed out before, the search for most dominant planes in these data structures is more time consuming. As this is the task that is needed when detecting planes these data structures appear not to be optimal. We decided to implement three of the fixed size designs for the accumulator. First, the simple array structure where φ and θ are uniformly discretized is the simplest way of discretizing the Hough Space. It is interesting to investigate whether the obvious flaws of this design come into effect in practical applications or if they are negligible. Second, out of the two designs that focus on symmetry with respect to the coordinate system we chose the cuboid design over the polyhedral design, since both designs seem to have similar characteristics and the cuboid design appears to be easier to manage. Third, the design of our accumulator ball with different discretization for each latitude slice of the unit sphere is evaluated against those other two designs.

Experimental Results

We reduce the experimental setup to a simple test case. A cube with a side length of 400 is placed around the origin. Each side consists of 10000 points which are randomly distributed over the entire face of the cube with a maximal noise of 10. Different rotations are applied to the cube to simulate different orientations of planes. The advantage of using this simple model is the existence of ground truth data for the actual planes. The cube possesses a perpendicular structure which is characteristic for most man-made indoor environments. As pointed out in Section 5.2 rotating the cube poses challenges to the accumulators as the sides are not symmetrically aligned with the coordinate axes anymore.

The first experiment is designed to investigate the ability of the accumulator designs to correctly detect planes in the given point cloud. For this purpose we apply the SHT to the cube. To achieve comparable results the number of cells for each accumulator needs to be approximately the same. The used setting is $N_\rho = 100$ with a maximal distance of 600. For the accumulator ball we use $N_\varphi = 45$ and $N_\theta = 90$. This means that the slice around the equator consists of 90 patches. The total number of cells is 257100. The accumulator cube has a total number of 264600 cells when using 21×21 cells on each face. The simple accumulator is discretized with $N_\varphi = 38$ and $N_\theta = 76$ leading to 288800 cells.

The demands towards the HT are twofold. First, the planes need to be easily detected, i.e., each plane is represented by exactly one dominant maximum in the accumulator. In the example this means that each of the six highest peaks corresponds to one face of the cube. Second, the

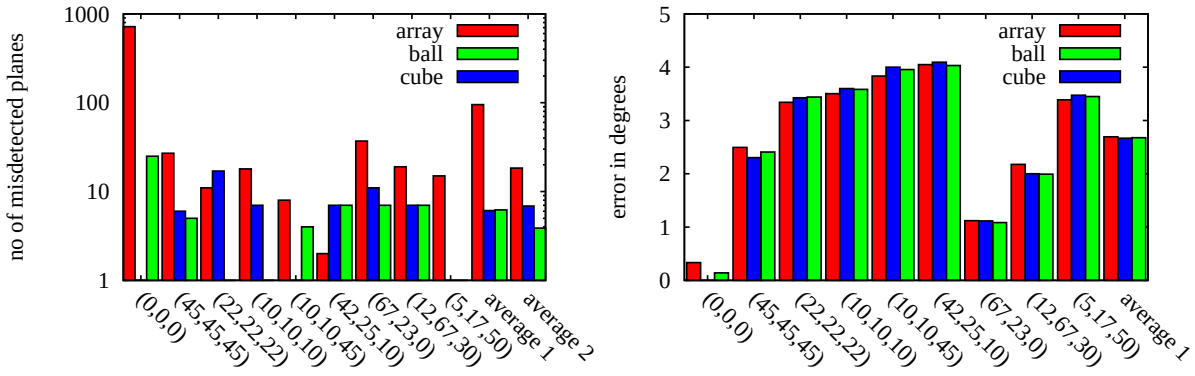


Fig. 5.12: Comparison of the different accumulator designs after applying the SHT. Left: Number of incorrectly detected planes on a logarithmic scale. Right: Angle error of the detected planes. Each set of bars represents one orientation of the cuboid laser scan model rotated by $(\alpha_x, \alpha_y, \alpha_z)$ as denoted underneath each set. α_x is the rotation around the x-axis, α_y around the y-axis and α_z around the z-axis

highest peak for each face is as close as possible to the ground truth of the plane. Fig. 5.12 shows an evaluation of the three different accumulator designs with respect to these aspects. On the left the ability to correctly detect all six planes is depicted. Nine different rotations are applied to the cube. The bars indicate the number of incorrectly detected planes, i.e., if the six highest peaks correspond to the six different faces of the cube, the error is zero. Each time the next highest peak corresponds to an already detected plane before every single face of the cube is represented by one peak, the error is incremented by one.

The results show clearly that the simple array structure has enormous problems to correctly identify the six cube faces. It totally fails when the cube is aligned with the coordinate system. This is due to the problem pointed out when introducing the accumulator designs in Section 5.2. The uneven sizes of the patches lead to an uneven distribution of peaks against favor of the planes parallel to the xy-plane. This comes mostly in effect when the cube is aligned with the coordinate system, when rotated the effects diminish but do not disappear.

The accumulator ball has also great difficulties to detect the perfectly aligned cube. The problems in this design are also planes parallel to the xy-plane. Due to the current design the slice closest to the z-axis is divided into several patches, i.e., more cells intersect at the z-axis than at the other parts of the accumulator. When the normal vector of a plane is the z-axis this leads to distribution of the votes over all these patches decreasing the count for each of those cells. This problem could be solved by specially treating the area around the pole. One way to do this is to create a circular cell around the pole that has the desired size of the patches and proceed with the rest of the sphere in the same manner as before. This way the pole will not be the intersection of too many cells. For the other test cases the ball and the cube show similar performance. The ball performs better in some test cases while the cube performs better in other test cases. The two last columns show the average performance of the accumulator designs. `average 1` is the average error over all nine test cases while `average 2` neglects the unrotated case. Comparison of these two average values shows that the cube performs extremely

good for the unrotated data set but lacks precision in the other cases. The ball design however outperforms the cube on average. Applying the small adaption mentioned above could work against the only weakness of the ball design.

The chart on the right of Fig. 5.12 shows the sum of the angle errors for the detected planes. For each side of the cube the cell with the highest vote is used and the angle between the normal vector and the ground truth normal vector of the plane is calculated as error. The results show only small negligible differences between the different accumulator designs. This indicates that once a plane is correctly detected the parameters are calculated eqally well with each accumulator design.

A more graphical analysis of the accumulator designs is shown in Fig. 5.13. The accumulators are plotted after applying 100000 iterations of the Randomized Hough Transform. For better visibility only the slice with $198 < \rho < 204$ is shown, i.e., the distance that the planes have. The votes are drawn in blue, the darker a cell, the more votes it has received. The depicted images show the results for the perfectly aligned cube and the cube rotated $45\,^\circ$ around each axis. The images point out, that for the accumulator array the two planes corresponding to the highest and lowest φ values do not show up. For the ball design the peaks show up, but are not as high as the peaks around the equator. In practice this means that the several cells along the equator have higher votes as the cells around the poles and are therefore earlier detected as planes. The bottom row shows the same experiments with the rotated cube. In the accumulator array six peaks show up. However, the peaks vary in color, which means they vary in the value of their votes and are therefore of different likeliness to be detected. The same holds true for the cube. The peaks close to the corners of the cube (the faces of the accumulator are marked in different colors) the peaks show lower values. These are the regions where the patches have the smallest size. The result is, that planes corresponding to those cells are less likely to be detected. For the ball design the peaks are most evenly colored in this scenario. This supports the previously mentioned assumption that the ball design is the best for detecting arbitrary planes due to its characteristic of having evenly sized patches in the Hough space. However, the flaw with respect to the pole of the unit sphere still needs to be taken care of.

5.4.2 Standard Hough Transform

For 2-dimensional data the Standard Hough Transform is one of the standard methods for detecting parametrized objects. But even there, enormous computational requirements have lead to the emerge of more and more methods to accelerate the Hough Transform without loss of precision. For 3-dimensional data the requirements increase drastically. In this section we are dealing with the question of applicability of the Standard Hough Transform in the 3D case. In the next section we investigate the loss pf precision when applying randomized or probabilistic methods of the Hough Transform.

Experimental Results

Again we use the laser scan model of the cube rotated by $(10, 10, 10)$ and apply the SHT to it using all three different accumulators. The SHT consists of two major steps. First, all points have to be transformed into Hough Space. Second, the maximal peaks in the accumulator have

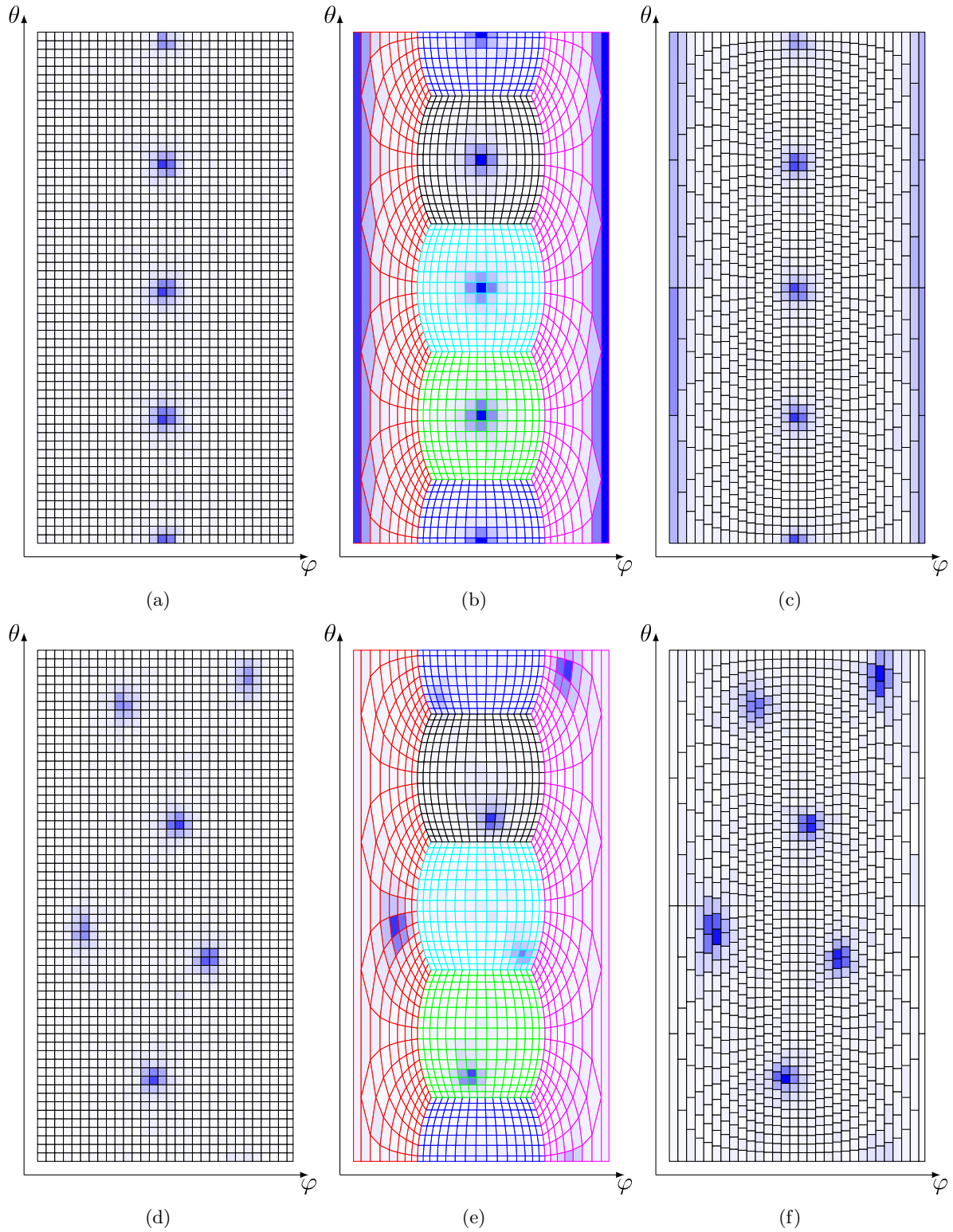


Fig. 5.13: Left to right: array, cube, ball design of the accumulator. Top: data set rotated by $(0, 0, 0)$. Bottom: data set rotated by $(45, 45, 45)$. The input data is the modeled cube.

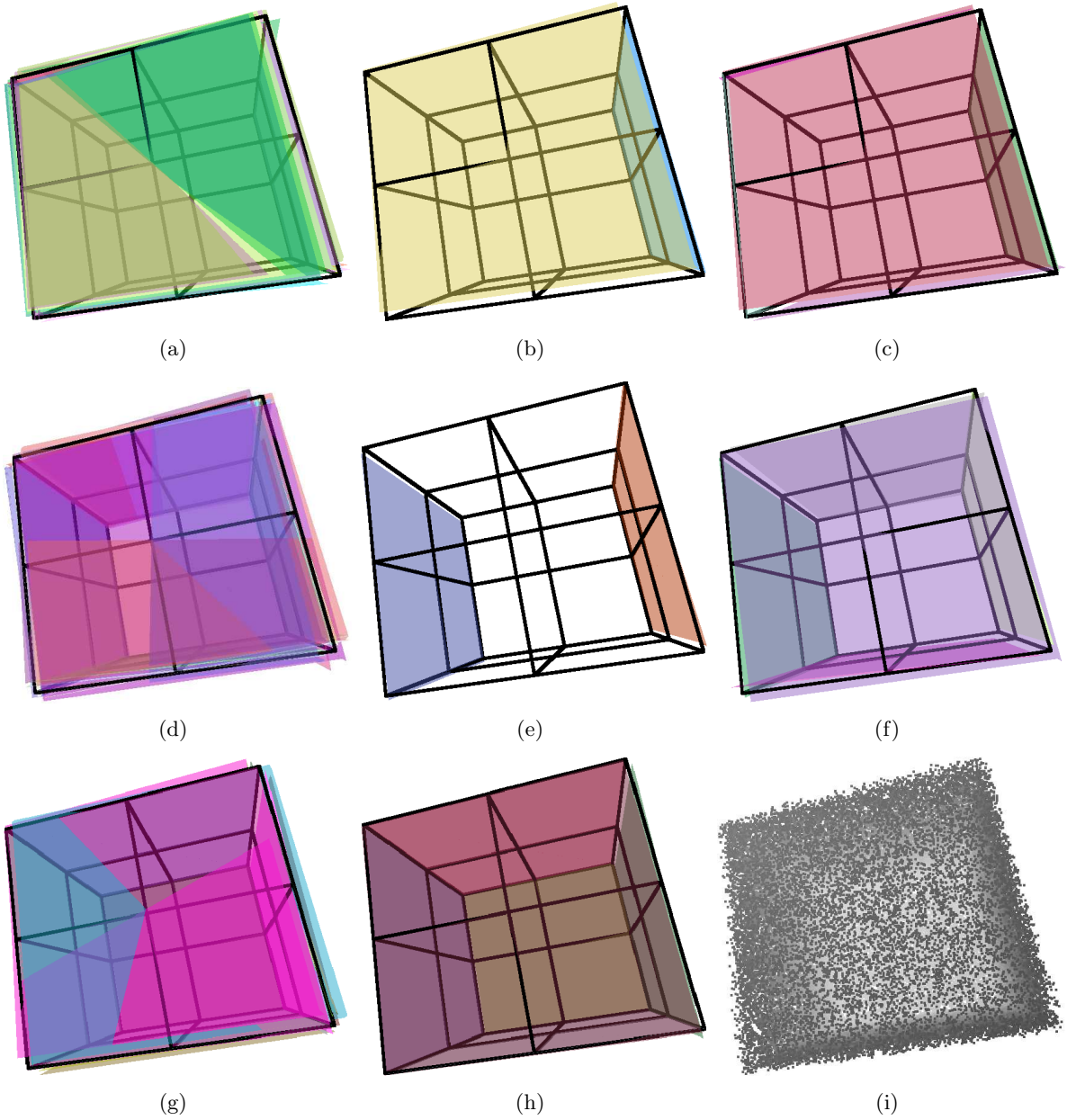


Fig. 5.14: Planes detected by the SHT using different accumulator arrays.
Accumulator array (a) - (c): (a) 20 planes with the highest score. (b) After peak search procedure, all planes with more than 90% of the maximal score. (c) The six planes with the highest score (more than 80% of the maximal score).
Accumulator cube (d) - (f): (d) 20 planes with the highest score (e) After peak search procedure, all planes with more than 90% of the maximal score. (f) The six planes with the highest score ($> 88\%$).
Accumulator ball (g) - (h): (g) 20 planes with the highest score. Note that the first six represent already all different faces of the cube. (h) After peak search procedure, all planes with more than 90% of the maximal score. The resulting planes correspond to the six faces of the cube.
(i): The data set used for the experiment, a cube with side length 400, placed around the origin. Each side consists of 10000 points randomly distributed with a maximal noise of 10.

to be found and a decision has to be made which of those peaks correspond to actual planes in the input data. As seen in Fig. 5.13 the votes for one plane spread over a small region in the Hough Space. Out of this region the best representation has to be picked and the other planes ignored in order to reliably detect planes. We implemented a simple peak search strategy that is applied after the SHT. Starting in one corner of the accumulator, we run over the complete space with a small window, in this experiment $8 \times 8 \times 8$ cells. Within this window all values but the highest one are set to zero. We are aware of the fact that this simple strategy might favor certain maxima but in our experiments this fact showed little influence on the results. For the accumulator cube we ran over each face separately. In the accumulator ball each slice consists of a different number of cells. This might cause problems when applying this windowed peak search. However, the division starts with $\theta = 0$ in each slice and the differences in size for two neighboring slices are only small. Therefore the window does not cover a regularly shaped region but still a connected region.

The resulting planes are shown in Fig. 5.14. The first row shows the results using the accumulator array, the second row the results using the accumulator cube and the third row using the accumulator ball, respectively. In the first column the 20 planes with the highest score using no peak search strategy are depicted. The second row is after applying the peak search strategy. For all accumulators all planes with a count up to 90 % of the highest score are considered to be actual planes. For the accumulator ball these planes are very close to the six faces of the cube model. For the accumulator cube only two planes are above the threshold. For the accumulator array the back and the bottom face are not among the top 90 % while the front appears three times. For the cube the threshold to correctly detect all six faces of the model is 88 %. For the array all six faces of cube have a score above 80 %.

5.4.3 Evaluation of Hough Transform Methods

The previous section shows the challenge in correctly selecting the planes from the data set in the accumulator. In this section we investigate on the different abilities of several HT methods to perform this task.

The methods under investigation are the Standard Hough Transform (SHT), the Probabilistic Hough Transform (PHT), the Adaptive Probabilistic Hough Transform (APHT), the Progressive Probabilistic Hough Transform (PPHT), and the Randomized Hough Transform (RHT). Detailed descriptions of those algorithms are given in Section 5.1.1 to 5.1.3. Instead of applying the Hough Transform to all points, the PHT randomly chooses a certain percentage of the input data and applies the Hough Transform only to those points. The APHT also applies the Hough Transform to randomly selected points from the input data. However, to improve the certainty of detecting planes, a stopping rule is introduced. Every time the Hough Transform is applied to a set of points, the maxima hit by those points are stored and compared to the previously hit maxima. If the order of maxima remains consistent for a certain time, the algorithm stops. In the PPHT points are randomly chosen and transformed into Hough Space until a certain threshold is hit in one of the the accumulator cells. The plane corresponding to the cell is considered as being detected and all points that belong to it are deleted from the data set. This is repeated until the number of points left falls below a threshold. The RHT is similar to the PPHT. However, instead of applying the Hough Transform to randomly chosen points,

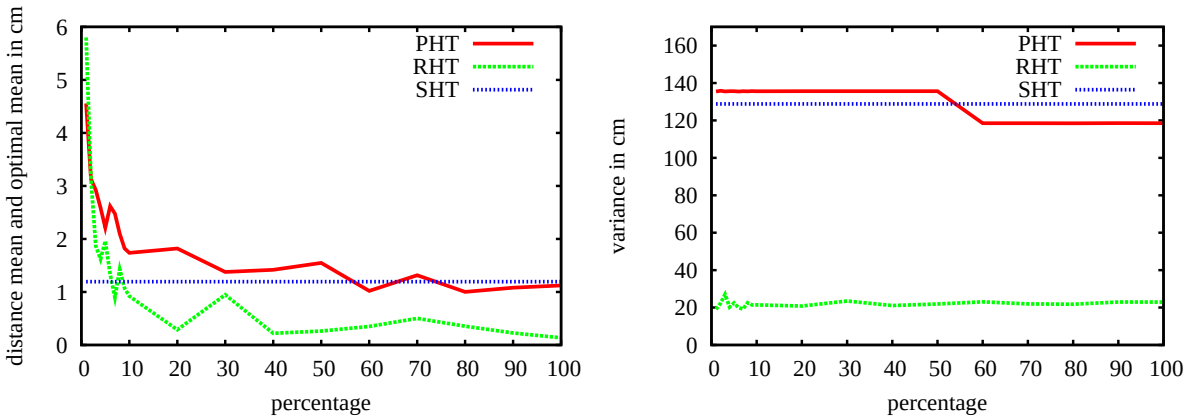


Fig. 5.15: Comparison of the different point selection strategies. For the SHT all points are used and transformed into Hough Space. For the PHT only x % of the points are chosen and the HT applied to them. For the RHT x % times three points are chosen and the cell that corresponds to the plane spanned by these three points is accumulated. Plotted are the mean and the variance of one face of the resulting accumulator space.

three points are randomly chosen and the only cell voted for is the one corresponding to the plane spanned by those three points. As in the PPHT, whenever the score of a cell overshoots a threshold, the points on that plane are deleted from the data set.

Experimental Results

To evaluate the different Hough methods we try to use the same setup for each method. The first experiment investigates how well the Hough methods detect one plane. We express this in terms of mean and variance of the result. Using the cuboid accumulator design and only one face of the laser scan cube model we consider only one face of the accumulator. The accumulator face is divided into 22×22 faces. The slice considered is the one with $198 < \rho < 204$. The mean is calculated as the sum of all accumulator cells weighed by the score of that cell and divided by the number of cells. The error plotted is the distance between the calculated mean and the supporting point of the optimal plane representation of the face. The variance illustrates the distribution of the values in the accumulator face in relation to the calculated mean.

For the evaluation it is necessary to divide the accumulation phase and the maximum search phase. Therefore we cannot evaluate all Hough methods according to this scheme. Instead we use only the PHT and the RHT. For the PHT we vary the number of points used. The RHT is slightly adapted. Instead of running until a threshold is hit, we pick P times three points and accumulate the cell of the corresponding plane. Where P is x percent of the number of points on the cube face.

The experiment is run 20 times for each setting and the average of the results is plotted in Fig. 5.15. Using only a small percentage the distance between the mean and the optimal normal vector is larger than for the SHT using both the PHT and the RHT. Beginning at 10% the RHT outperforms the SHT. Starting from 30% the error of the PHT is close to that of the SHT. The variance of the SHT and the PHT are close together. The RHT has a significantly

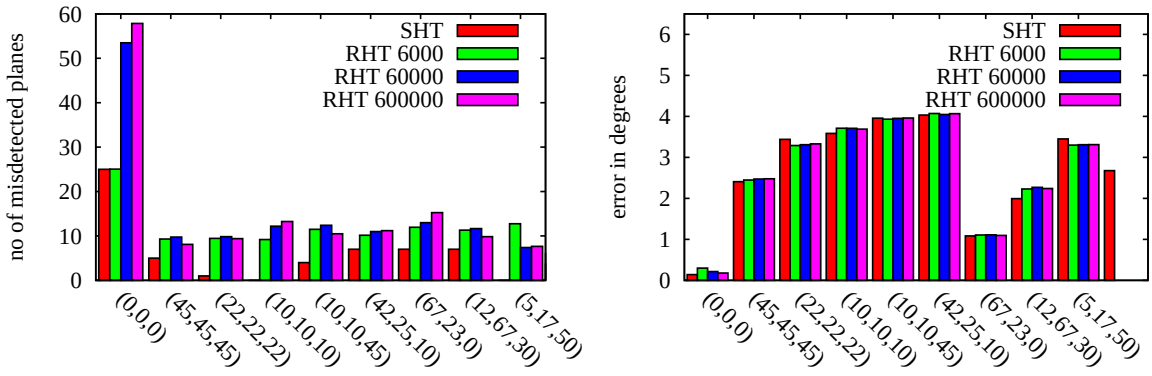


Fig. 5.16: Comparison of the RHT with the SHT when applying it on the test cube. For RHT6000 6000 triples of points are selected, for RHT60000 60000 and for RHT600000 600000, respectively. The experiments were carried out on the modelled cube scan rotated by $(\alpha_x, \alpha_y, \alpha_z)$.

smaller variance. Interpreting the results suggests that applying the PHT with at least 30% of the points leads to results that sufficiently represent the input data. The results for the RHT have to be considered very carefully. The RHT benefits from the simplicity of the input data. Since not the complete Hough Transform is calculated for each point, the RHT has hardly any variance if using only one plane as input. If the input data consists of several planes, however, selecting three points from different planes leads to more erroneous results. Due to that a more differentiated test of the RHT is needed.

We use the adapted RHT on the complete cube. As when testing the accumulator designs, we again use nine different rotations of the cube and count the number of incorrectly detected planes and the angle error of the detected normal vectors.

The results of applying different numbers of iterations of the RHT on the points from the cube model are depicted in Fig. 5.16. The accumulator used for this evaluation is the ball structure with $N_\varphi = 45$, $N_\theta = 90$ and $N_\rho = 100$. Each test is run 20 times and the average is plotted. The differences of the angle error are again within a negligible range. The number of incorrectly detected planes differs depending on the rotation of the point cloud. In all cases the SHT performs noticeably better than the RHT. However, in its original version the RHT (cf. Section 5.1.3) has a phase where the accumulator is reset and the points that belong to an already detected plane are deleted. This should lead to an improvement of the ability to detect all planes in the data set. To verify this, we tried the same experiment with the original version of the RHT that includes an intermediate phase where the points are deleted.

The results for the original RHT are shown in Fig. 5.17 along with results on the same experiment for the PPHT. The thresholds for the maximal accumulator counts for RHT D and PPHT are set to 1000. The results show that when deleting the detected planes in between the accuracy of the HT with respect to finding each plane exactly once is significantly improved. In most cases the first six detected planes correspond to six different faces of the cube. Noticeable is, that for the PPHT also the error in the orientation of the normal vector is significantly smaller. In practice the difference does not play an important role if plane fitting is used in

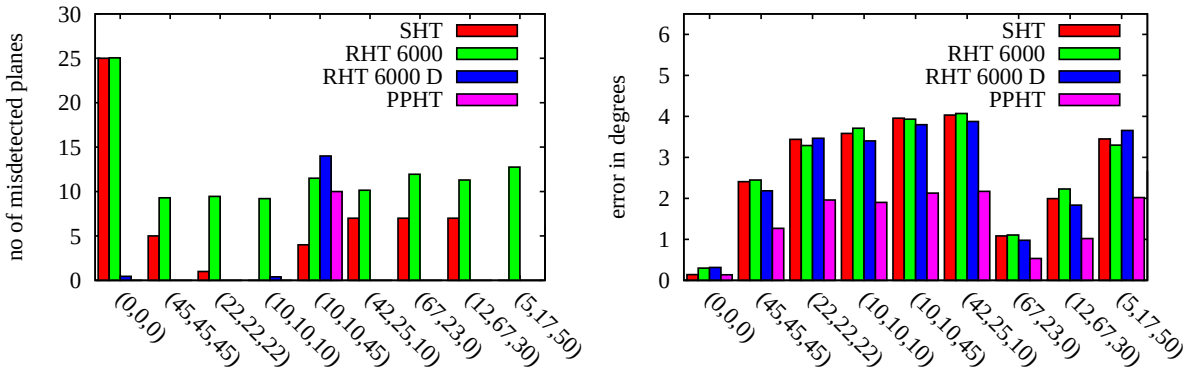


Fig. 5.17: Comparison of different Hough methods with respect to their ability to correctly detect all planes in a data set and their accuracy. Test were performed on the rotated cube.

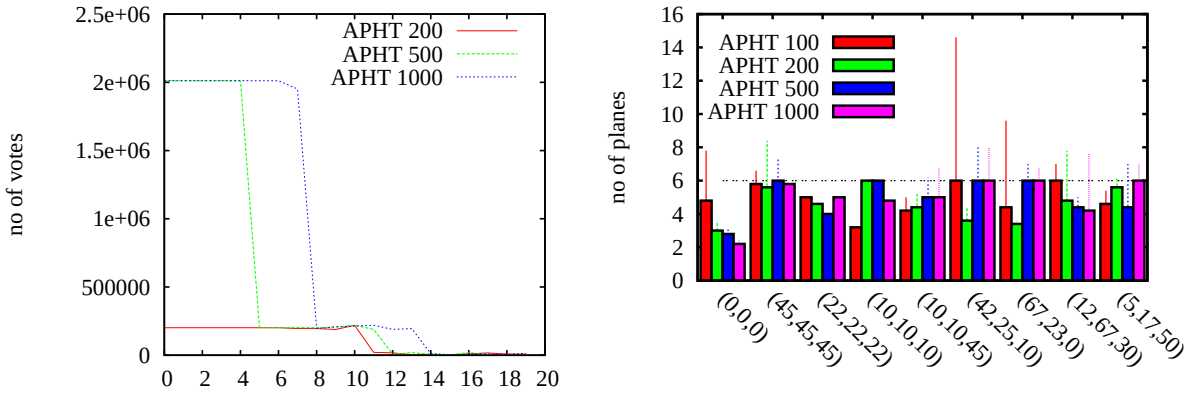


Fig. 5.18: Results for running the APHT with different maximum stability counts on the cube data set.

order to diminish the negative effects of the discretization.

The last method we implemented is the APHT. Like the RHT and PPHT the APHT includes a stopping rule for when to stop transforming points into Hough space. The difference is that instead of detecting a single peak the stability of many peaks is analyzed. The setup for detecting an unknown number of planes is to set a maximum stability count at which the algorithm stops. The Hough Transform is applied to a small number of points. For each point the cell with the maximum count is stored in a list. Comparing those lists over time leads to stability measurements. Once a number of cells reaches the maximum stability count that is set as stopping rule, these planes are considered to be detected in the data set. For a more detailed explanation see Section 5.1.2.

Fig. 5.18 shows a brief analysis of the APHT. The graph in the left shows an exemplary distribution of accumulator counts when using a list of 20 maxima and different maximum stability counts as stopping rule. It becomes clear that the counts between the detected planes

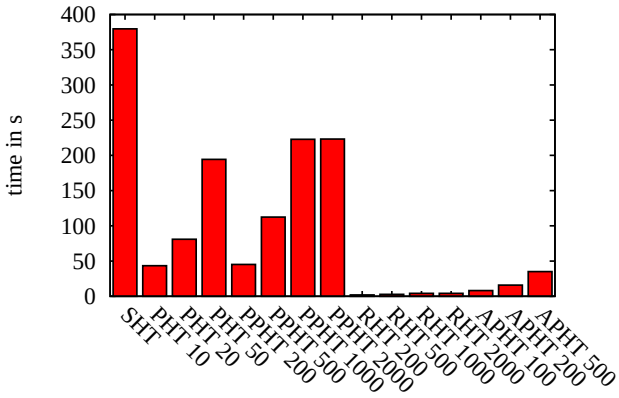


Fig. 5.19: Comparison of the runtime for the different Hough methods

and the not detected planes become very distinctive. However, in this run the algorithm detects 8 planes for the cube data set. The right image shows a more thorough evaluation. The lines indicate the number of planes detected by the algorithm averaged over 20 runs. The boxes show the number of planes out of this list that are the best representation for one of the cube faces. It becomes clear that the algorithm gives a close prediction of the actual planes but fails to correctly identify all planes in almost any case.

After evaluating the quality of the results achieved by the different methods of the Hough transform the remaining question is how far these methods improve the runtime of the algorithm. For this we run each algorithm 20 times, measure the time needed and plot the average time. For the variations we use different settings. The PHT is run with 10, 20 and 50 percent of the points. For the RHT and the PPHT maximal counts of 200, 500, 1000 and 2000 are used. For the APHT we used a maximum stability count of 100, 200 and 500. Since the RHT and PPHT have intermediate phases for deleting the points for one detected plane we include this phase into the measurements for all algorithms. This means the runtime includes the voting phase in the accumulator, the maximum search, plane fitting and creation of the convex hull of the planes.

Fig. 5.19 shows the runtimes for the different Hough algorithms. All variations drastically improve the runtime of the HT compared to the SHT. As expected, the runtime for the PHT decreases proportional to the decrease of points. The PPHT yields the best quality results. The runtime is similar to the PHT. However, the experiments that lead to the results of the quality evaluation were done with a maximal accumulator count of 100 which leads to even shorter runtimes as depicted in the graph. Considering that the PHT needs to be run with at least 30% of the points the PPHT is considerably faster. The APHT needs astonishingly short runtimes. Nevertheless, in the configurations used in the experiments the APHT was not able to reliably detect all six planes. The RHT outperforms all the other HT methods by far concerning runtime. Considering that the quality of the detected planes was only slightly behind the PPHT the RHT seems to be the method of choice for detecting an unknown number of planes in a laser scan map in reasonable time.

Chapter 6

Improvement of the Plane Model

After planes have been extracted from a point cloud using the Hough Transform, several kinds of errors are still present in the location and orientation of the planes. In case the point cloud consists of several laser scans that were merged by some matching algorithm, a single plane in the environment is found several times by the Hough Transform. Errors in the matching process lead to misaligned point clouds so that the planes in multiple scans representing the same actual surface are not parallel to each other The plane parameters may also be estimated incorrectly due to measurement errors from the laser scanner itself. Especially systematic errors such as large scale deformations can lead to errors such as floors and walls not being perpendicular to each other.

This section presents a novel algorithm to improve the position of a given set of planes that contains the previously discussed errors. The algorithm consists of two main parts. First, geometrical relations between planes as they appear in an indoor environment are established. These relations are used to correlate certain planes to a limited number of prototype planes. Second, these correlations are used to optimize all plane parameters and afterwards join planes that correspond to a single real plane.

6.1 Classification

This section discusses several approaches to recover the elementary planes, i.e., floors, walls, etc., in a structured indoor environment. First we give a problem description.

Given: A set of n planes $P = \{p_1, \cdots, p_n\}$, with $p_i = (\boldsymbol{n}_i, \rho_i)$.

Wanted: An optimal classification of P into four classes X,Y,Z and U, i.e., $X \cup Y \cup Z \cup U = P$ and all classes are pairwise disjunct.

In a 3-dimensional environment, there are 3 classes of planes that are of interest. The class Y corresponds to floors and ceilings, X corresponds to walls and Z corresponds to another set of walls that is perpendicular to X. Clearly, most buildings are structured in this fashion.

The goal is to assign each plane p_i to at most one of the three prototype X,Y,Z, i.e., $p_i \in X$, $p_i \in Y$ or $p_i \in Z$. However, a plane may not necessarily be assigned to a prototype at all.

In this case the plane is unassigned, i.e., $p_i \in U$. There are usually a number of unassigned planes in P. These planes mostly originate from small planar objects within the scans. They are therefore regarded as irrelevant to the global plane model. The optimality of classifications is defined solely by the ground truth. A classification is optimal if and only if every plane $p_i \in X$ corresponds to a wall in the real environment, every $p_i \in Y$ corresponds to a floor or ceiling in the environment and so on.

Before discussing the approaches to solve this problem we describe a pre-processing step that will be used in the following algorithms. The pre-processing establishes a set of relations R between two planes p_i and p_j. As this thesis is only concerned with indoor environments, we restrict the possible relations between planes to 2 simple cases. Assuming a plane i is given by the normal $\boldsymbol{n}_i$ and the distance ρ_i with $|\boldsymbol{n}_i| = 1$, then our binary relations are given by:

$o_{i,j} \in R$**:** Two planes p_i and p_j are perpendicular to each other. Walls should be perpendicular to the floor and the ceiling. In this case the constraint is described by:

$$\boldsymbol{n}_i \boldsymbol{n}_j = 0.$$

$p_{i,j} \in R$**:** Planes p_i and p_j are parallel. Floor and ceiling for example should be identified as parallel. The constraint is given by:

$$\boldsymbol{n}_i = \boldsymbol{n}_j.$$

Due to measurement errors the constraints will never be met exactly. Relations are therefore estimated in a simple fashion by utilizing two thresholds α and β. If $\measuredangle(\boldsymbol{n}_i, \boldsymbol{n}_j) < \alpha$ then p_i and p_j are said to be parallel. Consequently, if $\measuredangle(\boldsymbol{n}_i, \boldsymbol{n}_j) > \pi - \beta$ then the planes i and j are seen as orthogonal. Depending on the thresholds α and β the estimated relations are more or less accurate. Any algorithm working with R needs to be robust to the errors that are bound to be produced by the threshold mechanism.

Note that no equivalence relation between planes has been formulated at this point in the algorithm. All ρ_i are optimized in the second step of the improvement algorithm. This occurs irrespectively of the orientation of the planes, so that the decision whether 2 planes are equivalent can be made later on. This further means that we essentially ignore the distances ρ_i in this part of the improvement algorithm.

6.1.1 A Prolog Program

As inspired by Nüchter [65] a simple Prolog program was implemented in order to solve the classification problem. The set of relations R between the planes is encoded in Prolog in the following fashion:

```
labeling(P0,P1,P2,P3,P4) :- parallel(P0,P1), ortho(P0,P2), ...
```

where the list is amended with a `parallel(Pi,Pj)` if $p_{i,j} \in R$ and with `ortho(Pi,Pj)` if $o_{i,j} \in R$. The semantic relations between the classes X,Y,Z and U are represented in Prolog by a set of Horn clauses:

```
parallel(floor, floor).
parallel(wall1, wall1).
parallel(wall2, wall2).
parallel(X,_) :- X == nofeature.
parallel(_,X) :- X == nofeature.
ortho(floor, wall1).
ortho(floor,wall2).
...
```

The semantic net was formulated like this to prevent Prolog from assigning the `nofeature` on its own. The `nofeature` assignment is only considered in case Prolog can find no feasible classification without it. In this case additional Horn clauses are generated that explicitly assign `nofeature` to the planes. This is done for any possible combination of `nofeature` assignments starting with the minimal number of `nofeatures`. Additional Horn clauses are generated for combinations that allow for an additional `nofeature` until Prolog is able to find a feasible classification. The generated Horn clauses are given by:

```
consistent_labeling(P0,P1,P2,P3,P4) :- comb([P0,P1,P2,P3,P4],[nofeature]),
                                       labeling[P0,P1,P2,P3,P4]).
```

where the combinations are generated by:

```
comb(_,[]).
comb([X|T],[X|Comb]) :- comb(T,Comb).
comb([_|T],[X|Comb]) :- comb(T,[X|Comb]).
```

The unification algorithm is finally started by the following inquiry:

```
consistent_labeling(P0,P1,P2,P3,P4).
```

6.1.2 Graph Coloring Approach

The problem of finding the correct assignments can be seen as a variant of the graph coloring problem. Given an undirected graph the goal is to find a vertex coloring, i.e., a labeling of the graph's vertices such that no two vertices that share an edge are labeled with the same color. This problem is a classic $\mathcal{NP}$-complete problem [53]. In our case the objective is to find the optimal 3-coloring of the graph that is given by the vertices P and the edges R. Each of the 3 colors correspond to one of the prototypes X,Y and Z. Of course, the traditional graph coloring problem does not allow for edges $p_{i,j}$ which necessitate the same coloring of the corresponding vertices p_i and p_j. Simply removing all edges $p_{i,j}$ would clearly result in an incorrectly colored graph. This can be compensated for by joining vertices p_i and p_j into the same node if $p_{i,j} \in R$. Computing the joined nodes is equivalent to finding all connected components in the graph that is created by removing all edges $o_{i,j}$. The newly created graph is then given to an approximate graph coloring algorithm. The algorithm is approximate since it may not necessarily compute the maximal graph coloring, i.e., there may be more nodes left uncolored than in the optimal solution. Uncolored nodes are categorized as unclassified, so this is not a disadvantage. The algorithm given in Algorithm 11 is greedy, considers the n nodes in the order $p_1, \cdots, p_n$ and

Algorithm 11 Coloring(i)

```
1: if i = n then
2:    return percentage of covered points
3: end if
4: Compute unused colors for node p_i
5: if no color is viable then
6:    return Coloring(i + 1)
7: end if
8: for available color c do
9:    p_i ← c
10:   if Coloring(i + 1) covers more than 95 % then
11:      return Coloring(i + 1)
12:   end if
13: end for
14: p_i ← min{c}
15: return Coloring(i + 1)
```

assigns the current node p_i the smallest color not used by any of its neighbors. In case no color is available p_i is left uncolored. The algorithm is further augmented with a backtracking procedure. If the solution calculated by the greedy algorithm is insufficient the algorithm backtracks to the latest choice it made and chooses the next higher color. A solution is seen as sufficient if the colored planes cover more than 95 % of all points that correspond to all planes. This is accurate enough to proceed with the improvement step (see 6.1(c) for an impression).

6.1.3 Clustering Algorithm

The above algorithms try to compute solutions to the classification problem that label a maximal number of planes. This is very time-consuming and also somewhat unnecessary. The set of planes as detected in a real data set usually contains a large number of planes that do not need to be labeled. Additionally, with a larger number of planes even computing the full set of relations R becomes increasingly expensive. A further observation is that the planes that we wish to classify usually aggregate around a few points in the spherical coordinate system. This is exploited by the clustering approach as explained in this section. First, the clustering algorithm from Section 5.3.3 is used to group similar clusters of planes together. The maximal angle between two data points is set to α. This is equal to computing the connected components in the graph of planes that is solely connected by parallel edges. For every cluster i a representative normal $\boldsymbol{n}_i'$ is computed by:

$$\boldsymbol{n}_i' = \frac{\sum_{j=1}^{m} m_j \boldsymbol{n}_j}{\sum_{j=1}^{m} m_j}, \tag{6.1}$$

where m_j is the number of points that lie on plane $\boldsymbol{p}_j$. For this to work correctly all normals $\boldsymbol{n}_j$ must lie on the same hemisphere. If necessary, e.g., ceilings and floors are usually on opposite sides of the equator, the normals are flipped. We assume that the major planes that are to be classified are united in their respective clusters. It then follows that we need to find 3 clusters that are orthogonal to each other. After ordering the representatives according to the total number of represented points it is a trivial task to find the biggest three $\boldsymbol{n}'_i$,$\boldsymbol{n}'_j$,$\boldsymbol{n}'_k$ for which:

$$\boldsymbol{n}'_i\boldsymbol{n}'_j = \boldsymbol{n}'_j\boldsymbol{n}'_k = \boldsymbol{n}'_k\boldsymbol{n}'_i = 0$$

holds. Planes belonging to clusters i,j and k are then classified as X,Y and Z. All remaining planes are classified as U.

6.2 Improvement Step

With the planes correctly classified, their poses are now optimized. This is done by minimizing the positional error

$$E(P) = \sum_{\boldsymbol{p}_i \in P\backslash U} \sum_{j=1}^{m_i} |\boldsymbol{n}_i \boldsymbol{x}_j - \rho_i| \,, \tag{6.2}$$

under the side conditions:

$$\begin{aligned} \boldsymbol{n}_i\boldsymbol{n}_j &= 0 && \text{if } i \text{ and } j \text{ are differently classified} \\ \boldsymbol{n}_i &= \boldsymbol{n}_j && \text{if } i \text{ and } j \text{ belong to the same class} \\ |\boldsymbol{n}_i| &\neq 1. \end{aligned}$$

For each i the m_i points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m_i}$ lie on plane p_i. If a plane has not been assigned a type X,Y or Z it will not be included in the global optimization as it has no effect on the pose of the other planes. The side conditions encode the information on the classification of the planes. Parallel planes must have the same normal, while two planes in different classes are necessarily orthogonal. In an additional requirement, the normals of each plane need to be normalized.

There is no closed-form solution to the above equation system. Nüchter [65] remedied this by introducing an additional error term that replaces the side conditions.

$$E'(P) = \sum_{\boldsymbol{p}_i \in P\backslash U} \sum_{j=1}^{m_i} |\boldsymbol{n}_i \boldsymbol{x}_j - \rho_i| + \gamma \sum_{p_i \in P\backslash U} \sum p_j \in P\backslash U c_{i,j}.$$

$c_{i,j}$ is given by

$$c_{i,j} = \min\{|\arccos(\boldsymbol{n}_i\boldsymbol{n}_j)|, |\pi - \arccos(\boldsymbol{n}_i\boldsymbol{n}_j)|\},$$

if the planes p_i,p_j are perpendicular and by

$$c_{i,j} = \left|\frac{\pi}{2} \arccos(\boldsymbol{n}_i\boldsymbol{n}_j)\right|,$$

if they are parallel. The last side condition can be eliminated by restating the normal $\boldsymbol{n}_i$ in polar coordinates as in Eq. (5.2).

This new error metric is then minimized by non-linear optimization algorithms, such as the Downhill Simplex Method or Powell's Method. While the minimum of $E'(P)$ will reduce the error as stated in Eq. (6.2), there is likely to remain a residual error in the $c_{i,j}$'s. This technically violates the side conditions to a small degree.

To improve these results a novel way of optimizing $E(P)$ is presented in this section. First the error metric is reformulated.

$$\begin{aligned} E(\boldsymbol{R}, P) = & \sum_{\boldsymbol{p}_i \in X} \sum_{j=1}^{m_i} \left| (\boldsymbol{R} \cdot \boldsymbol{n}_x)\, \boldsymbol{x}_j - \rho_i \right| \\ & + \sum_{\boldsymbol{p}_i \in Y} \sum_{j=1}^{m_i} \left| (\boldsymbol{R} \cdot \boldsymbol{n}_y)\, \boldsymbol{x}_j - \rho_i \right| \\ & + \sum_{\boldsymbol{p}_i \in Z} \sum_{j=1}^{m_i} \left| (\boldsymbol{R} \cdot \boldsymbol{n}_z)\, \boldsymbol{x}_j - \rho_i \right|, \end{aligned} \tag{6.3}$$

where $\boldsymbol{R}$ is a 3×3 rotation matrix. By introducing the basis vectors $\boldsymbol{n}_x$,$\boldsymbol{n}_y$ and $\boldsymbol{n}_z$ into the equation all side conditions are eliminated. Furthermore, unlike $E'(P)$ this equation is still equivalent to the original error metric (6.2), i.e., their respective minima are the same. We seek to minimize $E(P)$ by finding the appropriate rotation matrix $\boldsymbol{R}$ and all ρ_i's. A rotation matrix is parametrized by 3 values. The number of parameters is therefore only $3 + n'$, where n' is the total number of classified planes, instead of $3 \cdot n'$. Given the optimized rotation $\boldsymbol{R}$ and ρ_i's the optimized normal n_i is given by:

$$\boldsymbol{n}_i = \boldsymbol{R} \cdot \boldsymbol{n}'_i,$$

where n'_i is the base vector for the appropriate class. This also allows us to simplify equation 6.3 to

$$E(\boldsymbol{R}, P) = \sum_{\boldsymbol{p}_i \in X \cup Y \cup Z} \sum_{j=1}^{m_i} \left| \left(\boldsymbol{R} \cdot \boldsymbol{n}'_i\right) \boldsymbol{x}_j - \rho_i \right|. \tag{6.4}$$

Unfortunately, a 3-dimensional rotation matrix is highly non-linear, so that this error metric is not easily minimized. However, by utilizing the linearization described in Section 3.2.5 an iterative minimization technique can be employed. The approximated rotation matrix is given by:

$$\boldsymbol{R} \approx \boldsymbol{I}_3 + \begin{pmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{pmatrix}.$$

Replacing this approximation within the error function (6.4) and rearranging the unknown

variables in a vector yields:

$$\begin{aligned}\left(\boldsymbol{R}\cdot\boldsymbol{n}_i'\right)\boldsymbol{x}_j-\rho_i &\approx \begin{pmatrix}1 & -\theta_z & \theta_y\\ \theta_z & 1 & -\theta_x\\ -\theta_y & \theta_x & 1\end{pmatrix}\boldsymbol{n}_i'\boldsymbol{x}_j-\rho_i\\ &= \boldsymbol{n}_i'\boldsymbol{x}_j+\begin{pmatrix}n_{i,y}'x_{j,z}-n_{i,z}'x_{j,y}\\ n_{i,z}'x_{j,x}-n_{i,x}'x_{j,z}\\ n_{i,x}'x_{j,y}-n_{i,y}'x_{j,x}\end{pmatrix}\begin{pmatrix}\theta_x\\ \theta_y\\ \theta_z\end{pmatrix}-\rho_i\\ &= \boldsymbol{n}_i'\boldsymbol{x}_j+\bar{\boldsymbol{x}}_{i,j}\boldsymbol{r}-\rho_i.\end{aligned}$$

The error metric is consequently approximated as:

$$E(\boldsymbol{r},P)=\sum_{\boldsymbol{p}_i\in X\cup Y\cup Z}\sum_{j=1}^{m_i}\left|\boldsymbol{n}_i'\boldsymbol{x}_j+\bar{\boldsymbol{x}}_{i,j}\boldsymbol{r}-\rho_i\right|.$$

Minimizing Eq. (6.3) is equivalent to minimizing:

$$\begin{aligned}E'(\boldsymbol{r},P) &= \sum_{\boldsymbol{p}_i\in X\cup Y\cup Z}\sum_{j=1}^{m_i}\left|\boldsymbol{n}_i'\boldsymbol{x}_j+\bar{\boldsymbol{x}}_{i,j}\boldsymbol{r}-\rho_i\right|^2\\ &= \sum_{\boldsymbol{p}_i\in X\cup Y\cup Z}\sum_{j=1}^{m_i}\left(\left(\boldsymbol{n}_i'\boldsymbol{x}_j\right)^2+\left(\bar{\boldsymbol{x}}_{i,j}\boldsymbol{r}-\rho_i\right)^2+2\boldsymbol{n}_i'\boldsymbol{x}_j\left(\bar{\boldsymbol{x}}_{i,j}\boldsymbol{r}-\rho_i\right)\right)\end{aligned}$$

Concatenating the parameters into a vector $\boldsymbol{X}=\left(\theta_x,\theta_y,\theta_z,\rho_1,\cdots,\rho_{n'}\right)^T$ this can be stated in matrix form as:

$$E'(\boldsymbol{X})=\boldsymbol{X}^T\boldsymbol{B}\boldsymbol{X}+2\boldsymbol{A}\boldsymbol{X}.$$

The minimum is then given by the solution to the linear equation system:

$$\boldsymbol{B}\boldsymbol{X}=\boldsymbol{A}, \tag{6.5}$$

where the symmetric matrix $\boldsymbol{B}$ and the vector $\boldsymbol{A}$ are given by:

$$\boldsymbol{B}_{[1:3,1:3]} = \sum_{\boldsymbol{p}_i \in X \cup Y \cup Z} \sum_{j=1}^{m_i} \bar{\boldsymbol{x}}_{i,j} \bar{\boldsymbol{x}}_{i,j}^T \tag{6.6}$$

$$\boldsymbol{B}_{[4:3+n',4:3+n']} = \begin{pmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & m_{n'} \end{pmatrix}$$

$$\boldsymbol{B}_{[1:3,4:3+n']} = \begin{pmatrix} -\sum_{j=1}^{m_1} \bar{\boldsymbol{x}}_{1,j} & \cdots & -\sum_{j=1}^{m_{n'}} \bar{\boldsymbol{x}}_{n',j} \end{pmatrix}$$

$$\boldsymbol{B}_{[4:3+n',1:3]} = \boldsymbol{B}_{[1:3,4:3+n']}^T$$

$$\boldsymbol{A}_{[1:3]} = \sum_{\boldsymbol{p}_i \in X \cup Y \cup Z} \sum_{j=1}^{m_i} \left(\boldsymbol{n}_i' \cdot \boldsymbol{x}_j \right) \bar{\boldsymbol{x}}_{i,j}$$

$$\boldsymbol{A}_{[4:3+n']} = \begin{pmatrix} -\sum_{j=1}^{m_1} \boldsymbol{n}_1' \cdot \boldsymbol{x}_j & \cdots & -\sum_{j=1}^{m_{n'}} \boldsymbol{n}_{n'}' \cdot \boldsymbol{x}_j \end{pmatrix}$$

Minimizing the linearized error metric by solving the linear equation system will not yield the exact solution that minimizes the original metric. Therefore, the linear optimization is iterated until it converges to a stable set of normals $\boldsymbol{n}_i'$. To increase the chance to converge to the global minima the starting estimate for the three normals $\boldsymbol{n}_x$,$\boldsymbol{n}_y$,$\boldsymbol{n}_z$ is calculated as follows:

$$\begin{aligned} \boldsymbol{n}_x &= \boldsymbol{n}_x' \\ \boldsymbol{n}_y &= \boldsymbol{n}' y - \left(\boldsymbol{n}_x \cdot \boldsymbol{n}_y' \right) \boldsymbol{n}_x \\ \boldsymbol{n}_z &= \boldsymbol{n}_x \times \boldsymbol{n}_y \end{aligned} \tag{6.7}$$

where $\boldsymbol{n}_x'$ and $\boldsymbol{n}_y'$ are the representatives of X and Y as given by equation (6.1). $\boldsymbol{n}_x$ is arbitrarily set to the representative of class X. $\boldsymbol{n}_y$ is the projection of $\boldsymbol{n}_y'$ onto the plane that is defined by $\boldsymbol{n}_x$. $\boldsymbol{n}_z$ is then consequently given by the cross product of $\boldsymbol{n}_x$ and $\boldsymbol{n}_y$. By construction all three normals are perpendicular to each other and somewhat approximate the classified planes. Starting with these estimates the equation system (6.5) is solved and the calculated angles are used to rotate the normals. With the rotated normals as new estimates, the process is repeated until convergence. The complete improvement step is summarized in Algorithm 12.

After the correct pose of each plane has been optimized a last relation between planes remains to be considered. Often times several planes will describe the same continuous surface that stretches over several planes. This property can best be identified in the corrected plane model. Two planes p_i,p_j are found to describe the same surface if their classification is equal and $\rho_i \approx \rho_j$. If two or more planes are found to be equal by a simple threshold method the planes are joined. The collective ρ' is computed as a weighted average by:

$$\rho' = \frac{\sum_{\boldsymbol{p}_i \in E} \sum_{j=1}^{m_i} \left(\boldsymbol{n}_i \boldsymbol{x}_j \right)}{\sum_{\boldsymbol{p}_i \in E} m_i},$$

where E is the set of equal planes.

Algorithm 12 Plane Model Optimization

1: Compute starting estimates $\boldsymbol{n}'_{i,0}$ for all normals $\boldsymbol{n}'_i$ (Eq. (6.7)).
2: **repeat**
3: Construct linear equation system $\boldsymbol{BX}_j = \boldsymbol{A}$ by using the normals from previous iterations (Eq. (6.6)).
4: Solve for $\boldsymbol{X}_j$ by inverting $\boldsymbol{B}$.
5: Compute $\boldsymbol{n}'_{i,j+1}$ by:

$$\boldsymbol{n}'_{i,j+1} = \boldsymbol{R}_{\theta_{x,j},\theta_{y,j},\theta_{z,j}} \boldsymbol{n}'_{i,j}$$

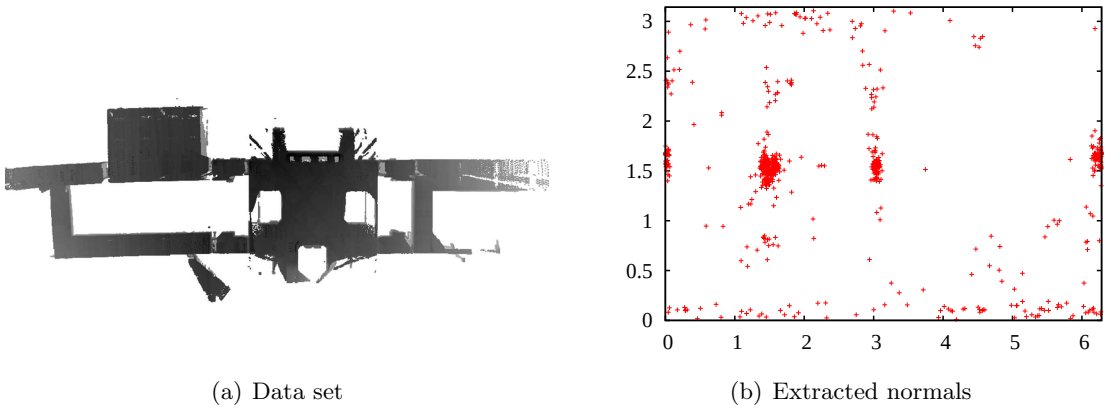
6: $j \leftarrow j + 1$
7: **until** $\theta_{x,j}, \theta_{y,j}, \theta_{z,j} \approx 0$

6.3 Experimental Results

Fig. 6.2(a) shows the performance of the 3 classification algorithms. Even though the Prolog program is the faster algorithm for a small number of planes in [65] it does not perform well when the amount of planes increases to a more realistic number. As Prolog exhausts every possible assignment when looking for an exact solution its computation time can reach extreme heights. For an unfavorably formed problem Prolog may require an hour, while it will find a solution under a second for a similarly sized but more favorable problem. The less exact vertex coloring algorithm is more flexible towards "hard" problems as it may simply stop when a good enough solution is found. The clustering algorithm clearly outperforms the other algorithms by a very large margin. The computation time remains relatively constant. Even when using several hundred planes as input the algorithm computes the clustering in less than a second. Evaluating the quality of the classification algorithms is done on a showcase data set using 63 laser scans from the fourth floor of the university building in Osnabrück. After extracting almost 700 planes, the three classification algorithms were started. Because the vertex coloring as well as the Prolog program computed a very similar classification the second classification was omitted for brevity. The results are depicted in Fig. 6.1(d). Many of the detected planes do not belong to either floors, ceilings or walls. The clustering algorithm is therefore preferable to the other approaches because it produces a more realistic labeling

The required computation time for the improvement step never exceeded 30 ms. The usual number of iterations required for convergence is 5. See Fig. 6.2 for an example.

A complete optimization of the plane model is demonstrated in Fig. 6.3. The algorithm was applied to only 4 scans where the Hough Transform detected 32 planes. A smaller data set was used to more easily demonstrate the effectiveness of the algorithm. After classifying the planes using the clustering approach, the plane model is optimized and planes are joined as necessary. The optimization results in only 6 planes accurately describing the environment.



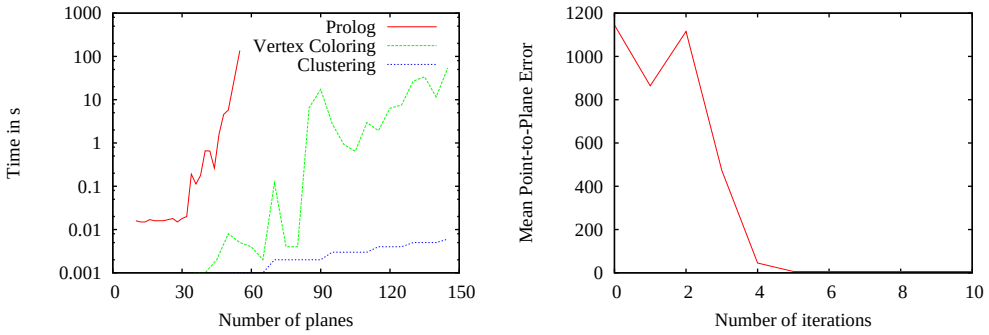
(a) Data set

(b) Extracted normals

(c) Classification by vertex coloring

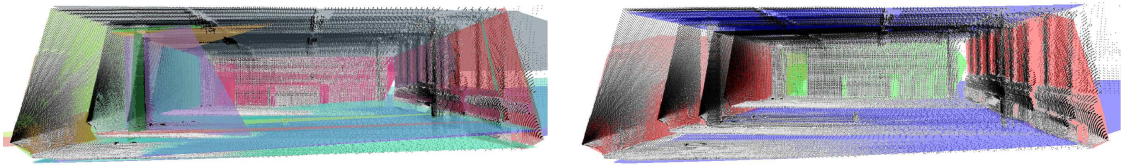
(d) Classification by clustering

Fig. 6.1: (a) The data set as used in the experiments consists of 63 laser scans with 81360 points each. (b) The Hough Transform found 690 planes. Their normals are plotted in spherical coordinates. The normals are concentrated around a few points similar to the pattern in an accumulator during the Hough Transform. (c) With the exception of one plane, the vertex coloring algorithm classified all planes. As Prolog usually computes similar solutions to the vertex coloring algorithm, this result is representative of the Prolog algorithm as well. (d) The classification of the planes as computed by the clustering approach. All normals have been flipped to the side of the unity sphere where the corresponding cluster has first been identified. The algorithm classified a total amount of 568 planes. 122 planes remain unclassified. These are mostly normals that lie between the 3 main clusters and are of dubious relevance to the optimization.



(a) Runtime of the classification (b) Convergence of the model improvement

Fig. 6.2: (a) Comparison of the running time of the algorithms for solving the classification problem. The Prolog program is only plotted till a total number of 55 planes is reached. Beyond this point the algorithm is prohibitively slow and does not allow further measurements. Each data point is the average computation time of the respective algorithm in 50 trials with randomly selected planes from a real data set (seen in Fig. 6.1(a)). Maximal and minimal times are not shown, as the variance of both the Prolog program and the Vertex Coloring algorithm is so large that they obscure the plot. The clustering approach vastly outperforms the more exact methods and is clearly the algorithm of choice.
(b) Demonstration of the convergence of the model improvement for an exemplary data set with 579 planes. The algorithm converges after only 5 iterations. The complete process took less than 10 ms.



(a) Detected Planes (b) Extracted normals

Fig. 6.3: The detected (left) and optimized (right) plane model for 4 scans of the data set in the data set from Fig. 6.1. Initially the Hough Transform detected 32 planes. There are several planes that describe the same surface, e.g., the green, yellow and blue planes describe the wall on the left. There are also a few planes that do not correspond to the architecture of the room, e.g., the small violet plane on the left side. In the optimized plane model planes describing the same surface are correctly joined while artifacts are filtered and planes are classified as floors and walls.

Chapter 7

Deforming Scans

The previous two chapters describe in detail two crucial elements in our map improvement strategy. By extracting planes using the Hough Transform, we aim to identify structures that are inherently straight and undeformed in the real environment. With information about the true underlying geometry we apply a Thin Plate Spline deformation on each scan. By this we hope to build a global model of the environment where all scans are consistent with each other.

This chapter explains how the Hough Transform as well as the Thin Plate Spline are used to improve the quality of a map. By combining these elements in different ways several algorithms emerge with diverse properties. Both the Hough Transform as a means of finding the global structure of the environment as well as the Thin Plate Spline as a ways of transforming the scans may be replaced by alternative techniques.

As the way of finding the global structure is more pivotal to the rest of the algorithm, we split the chapters along those lines. Section 7.1 lays out several algorithms that use the Hough Transform to identify planes. Section 7.2 gives an alternative algorithm that does not depend on specially structured environments.

7.1 Deformation using Geometrical Information

This section outlines a family of map improvement algorithms. They all have in common that they establish their non-rigid deformation by help of plane information obtained by the Hough Transform. They differ in the specific interpretation of that information, i.e., how the desired point positions are computed. The different algorithms are laid out and evaluated in the following sections.

Before any improvement can be done the scans have to be rigidly registered as best as possible. Only when the map is already aligned well enough can the further steps be realized reliably. To do this we first employ the ICP algorithm to align the scans sequentially. After the pairwise matchings have been computed the global rigid alignment according to Lu and Milios [15] takes over to create a globally consistent map. For each scan the Hough Transform is executed to find a set of planes. These planes are then optimized so that they comply with a general geometrical model and fit the registered map as best as possible. This is done using the scheme laid out in Chapter 6. From the so produced structures we extract global points, whose

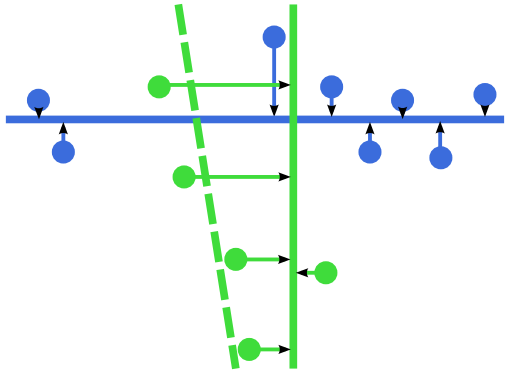


Fig. 7.1: Close-up view of the intersection of two planes. The green plane was first detected at the dashed position. The green points near the intersection have incorrectly been attributed to the green plane. The resulting correspondences are contradicting each other as seen by the overlapping projections.

positions correspond to the true underlying geometry. A summary is given in Algorithm 13.

This last step of the algorithm is mutable and allows two configurations of the non-rigid deformation and global point extraction. The most obvious choice for the computation of global point coordinates is to exploit the point-to-plane correspondences, for example by choosing the projection of a point to its plane as its desired global point. This information can be fed into the TPS or the slice based deformation as in Section 7.1.2.

Prior to using the point-to-plane information in either the TPS or the slice-based deformation some of the correspondences need to be removed. Fig. 7.1 illustrates the situation after step 3 in the algorithm. The order in which the planes are detected determine the correspondence of a point in a corner of the environment. This is undesirable since highly contradictory desired point positions emerge. These points are identified and removed before the scans are deformed to avoid unnecessary warp in those places. All points that could have been assigned to two or more planes are considered dubious and are filtered out. It must be pointed out that the removal procedure takes place before the plane positions have been optimized, i.e., between step 2 and 3. Applying it after step 3 would invariably result in errors. Points may not be recognized to be problematic when one of the corresponding planes has moved away from it.

Algorithm 13 Map Improvement using Plane Extraction

1: Rigidly register all scans using the globally consistent alignment framework as in Chapter 3.
2: Use Hough Transform to extract planes in every scan (Section 5.1).
3: Improve the Plane Model (Section 6).
4: Extract global point information (Section 7.1.1 and 7.1.2).
5: Deform each scan accordingly using either the TPS (Chapter 4) or the minimization in Section 7.1.2.

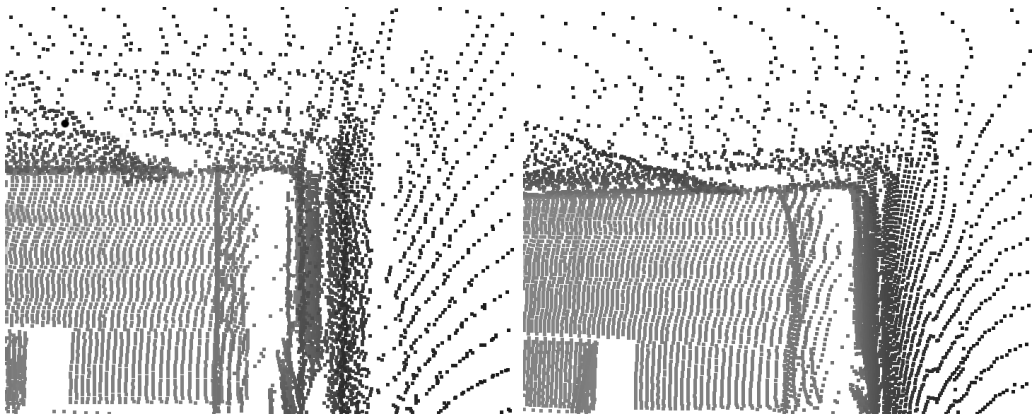


Fig. 7.2: Comparison of the initial (left) and the deformed (right) scan after the optimization using points on all detected planes. Due to contradictory information on perpendicular planes a twisting warp is introduced into the scan. This can be clearly seen on the pillar in the center of the picture that is warped after the optimization.

7.1.1 Deformation with TPS

After filtering the points optimizing the plane model we choose the projection of a point to its plane as the desired global point. With each point having a corresponding desired point a TPS deformation that transforms the scan could theoretically be created. However the number of points is usually too high to compute the TPS. In order to handle the computational burden we evenly subsample the points over the whole scan. The data cloud is represented as an octree and a maximal number of points n is randomly selected from the smallest octant with a side length of d. This achieves a maximal point density of $\frac{p}{d^3}$ over the whole scan. We aim to use approximately 5000 points for each scan. Of these we randomly choose 20 % to represent the basis function subset in the TPS approximation as described in Section 4.3.2.

Experimenting with this algorithm on a few laser scans soon reveals a problem. As demonstrated in Fig. 7.2, new warp is introduced near the intersection of planes.

Even though points near plane intersections are filtered, the remaining points do not take into account the deformation that takes place on planes that are perpendicular. Therefore the resulting deformation twists points near the intersection of these planes. The procedure is modified to accommodate this by transforming the scan in multiple steps. First, only points from floors and ceilings are used to form a TPS functional. After the scan has been deformed using this TPS, the same is done with points from the two remaining sets of planes representing the walls. This produces scans similar to Fig. 7.3.

Separating the deformation of perpendicular planes from each other forces the TPS functional to generalize the deformation across the scan without adding additional distortions. There remains one fundamental problem with directly feeding point-to-plane correspondences into the TPS. Small features like the lamps on ceilings are flattened so that flat surfaces emerge. This is partly due to points from these features being attributed to the plane the feature is close to. However, even when only correct correspondences are used the same effect occurs. Due to noisy

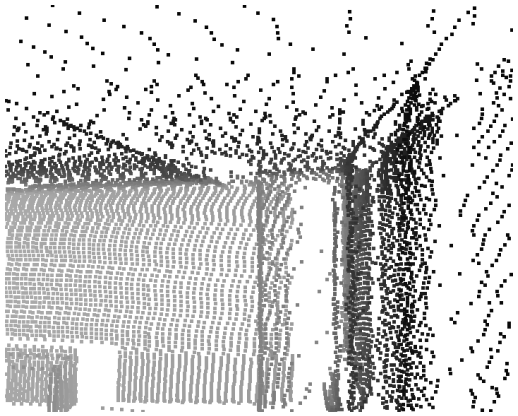


Fig. 7.3: The same scan as in Fig. 7.3 optimized with separate deformations. The twisting warp in the corners of the room are absent and the ceilings and planes are flattened.

point measurements and the systematical errors that we wish to correct, there are points on both sides of a plane. The TPS then generalizes this and everything in the surrounding area of a plane is projected onto that surface. Combining the Thin Plate Spline deformation with information derived only from planes will therefore always lead to the obfuscation of detail in the laser scans.

7.1.2 Deformation without TPS

As seen in the previous section, eliminating deformations using the Thin Plate Spline in a highly structured scan from an office environment is problematic. Deforming a point cloud so that totally planar areas emerge is usually ineffective due to the nature of the TPS. This is especially true when trying to correct errors as seen in Fig. 7.4.

Errors such as these are the result of irregular servo movements of rotating laser scanners. A 3-dimensional point cloud is assembled from several 2-dimensional laser scans taken at different servo positions. This is done by rotating the laser scan by the angle of the servo. If at the time of acquiring a particular range scan the servo position differs from its assumed position the scan is inserted incorrectly. These individual 2D laser scans are also referred to as slices.

We propose a novel optimization technique to improve the scan quality for these type of scanners. Again, we assume that in a scan a number of planes $p_i = (\boldsymbol{n}_i, \rho_i)$ have been identified. Without loss of generality the rotation axis of the scanner is assumed to be the x-axis. If this is not the case all planes and points of the 3D scan must be transformed accordingly before optimization. For each slice, we aim to compute the correct orientation the scanner had when it was taken by minimizing the summed point-to-plane distances.

Not all of the planes p_i can be used to improve the scan quality. Planes that are perpendicular or close to perpendicular to the scanner axis do not sufficiently restrict the angle of the slices. Using a pitch-angle scanner as on the Kurt3D platform this occurs with walls to the right and left of the robot. These planes are removed before optimization by a simple threshold on the angle between the plane normal and the scanner axis.

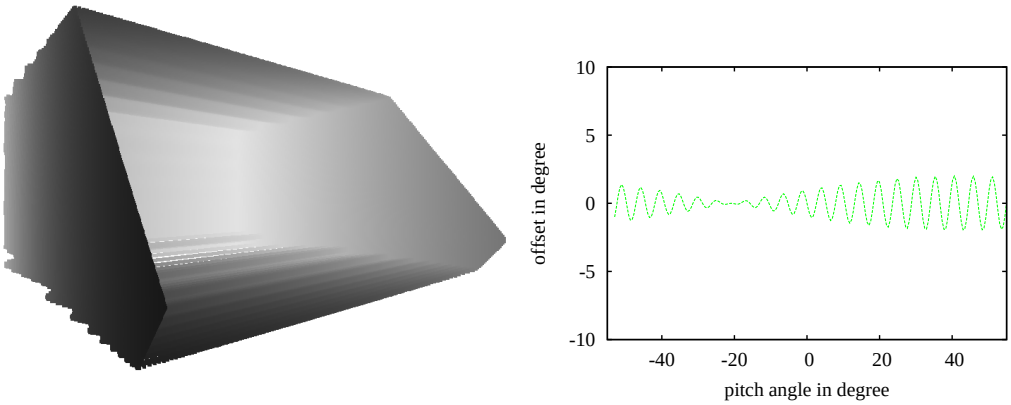


Fig. 7.4: Left: A simulated scan of a room with randomly generated servo errors that occur with a pitch-angle laser scanner. Right: The offset that was generated to simulate an oscillation of the rotating scanner. The offset for a slice with the assumed angle φ was generated by the function $E(\varphi) = A\sin(p_1\varphi)\sin(p_2\varphi + P)$. Here, p_1 and p_2 control the periodicity of the function and remain constant. The parameters A and P control the amplitude and the phase offset, respectively.

For each slice j consisting of the points $\boldsymbol{x}_j$ we seek an α that minimizes the error metric

$$E = \frac{1}{n}\sum_{p_i}\sum_{\boldsymbol{x}_j \in p_i} \left|(\boldsymbol{R}_\alpha \boldsymbol{x}_j)\,\boldsymbol{n}_i - \rho_i\right|,$$

where n is the total number of points associated to planes and $\boldsymbol{R}_\alpha$ is the rotation matrix describing a rotation around the x-axis by α. The α that minimizes the error is then used to optimally bring the slice into the correct position.

The error metric is minimized in an iterative fashion by using the small angle approximations $\sin(\alpha) \approx \alpha$ and $\cos(\alpha) \approx 0$. The resulting approximated error

$$E \approx \frac{1}{n}\sum_{p_i}\sum_{\boldsymbol{x}_j \in p_i} \left|\boldsymbol{x}_j\boldsymbol{n}_i + \alpha\,(n_z x_y - n_y x_z) - \rho\right|$$

is minimized by α as given by:

$$\alpha = \frac{1}{n}\sum_{p_i}\sum_{\boldsymbol{x}_j \in p_i} \frac{\rho_i - \boldsymbol{x}_j\boldsymbol{n}_i}{n_{i,z}x_y - n_{i,y}x_z}.$$

The improvement algorithm is evaluated on real as well as on simulated data. The computation time of the improvement step in all examples was less than 1 ms per scan on modern hardware.

We applied the algorithm to every one of the 63 scans from the fourth floor of the AVZ university building in Osnabrück. The improvement is remarkable. The errors that produce wave-like effects on floor and ceiling were reduced and the scans were visibly improved. Fig. 7.5 shows a representative scan.

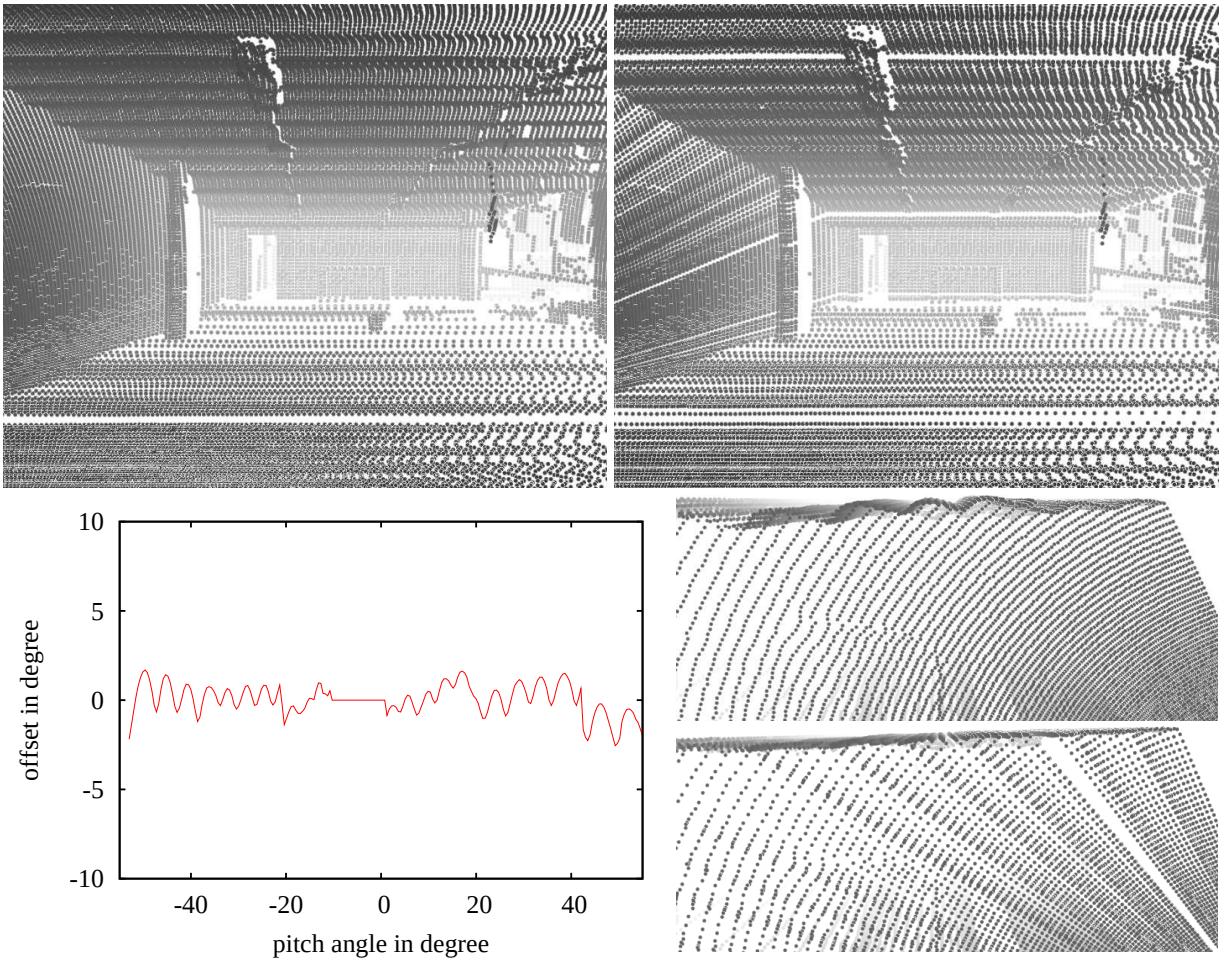


Fig. 7.5: Depiction of a scan of the soccer field on the fourth floor in the AVZ building in Osnabrück before and after improvement. Top Left: The initial scan with incorrectly aligned 2D laser scans. Top Right: The corrected scan with a straight floor and ceiling, where details are more clearly distinguishable. Due to the oscillation of the laser scanner the points are unevenly distributed, making the corrected scan look slightly disjointed. Bottom left: The angle by which each 2D scan is rotated behaves sinusoidal. In our experimentations with real data the amplitude and angular phase shift differ each time, while the oscillation period remains relatively constant over all 3D scans. Bottom Right: Close-up view of the ceiling of the initial scan and the corrected scan.

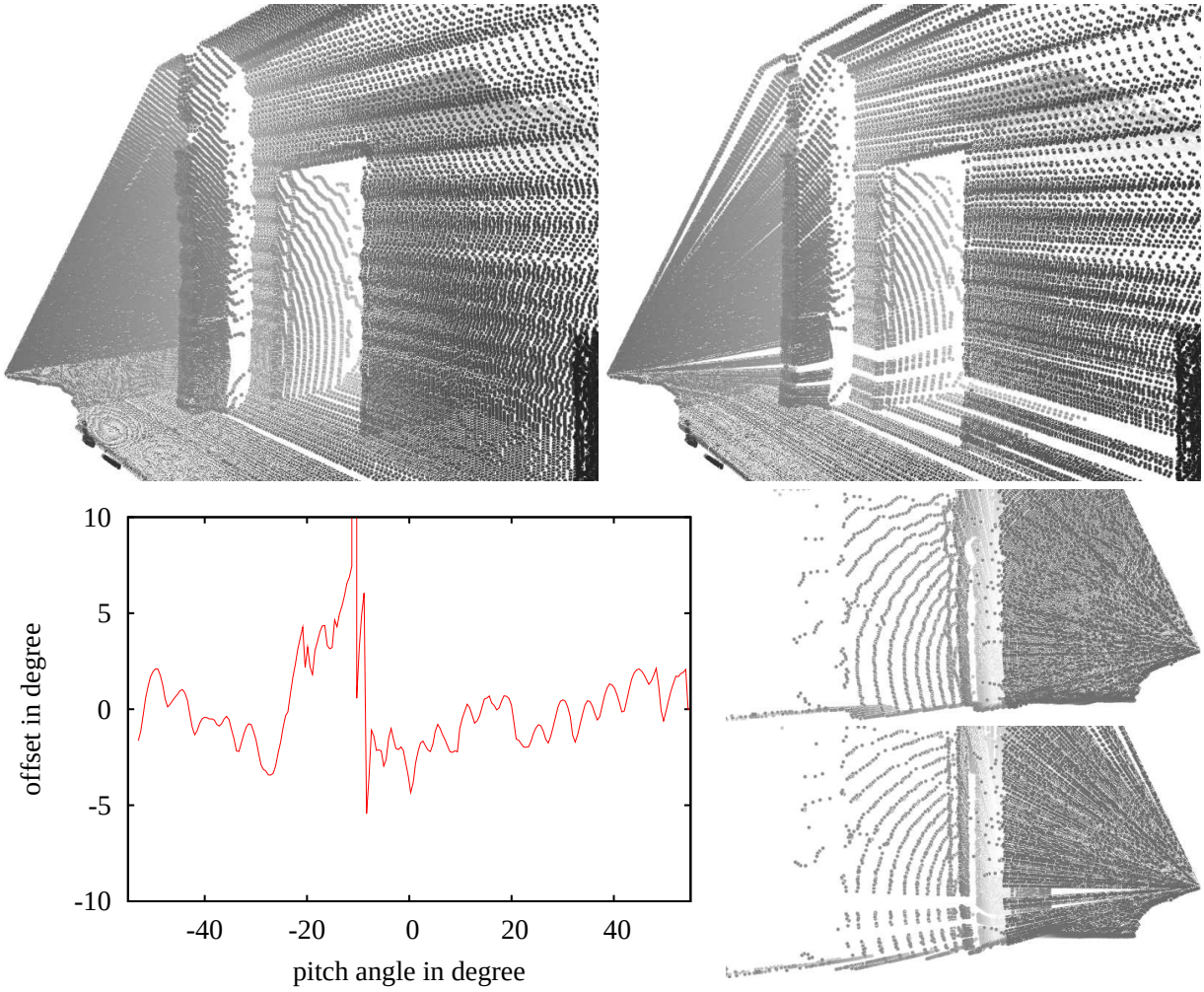


Fig. 7.6: Depiction of a second scan of the soccer field on the fourth floor in the AVZ building in Osnabrück before and after improvement. In this scan the wall in front of the scanner is more prominent while other planes are much less apparent. In this particular scan the improvement algorithm introduced some new error into the scan. Top Left: The initial scan with incorrectly aligned 2D laser scans. Top Right: The corrected scan where new errors in the floor scans appear. 2D scans that measured the floor behind the door have considerably more points that stem from the wall to the left of the scanner. Some of the points have incorrectly been assigned to the floor so that they negatively influence the error metric. Slices that are dominated by the wall in front of the robot are again correctly aligned by the improvement so that the pillar is much more correctly characterized by the 3D scan. Bottom left: The improvement angle is unusually large and erratic for those slices where the error occurred. Bottom Right: Close-up view of the introduced error in the floor. While the optimization in the lower picture clearly improves the floor close to the robot's position the slices immediately behind the door are incorrectly positioned.

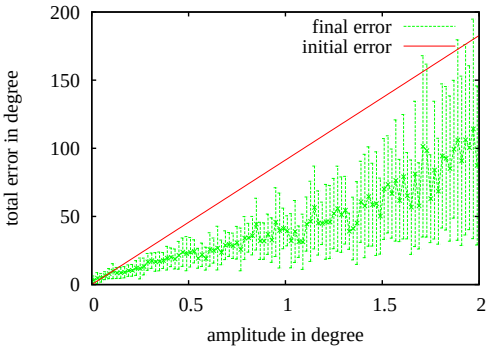


Fig. 7.7: Results of the improvement using simulated data. The error bars illustrate the variance for each experiment. With increasing initial error (red line) the accuracy of the improvement decreases. The quality of the planes as detected by the Hough Transform suffers from the generated errors. Consequently the error of the improvement increases even when a visually improved scan with straight floors and ceilings is produced.

In some cases however (in 8 of 63 scans) the algorithm encountered difficult geometry as in Fig. 7.6 and parts of the scans were badly misaligned. This often occurred on slices that were acquired shortly before the scanner reached a parallel position to the floor. Points from these slices tend to be mistaken for points stemming from the floor. The following improvement steps then find an α that brings these points closer to the floor so that the slice is misaligned. However, even in these worst cases, most slices of the scan were brought into a visually more appealing position.

The actual oscillating behavior of the laser scanner can be observed when plotting the α's against the slice number as in Fig. 7.5 and 7.6. The amplitude of oscillation does not only change from scan to scan but also from oscillation to oscillation. Conversely, the period of oscillation remained almost constant over all 63 scans.

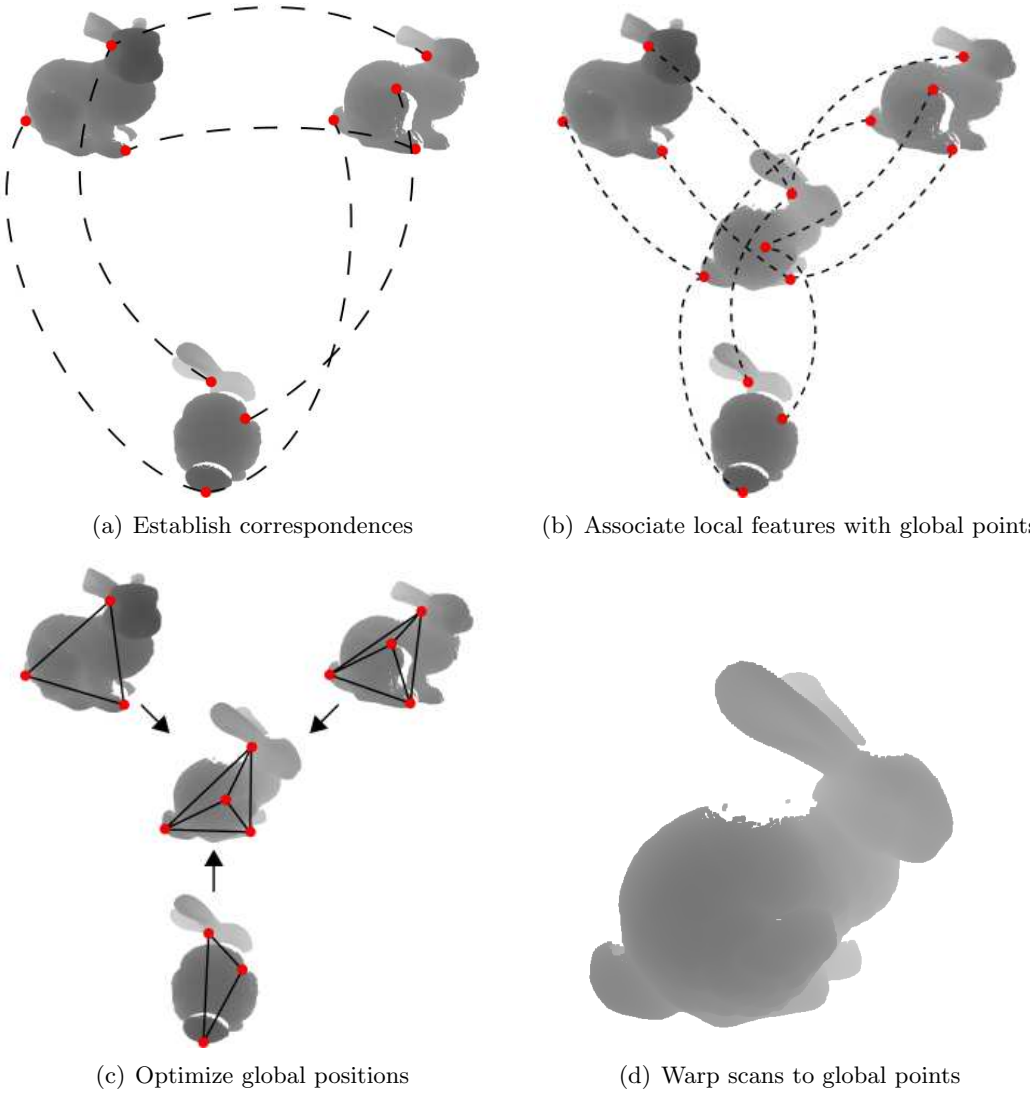
To objectively demonstrate the effectiveness of the improvement algorithm we generate simulated scans where a variable amount of oscillation error is present. Such a simulated scan as well as the generated oscillation error is presented in Fig. 7.4. For different settings for the amplitude A in the error function $E(\varphi) = A\sin(p_1\varphi)\sin(p_2\varphi + P)$ we randomly generated 20 scans with a random phase offset $P \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. The mean error of the optimized scans, i.e., the summed difference between the estimated and the correct α offset is plotted against the amplitude A in Fig. 7.7. Clearly, the effectiveness of the improvement suffers from higher initial errors. This is mostly due to the plane detection algorithm extracting less accurate planes the higher the errors are.

7.2 Deformation without Geometrical Information

In the previous section we focused on structured environments where plane information can be extracted. This section will discuss a strategy to improve the map quality of any structured and unstructured environment. It was first described by Brown and Rusinkiewicz [18, 19]. We propose an improvement on the given algorithm and evaluate it on simulated as well as on real world data.

The proposed algorithm is an extended scan matching procedure where non-rigid deformations are allowed. Rigid sequential scan matching is problematic due to the summation of errors. This problem is even more pronounced in non-rigid scan matching. Small errors in the matching process usually accumulate and result in large scale incorrect warps that are applied to scans later in the sequence. Even when registering a scan to all previously aligned scans, non-rigid deformations result in incorrect behavior. Imagine a loop-closing scenario where a scan closes the gap in a sequence. The scan is naturally stretched to "best" fit the scan regardless of any discrepancy between the size of the gap in the map and in the environment. This is due to the ability of non-rigid deformations to register a scan with any set of corresponding points. Unfortunately there is no trivial way to perfectly discern whether the deformed scan is actually desirable because the point-to-point errors of the correspondences are fully minimized. The proposed algorithm is therefore a global scan matching algorithm that computes the non-rigid alignment of all scans simultaneously.

In general environments plane information is not a sufficient indication for the desired positions of features in laser scans. The algorithm does not rely on any special type of feature. Features are simply selected as a random subset of a scan. Computing their correct positions is done via a two step process. First, the correspondences between features in pairs of scans are established by a variant of the Iterative Closest Points (ICP) algorithm. Second, for each of the features a desired point is computed from the correspondences by globally minimizing an error metric. Now that each scan has a set of features with associated desired points a thin plate spline deformation is calculated and each scan is deformed accordingly. Fig. 7.8 gives an overview over the complete process.



(a) Establish correspondences

(b) Associate local features with global points

(c) Optimize global positions

(d) Warp scans to global points

Fig. 7.8: The individual steps of the global non-rigid alignment algorithm. First, the pairwise correspondences between features selected in the laser scans are computed. From the features and their correspondences a number of global points are identified. These are usually only a smaller subset because some features may have no suitable correspondence, or multiple features may refer to the same global point. The positions of the global points are then optimized using the distances between the local features as estimates. After a consistent global point set is computed, the scans are warped to fit the point set.

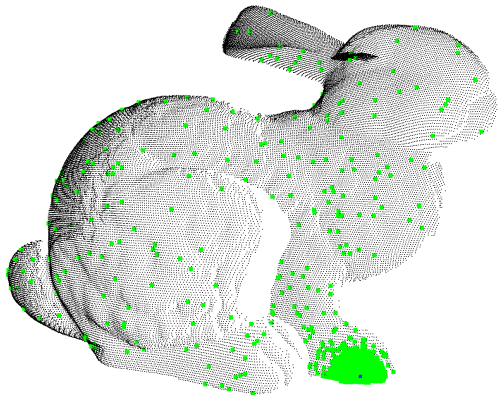


Fig. 7.9: The points from the source scan as selected by the modified selection process. The points (green squares) are selected in the vicinity of the feature point (blue/dark square) on the paw.

7.2.1 Finding Features and their Correspondences

Feature points should ideally cover the entire model and should be chosen so as to yield stable correspondences between scans. It is intuitive to pick features at significant places in the environment, e.g., the eye of the Stanford Bunny. This would however require some knowledge about the geometry of the entire environment and would not guarantee a sufficient coverage.

Features are therefore selected on each laser scan. Correspondences for these features are then established in every other scan. The features are selected using uniform random sampling to ensure coverage everywhere in the environment. As the feature points will constitute the control points that define a TPS deformation, the number of points that are required depends on the degree of warp in the point clouds. In practice, the amount of warp in laser scans is relatively small so that a fixed percentage of points from each scan (approx. 1 %) suffices.

After the feature points $\boldsymbol{m}_i$ have been selected for each scan the next step is to find the correspondences. For each $\boldsymbol{m}_i$ a corresponding point $\boldsymbol{m}_j$ on each scan that overlaps it is sought-after. In order to efficiently find many thousands of correspondences for points distributed across many scans, a novel variant of the ICP algorithm is used. ICP is usually seen as a tool to find the correct alignment of two point clouds. However, it also estimates the correspondences of those points. It is therefore employed as a correspondence estimator. Instead of roughly estimating correspondences for all points we wish to compute the most precise estimate for each feature $\boldsymbol{m}_i$. This is equivalent to increasing the accuracy of the alignment in areas close to $\boldsymbol{m}_i$. This is achieved by locally weighted ICP, which gives higher importance to points surrounding the feature. After the algorithm converges, the closest point to m_i on the second scan is used as the corresponding point.

The locally weighted ICP is based on the standard ICP baseline algorithm with a point-to-point error metric and closest-point computations using k-D trees for speed up. The only change to the standard algorithm is in the point-selection stage. The set of corresponding points that are considered in the minimization step is sampled using a probability distribution that makes it more likely that points near $\boldsymbol{m}_i$ are used. It is a decreasing function of the distance from the

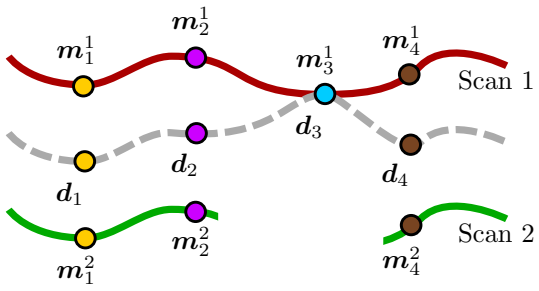


Fig. 7.10: When computing the global points, care must be taken to choose mutually consistent positions that prevent unnecessary warp. Depicted here are the global points as calculated by simply averaging over the correspondences. Because the blue feature $\boldsymbol{d}_3$ has no correspondence on the second scan, its position is inconsistent with that of the other features.

feature point in question:

$$p(\boldsymbol{x}) = \frac{1}{|\boldsymbol{x} - \boldsymbol{m}_i|}.$$

The probability distribution is normalized and numerically inverted to transform the random variable into samples to be selected. This is called the inversion method for sub-sampling [26].

Fig. 7.9 shows the results of the modified corresponding point selection process. The points used for ICP are selected in the vicinity of the feature points, leading to lower alignment errors in those regions, and therefore more accurate correspondences.

For each of the selected feature points $\boldsymbol{m}_i$ a separate locally weighted ICP is run, and the nearest point on the target range scan is selected as the correspondence to $\boldsymbol{m}_i$. Because an initial registration of the laser scans has already been performed, only few iterations of the locally weighted ICP are required for convergence. In addition, the computationally most expensive part of the ICP, i.e., building the k-D tree is done only once per scan. The computation of the correspondences is therefore not as time consuming as it may appear at first.

7.2.2 Computing the Global Point Positions

For any given feature i, we denote the location of its correspondence on scan k by $\boldsymbol{m}_i^k$. Note that these points are not simply the relabeled features $\boldsymbol{m}_i$ that were selected before the correspondence computation in the previous section. They also include the points that were found to correspond to the initially selected features from all overlapping scans. Points for which no adequate correspondence could be found are consequently not included. In order to calculate a consistent deformation for all scans, a global position $\boldsymbol{d}_i$ for each feature is computed. These positions are computed based on the known correspondences $\boldsymbol{m}_i^k$ in such a way as to minimize unnecessary deformations as in Fig. 7.10.

In each scan k the configuration of the global point positions should be as similar to the arrangement of the corresponding local features as possible. Under the assumption that the scans contain no deformation or errors of any kind and the correspondences have been computed

correctly, the arrangements of both points are identical. If the scans are slightly deformed this correlation will only hold approximately.

There are two naive ways of estimating the global points $\boldsymbol{d}_i$'s that give some insight into this relationship. First, the position of each global point i can be set to the location where the feature i was first selected. That is, if a feature i was originally selected on the laser scan k then $\boldsymbol{d}_i = \boldsymbol{m}_i^k$. In case the scans have not been perfectly aligned, this would result in a situation that is similar to the non-rigid loop closing problem laid out initially. The global point positions would not only be mutually inconsistent but also dependent on the sequence in which the original features have been selected. The second method is to place the position of $\boldsymbol{d}_i$ at the mean of $\boldsymbol{m}_i^k$ over all k's, i.e.:

$$\boldsymbol{d}_i = \frac{1}{n} \sum_k \boldsymbol{m}_i^k.$$

This, too, has undesired consequences. For example, one set of global points may have been estimated from features found on scan 1 and 2 while another set of global points only from features on scan 2. Then it is clear that there will be a jump between one set and the other. See Fig. 7.10 for a visual depiction of the problem.

Guided by the intuition that global point positions should be affected by their geometric configuration relative to neighboring features, Brown and Rusinkiewicz optimize the positions $\boldsymbol{d}_i$ by attempting to preserve their relative distances as indicated by the computed correspondences within the scans. Consequently the points are no longer represented as simple positions in a global coordinate system but relative to each other. A "spring" with non-zero rest length is placed between all pairs of features i and j. Assuming feature i was originally selected on scan k and feature j on l, the rest length is set to

$$\frac{1}{2}\left(\left|\boldsymbol{m}_i^k - \boldsymbol{m}_j^k\right| + \left|\boldsymbol{m}_i^l - \boldsymbol{m}_j^l\right|\right).$$

Then an error metric is formulated:

$$E = \sum_k \sum_i \sum_{\substack{j \\ j>i}} w_{i,j} \left(|\boldsymbol{d}_i - \boldsymbol{d}_j| - \left|\boldsymbol{m}_i^k - \boldsymbol{m}_j^k\right|\right)^2. \tag{7.1}$$

When computing this weighted sum it is necessary to consider that the inner sum over i and j is dependent on k. This means the inner sum is not over all i, j but over all i, j so that:

- Both features i and j were originally selected on k, or
- Feature i was selected on k and feature j, selected in some other scan has a valid correspondence m_j^k on k.

See Fig. 7.11 for a closer look on how the sum is computed.

The error metric can be conceptualized as a measure of the discrepancy between the distances of the global points and the distances of the corresponding local features. Minimizing this discrepancy clearly leads to a geometric configuration of the global points that closely resembles

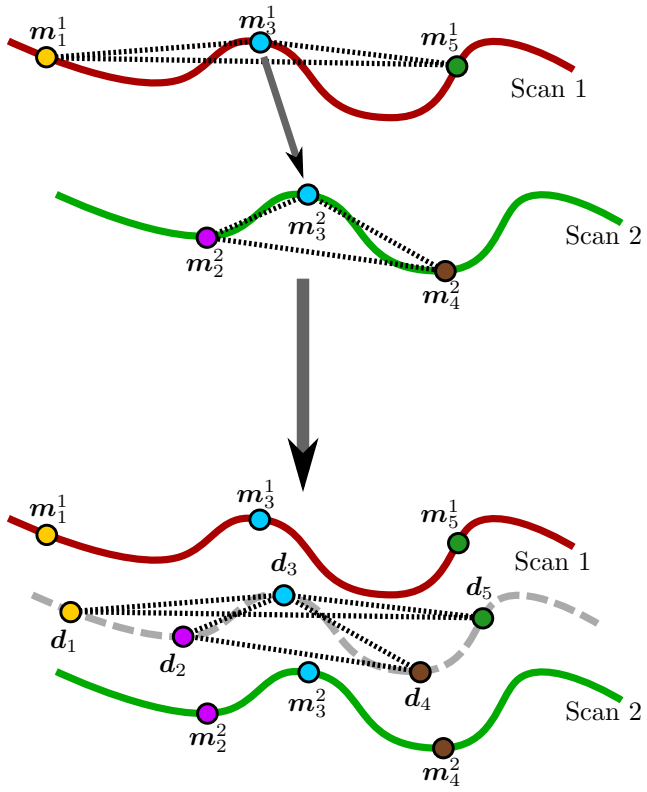


Fig. 7.11: Global point positions as computed by the error metrics. Top: Local feature points are represented relative to each other. This can be interpreted as springs between the features with a rest length based on the distances in the range scan. Each of the springs is one summand in the error metric. Bottom: Global point positions are then adjusted to minimize the total spring energy. In this example only the feature $\boldsymbol{d}_3$ is constrained by possibly conflicting springs as no other feature has correspondences on both scans.

the features $\boldsymbol{m}_i^k$. The error metric is minimized by moving one global point at a time using gradient descent. The derivative of E with respect to d_i is given by $\nabla \boldsymbol{E}_i$:

$$\nabla \boldsymbol{E}_i = \sum_k \sum_j \frac{2\left|\boldsymbol{d}_i - \boldsymbol{d}_j\right| - \left|\boldsymbol{m}_i^k - \boldsymbol{m}_j^k\right|}{\left|\boldsymbol{d}_i - \boldsymbol{d}_j\right|} \cdot \left(\boldsymbol{d}_i - \boldsymbol{d}_j\right)$$

Though gradient descent is not guaranteed to converge to a global minimum it is efficient and robust.

We propose a novel alternative to the optimization of the global points with the aim of considerably improving the performance of the algorithm. The approach is similar to the previously described optimization of Brown and Rusinkiewicz. Instead of only considering the distance between feature points we will consider the relative orientation of each feature point to each other. In addition, the problem is modeled in probabilistic terms so that more complex "weights" are allowed. Interestingly this different approach enables us to give a globally optimal solution and

compute it much more efficiently.

We view each scan k as a set of measurements of the features $\boldsymbol{m}_i^k$ within that scan. To be more precise, for any feature pair $\boldsymbol{m}_i^k$,$\boldsymbol{m}_j^k$ we assume scan k measured the relative displacement of $\boldsymbol{d}_j$ from $\boldsymbol{d}_i$, i.e., $\boldsymbol{D}_{i,j} = \boldsymbol{d}_i - \boldsymbol{d}_j$. The measurement $\bar{\boldsymbol{D}}_{i,j}^k = \boldsymbol{m}_i^k - \boldsymbol{m}_j^k$ is associated with an uncertainty $\boldsymbol{C}_{i,j}^k$ that replaces the weights $w_{i,j}$ in Eq. (7.1).

Given all the measurements, we wish to maximize the likelihood of the global point positions by minimizing the following Mahalanobis distance.

$$\begin{aligned} E &= \sum_k \sum_{i,j} \left(\boldsymbol{D}_{i,j} - \bar{\boldsymbol{D}}_{i,j}^k\right)^T {\boldsymbol{C}_{\boldsymbol{i,j}}^{\boldsymbol{k}}}^{-1} \left(\boldsymbol{D}_{i,j} - \bar{\boldsymbol{D}}_{i,j}^k\right) \\ &= \sum_k \sum_{i,j} \left(\boldsymbol{d}_i - \boldsymbol{d}_j - \bar{\boldsymbol{D}}_{i,j}^k\right)^T {\boldsymbol{C}_{\boldsymbol{i,j}}^{\boldsymbol{k}}}^{-1} \left(\boldsymbol{d}_i - \boldsymbol{d}_j - \bar{\boldsymbol{D}}_{i,j}^k\right). \end{aligned} \tag{7.2}$$

Under the assumption that ${\boldsymbol{C}_{\boldsymbol{i,j}}^{\boldsymbol{k}}}^{-1} = w_{i,j}\boldsymbol{I}_3$ minimizing Eq. (7.2) is equivalent to minimizing:

$$E = \sum_k \sum_i \sum_{\substack{j \\ j>i}} w_{i,j} \left((\boldsymbol{d}_i - \boldsymbol{d}_j) - \left(\boldsymbol{m}_i^k - \boldsymbol{m}_j^k\right)\right)^2,$$

which is a restated Eq. (7.1). Obviously, the problem is now also very similar to the global optimization problem in our previous work [15]. The major difference is that in this case no linearization is required and the optimal solution can be calculated in a single step. As we have no further information other than a weight $w_{i,j}$ we can assume that $\boldsymbol{C}_{\boldsymbol{i,j}}^{\boldsymbol{k}}$ as well as its inverse is a diagonal matrix as above. This means that the minimum to Eq. (7.2) can be even more easily computed, because the feature dimensions are independent of each other. In order to derive the minimum of the Mahalanobis distance, the global point positions are concatenated into the $(n-1) \times 3$-dimensional matrix $\boldsymbol{D} = (d_2, \ldots, d_n)^T$. The first feature is not included as it defines the reference point by which all other features will be transformed. Eq. (7.2) is now represented in matrix form:

$$E = \left(\bar{\boldsymbol{D}} - \boldsymbol{H}\boldsymbol{D}\right)^T \boldsymbol{C}^{-1} \left(\bar{\boldsymbol{D}} - \boldsymbol{H}\boldsymbol{D}\right),$$

where $\boldsymbol{C}$ is a diagonal matrix containing all $w_{i,j}$ and $\bar{\boldsymbol{D}}$ is the concatenation of all $\bar{\boldsymbol{D}}_{i,j}$. $\boldsymbol{H}$ is an incidence matrix representing the graph created by the edges i, j. The minimum $\boldsymbol{D}$ is given by:

$$\begin{aligned} \boldsymbol{D} &= \left(\boldsymbol{H}^T\boldsymbol{C}^{-1}\boldsymbol{H}\right)^{-1} \boldsymbol{H}^T\boldsymbol{C}^{-1}\bar{\boldsymbol{D}} \\ &= \boldsymbol{G}^{-1}\boldsymbol{B}, \end{aligned}$$

where $\boldsymbol{G}$ and $\boldsymbol{B}$ are given by:

$$\begin{aligned} G_{i,i} &= \sum_k \sum_j w_{i,j}^{-1} \\ G_{i,j} &= \sum_k w_{i,j}^{-1} \qquad \text{for} \quad (i \neq j) \\ B_i &= \sum_k \sum_{i,j} w_{i,j}^{-1} \left(\boldsymbol{m}_i^k - \boldsymbol{m}_j^k\right). \end{aligned}$$

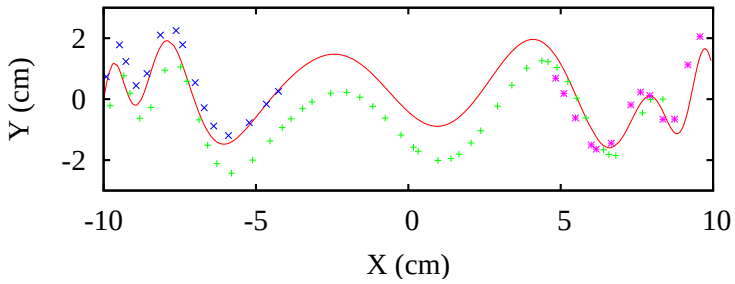


Fig. 7.12: The simulated environment with 3 simulated scans. The solid line represents the environment that is to be estimated. The points above and below constitute the scans. Rigid transformation errors simulating a misalignment in the registration process have been applied.

The alternative with more complex covariances can be derived analogously with the covariances $\boldsymbol{C}_{i,j}^{k}$ in place of the $w_{i,j}$ and $\boldsymbol{D}$ as a $(3n-3)$-dimensional vector.

Estimating the optimal positions of the global points is therefore reduced to computing the inverse of an $(n-1) \times (n-1)$ matrix. As this is in the order of $O(n^3)$ this algorithm is much more efficient than the gradient descent method. As a further bonus, the covariance of each global point d_i is given by the inverse of $\boldsymbol{G}$. This measure of certainty can be used in the TPS functional as laid out in Chapter 4.4.2. Using an approximating TPS instead of the interpolating TPS would then trade accuracy near uncertain global points in favor of rigidity. In theory, iterating the correspondence computation with the global point estimation could then converge to an optimally registered global map.

7.2.3 Mapping to the Global Points

After the global points have been estimated a mapping for each scan is computed. For each scan a thin plate spline deformation may be calculated that maps the local features to the corresponding global points. In case the feature points are distributed well enough the TPS should approximate the inherent deformation in the scan similar to Chapter 4.3. All deformed scans then constitute a consistent global model of the environment.

7.2.4 Experimental Evaluation

This section describes the evaluation of the two global point estimation techniques as presented in the previous chapters. In order to objectively assess the quality of the different error metrics an artificial 2-dimensional test scenario is employed. The complete non-rigid alignment is also qualitatively evaluated on a real data set.

First, a number of global points that lie on a curve as seen in Fig. 7.12 are generated. A set of 2-dimensional range scans that sampled the curve are simulated by choosing a continuous set of global points. The selected stretches of global points vary in length and position. Three types of errors are applied to the scans. To each of the local scan points a positional error, whose maximal extent can be controlled, is applied. Each scan is also transformed with a randomly generated rigid transformation. The maximal extent of the translatory as well as the rotational error can

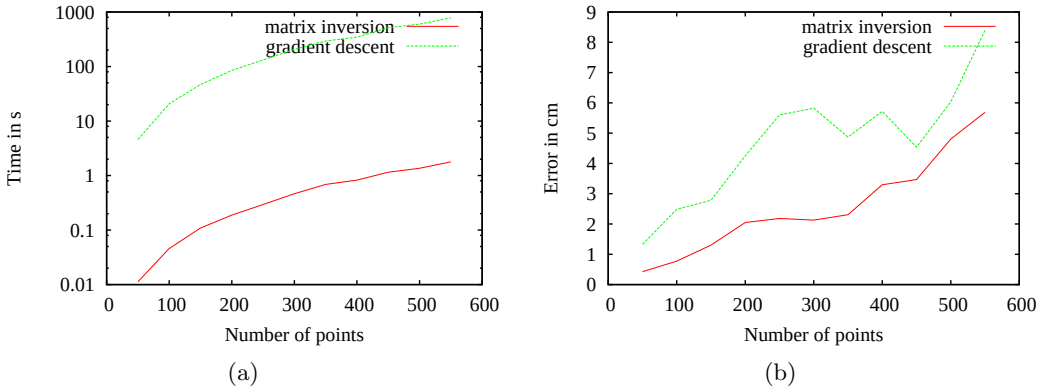


Fig. 7.13: Evaluation of the estimation algorithms on the synthetic data set with a variable number of points. For this plot a randomly generated rigid transformation was applied to the simulated scans. Maximal translatory error is 1 cm and the maximal rotational error is 2 °. Left: The computation time required for both algorithms in a plot of logarithmic scale. Both algorithms need increasingly more time to calculate the minimum. Compared to the matrix inversion the gradient descent requires exceedingly more time to converge to a minimum. Using only 500 points, the gradient descent is about 10 min slower than the matrix inversion. Right: Comparison of the accuracy of both algorithms. With increasing number of points the total error naturally rises linearly. The error of the matrix inversion lies below the error of the gradient descent.

be controlled. Finally, for each scan a non-rigid deformation is simulated by transforming a point $\boldsymbol{p}$ as follows:

$$p'_y = p_y + A \sin L.$$

Here, A is a randomly generated value whose absolute value can be controlled by a parameter. A is the amplitude and therefore represents the extent of the deformation. As the correct positions of the global points as well as the correspondences of the local features to the global points are known, the estimation algorithms can easily be evaluated. After the algorithms have finished minimizing their respective error metric, the error of the estimated global point positions is computed as follows:

$$E = \sum_i \left| \boldsymbol{d}_i - \bar{\boldsymbol{d}}_i \right|,$$

where $\bar{\boldsymbol{d}}_i$ is the estimation of the ith global point $\boldsymbol{d}_i$.

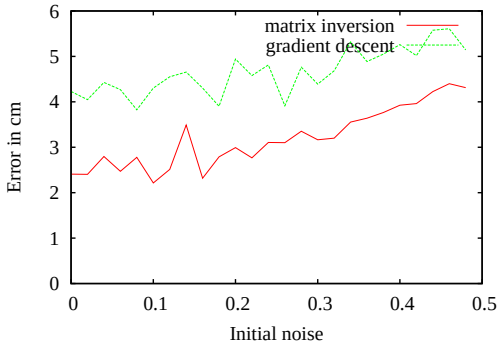
Each of the following experiments was repeated 20 times with new randomly generated initial errors and scans. Both algorithms received the same input. Plotted is the mean of the error of the 20 experiments.

The first point of interest is the efficiency of both algorithms. As seen in Fig. 7.13 the newly proposed estimation algorithm is much faster than the gradient descent. Also shown is the error of the estimated global points in relation to their total number. As expected the error increases with more points. Even though the error of the matrix inversion is lower than the error of the

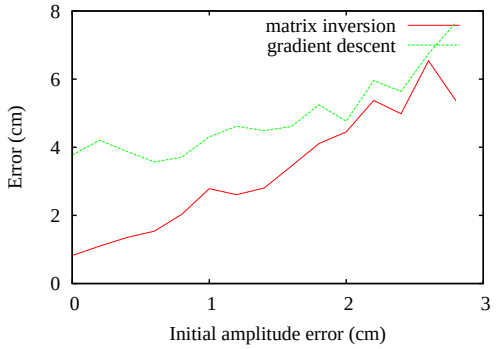
gradient descent, these results may not be meaningful as only one set of parameters for the error generation was used.

To evaluate the accuracy of the two estimation techniques more precisely, experiments for all 3 types of error were conducted. The results are shown in Fig. 7.14. For small and proportional errors the matrix inversion usually fares better. The accuracy of the novel estimation algorithm decreases when more rotational error is introduced in the scans. This can be explained by the fact that the old error metric only accounts for distances between points while the new metric encodes the relative position of the points. Simply rotating a scan would lead to an increase of the new error metric. This does not happen with the error metric of Brown and Rusinkiewicz.

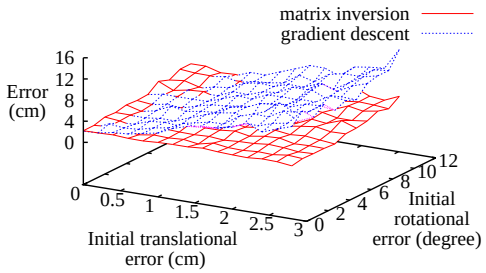
The full non-rigid registration algorithm was also tested on the bunny data set from the Stanford 3D Scanning Repository [73]. For the visualization of the resulting model each of the first 3 laser scans are depicted in a separate color. The non-rigid registration is compared to the rigid alignment in Fig. 7.15. The rigid registration results in a model with slight misalignment. This can be seen in the large monochromatic regions where a single scan is superimposed over the rest of the model. The non-rigid registration produces a more brindled bunny. This indicates a closer matching of the scans. Note that the blue scan does not cover the entire body of the bunny and does therefore not show in the front of the model. Fig. 7.16 shows a different view of the model, as well as a closer look at the bunny's tail. The cross-section of the tail clearly shows that all three scans are closer together using the non-rigid registration procedure.



(a) Noise

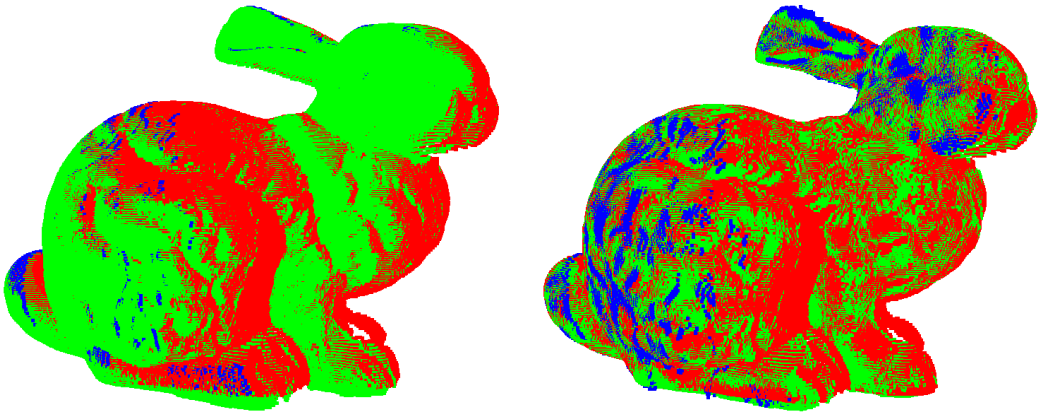


(b) Non-Rigid errors



(c) Rigid errors

Fig. 7.14: Evaluation of the global estimation algorithms with different amount and types of error. All experiments were carried out with 100 global points. Unless an error parameter is explicitly varied in an experiment the maximal errors are constant and set as follows. The positional error for each individual feature point is ≤ 0.25 cm. For each simulated scan the maximal translatory error is 1 cm and the maximal rotational error 2 °. The non-rigid error is simulated with a maximal amplitude error of 1 cm. All errors are total and are not normalized by the number of global points. The mean error was therefore never higher than 1.6 mm per point. Left: Depiction of the error of both algorithm with varying amount of noise that was applied to the local features. Right: In this test the parameters pertaining to the non-rigid deformation were varied. The novel error metric fares worse when the amplitude increases, but is still more accurate when the error is of realistic extent.
Bottom: The parameters controlling the rigid transformation errors were varied. The error of the novel estimation technique increases with the rotational error of the scans while Brown's gradient descent has more trouble with translatory errors. On the whole the matrix inversion did best, however in experiments with small translatory and high rotational errors the gradient descent computes better estimates.



(a) (b)

Fig. 7.15: The fully registered scans for rigid (left) and the newly proposed non-rigid registration (right). Each scan is shown in a different color. The rigid alignment has large areas where one scan obscures the others and scan intersections between those areas. In the non-rigid registration all 3 scans are equally dominant and contribute to the surface of the object. As the mapping is more exact, no scan conceals another over a large area. Both error metrics as described in this thesis result in visually indiscernible alignments.

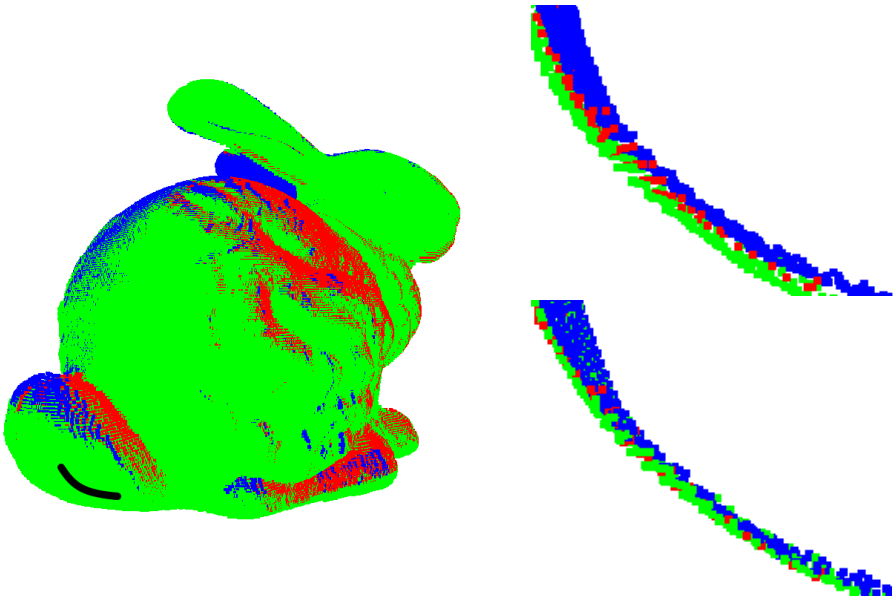


Fig. 7.16: A close-up view of the tail of the Stanford bunny. The cross sections on the right are approximately half a centimeter thick. The top cross section shows the result of a rigid alignment while the bottom one shows non-rigid alignment. Non-rigid alignment achieves a better matching as seen by the closer grouping of the three scans in the bottom. Rigid matching necessarily leads to errors when warp is present in the laser scans as seen in the top.

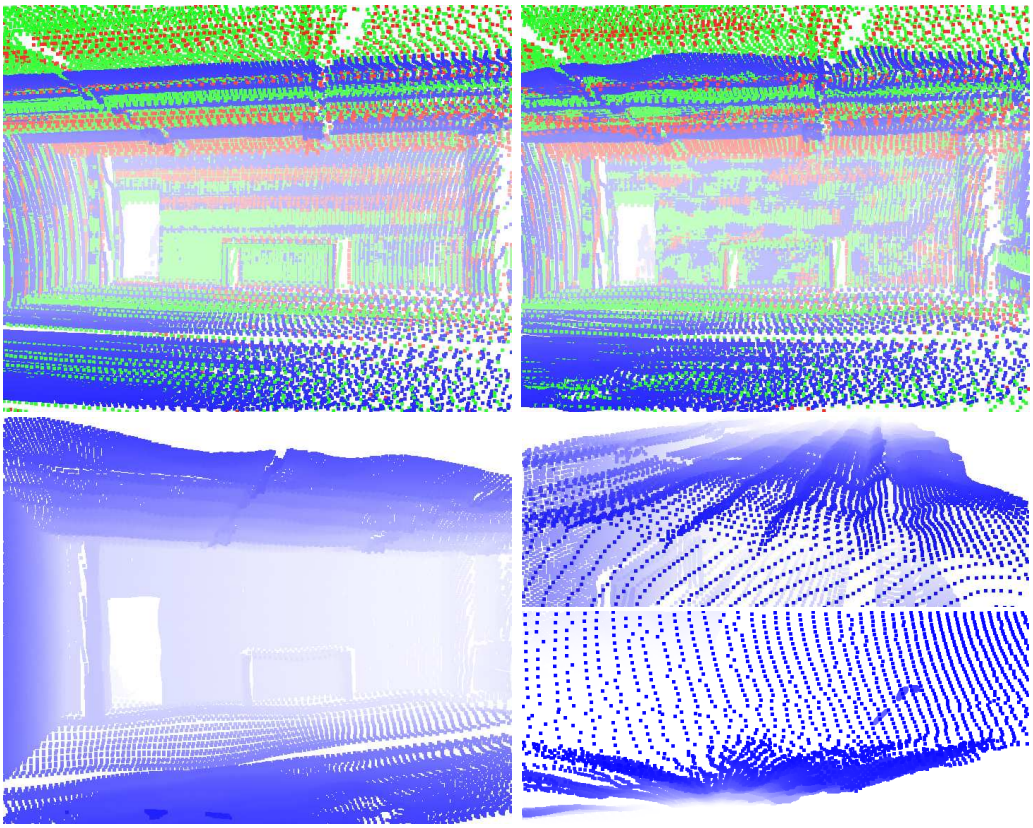


Fig. 7.17: Results of the non-rigid alignment by Brown and Rusinkiewicz. Top: Comparison of the initial rigid alignment and the resulting non-rigid alignment. As with the Stanford Bunny, the non-rigid alignment results in a closer matching. Bottom: A single scan that was deformed. The systematical errors of the 3D scanner are reduced. However, new warp on the same scale is introduced.

The algorithm was also used to improve the same data set that was used in Section 7.1. The algorithm has not been previously tested in an office environment with many flat surfaces. We wish to compare the algorithm to the optimization techniques in the earlier sections. The results are depicted in Fig. 7.17.

Comparing the initial with the optimized scans reveals that the quality of registration, i.e., the point-to-point distance is notably reduced. This is again seen by the equal distribution of the colors, similar to the example in Fig. 7.15. Contrary to the Stanford Bunny or the examples depicted in the original papers [19], [18] an improvement in the scans has not necessarily taken place. When looking at a single scan we see that the systematical errors due to the scanner setup are not eliminated evenly. Floor and ceiling are not planar but are instead still wavy. Unlike the algorithm in Section 7.1.1 no details are obscured. However, the resulting scans clearly do not achieve the quality of those in Section 7.1.2.

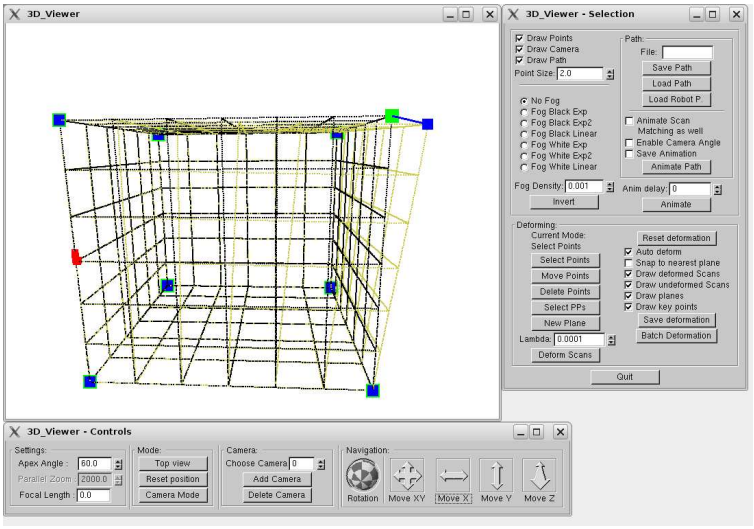


Fig. 7.18: A screenshot of the new GUI for the `show` program. On the left side the display shows the original (black points) and the deformed scan (yellow points). The large rectangles illustrate the TPS deformation (where blue is the desired position for the corresponding green point). Selected points are shown in red. The right control panel has several options that control what is displayed and how the selection process behaves.

7.3 Manual Deformation

In the course of this master's thesis a graphical user interface (GUI) was developed that enables the user to define arbitrary TPS functionals and to deform 3D scan data with said TPS functional. The GUI was integrated into the already existent program `show` and can be seen in Fig. 7.18.

The GUI is motivated by several factors. First, to give an impression of the effects of a TPS deformation on real 3-dimensional laser scan data. Second, to display the planes detected by the Hough Transform. Third, to be able do define a TPS that correctly deforms a large map comprising several laser scans (see Fig. 7.19). Fourth, to batch-deform a set of scans that share a common deformation due to an error in the data acquisition process.

The main advantage of the GUI is the ability to manually define a TPS by selecting landmarks and giving specific destination points for each landmark. This manually formulated deformation can then for example be used to remedy errors that occur repeatedly on the same laser scanner. The GUI also allows the user to deform a large scale map comprised of several dozen or more laser scans. Due to small imperfections in the individual scans the rigid registration process often results in a curved map as seen in Fig. 7.19. This curved map can be straightened out with a few carefully selected landmarks.

Landmark selection can be done in a variety of ways. First of all, the deformations from the previous optimization algorithms can be read from file. Second, new landmarks can be selected from the scans and moved either freely or along planes. These planes can also be either read from file or be defined in the GUI.

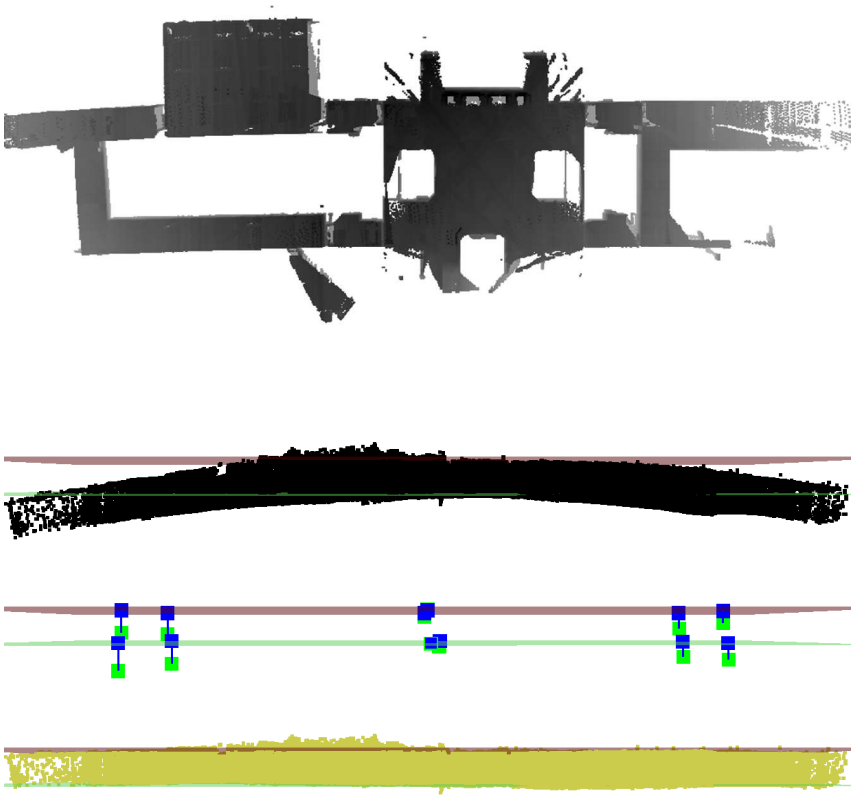


Fig. 7.19: Manual Deformation of a large scale map with over 60 laser scans. The initial map is slightly curved. The GUI allows us to carefully select 12 landmarks that are spread across the map. These are then moved to the desired position. The resulting map is straight as intended.

Chapter 8

Conclusions and Outlook

This thesis addresses the quality of maps assembled from laser range scans. Several algorithms that optimize the alignment quality have been implemented and evaluated. From the beginning we set out to combine the Hough Transform as a plane detection method with the non-rigid TPS deformation in order to improve the quality of maps from structured environments. This specific configuration turned out to be problematic due to several reasons.

The Thin Plate Spline has the nature of a universal function approximator. The TPS will therefore contain a large generalization when it is formed only by points lying on planes. On the one hand this is a nice feature. Only a small proportion of the data cloud is necessary to deform a laser scan so that walls etc. are flattened. On the other hand small scale details near planes fall victim to this generalization property of the TPS. These too get flattened by the deformation. Applying a TPS on a scan acquired in an office environment is problematic in itself. In the algorithm of Brown and Rusinkiewicz 7.2 the global points are correct with a high certainty. Even then the ensuing deformation warps the scans in an awkward way. Although the registration accuracy has improved, the optical quality of the individual scans has not. This is not the case when using laser data acquired from statues and similar structures as in [19], [18]. We believe this is due to two factors. First, the initial warping error that is present in the Kurt3D laser scans is of a larger magnitude and higher frequency. Second, any smooth wave-like error in an office scan is much more apparent than it is in a scan of, e.g., the Stanford Bunny. As we expect straight and plane surfaces in an office environment any aberration from that immediately catches ones eye.

Still, the TPS achieves good results in improving the accuracy of laser scan alignment when used in conjunction with the global point positioning of Brown and Rusinkiewicz. The novel technique for optimizing the global point positions as presented in this thesis surpasses the one of Brown and Rusinkiewicz in both accuracy and speed. In fact, the novel technique is able to find the global optimum of its error metric. The old optimization requires exponentially more iterations when the number n of global points increases. The complexity of the new technique with $O(n^3)$ is therefore much less extreme.

It is interesting to note that no matter how the global points are computed, either by plane extraction or by error metric the TPS is not a necessary part of the improvement algorithm. Any transformation may be used that can be calculated by local to global feature correspondences.

This is demonstrated by the novel slice based deformation in Section 7.1.2 to great effect. In conjunction with dense point-to-plane information it is capable of very accurately correcting the deformation of a scan. The computation of the deformation itself is very fast, so that it may be used online. In fact we found that the plane extraction coupled with the slice based deformation is the most effective and reliable combination for correcting laser scans. The mapping between local features and global points may also be a rigid body transformation. In this case the algorithms are solutions to the SLAM problem, where the poses of the individual scans are sought after.

Under the assumption that the points for which desired positions are computed are not merely a subsample of each point cloud but instead constitute all of it, no further mapping of the scans is necessary. The point clouds are naturally aligned perfectly to a globally consistent model of the environment. In this case the proposed algorithm is consequently a solution to the non-rigid SLAM problem similar to algorithms that estimate not only the pose of each scan but also the position of each point in the point cloud itself [95]. However unlike these algorithms its formulation is much more simple and intuitive. This is especially interesting when we consider the novel global point positioning algorithm as proposed in Chapter 7.2. The algorithm is absolutely independent of any representation of rotation and is therefore always able to find the global optimum, while rotations are frequently a source of problems.

During the course of the master's thesis we also implemented and compared algorithms that aim to classify and optimize plane models. In addition to the two approaches to classify planes in accordance to a semantic net proposed in Nüchter [65] we implemented a novel algorithm that is robust to errors and scalable to models consisting of a large number of planes. Whereas previous algorithms reach their limit at a few dozen planes our clustering approach can handle hundreds of thousands of planes. After the planes have been classified our novel optimization computes the improved plane positions in only a few ms by repeatedly solving a linear equation system.

Detection of prominent structures in laser scans is a task that finds application when solving different problems. For the 3D case planes are probably the most common structures. In this thesis we applied a common algorithm for detecting parametrized 2D objects, the Hough Transform, to the problem of detecting planes. Reliable planes are the basis for both algorithms presented in this thesis to improve the quality of laser scans. Even though a thorough evaluation of the Hough Transform in the 2D case can be found in literature, the application for 3D data is rare. We solve the problem of parametrizing planes and discretizing the Hough Space in a reasonable fashion by introducing a new structure for the accumulator. Furthermore, we evaluate several variations on the Hough Transform as they exist for the $2D$ case. Runtime measurements show that with the increasing amount of data in 3D applications the need for faster methods grows. Our results show that even though the SHT appears to be too time consuming in practice, the RHT yields good results in short time and is therefore a valuable method for the detection of planes in point clouds.

This master's thesis also thoroughly explored the current state of knowledge in the area of Thin Plate Splines. Aside from implementing known techniques to approximate and speed up the computation of the TPS we proposed the use of the LDL^T decomposition for the use with the TPS. We also were the first to evaluate these against each other based on both accuracy and performance in both 2 and 3 dimensions. Even though the LDL^T decomposition is the

fastest inversion method for the TPS in our experimentations, the difference to the standard matrix inversion is minor. This suggests that the TPS matrix is inherently difficult to invert. The approximation technique with a basis function sub-sampling in Chapter 4.3.2 is significantly slower than the simple sub-sampling in all cases. However, the memory requirements of the basis function approximation are on par with the simple sub-sampling. This allows us to maximize the number of landmarks used so that a much higher accuracy in deformation is achieved. The TPS is also capable of incorporating information about landmark uncertainty and orientation information. The first property is especially useful in combination with the new global point positioning to further improve registration accuracy.

There still remains more work to be done. The best alternative for correcting systematical errors in laser scanner, i.e., the slice based deformation can be made even more robust by feeding it more accurate point-to-plane correspondences. Applying a good point filtering technique to remove incorrect points should achieve this goal.

While combining the TPS with plane extraction has been shown to be problematic, manually created deformations as in Section 7.3 show that it is feasible to define a TPS function that correctly deforms a large scale map using very sparse landmarks. Computing these landmarks is a possible area of future work. By using plane extraction it is possible to find corners or other significant features of the environment. Early experiments have shown that the challenge lies not only in reliably finding features but also in choosing the relevant features.

We believe that our proposed global point positioning merits future investigation into the application of the algorithm to non-rigid SLAM problems. In its current state the new error metric does not allow for the rotation of a scan to change without incurring a penalty. Therefore the rigid alignment of the scans has to be accurate before the non-rigid alignment is started. The error metric could easily be changed to incorporate a separate rotation for every scan. This would mean however that it could not be as easily globally minimized anymore. Incorporating rotations would likely necessitate a linearization and therefore iterating the matrix inversion.

Bibliography

[1] B. Allen, B. Curless, and Z. Popovic. Articulated Body Deformation from Range Scan Data. In *Proceedings of ACM SIGGRAPH 2002*, 2002.

[2] B. Allen, B. Curless, and Z. Popovic. The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans. *ACM Transactions on Graphics*, 22:587–594, 2003.

[3] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. AVENUE: Automated Site Modelling in Urban Environments. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM '01)*, Quebec City, Canada, May 2001.

[4] N. Archip, S. Tatli, P. R. Morrison, F. A. Jolesz, S. K. Warfield, and S. G. Silverman. Non-Rigid Registration of Pre-Procedural MR Images with Intra-Procedural Unenhanced CT Images for Improved Targeting of Tumors During Liver Radiofrequency Ablations. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2007.

[5] K. S. Arun, T. S. Huang, and S. D. Blostein. Least Square Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698 – 700, 1987.

[6] M. Atiquzzaman. Multiresolution Hough Transform – An Efficient Method of Detecting Patterns in Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11), 1992.

[7] M. Atiquzzaman. Pipelined Implementation of the Multiresolution Hough Transform in a Pyramid Multiprocessor. *Pattern Recognition Letters*, 15(9):841–851, 1994.

[8] U. Bauer and K. Polthier. Detection of Planar Regions in Volume Data for Topology Optimization. *Lecture Notes in Computer Science*, 2008.

[9] A. M. Bazen and S. H. Gerez. Thin–Plate Spline Modelling of Elastic Deformations in Fingerprints. In *Proceedings 3rd IEEE Benelux Signal Processing Symposium*, pages 1859–1867. Elsevier Science, Oxford, 2002.

[10] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[11] P. Besl and N. McKay. A Method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.

[12] M. Block and R. Rojas. Entzerrung von Textdokumenten unter Verwendung von Thin-Plate-Spline. Technical Report B-08-13, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 2008.

[13] F.L. Bookstein. Principal Warps: Thin Plate Splines and the Decomposition of Deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567 – 585, June 1989.

[14] F.L. Bookstein and D.K. Green. A Feature Space for Edgels in Images with Landmarks. *Journal of Mathematical Imaging and Vision*, 1993.

[15] D. Borrmann and J. Elseberg. Global konsistente 3D Kartierung am Beispiel des Botanischen Gartens in Osnabrück. Bachelor's thesis, Universität Osnabrück, 2006.

[16] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally Consistent 3D Mapping with Scan Matching. *Journal of Robotics and Autonomous Systems*, 56(2):130–142, 2008.

[17] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. The Efficient Extension of Globally Consistent Scan Matching to 6 DOF. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '08)*, Atlanta, GA, USA, June 2008.

[18] B. J. Brown and S. Rusinkiewicz. Non-Rigid Global Alignment using Thin-Plate Splines. *ACM Transactions on Graphics*, 26(31):16, 2005.

[19] B. J. Brown and S. Rusinkiewicz. Global Non-Rigid Alignment of 3-D Scans. *ACM Transactions on Graphics*, 26(3):21, 2007.

[20] A. Censi and S. Carpin. HSM3D: Feature-Less Global 6DOF Scan-Matching in the Hough/Radon Domain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009. (accepted).

[21] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image and Vision Computing*, 10(3):145–155, 1992.

[22] D. Chetverikov, D. Svirko, D. Stepanov, and Pavel Krsek. The Trimmed Iterative Closest Point Algorithm. *16th International Conference on Pattern Recognition (ICPR'02)*, 03:545–548, 2002.

[23] H. Chui and A. Rangarajan. A New Point Matching Algorithm for Non-Rigid Registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.

[24] L. D. Cohen and I. Cohen. Deformable Models for 3D Medical Images using Finite Elements and Balloons. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992.

[25] D. M. Cole and P. M. Newman. Using Laser Range Data for 3D SLAM in Outdoor Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06)*, Florida, USA, 2006.

[26] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[27] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229 – 241, June 2001.

[28] J. Duchon. Interpolation des Fonctions de deux Variables suivant le Principe de la Flexion des Plaques Minces. *RAIRO Analyse Numerique*, 10:5 – 12, 1976.

[29] R. O. Duda and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. Technical Note 36, Artificial Intelligence Center, SRI International, 1971.

[30] R. A. Finkel and J. L. Bentley. Quad Trees - a Data Structure for retrieval on Composite Keys. *Acta Informatica*, 4(1):1–9, 1974.

[31] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381 – 395, 1981.

[32] J. Folkesson and H. Christensen. Outdoor Exploration and SLAM using a Compressed Filter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, pages 419–426, Taipei, Taiwan, September 2003.

[33] U. Frese. Efficient 6-DOF SLAM with Treemap as a Generic Backend. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, Rome, Italy, April 2007.

[34] U. Frese and G. Hirzinger. Simultaneous Localization and Mapping – A Discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17 – 26, Seattle, USA, August 2001.

[35] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[36] C. Früh and A. Zakhor. 3D Model Generation for Cities Using Aerial Photographs and Ground Level Laser Scans. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR '01)*, Kauai, Hawaii, USA, December 2001.

[37] G. Donato and S. Belongie. Approximate Thin Plate Spline Mappings. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 21–31, London, UK, 2002. Springer-Verlag.

[38] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically Stable Sampling for the ICP Algorithm. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling*, Canada, October 2003.

[39] A. Georgiev and P. K. Allen. Localization Methods for a Mobile Robot in Urban Environments. *IEEE Transaction on Robotics and Automation*, 20(5):851 – 864, October 2004.

[40] D. Hähnel, S. Thrun, and W. Burgard. An Extension of the ICP Algorithm for Modeling Nonrigid Objects with Mobile Robots. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[41] M. Hebert, M. Deans, D. Huber, B. Nabbe, and N. Vandapel. Progress in 3–D Mapping and Localization. In *Proceedings of the 9th International Symposium on Intelligent Robotic Systems, (SIRS '01)*, Toulouse, France, July 2001.

[42] M. Hofer and H. Pottmann. Orientierung von Laserscanner-Punktwolken. *Vermessung & Geoinformation*, 91:297 – 306, 2003.

[43] B. K. P. Horn. Closed–form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America A*, 4(4):629 – 642, April 1987.

[44] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed–form Solution of Absolute Orientation using Orthonormal Matrices. *Journal of the Optical Society of America A*, 5(7):1127 – 1135, July 1988.

[45] Paul V. C. Hough. Method and Means for Recognizing Complex Patterns. US Patent 3069654, December 1962.

[46] M. F. Hutchinson. Interpolation of Rainfall Data with Thin Plate Smoothing Splines. *Journal of Geographic Information and Decision Analysis*, 2(2):139 – 167, 1998.

[47] L. Ikemoto, N. Gelfand, and M. Levoy. A Hierarchical Method for Aligning Warped Meshes. In *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2003.

[48] J. Illingworth and J. Kittler. A Survey on the Hough Transform. *Computer Vision, Graphics, and Image Processing*, 44:87–116, 1988.

[49] H. Kälviäinen and P. Hirvonen. Connective Randomized Hough Transform (CRHT). In *9th Scandinavian Conference on Image Analysis*, pages 1029–1036, June 1995.

[50] H. Kälviäinen, P. Hirvonen, L. Xu, and E. Oja. Probabilistic and Non-Probabilistic Hough Transforms: Overview and Comparisons. *Image and Vision Computing*, 13(4), May 1995.

[51] R. Katz, N. Melkumyan, J. Guivant, T. Bailey, J. Nieto, and E. Nebot. Integrated Sensing Framework for 3D Mapping in Outdoor Navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, Beijing, China, October 2006.

[52] N. Kiryati, Y. Eldar, and A.M. Bruckstein. A Probabilistic Hough Transform. *Pattern Recognition*, 24(4):303–316, 1991.

[53] S. Knust. *Graphenalgorithmen*. Selbstverlag der Universität Osnabrück, 2007.

[54] P. Kohlhepp, M. Walther, and P. Steinhaus. Schritthaltende 3D-Kartierung und Lokalisierung für mobile Inspektionsroboter. In *Proceedings of the Autonome Mobile Systeme 2003, 18. Fachgespraeche*, December 2003.

[55] R. Lakaemper and L. J. Latecki. Extended EM for Planar Approximation of 3D Data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[56] H. Li, M. A. Lavin, and R. J. L. Master. Fast Hough Transform: A Hierarchical Approach. *Computer Vision, Graphics, ans Image Processing*, 36:139–161, 1986.

[57] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333 – 349, April 1997.

[58] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality Image Registration by Maximization of Mutual Information. *IEEE Transactions on Medical Imaging*, 1997.

[59] M. Magnusson and T. Duckett. A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles. In *Proceedings of the Second European Conference on Mobile Robotics (ECMR '05)*, pages 86 – 91, Ancona, Italy, September 2005.

[60] K. Mardia, J.T. Kent, C.R. Goodall, and J. Little. Kriging and Splines with Derivative Information. *Biometrika*, 1996.

[61] J. Matas, C. Galambos, and J. Kittler. Progressive Probabilistic Hough Transform. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 256–265, 1998.

[62] N. J. Mitra, S. Flöry, M. Ovsjanikov, N. Gelfand, L. Guibas, and H. Pottmann. Dynamic Geometry Registration. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, Barcelona, Spain, 2007.

[63] N. J. Mitra, A. Nguyen, and L. Guibas. Estimating Surface Normals in Noisy Point Cloud Data. *International Journal of Computatinal Geometry and Applications*, 14:261–276, 2004.

[64] J. Montagnat and H. Delingette. Globally Constrained Deformable Models for 3D Object Reconstruction. *Signal Processing*, 1998.

[65] A. Nüchter. *Semantische dreidimensionale Karten für autonome mobile Roboter*. PhD thesis, University of Bonn, 2006.

[66] A. Nüchter. *3D Robotic Mapping. The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom.*, volume 52 of *Springer Tracts in Advanced Robotics*. Springer Verlag, 2009.

[67] A. Nüchter, J. Elseberg, P. Schneider, and D. Paulus. Linear Solutions to the Scan Registration Problem. *Computer Vision and Image Understanding (CVIU)*, 2009 (submitted).

[68] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Heuristics-Based Laser Scan Matching for Outdoor 6D Slam. In *KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings Springer LNAI vol. 3698, pages 304 - 319*, Koblenz, Germany, September 2005.

[69] G. Olafsson and E. T. Quinto (Editor). *The Radon Transform, Inverse Problems, and Tomography*. American Mathematical Society, 2005.

[70] J. O'Rourke. Dynamically Quantized Spaces for Focusing the Hough Transform. In *IJCAI*, pages 737 – 739, 1981.

[71] H. Pottmann, S. Leopoldseder, and M. Hofer. Simultaneous Registration of Multiple Views of a 3D Object. *ISPRS Archives*, 34(3A):265 – 270, 2002.

[72] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[73] The Stanford 3D Scanning Repository. `http://graphics.stanford.edu/data/3Dscanrep/`.

[74] K. Rohr, M. Fornefett, and H. S. Stiehl. Approximating Thin-Plate Splines for Elastic Registration: Integration of Landmark Errors and Orientation Attributes. In *Proceedings of the 16th International Conference on Information Processing in Medical Imaging*, 1999.

[75] P.J. Rousseeuw and B.C. van Zomeren. Unmasking Multivariate Outliers and Leverage Points. *Journal of the American Statistical Association*, 85:633–651, 1990.

[76] S. Rusinkiewicz. 3D Scan Matching and Registration, 2005. ICCV 2005 Short Course.

[77] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modelling (3DIM '01)*, Quebec City, Canada, May 2001.

[78] D. Salomon. *Curves and Surfaces for Computer Graphics*. Springer Science+Business Media Inc., 2006.

[79] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 2007.

[80] V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg. Automated 3D Reconstruction of Interiors with Multiple Scan–Views. In *Proceedings of SPIE, Electronic Imaging '99, The Society for Imaging Science and Technology/SPIE's 11th Annual Symposium*, San Jose, CA, USA, January 1999.

[81] SICK Laser Range Finder. `http://www.sick.de`.

[82] K. R. Sloan. Dynamcially Quantized Pyramids. In *IJCAI*, pages 734 – 736, 1981.

[83] A. J. Smola and B. Scholkopf. Sparse Greedy Matrix Approximation for Machine Learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000.

[84] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. 6D SLAM - Preliminary Report on Closing the Loop in Six Dimensions.

[85] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. A 3D Laser Range Finder for Autonomous Mobile Robots. In *Proceedings of the 32nd International Symposium on Robotics (ISR '01)*, Seoul, Korea, April 2001.

[86] S. Thrun. Robotic Mapping: A Survey. *CMU-CS-02-111*, February 2002. School of Computer Science - Carnegie Mellon University. Pittsburgh, PA 15213.

[87] S. Thrun, W. Burgard, and D. Fox. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning and Autonomous Robots*, 31(5):1 – 25, October 1997.

[88] S. Thrun, D. Fox, and W. Burgard. A Real-time Algorithm for Mobile Robot Mapping with Application to multi robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 00)*, San Francisco, CA, USA, April 2000.

[89] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. F. Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *Machine Learning and Autonomous Robots*, 23(7 – 8):693 – 716, July/August 2004.

[90] S. Thrun, M. Montemerlo, and A. Aron. Probabilistic Terrain Analysis For High-speed Desert Driving. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.

[91] G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

[92] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-D Location Parameters using Dual Number Quaternions. *CVGIP: Image Understanding*, 54:358 – 367, November 1991.

[93] D. S. Watkins. *Fundamentals of Matrix Computations*. Wiley-Interscience, 2. edition, 2002.

[94] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for Structured Environment Reconstruction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pages 2089 – 2094, Edmonton, Alberta Canada, August 2005.

[95] J. A. Williams, M. Bennamoun, and S. Latham. Multiple View 3D Registration: A Review and a New Technique. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1999.

[96] J. Wills, S. Agarwal, and S. Belongie. A Feature-based Approach for Dense Segmentation and Estimation of Large Disparity Motion. *International Journal of Computer Vision*, 68(2):125–143, 2006.

[97] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 4204 – 4209, New Orleans, USA, April 2004.

[98] L. Xiaoguang and A.K. Jain. Deformation Modeling for Robust 3D Face Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1346 – 1357, 2008.

[99] L. Xu, E. Oja, and P. Kultanen. A new Curve Detection Method: Randomized Hough Transform (RHT). *Pattern Recognition Letters*, 11:331–338, 1990.

[100] A. Ylä-Jääski and N. Kiryati. Adaptive Termination of Voting in the Probabilistic Circular Hough Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9), 1994.

[101] G. Yu, M. Grossberg, G. Wolberg, and I. Stamos. Think Globally, Cluster Locally:A Unified Framework for Range Segmentation. In *Proceedings of the 4th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '08)*, Atlanta, USA, 2008.

[102] T. Zaharia and F. Preteux. Shape-based Retrieval of 3D Mesh Models. In *Proceedings of the IEEE International Conferenc on Multimedia and Expo*, 2002.

[103] H. Zhao and R. Shibasaki. Reconstructing Textured CAD Model of Urban Environment Using Vehicle-Borne Laser Range Scanners and Line Cameras. In *Second International Workshop on Computer Vision System (ICVS '01)*, pages 284 – 295, Vancouver, Canada, July 2001.

The chapters were written by:

Introduction	Dorit Borrmann
State of the Art	Jan Elseberg
Mapping	Dorit Borrmann
The Thin Plate Spline	Jan Elseberg
Plane Extraction	Dorit Borrmann
Improvement of the Plane Model	Jan Elseberg
Deforming Scans	Jan Elseberg
Conclusions and Outlook	Jan Elseberg

Proclamation

Hereby we confirm that we wrote this thesis independently and that we have not made use of any other resources or means than those indicated.

Osnabrück, August 2009