FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Global Registration of Range Scans with Match Tolerance Verification

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Florian Laager, BSc.
Matrikelnummer 0525859

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Mag. rer. soc. oec. Stefan Ohrhallinger, PhD

Wien, TT.MM.JJJJ

(Unterschrift Verfasser)

(Unterschrift Betreuer)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Global Registration of Range Scans with Match Tolerance Verification

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Florian Laager, BSc.
Registration Number 0525859

to the Faculty of Informatics
at the Technische Universität Wien

Advisor: Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Mag. rer. soc. oec. Stefan Ohrhallinger, PhD

Vienna, TT.MM.JJJJ

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Florian Laager, BSc.
Klara-Blum-Gasse 4/11/2, 1220 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Danksagung

Weder der Abschluss einer Diplomarbeit, und noch weniger der eines ganzen Studiums, ist ohne die Hilfe vieler Menschen möglich. Hier möchte ich gerne ein paar Zeilen dazu nutzen, diesen Menschen meinen Dank und meine Anerkennung auszudrücken, wobei das volle Ausmaß hiermit nicht annähernd darstellbar ist.

Von ganzem Herzen will ich meiner Familie für ihre überschäumende Liebe und Unterstützung danken. Meinem Bruder, Stefan, dafür dass er immer da ist und mich in allem nur Vorstellbaren unterstützt. Meinen Eltern, Christa und Kurt, für ihre ständige Geduld und Gutmütigkeit und dass sie mich auf die Universität geschickt haben und gesagt haben dass ich das packe wenn ich anderer Meinung war. Ein großer Dank geht auch an meinen kürzlich verstorbenden Großvater, Alois, für all die interessanten Konversationen und das Formen meiner Persönlichkeit, und dafür mich immer zu fragen: “Na, bist’ schon fertig?”. Ich bin’s.

Es gibt keine Worte um meine Anerkennung und Dankbarkeit meiner Freundin, und bald Frau, Sabrina, auszudrücken. Danke für deine unendliche Geduld und Hartnäckigkeit wenn meine Prokrastination wieder ein neues Level erreicht hat. Danke, dass du mir immer Raum gegeben hast mich auf die Diplomarbeit zu konzentrieren. Und natürlich auch für’s Korrekturlesen. Danke, danke, danke.

Danken möchte ich auch gerne meinen Betreuern, Stefan Ohrhallinger und Michael Wimmer, für die riesen Hilfe. Angefangen bei der Hilfe dabei meine ersten Ideen zu formulieren bis hin zu deren unglaublichen Korrekturen am Ende. Danke auch an Stefan Ohrhallinger für die vielen E-Mails und Skypekonversationen dazwischen. Ich hätte es ohne eure Hilfe nicht geschafft.

Es gibt noch so viele andere Menschen deren Namen sich einen Platz auf diesen Seiten verdient haben. Freunde, alt und neu, für ihre Begleitung auf dem Weg hierher. Professoren dafür dass sie ihr unglaubliches Wissen geteilt haben. Danke auch an Arbeitskollegen für ihre Inspiration zu strukturierterem Denken und Arbeiten.

Vielen Dank euch allen.

Acknowledgements

The completion of a diploma thesis, much less of a whole study, cannot be achieved without the help of many people. I would like to spend a few lines to show my appreciation and thank these people and give them credit, unfortunately much less than they would actually deserve.

From the bottom of my heart I want to thank my family for their overflowing love and support. My brother, Stefan, for always being there and supporting me in every way imaginable. My parents, Christa and Kurt, for their patience and benevolence on so many occasions, and for making me go to university and telling me I could do it when I thought I'd not be cut out for it. A big "thank you" also to my late grandfather, Alois, for all the interesting conversations and shaping my personality, and never ceasing to ask "So, are you done yet?". I am now.

There are no words to express my appreciation for my girlfriend, soon to be wife, Sabrina. Thank you for your undying patience and tenacity when I reached new levels of procrastination. For taking so many things off my plate so I could focus on the thesis. And last but not least for proof reading this thesis. Thank you, thank you, thank you.

I would also like to thank my supervisors, Stefan Ohrhallinger and Michael Wimmer, for their immense help. Starting from helping me formulate the first ideas until their amazing corrections at the end. To Stefan also for all the e-mails and Skype chats in between. I wouldn't have made it without your help.

There are so many more people whose names deserve to be on these pages. Friends, old and new, for accompanying me along the way. Professors for sharing their incredible amounts of knowledge. Colleages from work for inspiring me to more structured thinking and working.

Thank you all so much.

Kurzfassung

Bei der Registrierung von Oberflächen handelt es sich um den Prozess, Punktkorrespondenzen zwischen multiplen Oberflächen zu finden und eine ausrichtende Transformation zu schätzen. Diese Transformation soll dann die beiden Oberflächen so aufeinander ausrichten, dass sie sich so gut wie möglich überlappen. Es existieren einige Ansätze, die hervorstechende Punkte identifizieren und einen lokalen Deskriptor bauen, der die lokale Nachbarschaft der identifizierten Punkte beschreibt. Korrespondenzen werden gefunden, indem die Deskriptoren der jeweiligen Punkte miteinander verglichen werden. Die globale Registierung, mit der sich die vorliegende Arbeit beschäftigt, zielt allerdings darauf ab Oberflächen auszurichten und dabei Wissen über den gesamten Überlappungsbereich einzusetzen.

Durch das Erscheinen von Endverbraucher-Tiefenscannern wie der Microsoft Kinect™ wurde die Forschung über Registrierung von Tiefenscans weiter vorangetrieben. Ein Objekt kann von mehreren Perspektiven gescannt werden und die Teilscans können dann aufeinander registriert werden und ein vollständiges virtuelles Objekt rekonstruiert werden. Ein anderer Anwendungsfall wäre im Bereich der Augmented Reality oder in der Robotik, wo Tiefenscans der Lokalisierung der Kamera in einer Referenzszene dienen.

Über Oberflächenregistrierung, lokal als auch global, existiert viel Material in der Literatur. Im Rahmen dieser Arbeit wird ein Algorithmus zur Registrierung von Tiefenscans vorgeschlagen der, anders als die von uns in der Literatur gefundenen Ansätze, eine ausrichtende Transformation nur dann extrahiert, wenn diese innerhalb eines Toleranzbereichs existiert. Ähnlich zu vielen Ansätzen identifizieren wir dazu zuerst hervorstechende Feature-Punkte auf der Oberfläche und erstellen einen globalen Deskriptor, der das räumliche Verhältnis der Punkte zueinander abbildet. Wir bauen ein Toleranzmodell für die Akzeptanz von ausrichtenden Transformationen, das auf dem Fehlermodell des verwendeten Scanners aufbaut.

Wir haben unseren Algorithmus an Tiefenscans unterschiedlicher Objekte angewendet und seine Fähigkeit zur Registrierung evaluiert. Weiters haben wir Tiefenscans, die mit der Microsoft Kinect™ v2 aufgenommen wurden, mittels unseres Algorithmus registriert und dessen Anwendbarkeit untersucht. Des Weiteren haben wir unseren Ansatz mit einem state-of-the-art Algorithmus verglichen um zu sehen, ob er entscheiden kann ob eine ausrichtende Transformation existiert oder nicht.

Abstract

Surface registration is the process of identifying correspondences between points on multiple surfaces and estimating an aligning transformation of one surface to the other so that corresponding surface patches overlap. Several approaches exist that identify salient points on the surface and extract a local descriptor of the neighborhood of those points. Correspondences are then found by comparing the descriptors. Global registration, however, aims at aligning surfaces using knowledge over the whole area of overlap.

Registration of range scans has seen a lot of research since the advent of commodity range scanners like the Microsoft Kinect™. An object can be captured from multiple perspectives and the scans can be registered to reconstruct the whole virtual object. Another usage can be found in the fields of augmented reality or robotics, where range scans are used to locate the camera within a reference scene.

A lot of work has been conducted on surface registration, both local and global. We propose an algorithm for registration of range scans that, other than what we found in the literature, provides an aligning transformation only if such a transformation exists within a certain tolerance. Similar to many approaches, we first identify salient feature points on the surface and build a global descriptor that incorporates the spatial relation between feature points. Using the error model of the capture device, we build a tolerance model for acceptable aligning transformations.

We applied our algorithm to range scans of objects of different types and evaluated its registration capabilities. Additionally, we registered range scans made with the Microsoft Kinect™ v2 using our algorithm and evaluated its applicability. Furthermore, we compared our approach to a state-of-the-art algorithm in global registration to see whether it is able to decide on whether an aligning transformation exists.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Figure 1.1: Several range scans, here shown as triangular meshes, being registered onto each other. First aligned roughly in a crude registration step, followed by fine registration. Image courtesy of Gelfand et al. [15].

How does a computing device know if two different images show one and the same scene if the only information it has are pixel values? If it does, is it able to stitch together the captured images in such a way that they form a whole? This would be helpful if one wanted to capture a panoramic image of a beautiful landscape scenery but the camera used can only cover so much area in one capture. This functionality is nowadays usually already built into an average digital camera or smartphone. What one would need to do is to take several images of the landscape such that in each image, there is a part of the scene that is also visible in another image. The camera will then *register* the images by finding *salient features* in them. It then aligns the images such that *matching* features in the images are positioned at the same pixel locations. This results in a panorama image.

The same is possible for images taken with a *range scanner*, a capture device that captures depth information at pixel locations. Now the depth information inherent to a range scan can be leveraged and geometrical properties of the captured scene can be used to extract *salient feature points*. Registration of range scans is used, for example, by autonomous robots equipped with

range sensors, so that they know where they are in a reference coordinate system. Consecutive camera frames are analyzed and related to each other so that the robot knows how it has moved in the time elapsed since the last frame. (See Durrant-Whyte et al. [12])

Another application of registration of range scans lies in the field of heritage preservation. An ancient artefact is seldom retrieved as a whole but in parts. To get an image of the whole object, range scans of each part can be used to produce a virtual 3D model of the whole object by aligning the scans and *"stitching"* them together, just like the example of the panorama image described earlier, but for 3D data.

In shape retrieval, also *similar* shapes can be retrieved by providing a reference shape as input to a database of shape data. The output would then generally be a list of matching shapes, 3D models for example. Range images can also be used as an input to find matching 3D models. The approach can be used in CAD and other domains that make heavy use of 3D models [38].

A lot of work has been conducted on automatic registration of surfaces. Algorithms generally consist of two steps: *crude registration*, where the surfaces are roughly aligned, and *fine registration*, which aims at optimising the aligning transformation in such a way that every point of one surface is as close as possible to its corresponding point on the other surface. These steps are depicted in Figure 1.1. Fine registration has been solved by the *Iterative Closest Point (ICP)* algorithm, which is an iterative process that first finds correspondences between points on both surfaces and then estimates a transformation using the corresponding points that minimizes an error metric over the whole surfaces. Many variants of the algorithm have been presented that differ in the way in which point distance is modeled, e.g., point-point distance or point-plane distance, or how the transformation is derived from point correspondences [33]. Although ICP can solve the problem of fine registration, the solution depends on whether or not the surfaces are already roughly aligned. It needs a first rough guess of the aligning transformation so that it does not converge to a locally minimal solution due to its iterative nature. *Crude registration* can be used to provide such a rough estimate. A lot of research has been conducted on crude surface registration of rigid and non-rigid surfaces, and methods exist that work on 3D meshes, unstructured point clouds and range scans [34].

1.2 Problem Statement

The algorithm presented in this thesis can be positioned in the domain of crude registration of range scans of rigid 3D surfaces.

Crude Registration, in literature also called *coarse registration*, is the process of roughly aligning two or more surfaces. In the discrete case, the given surfaces can be modelled as point clouds or meshes. The general problem of registering surfaces consists of finding an aligning transformation and then verifying the quality of the alignment. The former can be modelled in multiple ways, e.g., in case of the brute-force approach, trying out all possible transformations with six degrees of freedom (6DOF). The prevalent approach is to model registration as a minimization problem and find correspondences between subsets of the given surfaces. These subsets are used to estimate a transformation that aligns them. The transformation is then verified using the whole surfaces by measuring the amount of residual error between the points of both surfaces after applying the transformation. This formulation, however, does not provide

evidence of whether or not the registered surfaces actually match. The subsets used for correspondence search can be chosen at random or can be selected using *feature detection* based on certain characteristics of the surface at their location. The correspondence search can also be used such that corresponding points on the other surface exhibit the same characteristic, or *feature descriptor*. This is also called *local registration*, since the correspondence search is only based on *local* information at a given point on a surface.

Another approach, *global registration*, consists of identifying correspondences between points of two surfaces by evaluating their spatial relation. This means finding a subset in one surface that exhibits the same spatial layout as a subset on the other surface. The algorithm presented in this thesis follows this approach.

A formal statement of the problem of rigid surface registration is the following. Let S be a surface captured with a range sensor from two different positions and let P_R and P_T be the sets of points sampled from S. Corresponding points $p \in P_R$ and $q \in P_T$ will not fall within the same infinitesimal space but instead into a volume of a certain size, $\|p - q\| \leq \xi$, because of *measurement error* and *sample displacement*. *Measurement error* here is based on noise of the sensor itself. *Sample displacement* is based on the fact that the sampling pattern "slides" over the surface as the scanner is moved or increasing sampling distance based on perspective. The aim of crude registration is to find a rigid transformation $\mathcal{T}$ that aligns the target point cloud P_T to a reference point cloud P_R such that a given error metric E is minimized, taking into account all mentioned sources of spatial displacements between corresponding points. The solution $\hat{\mathcal{T}}$ is then

$$\hat{\mathcal{T}} = argmin_{\mathcal{T}}(\sum_{i,j} \|p_i - \mathcal{T} * q_j\|_E)$$

As can be seen, usually several transformations have to be evaluated if correspondences are not known in advance. The problem with extracting several transformations and calculating an error measure for each one, then returning the transformation with the smallest error, is that it is not guaranteed that using the best transformation will lead to successful fine registration, or that the surfaces match at all. This happens for example if the surfaces exhibit symmetries at their overlap. Since this technique will always report a minimal error, it would also do so even in case of geometric differences still existing between surfaces, because it always aims at a "best" alignment.

1.3 Aim of this work

To be able to know whether or not two surfaces captured with a range sensor actually match or not can save computation time on fine registration that would not make sense if no match was given at all. The aim of this thesis is to propose a method to incorporate the error model of a range sensor into the crude registration process of range scans and answer the above question within an uncertainty threshold that is based on the error model of the range sensor and the geometrical properties of the scanned surface. At the same time, if possible, our method will provide an estimate of the aligning transformation. As an optional additional step, a fine registration using ICP can be applied afterwards to further align the surfaces, if necessary, and to guarantee convergence to the global minimum.

1.4 Methodological Approach

We follow the prevalent approach to crude registration, by first preprocessing the range scans to extract necessary point correspondences, which are then used to construct transformation candidates. To reduce the search space of correspondences, we reduce the number of points by identifying salient features within the captured range scans. One surface characteristic that is invariant to rigid transformation is *local surface curvature*, so we choose local curvature extrema as the salient feature characteristic. Since range scans are subject to noise, which has high impact on local curvature, we estimate the local curvature at each point by fitting a quadratic surface to its local neighborhood. The size of the used neighborhood depends on the spatial error of the evaluated point and is based on two factors. First, the sensor noise model of the range sensor which was used for capturing the scans, and secondly, on discretization artefacts introduced by sampling the captured surface in pixels.

Other than most works on surface registration, we do not encode the local neighborhood of the salient features in a feature descriptor per point, but rather incorporate the spatial relation between triples of points into a *global descriptor* of the whole range scan. The global descriptor also takes the pointwise error model into account.

Transformation candidates are calculated using corresponding point triples of both range scans and are verified by matching the whole scans while, again, considering spatial point errors in the process. This either leads to a verified match within a tolerance based on the sensor's noise model and discretization artefacts introduced during sampling of the scans or a negative answer if the two scans do not overlap or do not match at all.

1.5 Structure of this work

The rest of this thesis is structured as follows. In Chapter 2 the basic concepts and related literature on surface registration are reviewed, with a special emphasis on different *feature detectors* and *feature descriptors*. In Chapter 3 our proposed implementation for global registration of range scans is presented, accompanied by a practical example using synthetic data. A critical evaluation of the important concepts of the implementation is described in Chapter 4, with a comparison to a state-of-the-art-method for crude global registration. The thesis is then concluded in Chapter 5 with a summary and an outlook on possible future work.

CHAPTER 2

Related Work

In this chapter an overview of the research on rigid surface registration is presented, with a special emphasis on crude registration. Since registration is an important field in many different research domains, like computer vision, computational geometry or medical imaging, a lot of research has been conducted. We will present the relevant literature structured as follows. Starting from a more general view on surface registration, we will split up the literature review into separated steps often found in the literature: *feature detection* and *feature description*.

In addition to the literature on algorithms, we will also provide a short coverage of the *Microsoft Kinect*^{TM} commodity range scanners and how they capture depth information.

2.1 Surface Registration

A recent review of crude registration methods is offered by Diez et al. in [10]. They also propose a standardized notation to counteract the fact that crude registration plays an important role in several different research domains, like computational geometry and computer vision, but similar ideas are named differently. They also mention volumetric data as registration input, since it is most common in medical applications where registration also plays an important role. The structure of this chapter was inspired by their registration pipeline definition. They highlight the relation between feature detection and description since features are often chosen because of the distinctiveness of their associated descriptor, and in most cases both steps are based on the local shape of the object. They also define the term *shape function* which is used to describe which surface characteristic is used to detect or describe a feature point.

In terms of applications Weise et al. [40] use a combination of coarse and fine registration which also leverages the texture information provided by an *RGB-D* camera, a camera that provides color as well as depth data, for in-hand 3D modelling at interactive frame rates. They also conduct geometrical, as well as textural, consistency checks for a registration result but do not incorporate the sensor error model into the checks.

Registration of fractured objects is presented by Huang et al. [21] where they segment the parts of an object to identify fracture surfaces. Features are then identified and correspondences

found. Registration is performed first in a pair-wise manner to create a matching-graph, and then multi-part registration is performed, using this graph, to get the final reassembled object.

2.2 Fine registration: the Iterative Closest Point algorithm

Fine surface registration, independent of representation and dimension, is presented by Besl and McKay in their seminal work on the *Iterative Closest Point (ICP)* algorithm, [5]. Several implementations of ICP exist in Open Source Software packages, e.g. [26, 29].

The key idea is to register 2 surfaces, the *data* surface, represented as point set P, against the *model* surface, transformed to point set X, iteratively, in the following 4 steps:

1. For points $p_i \in P$, find their closest points $y_i \in X$
2. Compute a transformation which best aligns p_i and y_i
3. Evaluate a mean square error metric
4. Re-iterate if maximum-threshold not satisfied

When a closest point y_i is found for each point p_i an aligning transformation is estimated by calculating a symmetric matrix $Q(\sum_{PY})$ of the following form:

$$Q(\sum_{PY}) = \begin{bmatrix} tr(\sum_{PY}) & \Delta^T \\ \Delta & \sum_{PY} + \sum_{PY}^T - tr(\sum_{PY}) I_3 \end{bmatrix} \tag{2.1}$$

where tr is the trace, $\Delta = [A_{23} A_{31} A_{12}]$ is the anti-symmetric matrix $A_i j = (\sum_{PY} - \sum_{PY}^T)$, T is the transpose operation of a matrix and I_3 is the $3x3$ identity matrix. $\sum_{PY}$ is the cross-covariance matrix of points p_i and their corresponding closest points y_i. The aligning transformation consists of a rotation matrix R and a translation vector $\mathbf{t}$. The rotation matrix is based on the unit eigenvector $\mathbf{q}_R = [q_0, q_1, q_2, q_3]^t$ corresponding to the maximum eigenvalue of the matrix Q. $\mathbf{t}$ is the vector difference between the mean position of y_i and the mean position p_i rotated by R,

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}, \tag{2.2}$$

$$R = \mu_{\mathbf{y}} - R\mu_{\mathbf{p}}. \tag{2.3}$$

They prove that their algorithm monotonically converges to a local minimum of the mean square error metric stated above. Leveraging this monotonicity property, they present an accelerated algorithm for finding the next registration transformation, formulated as a unit quaternion vector q_{0-6}, based on either line based or parabola based deduction in the 7-space of transformations. This typically provides them with a reduction of iterations needed from 50 to between

15 and 20 iterations for a given mean square error tolerance. The convergence toward a local minimum means that the resulting aligning transformation is not necessarily the correct one.

They address global matching by relying on a set of initial states. Depending on the first two moments of geometric variation of the shapes and coverage they limit the set of initial sets to rotations. If this is not enough, they propose sampling the upper hemisphere of the unit 4-sphere of possible quaternions based on rotation groups of the regular polyhedra. One can easily deduce that its sensitivity to an initial alignment state is a limitation, although once the correct initial state is found, global fine registration can be achieved using ICP.

Multiple improved versions of *ICP* have been proposed that address different concerns. In [18] Greenspan et al. propose nearest neighbor pre-processing of the model point set in order to speed up closest-point search by looking at local neighborhoods of the closest points of the previous iteration. Jost et al. [24] use invariant point features to improve the point distance metric for better correspondence. Features used are spherical harmonics, curvature and moments. A similar approach is also followed in Godin et al. [16] such that they incorporate color information for better point correspondences to also handle geometrical symmetries.

ICP has been used in Newcombe et al. [30] to continuously build an implicit surface registration of a complex scene in real-time using sensor data from a Microsoft Kinect^{TM} device. Their work is targeting the field of *simultaneous localisation and mapping (SLAM)*, which is often used to give autonomous robots the ability to sense their environment and localise themselves within it.

Using the data streams of a handheld Kinect sensor, they continuously track its 6 degrees-of-freedom (6DoF) pose and integrate depth measurements into a global dense volumetric model of the scene in real-time using the highly parallel computing capabilities of a graphics processing unit(GPU). They use ICP on a multi-scale representation of a range scan for scan alignment to estimate the sensor's pose relative to the global surface.

They preprocess each scan frame with a discontinuity preserving bilateral filter to decrease the amount of noise and therefore gain better per-point normals. They assume only small movement between consecutive depth frames, which lets them use ICP with a *projective data association* algorithm to obtain correspondence and the point-plane metric for pose optimisation.

2.3 Feature detection

Some approaches in surface registration rely on the identification of salient, or distinctive, feature points. One of the reasons for that is that it is important to reduce the number of points for the correspondence search. We now want to present an overview of different feature detection methods available in the literature.

Feature detection and description are closely related since often the distinctiveness of a feature descriptor in a given set of points is used to map a saliency measure. This step is also called *filtering*, since "unimportant" points will be filtered.

In an attempt to investigate *saliency* as such, Chen et al. [8] analyse *Schelling points*, those points on surfaces that where identified by most people based on perceived saliency, in a survey they conducted. The idea was to find out what makes up a *significant point*. They concluded that these points are usually highly symmetric across an object and often relate to points with

significant local curvature, points on symmetry axis or center points of segments. They also fit an analytical model to their collected observations which can predict Schelling points on 3D meshes.

If point normals are available, *Normal Space Sampling (NSS)* can be used. Introduced by Rusinkiewicz and Levoy in [33], they group points into "buckets" according to the angles between their normal vectors (considered in the unit sphere) and the coordinate axes. To filter points, they sample uniformly across the resulting buckets, providing a downsampling of the points with more "frequent" normal vectors. Diez et al. build upon this idea in [11] by grouping buckets hierarchically to speed up correspondence search during matching.

Similar to the approach presented in this thesis Ho et al. present a curvature scale space in [19]. At each surface point p_i, they estimate its local *curvedness*, the amount of local curvature, expressed as a positive number. They estimate the local curvature at different scales by fitting a manifold to the local neighborhood of different sizes. They identify salient feature points based on the *confidence* in local curvedness extrema. In Section 3.6 we provide a comparison between *curvature scale space* and the approach used for our algorithm, for identifying feature locations in noisy data.

Another approach based on estimated normals by Ioannou et al. is proposed in [22]. They introduce the multi-scale *Difference of Normals (DoN)* operator. It is based on the idea of the *Difference of Gaussians* multi-scale operator used for 2D image recognition proposed by Lowe et al. [27]. They estimate normals in range data using principal component analysis (PCA). The Difference of Normals operator at point $\mathbf{p}$ is formulated in its basic form as

$$\boldsymbol{\Delta}_{\hat{\mathbf{n}}}(\mathbf{p}, r_1, r_2) = \frac{\hat{\mathbf{n}}(\mathbf{p}, r_1) - \hat{\mathbf{n}}(\mathbf{p}, r_2)}{2}, \tag{2.4}$$

with $\hat{\mathbf{n}}(\mathbf{p}, r_i)$ denoting the estimated normal vector for support radius r_i and $r_1 < r_2$. Choosing support radii is a matter of experimentation and results in features of different scales being filtered.

Other related work in the field of feature detection are approaches which apply prominent algorithms from the 2D image processing domain to 3D data. *Harris 3D* by Sipiran and Bustos ([36]) aims at applying the well known Harris operator on a 2D projection of the point data and considers those points as feature points that exhibit the highest Harris response.

2.4 Feature description

Once salient feature points are identified, a *feature descriptor* can be used to aid the correspondence search between data and model surface. A feature descriptor encodes information about the neighborhood around a feature point. What follows is a review of feature description approaches most found in the literature.

Spin Images, a localised description of the global shape of an object, was proposed by Johnson et al. in [23]. It can be modified to encompass only local information as well. The description is relative to an oriented basis point, i.e. a point and the surface normal at this point. Because they need reliable normals to establish point orientation they need the shape to be represented as a polygonal surface mesh.

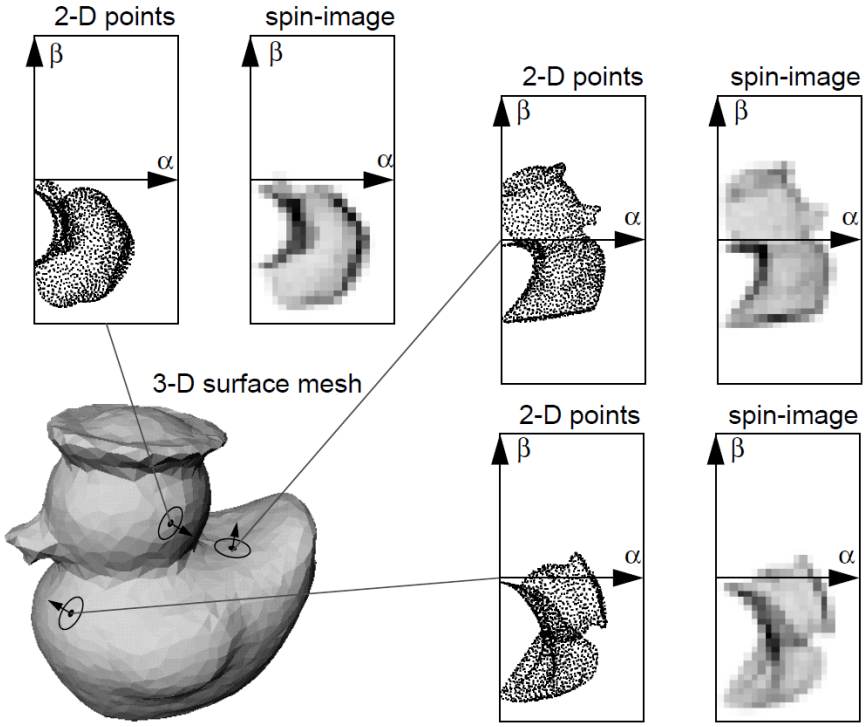


Figure 2.1: Concept of *spin images*. 3 spin images constructed from 3 different basis points on the surface shown on the lower left. Image courtesy of Johnson et al. [23]

.

A spin-image is a 2D array of bins (α, β) that encodes aggregated point positions on the surface relative to the oriented basis point by "spinning" a sheet around the surface normal at the basis point. For each other point, the value at the bin at its radial coordinate α and the elevation coordinate β is increased. The concept is depicted in Figure 2.1.

They base the bin size on the mesh resolution. The amount of the neighborhood encoded in a spin image can be controlled by 2 parameters, *image width* and *support angle*. Image width controls the number of bins in both dimensions and defines together with the bin size the *support distance* D_s. Image width is used to limit the amount of clutter encoded within a spin-image, the support angle is used to reduce self-occlusions possibly not visible from a capture perspective. The support angle is used in the following way.

Given a base oriented point a with normal $\mathbf{n}_a$, a point b with normal $\mathbf{n}_b$ will be accumulated in the spin-image of a if

$$acos(\mathbf{n}_a, \mathbf{n}_b) < A_s \tag{2.5}$$

where A_s denotes the support angle.

This formulation basically means that points with similar normals will be accumulated. For highly concave objects self-occlusions cannot entirely be removed but in combination with D_s highly reduced. Matching of spin-images rests on the fact that spin-images from uniformly

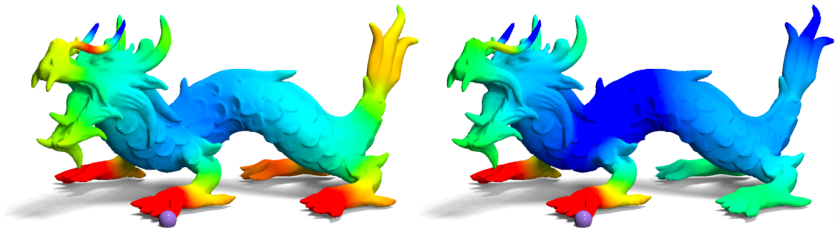


Figure 2.2: Illustration of difference of HKS between the point marked by the purple sphere and other points on the object's surface over different temporal intervals $[t_1, t_2]$. The difference increases as the color changes from red to green to blue. Left: both t_1 and t_2 are small, right: t_1 is small but $t2$ is big. Image courtesy of Sun et al. [39]

.

sampled surfaces are linearly related. This means that surfaces need to be resampled before in order to enforce uniform sampling. Similarity of spin-images is measured using the linear correlation coefficient. For performance reasons, they also present a way to compress spin-images of model data for later matching using PCA, and show how to match uncompressed spin-images with compressed ones.

Sun et al. [39] present *Heat Kernel Signature (HKS)* to both describe the geometry around a point p and detect salient points on a surface using those descriptors for manifolds. HKS is based on the *heat kernel* $k_t(x, y)$, associated with the *Laplace-Bertrami operator* [39], that describes how much heat flows from point x to point y in time t. HKS, however, is limited to the temporal domain and thus more compact. They state that HKS is informative in that it characterizes the local shape up to isometry and holds all information about the intrinsic geometry. They achieve a multi-scale description by varying the time component t. In this way, they get point descriptors for a point x that hold information about the structure of the local neighborhood with small t and characteristics of the global structure relative to x with higher values for t. HKS for different scales is shown in Figrure 2.2.

They define the HKS at a point p as

$$HKS(x) : \mathbb{R}^+ \rightarrow \mathbb{R}, HKS(x, t) = k_t(x, x), \tag{2.6}$$

where $k_t(x, x)$ denotes the heat kernel for a point x which is computed as

$$k_t(x, x) = \sum_i e^{-\lambda_i t} \phi(x)^2. \tag{2.7}$$

Here λ_i and ϕ_i denote the i'th eigenvalue and eigenfunction of the *Laplace-Bertrami operator* of the manifold. Sun et al. show that, in the discrete case, the Laplace-Bertrami operator can be substituted by the Laplace operator of the mesh.

We could have also mentioned HKS in Section 2.3 because it can also be used to identify salient points on a manifold. Sun et al. also show that in case of discrete surfaces (meshes) HKS is closely related to *Gaussian curvature*, a fact they use to detect salient points as follows.

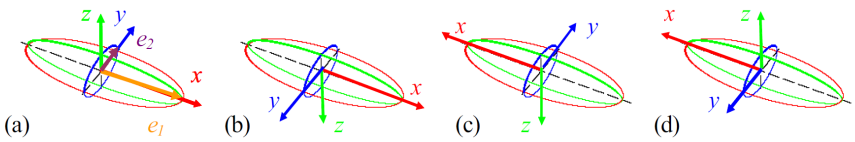


Figure 2.3: The *intrinsic reference frame* based on the principal axis e_1 ans e_2. a) shows the primary one whereas b)-d) depict reference frames based on a) to cover all possibilities of primary axis directions. Image courtesy of Zhong et al. [42]

For each point p, it's HKS is computed. A point is considered salient if it exhibits a local maximum in HKS compared to it's 2-ring neighborhood.

The difference between to HKS is defined as

$$d_{[t_1,t_2]}(x,x') = (\int_{t_1}^{t_2} (\frac{|k_t(x,x) - k_t(x',x')|}{\int_M k_t(x,x)dx})^2 d\log t)^{\frac{1}{2}}. \tag{2.8}$$

In practice, matching HKS's of two points x and x' at a scale interval $[t_1, t_2]$ is done by creating a vector by uniformly sampling their HKS on a logarithmic temporal scale and computing the $L-2$ norm of the vectors.

Although HKS provides stable multi-scale properties, it is limited to meshes and therefore not applicable to point clouds. The reported computation times for calculating the matrix and extracting the necessary 300 eigenvalues for each point lay in the minutes, which proved to be rather time consuming compared to other techniques.

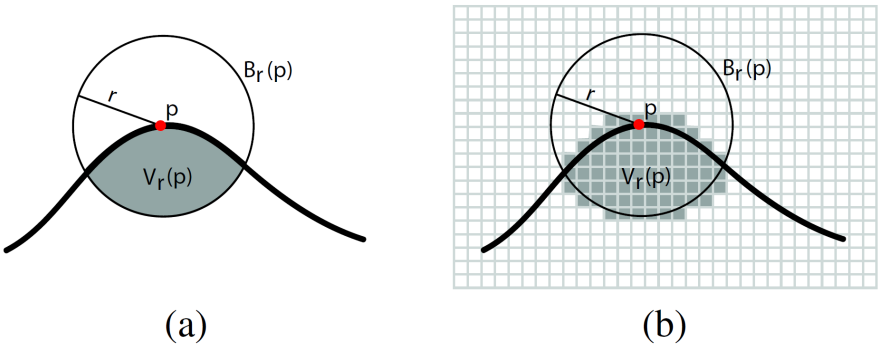
Zhong et al. [42] present *Intrinsic Surface Signatures (ISS)* to describe the semi-local neighborhood around a basis point p_i. They compute a 3D occupational histogram using a *discrete spherical grid* ([20]) for a spherical neighborhood of radius $r_{feature}$ around each basis point. Each point p_i is assigned a weight which is inversely related to the number of points in the spherical neighborhood of p_i with radius $r_density$:

$$w_i = \frac{1}{\|\{p_j : \|p_j - p_i\| < r_{density}\}\|}. \tag{2.9}$$

They achieve view independence of the ISS by setting up an *intrinsic reference frame* F_i at each basis point p_i. The intrinsic reference frame consists of a coordinate system which is estimated with PCA using a weighted scatter matrix with weights computed with 2.9. With eigenvectors e_1, e_2, e_3 of the scatter matrix that correspond to the 3 biggest eigenvalues the axes of the coordinate system are formed via e_1, e_2 and their cross product and the basis point p_i which represents the origin. Orientational symmetries are also handled by storing 4 versions of each intrinsic reference frame, as depicted in Figure 2.3.

With K denoting the number of spherical grid cells, the ISS for a point p_i is a vector $f_i = (f_{i0}, \cdots f_{iK-1})$ where each element f_{ij} is computed as the sum of the weights of all points in cell j plus the intrinsic reference frame F_i.

They match two point clouds by point-to-point comparison of ISS's to find point pairs. To calculate an aligning transformation they apply a voting scheme over all transformations which



(a) (b)

Figure 2.4: Illustration of calculating the *Integral Volume Descriptor* for the 2D case. (a) Gelfand et al. [15] encode the intersection of a sphere of radius r centered at point **p** with the interior of the surface. (b) shows the discretization into a volume grid of cell size ρ.

align a pair of intrinsic reference frames of matching ISS. Two points match if their ISS's are similar and they satisfy the transformation with the maximum vote.

They formulate their measure of match between two point clouds P and Q as

$$E_{comprehensive} = \frac{E_{residual}(P,Q)}{Similarity(P,Q)}. \tag{2.10}$$

$E_{residual}$ specifies the positional residual between matching point pairs and $Similarity$ measures the number of matching point pairs N_c versus the geometric mean of the numbers of basis points of each point cloud N_P and N_Q,

$$Similarity(P,Q) = \frac{N_c}{\sqrt{N_P \cdot N_Q}}. \tag{2.11}$$

This measure of match is still based on local descriptors and does not provide a concrete answer on whether or not two point clouds match globally, and together with the fact that an intrinsic reference frame has three siblings based on symmetries, adds another factor of uncertainty.

Gelfand et al. [15] argue that differential surface properties like normals or curvature are very sensitive to noise. They propose an integral measure as their local surface descriptor. They present the *Integral Volume Descriptor*. Given a sphere $B_r(\mathbf{p})$ with radius r centered at surface point **p**, they count the number of voxels which lie inside the surface boundary. They setup an occupancy grid and intersect it with the sphere. The descriptor is then the amount of voxels of the sphere that fall inside the surface. See Figure 2.4 for the 2D case. The volumetric descriptor, assuming a simply-connected intersection between the convolution kernel $B_r(\mathbf{p})$ and the surface S, is related to the *mean curvature* H as follows,

$$V_r(\mathbf{p}) = \frac{2\pi}{3}r^3 - \frac{\pi H}{4}r^4 + O(r^5). \tag{2.12}$$

They also use their descriptor to identify salient points. After calculating the descriptor for every point of the data surface with cardinality N, they create a histogram and select the k least populated bins such that the total number of points in them is less then $0.01N$, N being the total number of points in the point cloud. They also limit the minimum distance of the selected points to a radius R_e which is based on a predefined noise model.

For scale invariance, they employ a scale-space strategy by calculating the volume descriptor for 5 radii r_i between 10ρ and 0.1 times the diameter of the shape. ρ is the resolution of the voxel grid. They consider a point $\mathbf{p}$ a feature only if it is selected from the volume descriptor histogram for at least 2 consecutive radii. By deriving the scale-space radii from the diameter of the shape, they are able to relate feature points of the data and model shape based on their scale.

They identify correspondence candidates on the model surface using an intrinsic metric, the *distance root mean squared error (dRMS)*. The dRMS error is computed by comparing all internal pairwise distances of the two point sets, and is defined as

$$dRMS^2(\mathbf{P}', \mathbf{Q}') = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\|\mathbf{p}_i - \mathbf{p}_j\| - \|\mathbf{q}_i - \mathbf{q}_j\|)^2. \tag{2.13}$$

This way they do not need to calculate a transformation to evaluate the model points for correspondence. For each point on the data surface, they identify a correspondence candidate set of points of the model data and use a branch-and-bound algorithm to find the best correspondences. They leverage the rigidity constraint to prune away correspondence candidate branches. Once the best correspondences for the feature points are found, they calculate an aligning transformation and repeat the process with an updated bound for the branch-and-bound algorithm until they reach a minimal error. This way however, even though a lot of pointwise correspondences are found for the feature points, there is no guarantee that the remaining data points have real correspondences in the model data.

For partial matching they introduce a *non present value* $\emptyset$ as a possible point correspondence setting in their correspondence search. To get the maximum number of valid correspondences, while still keeping $dRMS$ as low as possible, they run their branch-and-bound algorithm with varying numbers k of assumed points missing in the model data.

2.5 Feature-agnostic alignment

Chen et al. [7] present *RANSAC-based DARCES*, an algorithm which searches for point correspondences by evaluating them with respect to a rigidity-constraint. Their output is a rigid transformation T_C which aligns the *data surface* with the *model surface*.

Different than the literature presented so far, they do not exploit local point features, like normals or curvatures, but instead propose uniform subsampling of the surface to reduce the number of data points as search space. They argue that no pre-processing is needed this way, and thus time saved.

In their basic algorithm, they identify three control points on the data surface: the *primary point* S_p, the *secondary point* S_s and the *auxiliary point* S_a. Three point correspondences are needed to calculate a *least mean square (LMS)* transformation with 6DoF. On the model surface,

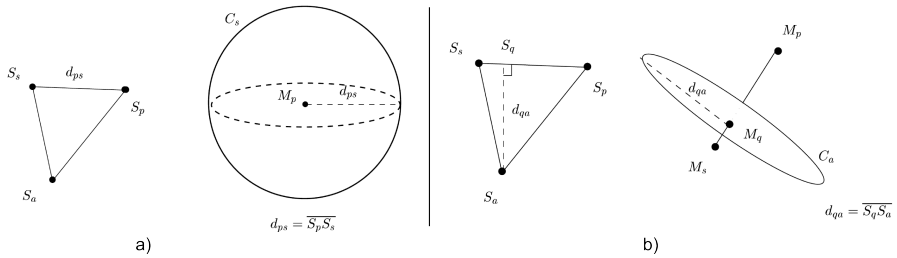


Figure 2.5: The *rigidity constrained search* in RANSAC-based DARCES for three control points S_p, S_s and S_a. a) shows the initial setup of the primary point S_p, the secondary point S_p and the auxiliary point S_a on the left. How the corresponding point M_s for the secondary point is found once a correspondence candidate M_p for the primary point is chosen, is depicted on the right side. A sphere of radius d_{ps} is centered at M_p and all points at the intersection border of the sphere and the surface are possible candidates for the secondary point. b) shows how to get the corresponding point of the auxiliary point. Image courtesy of Chen et al. [7].

they start from a randomly selected point M_p as the correspondence for S_p and select two more points such that they form a similar triangle as the one spanned by S_p, S_s and S_a, as depicted in Figure 2.5. If they can identify such three points, they use the correspondence pairs to calculate a transformation which is then used to transform all the remaining subsampled points $S_{r,i}$ of the *data surface*. The alignment quality is then measured as the number n_0 of successfully aligned $S_{r,i}$. They call a point successfully aligned if its distance to the *model surface* is smaller than a given threshold but do not provide further details on which metric they use or whether they base their threshold on a specific model.

They use the first three control points to calculate a transformation T_C and then transform each of the remaining control points and evaluate the rigidity constraint. If the rigidity constraint is violated by one of the transformed control points they discard the transformation and proceed with the next possible three point correspondence. In this way, they do not check all the points of the data surface for correspondence for each three point correspondences found in the model data.

In addition, they provide explanation on how to use more than three points. Arguably, using more than three points speeds up correspondence search in the case when the data surface is fully contained within the model data.

The RANSAC-based approach rests on the verification process. Several control point sets are used to find correspondences. Verification is done by measuring the number n_0, or *overlapping number*, of points of the *data surface* that have a corresponding *model data* point. The transformation with highest n_0 is considered an aligning transformation. By its optimisation-nature, their verification does not guarantee an aligning transformation since only the "best" transformation is accepted and manual verification is still needed. This is the most commonly used verification procedure found in the literature. They only guarantee the optimal solution in case of noiseless data, and when the data surface is fully contained within the model data, since they can evaluate all possible transformations.

The algorithm presented in this thesis was heavily inspired by *RANSAC-based DARCES*

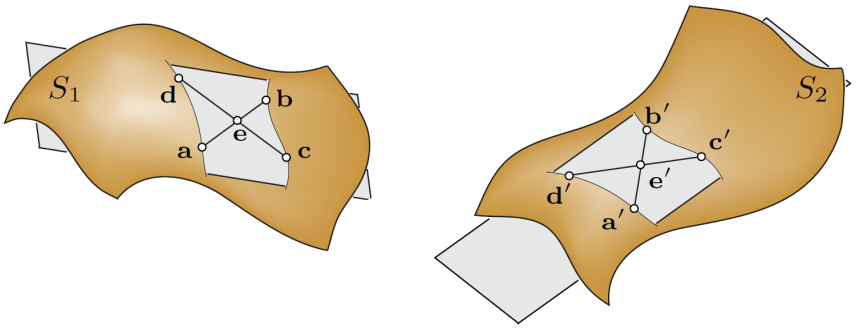


Figure 2.6: The concept of congruent 4-points by Aiger et al. [1]. Using a quadrilateral base of S_1, the idea is to find a 4-point set in S_2 such that specific ratios are preserved.

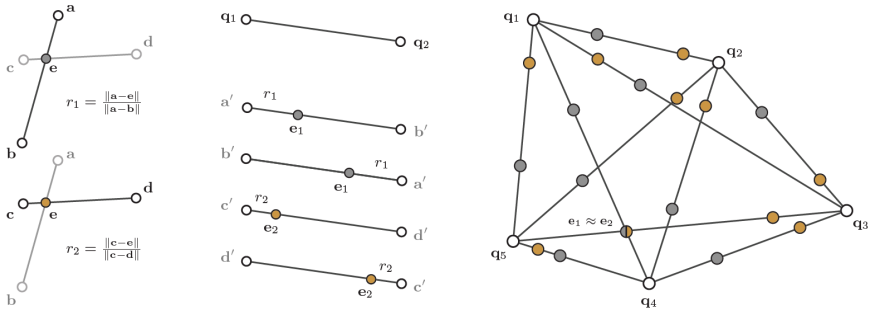


Figure 2.7: Identifying congruent 4-points in a target surface Q. Given a base of 4 roughly coplanar points a, b, c, d calculate ratios r_1 and r_2 (left). With a point pair $(q_1, q_2) \in Q$, calculate intersection point candidates for each ratio for both directions (middle). 2 point pairs for which intersection points for the correct ratio coincide can be considered for congruency test (right).

such that it also identifies triples of points and identifies correspondences between triples points based on a rigidity-constraint.

Aiger et al. [1] follow a similar approach. They extract correspondences between *4-point sets*, four (roughly) coplanar points, of each surface. The key idea is the following: Under affine transformations certain ratios are preserved. Given two point clouds P and Q, they follow a RANSAC approach and select from P a random *quadrilaterial base*, ie. 4 points $B \equiv \{a, b, c, d\}$ which are roughly coplanar and roughly span the estimated overlap of the surfaces. Using this *base*, they find congruent 4-point sets in Q. For each such a candidate, they estimate a transformation which aligns it to the base of P. Figure 2.6 shows the concept.

The 4-point set exhibits the following properties. The intersection point e of the lines ab and cd is used to calculate ratios $r_1 = \frac{\|a-e\|}{\|a-b\|}$ and $r_2 = \frac{\|c-e\|}{\|c-d\|}$. The interesting finding is that these ratios are preserved under affine transformations.

With distances $d_1 = \|ab\|$ and $d_2 = \|cd\|$, they identify subsets $R_1, R_2 \subseteq Q$ such that

$$R_1 \equiv \{(q_i, q_j) | q_i, q_j \in Q, \|q_i - q_j\| \in [d_1 - \epsilon, d_1 + \epsilon]\} \tag{2.14}$$

$$R_2 \equiv \{(q_i, q_j) | q_i, q_j \in Q, \|q_i - q_j\| \in [d_2 - \epsilon, d_2 + \epsilon]\} \tag{2.15}$$

with ϵ being a user defined inaccuracy measure. To all those point pairs, they apply ratios r_1 and r_2 for both directions, which gives them 4 intersection point candidates per point pair. The combinations of pairs of R_1 and pairs of R_2, which have coinciding intersection point candidates, form congruent 4-points that can be used to estimate a transformation which aligns them to B. The concept above is depicted in Figure 2.7. Each extracted transformation is applied to Q until a *Largest Common Pointset (LCP)* between P and Q is found.

In principal, their approach is not dependent on point features or the use of other surface characteristics, e.g. color, but can be enriched by them to aid the correspondence search. They also show how to incorporate pointwise normals in their approach, as an example.

The point to note about 4-point congruent sets is that, although only 3 points would be necessary to estimate a 6DoF transformation, using quadrilaterals as base, they can reduce the runtime complexity of the correspondence search to $O(n^2)$ because they only need to identify point pairs.

An improved version of the 4-PCS, aiming at reducing the time complexity, was proposed by Mellado et al. [28]. They tackle the most computationally complex part of the approach by Aiger et al., which is the extraction of point pairs which are d_1 and d_2 apart. They formulate it as an *incidence problem* of points and spheres. They apply a grid based approach in which they recursively rasterize spheres of radius d_1 and d_2. The spheres are centered at points $q_i \in Q$. They then subdivide cells which have an incidence with the spheres. They adaptively index points based on the grid cell they are in, and build point pairs between the centers of the spheres and the points in the cell reached by the sphere. This gives them optimal linear time.

For our comparison in Section 4.2 we chose *Super 4-PCS* due to its global and featureless nature.

2.6 Commodity ranger scanners: the Microsoft Kinect™

The algorithm for this thesis was implemented while having a Microsoft Kinect™ device of the first and second generation as a data source in mind. The differences in how they acquire depth information are important to know how they affect the noise of point measurements. A detailed description will be provided in Section 3.4, but the high-level differences in capturing depth information are as follows.

Kinect v1 measures depth based on triangulation. An infrared speckle pattern is projected into the scene and its reflection captured by an infrared sensor. The expected pattern at a reference plane of known distance is stored within the device. Since the infrared sensor does not coincide with the projector, it will observe displacements of the projected pattern at objects of different distance to the projector. This disparity can be used to calculate the distance of the captured object using triangulation. The fact that depth measurements are based on triangulation leads to a depth measurement error as a quadratic function of the measured distance, [25].

Kinect v2 measures depth using the *Time-of-Flight* principle. It measures how long it took for the infrared light projected into the scene to arrive back at the infrared sensor after being reflected by objects. This is done by correlating the incoming signal phase via a known phase

modulation frequency with light emitted by the infrared laser projector, [35]. This leads to a linear depth measurement error function as reported by Pagliari et al. [31].

2.7 Summary

In this chapter a literature review in the field of surface registration was provided, focusing on feature detection and feature description algorithms. We first covered the *ICP* algorithm as a solution for fine registration. After that we presented multiple solutions for feature detection and description. Feature-agnostic approaches, which do not depend on surface characteristics at a given point, were also presented. We furthermore provided a short summary of how depth is captured in commodity range scanners. In the next chapter a detailed description of our algorithm will be covered.

CHAPTER 3

Tolerant global registration

In this capter a detailed description of the approach taken for this thesis is provided. First, the most important aspects used in the description are defined, followed by each step of the algorithm accompanied by the registration of two synthetic range scans of the Stanford Bunny as an example. Code was written in the *C++* programming language[1] using the *OpenCV*[2] library for its well documented matrix calculation and image processing-routines.

3.1 Definitions

The spatial error of a measurement made with a range sensor is usually modelled as a Gaussian distribution. We employ the *error sphere* as a simplified assumption of the error model, instead of an ellipsoid reflecting the cut-off of the Gaussian distribution modelling the error at a given position.

A ***point triple*** is a set of three points a, b and c that together make up a triangle with edges ab, bc and ca. Each point of the triple is a point of the point cloud that results from reprojecting the range scan to 3D and therefore subject to the noise model of the sensor that captured the point cloud. Hence the length of each edge is specified as an interval $\left[l_{min}, l_{max}\right]$ with

$$l_{min} = \|ab\| - (r_a + r_b) \tag{3.1}$$

$$l_{max} = \|ab\| + (r_a + r_b)\,, \tag{3.2}$$

r_p being the radius of the *error sphere* centered at triangle point p.

Only three point correspondences, the position of these points being subject to sensor noise, are used to calculate a transformation candidate that minimizes the distance between corresponding point pairs of both point clouds A and B. This leads to a transformation $\hat{T}$ that minimizes an error function, e.g., least squares, that only takes into account those point pairs of corresponding

[1] `https://isocpp.org`
[2] `http://opencv.org`

point triples used to estimate it. Effectively, this means that all other points of the point cloud A, after transformation, are not as close to their corresponding points of B as they could be if the error function would have been minimized globally, i.e. over all points of the point cloud. Given a candidate for an aligning transformation $\hat{T}$ estimated using corresponding point triples t_a and t_b consisting of points a_i and b_i with their respective error sphere radii r_{ai} and r_{bi}, a point p of A transformed by $\hat{T}$,

$$\hat{p} = \hat{T} * p, \tag{3.3}$$

has to be evaluated for correspondence incorporating a ***transform tolerance*** $\hat{r}_{tilt}$. The *transform tolerance* models the fact that the points of t_a used to estimate $\hat{T}$ are inaccurate measurements and, would they be displaced withing their error spheres, a different transformation $\hat{T}_e$ would have been estimated and p would be transformed to a different position,

$$\check{p} = \hat{T}_e * p. \tag{3.4}$$

Transform tolerance $\hat{r}_{tilt}$ of p is therefore

$$\hat{r}_{tilt} = \|\hat{p} - \check{p}\| . \tag{3.5}$$

In this way, we incorporate inaccuracy awareness into our transformation verification process. For further details on calculating the *transform tolerance* see Section 3.11.1.

3.2 Algorithm Overview

Here we provide an overview of all the steps involved in the registration process, followed by an overview of both Microsoft Kinect™ devices and their characteristics. A detailed explanation of every step of our algorithm is then provided, starting at Section 3.3, which defines the involved parameters and provides a short explanation of their usage. We will now list each step of our algorithm, see Figure 3.1 for an illustration:

- *3D position and error calculation*: The *3D point clouds* are projected from the range scans and *pointwise spatial error radii* are calculated using a scanner-dependent error model, taking into account *discretization* due to pixelation and depth measurement noise. See Section 3.4.
- *Occlusion detection*: We identify points that lie on the *object's silhouette* and boundaries of *self occlusion* in the image domain using a 2D image filter. See Section 3.5.
- *local curvature estimation*: We use the information gathered from *occlusion detection* so we can reliably fit a quadratic surface to a point's local neighborhood and evaluate the curvature of the surface. See Section 3.6.
- *Feature identificatin*: We then identify *local curvature maxima* as salient feature points. See Section 3.7.

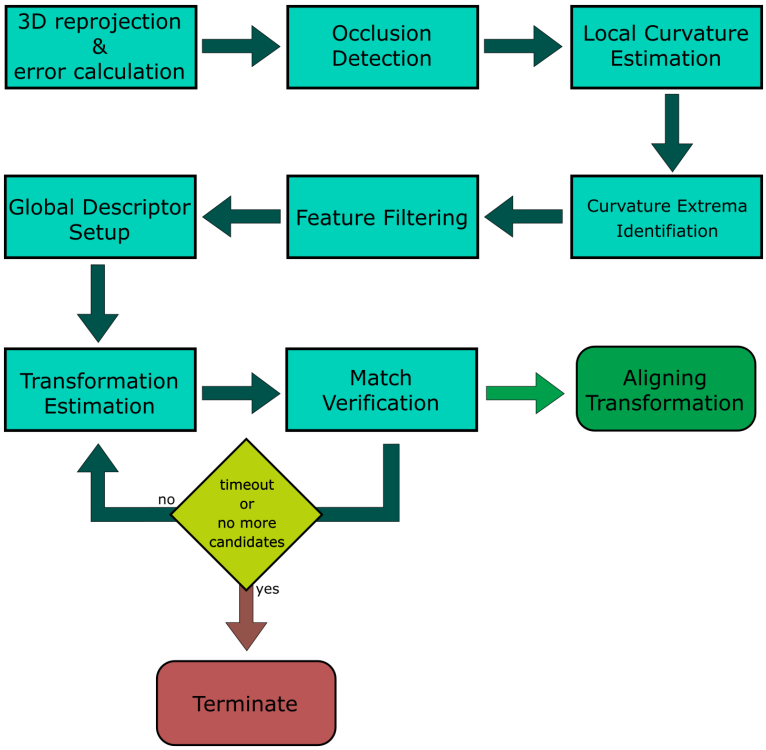


Figure 3.1: Flow chart of the proposed algorithm which highlights the main steps involved. The algorithm terminates either with an aligning transformation or if no more transformation candidates are available or the process times out after a specified period of time.

- *Feature filtering*: We filter feature points based on multiple criteria to shrink the search space of correspondence search. See Section 3.8.
- *Global descriptor setup*: We set up our global descriptor based on *point triples* that incorporate the previously computed pointwise spatial error radii, See Section 3.9.
- *Transformation estimation*: Global descriptors of the *target* and *reference* range scans are iteratively matched by randomly selecting a point triple of the reference scan and finding matching point triples in the target scan by comparing the edge lengths between the points constituting each point triple using a *sweep and prune* approach. A candidate for an aligning transformation is calculated with *Singular Value Decomposition (SVD)* using the three point correspondences of the matching triples. See Section 3.10.
- *Match verification*: The estimated transformation is verified by first testing *Invalid Free Space violations* and *Invalid Occupied Space violations* of every point of the target scan against a subset of the reference scan that in image space is covered by the evaluated target point's *transform tolerance volume*. If that test passes, the transformation is again verified by aligning the reference scan to the target scan and only then is a transformation consid-

ered an aligning transformation. This is in contrast to other work found in the literature, where only a measure of match quality can be given based on stochastic estimates. See Section 3.11.

Transformation estimation and *match verification* are done iteratively, and our algorithm terminates either if a match could be verified or if no more matching point triples are left to estimate another transformation candidate. Additionally, we introduced a *timeout* parameter, which can be used in case a decision needs to be made in a certain period of time.

3.3 Parameters

Before entering the algorithm implementation in detail, a short description of the available parameters is shown below.

- **Occlusion threshold (τ_o):** A global threshold on the depth difference between neighboring pixels of the 2D range scan to identify points at boundaries of the object or self occlusion. Specified in scene dimensions.
- **Curvature absolute minimum threshold (κ_{min}):** A global threshold to filter points with low curvature response.
- **Maximum triple edge length (λ_{max}):** Threshold used to limit the size of a point triple expressing the expected diameter of overlap between scans. Specified in scene dimensions.
- **Minimum triple edge length (λ_{min}):** Threshold to choose wider point triples to reduce the possible transform tolerance. More details will be shown in Section 3.11.1. Defaults to 0.

3.4 Calculating 3D point positions and pointwise errors

The presented algorithm processes a point cloud in 3D space. The first data preparation step described here will set up an association of each pixel that holds valid depth information with a pair of 3D point and error radius. The input for this step is a range image. Its output is a list of 3D points together with their error radii. Each point is associated with a pixel in the original range image.

As covered in Chapter 2, different surface-registration techniques work on data of varying representations. It can be in the form of a 3D mesh, which is a essentially a connected graph with nodes V as 3D positions and edges E connecting neighboring nodes, an unstructured point cloud with no neighborhood information, or a *range image*, which can be seen as an image where the value at each pixel describes the distance of the first object visible to the scanning device at that particular pixel at the time of capture, measured as the distance to a plane perpendicular to the viewing direction of the range sensor. Our implementation rests heavily on the fact that, using a range image, its 2D nature provides an inherent neighborhood information between points,

which is not available when using an unordered 3D point cloud. Furthermore, dealing with a range scan in terms of an image makes it possible to leverage image-processing routines, like *edge detection*, that are well known and have good software support. Therefore, when registering two range scans, we operate on two images I_R and I_T captured by one or two distinct range sensors. Each pixel of $I_{R,T}$ that covers the captured object holds a distance in millimeters. A value of 0 at a pixel indicates either a measured distance outside of the sensor's reliable range or that the sensor itself reported invalid data for that pixel, so only non-zero pixels will be processed.

Given the input in the image domain, the goal of the data preparation stage is to derive the 3D position of a point corresponding to a pixel as well as its *spatial error sphere*. The latter is the volume around the measured point in which the ground-truth surface point could actually lie. The aim is to establish a correlation of a pixel to a 3D point together with an error measure, $p \sim (\mathbf{p}, r_e)$, with p being the image domain pixel, $\mathbf{p}$ being the 3D point associated with it and r_e the radius of the *error sphere* centered at $\mathbf{p}$.

3D position

To derive the 3D information of a pixel p, it has to be reprojected to 3D space. To do this, the intrinsic parameters, i.e., focal length for both image axes f_x, f_y and radial distortion parameters k_2, k_4 and k_6, of the scanning device the scan was captured with have to be known. To incorporate the spatial uncertainty of the scanned point into the pixel-point mapping, two sources of spatial error were identified, *discretization* due to pixelation and *sensor noise*.

To be able to establish correct correspondences, we need to map pixels of different range sensor outputs to the same coordinate system relative to the scanning device. This coordinate system is defined by the three axis vectors $\mathbf{x}_c$, $\mathbf{y}_c$ and $\mathbf{z}_c$. We chose the prevalent right-handed coordinate system in computer graphics with $\mathbf{y}_c$ pointing to the top of the device, $\mathbf{z}_c$ pointing in the opposite viewing direction and $\mathbf{x}_c$ resulting from the cross product of $\mathbf{y}_c$ and $\mathbf{z}_c$. Given an image pixel $p = (\hat{x}, \hat{y}, d)$, $\hat{x}$ and $\hat{y}$ denoting the 2D image pixel coordinates and d the measured distance in millimeters, the reprojection of p to $\mathbf{p} = (x, y, z)$, denoted as $P^{-1} : p \rightarrow \mathbf{p}$, is given by

$$\begin{aligned} x_{\mathbf{p}} &= \hat{x}_n * \delta * d \\ y_{\mathbf{p}} &= -1 * \hat{y}_n * \delta * d \\ z_{\mathbf{p}} &= -d \end{aligned} \tag{3.6}$$

with normalized pixel coordinates $p_n = (\hat{x}_n, \hat{y}_n)$

$$\begin{aligned} \hat{x}_n &= (\hat{x} - c_x)/f_x \\ \hat{y}_n &= (\hat{y} - c_y)/f_y. \end{aligned} \tag{3.7}$$

$f_{x,y}$ denote the focal length in pixels and $c_{x,y}$ the center of projection, also given in image-space coordinates. δ denotes the radial distortion factor for pixel $(\hat{x}, \hat{y})$ and radial distortion

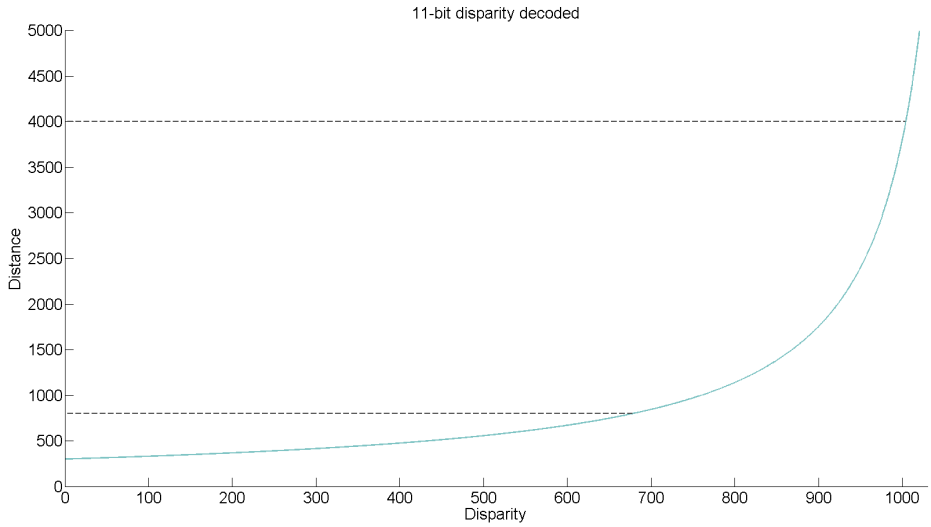


Figure 3.2: Decoded disparity values given in pixels around the reliable depth range of 800mm to 4000mm of the Microsoft Kinect^{TM}v1. Disparity is given in pixels on the x-axis. Decoded distance is represented on the y-axis and given in millimeters.

parameters k_2, k_4 and k_6 given by

$$\delta = 1 - \mathbf{k} \cdot \mathbf{r}^T, \tag{3.8}$$
$$\mathbf{k} = \begin{pmatrix} k_2 \\ k_4 \\ k_6 \end{pmatrix}, \mathbf{r} = \begin{pmatrix} \|p_n\|^2 \\ (\|p_n\|^2)^2 \\ (\|p_n\|^2)^3 \end{pmatrix}$$

Error

The error r_e of $\mathbf{p}$ is calculated as

$$r_e = max(r_d, r_n) \tag{3.9}$$

r_d is the error based on *discretization* and r_n the error based on *sensor noise*.

Discretization Error

r_d is measured as the maximum Euclidean distance in 3D space between points $\mathbf{p}$ and $\mathbf{q}_i$ of pixel p and pixels q_i corresponding to the 8-neighborhood of p, i.e.

$$r_d = max(\|\mathbf{p} - \mathbf{q}_i\|). \tag{3.10}$$

As described in Section 2.6, d_z is a special case when operating on data obtained with the Microsoft Kinect^{TM} v1 sensor, because the value at a depth measurement pixel retrieved using

the *OpenKinect*[3] driver gives depth encoded in 11-bit precision integer values d_{raw}, and to get to the real depth value, the raw data has to be decoded with

$$d = 1000.0/(d_{raw} * -0.0030711016 + 3.3309495161), \tag{3.11}$$

which gives better resolution at close distances but precision decreases with increasing distance as can be seen in Figure 3.2. The reliable depth range is reported [4] within 800 to 4000 millimeters. This can lead to higher discretization errors based on depth precision than on pixel neighborhood. Therefore, to calculate the possible discretization error for a measured depth value d, the corresponding encoded value $\hat{d}$ for d is increased by 1 and decoded again, which gives d_1. The discretization error for the given pixel is then

$$r_d = max(\|\mathbf{p} - \mathbf{q}\|, d_z) \tag{3.12}$$

$$d_z = \|d - d_1\|. \tag{3.13}$$

Sensor Noise

The error based on sensor noise r_n is calculated as a function of $\hat{x}, \hat{y}$ and d using a model for each axis provided by Choo et al. [9] in case of the Kinect v1 sensor and as a function of d as described in Fankhauser et al. [14] for the Kinect v2 device. Both use a quadratic model with estimated parameters. For the sake of completeness we want to present them here.

The noise model for the Kinect v1 is based on a quadratic surface fitted to consecutive measurements. For the z axis the model can be used directly, but for axes x and y the image is divided into 8x8 tiles and

$$\hat{x}_r = \left\lceil \frac{\hat{x}}{640/8} \right\rceil \quad \hat{y}_r = \left\lceil \frac{\hat{y}}{480/8} \right\rceil. \tag{3.14}$$

The *sensor noise* at pixel p along axis $\mathbf{a} \in \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ is then calculated with

$$f(\hat{x}, \hat{y}, d, \mathbf{a}) = \beta_{1,\mathbf{a}}\hat{x}^2 + \beta_{2,\mathbf{a}}\hat{y}^2 + \beta_{3,\mathbf{a}}d^2 + \beta_{4,\mathbf{a}}\hat{x}\hat{y} + \beta_{5,\mathbf{a}}\hat{x}d + \beta_{6,\mathbf{a}}\hat{y}d + \beta_{7,\mathbf{a}}\hat{x} + \beta_{8,\mathbf{a}}\hat{y} + \beta_{9,\mathbf{a}}d + \beta_{10,\mathbf{a}} \tag{3.15}$$

with coefficients $\beta_{i,\mathbf{a}}$ given by Table 3.1.

The model provided by Fankhauser et al. [14] for Kinect v2 only takes into account the depth component d but incorporates an angular component, which denotes the angle θ between the surface normal at pixel p and the vector pointing form p to the position of the sensor.

With d_m denoting the measured distance in meters, the *sensor noise* can be computed with

$$f(d_m, \theta) = 1.5 - 0.5d_m + 0.3d_m^2 + 0.1d_m^{\frac{3}{2}} \frac{\theta^2}{(\frac{\pi}{2} - \theta)^2}. \tag{3.16}$$

Assuming a uniform distribution of surface normals of the scanned objects, θ was used as a conservative constant with $60°$ in radians (approx. 1.047), so normals did not have to be computed. The behavior of the used noise model is depicted in Figure 3.3.

[3] https://openkinect.org

[4] https://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges

	x	y	z
β_1	1.1225e-01	6.3038e-01	2.0000e-05
β_2	6.3801e-01	2.6496e-01	2.0000e-05
β_3	3.5751e-06	1.3279e-06	1.2500e-6
β_4	4.0645e-03	1.5000e-02	2.0000e-06
β_5	0-1.4951e-04	9.0174e-05	3.5000e-09
β_6	07.0336e-05	3.3417e-04	3.5000e-09
β_7	0-5.6762e+00	-5.9320e+00	-1.0002e-02
β_8	0-8.0153e-01	-2.4411e+00	-1.002e-02
β_9	0-3.1496e-03	3.1239e-03	-1.5025e-03
β_{10}	1.2996e+01	1.0995e+01	1.4515e+00

Table 3.1: Coefficients for the quadratic model to calculate noise along the corresponding axis as described in Choo et al. [9]. Each column represents the error along the given axis.

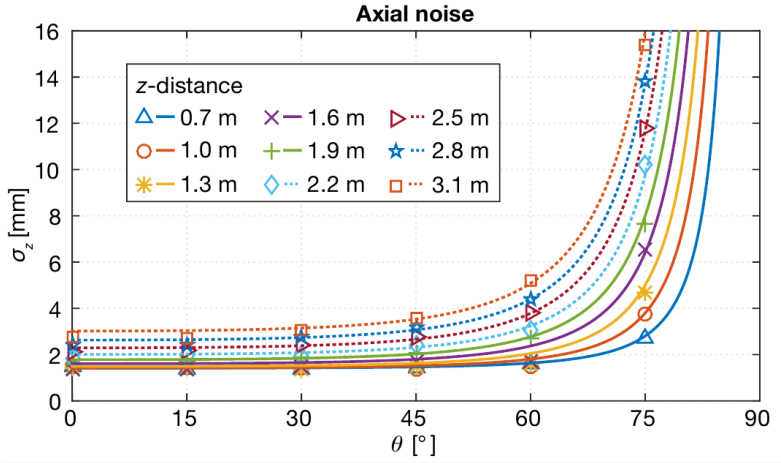


Figure 3.3: Effect of surface angle and distance on measurement error of the Microsoft Kinect™ v2. Image courtesy of Fankhauser et al. [14].

3.5 Occlusion detection

After calculating 3D position and error radius for each pixel, the next step will mark each pixel if it is part of an *occlusion boundary*. Since a range scanner can only store the distance information of the nearest object it "sees" at each pixel, all other objects, or parts of objects, behind the nearest one along the projection direction are said to be *occluded* at that pixel. *Occlusions* have to be taken into account because of two reasons, which are depicted in Figure 3.4.

The first reason, as shown in Figure 3.4 a, is to get reliable estimates for the local curvature at a surface point. Curvature is a differential quantity of a surface and *occlusion boundaries* mark discontinuities. When fitting a quadratic surface to a local neighborhood, care has to be taken to not select points for the support neighborhood of that point that lie across *occlusion boundaries*

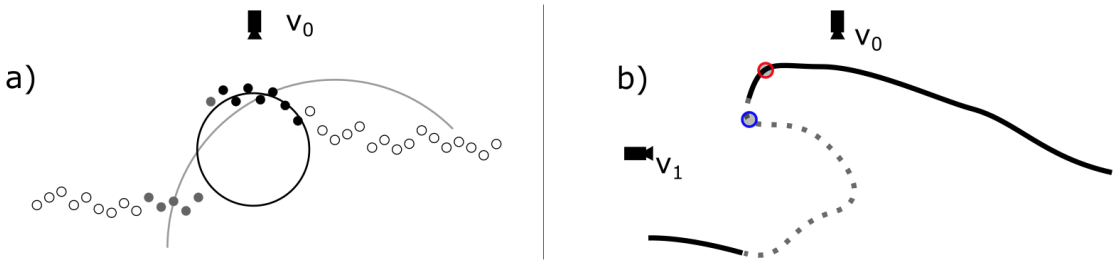


Figure 3.4: Occlusions have to be taken into account for reliable curvature estimation as depicted in a). Capture direction is depicted as v_0. Had the points in grey also been used for fitting a quadratic surface, the resulting curvature would be less reliable. Using only the points in black leads to reliable local curvature estimates. Also, as shown in b), when selecting local curvature maxima as reliable features from capture v_0, we cannot use points at occlusion boundaries since local curvature could increase from another perspective, i.e., v_1. A potential local maximum is shown in red. A surface patch that was missing during capture is depicted as a dotted line. A more reliable local extremum, shown in blue, would be visible from another perspective.

Figure 3.5: Range scan image with occlusion pixels marked in red.

and therefore do not belong to the same object or surface patch.

The second reason is that capturing the same object from multiple viewing positions may expose features that are occluded from one point of view but are visible from another. We identify features as local extrema of surface curvature. If such extrema lie on an occlusion boundary they are not reliable. This is because it is possible that curvature would increase even further along parts of the surface that are occluded.

Since a range scan is an image with the depth of the captured point as the value at each pixel, occlusion detection is implemented as a simple edge detection in screen space using a *Sobel filter* to calculate the gradient magnitude at each pixel. The image is convolved with the *Sobel kernel* to calculate first derivatives, or *gradients*, along the horizontal and vertical direction, [37]. The gradient magnitude is then thresholded using an *occlusion threshold* τ_o

mentioned in Section 3.3, which states how much neighboring pixels can differ in distance to the camera and still be assumed to belong to a continuous surface patch. This parameter has to be explicitly specified.

Formally, whether or not a pixel coordinate (x, y) lies on an object's silhouette is given by $G(x, y) > \tau_o$ for a given occlusion threshold τ_o, with G defining the gradient magnitude of each pixel of range image I_r, i.e.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I_r, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I_r$$
$$G = \sqrt{G_x^2 + G_y^2}. \tag{3.17}$$

Detected occlusions for one scan of the Stanford bunny can be seen in Figure 3.5.

3.6 Local Curvature Estimation

Since we identified local surface curvature extrema, the next step estimates local curvature. We will end up with a *curvature image* C which, for each pixel location p of the original range scan, holds the local curvature value c at p.

With occlusion pixels detected, the next step consists of estimating the local curvature at each non-occlusion pixel corresponding to a point of the scanned object, by fitting a quadratic surface to the local neighborhood of that point. We chose the sphere as an approximation of the local surface patch. This is a trade-off between accuracy and simplicity. Although techniques like the one by Cazals et al. [6] exist to fit arbitrary quadratic surfaces, the sphere geometrically better matches the object features we want to detect, like bumps and dents. Also, noise would have a bigger impact on the fitted surface when fitting arbitrary quadratic surfaces. The sphere, on the other hand, has led to a good representation of local curvature behavior.

With the error radius r_e at each point $\mathbf{p}$ known, the size of the local neighborhood $N_\mathbf{p}$ of $\mathbf{p}$ is simply given as $4 * r_e$. The factor 4 was chosen empirically and led to reliable local curvature estimates. To be able to leverage the 2D neighborhood information of the scan data, it is necessary to transform the error radius given in object space to the 2D image domain. To get the image-space equivalent of $N_\mathbf{p}$, $\hat{N}_\mathbf{p}$, a sphere S_e with radius $4 * r_e$ centered at $\mathbf{p}$ is projected to the image space of the range scan using the inverse of the re-projection defined in 3.6 with the characteristics of the scanning device as described in 3.4.

It is now necessary to remove pixels of the image-space neighborhood $\hat{N}_\mathbf{p}$ which can only be reached by crossing *occlusion boundaries* found during occlusion detection described in Section 3.5. This is achieved by applying the *floodfill* algorithm starting at pixel location p corresponding to the point $\mathbf{p}$ in object space. This yields a floodfilled neighborhood $\hat{N_{ff}}$.

Pixels in $\hat{N_{ff}}$ can still correspond to points in object space that lie outside S_e, so the sphere has to be fitted to all points

$$\mathbf{p}_i \in N_{fit} = \{\mathbf{p} \sim p \in \hat{N_{ff}} \setminus p_i, \|\mathbf{p}_i - \mathbf{p}\| < r_e\}. \tag{3.18}$$

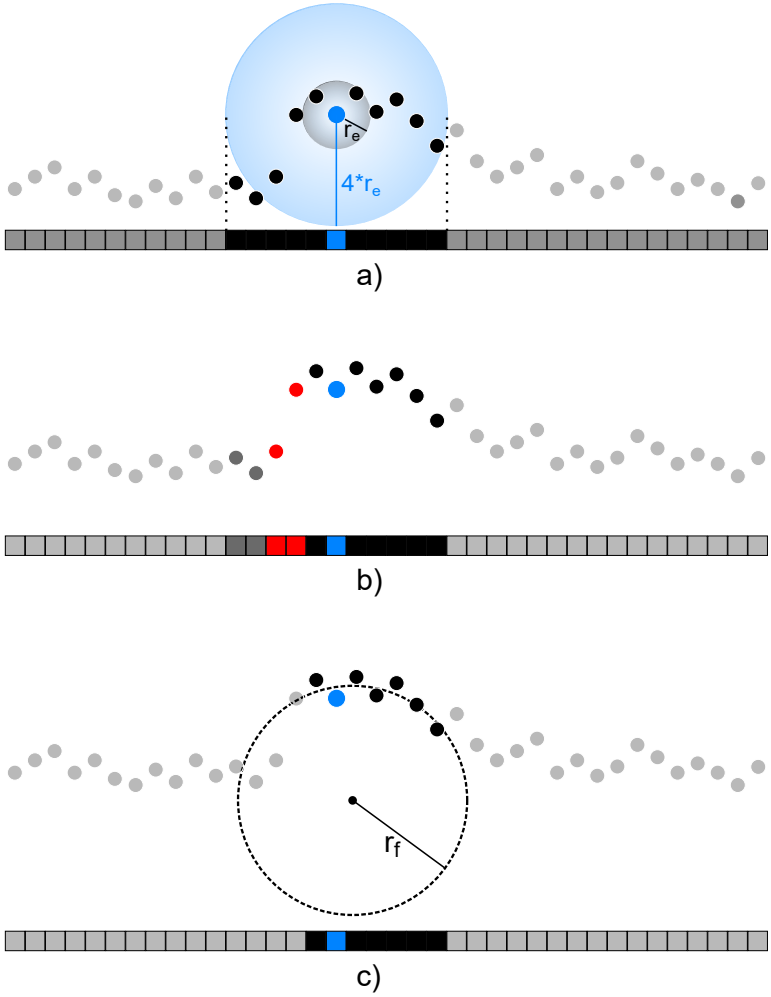


Figure 3.6: Sphere fitting procedure (assuming orthogonal projection for simplicity) for one point p (blue). The 3D neighborhood determined by the point error of p by splatting a sphere of radius $4 * r_e$ to image space (a). Floodfill with p as seed pixel is performed to filter out occlusion points (b). A sphere is fitted to the remaining neighborhood (c).

Fitting a sphere to points of N_{fit} is done using an extension of the work of Al-Sharadqah et al. [2] on algebraic circular fitting to 3D space formulated as an *Eigenproblem* that provides a smaller regression bias than previous techniques. See Prieler et al. [32] for a more detailed explanation of the method. The fitting process yields a sphere of radius r_f, so the local curvature estimate c at point **p** is given as

$$c = 1/r_f. \tag{3.19}$$

Since at least 4 points that are not co-planar are necessary to define a sphere in 3D space, neighborhoods of $|N_{fit}| < 4$ yield $c = 0$. The 2D case of the process of sphere-fitting to a local

Figure 3.7: Range scan image with estimated curvature

neighborhood of a point **p** is shown in Figure 3.6

Figure 3.7 depicts the estimated curvature for each pixel of the used range scan.

3.7 Local maximum detection

To be able to setup a global descriptor, a list of feature locations in range-scan image coordinates is extracted next. To identify salient feature points as local maxima of the curvature response, the curvature image C of the range scan image is filtered with a 3×3 kernel ω centered at p. A pixel location p is identified as a local maximum if $c(p) \equiv max(c_\omega)$.

The 2D nature of this formulation provides the possibility to fully leverage the neighborhood information inherent to the range-scan image pixels and take advantage of parallelization techniques for image processing routines, since every pixel can be processed independently of the other.

Local maxima which are located at pixels p_m of the silhouette of the object and occlusion boundaries are ignored since there is no supportive information available whether they really correspond to a local curvature maximum of the surface or not, because captured from a different perspective the curvature might still increase in the neighborhood of $\mathbf{p} \sim p_m$, as described in Section 3.5.

Since discretization artefacts have not been accounted for until now, it is possible that many local maxima will be identified based on discretization. Also, using a wide kernel during curvature estimation damps the impact of noise, but local extrema based on noise will still be apparent. Also, at a higher sampling density more local extrema can be detected since smaller details are captured. The curvature fitting kernel radius is defined in 3D space and is therefore independent of sampling density. Increasing sampling densities could, however, still introduce additional local extrema in the curvature response. These will still include the local extrema identified at lower sampling densities as long as the feature is not missed entirely during sampling. Figure 3.8 shows the effect of higher sampling density on the amount of features detected for 2 scans made of the Stanford bunny that only differ in capture distance.

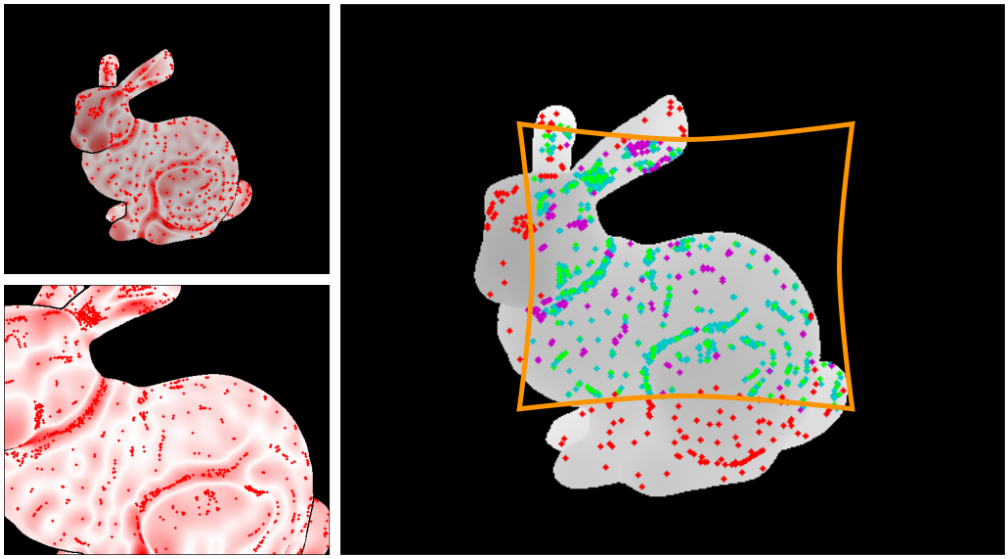


Figure 3.8: Comparison of identified local curvature extrema at different capture distances. The left side shows both scans with identified local curvature maxima as red dots. The right side shows the scan made from a higher distance (left upper subfigure) with the covered area of the closer scan (left lower subfigure) marked in orange. **Red** local extrema were identified in the farther scan but not in the close scan. Local extrema identified in the close scan but not in the farther scan are marked in **purple**. **Green** extrema depict spatially matching locations in the farther scan. **Turquoise** points mark local extrema identified in the close scan which have corresponding points in the farther scan (green). Local extrema were considered corresponding if their spatial error spheres intersect.

The local maximum detection step yields a list M of pixel locations $\{p_i\}$. In Figure 3.9, local curvature maxima of the Stanford bunny range image are marked. Until now the amount of feature points would lead to an unfeasible number of feature triples later, so the next step filters unreliable extrema.

3.8 Feature Filtering

A drawback of local curvature is that its behavior is highly sensitive to noise. Many of the local curvature maxima identified in the previous step will therefore be based on noise instead of the actual surface. To reduce the amount of point triples set up in Section 3.9, the number of features used has to be reduced. The output of the filtering step will be a reduced list of feature locations.

By using a wide kernel for fitting a spher, we reduced the impact of noise on the local curvature estimate. The effect of wider fitting kernels on estimated curvature can be seen in Figure 3.11 on the right. Curvature peaks still remain, but differences between adjacent curvature peaks and valleys based on noise are heavily reduced.

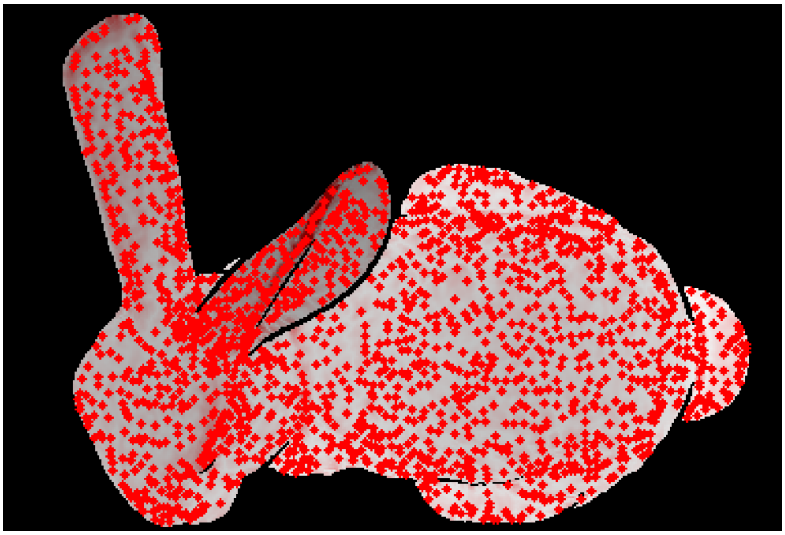


Figure 3.9: Range-scan image with marked local curvature maxima. As can be seen, noise and small geometrical details led to many more local curvature extrema being detected than would be necessary for efficient registration.

3.8.1 Curvature Scale Space – not satisfactory

To gain a resolution-independent representation of the local curvature at a surface point, Ho et al. [19] propose a *curvature scale space*. They perform curvature estimation similar to the procedure described above on neighborhoods of varying sizes (or scales) k_i, hence the term *scale space*. It is based on the idea of *scale invariant feature transform (SIFT)* to detect scale invariant salient feature points as local maxima of a *Difference of Gaussians* scale space with an image blurred with a Gaussian kernel with increasing σ, [27].

This approach was also attempted by varying the factor to calculate the size of $N_{\mathbf{p}}$. For each scale k the size of $N_{\mathbf{p}}$ is given by $k * r_e$ and the resulting curvature as $c(k, \mathbf{p})$. Broadening the fitting kernel is approximately equivalent to further smoothing the scanned surface to reduce the impact of sensor noise, so a wider fitting kernel yields a curvature response which is less perturbed by noise.

Exploiting the curvature scale space, a salient feature point is then identified as a local maximum across several scales. A point $\mathbf{p}$ refers to a feature point if a local maximum was detected at the corresponding pixel location p of point $\mathbf{p}$ at consecutive scales k_{i-1}, k_i and k_{i+1}. This approach, however, did not yield satisfactory results in terms of the number of detected feature points since local maxima were not stable enough across scales, as can be seen in Figure 3.10 for a 2D example.

3.8.2 Filtering features based on curvature confidence – not satisfactory

One approach to filtering feature locations was to look at the distribution of curvature values in a local neighborhood of pixel p of the same size as the fitting kernel, similar to the method

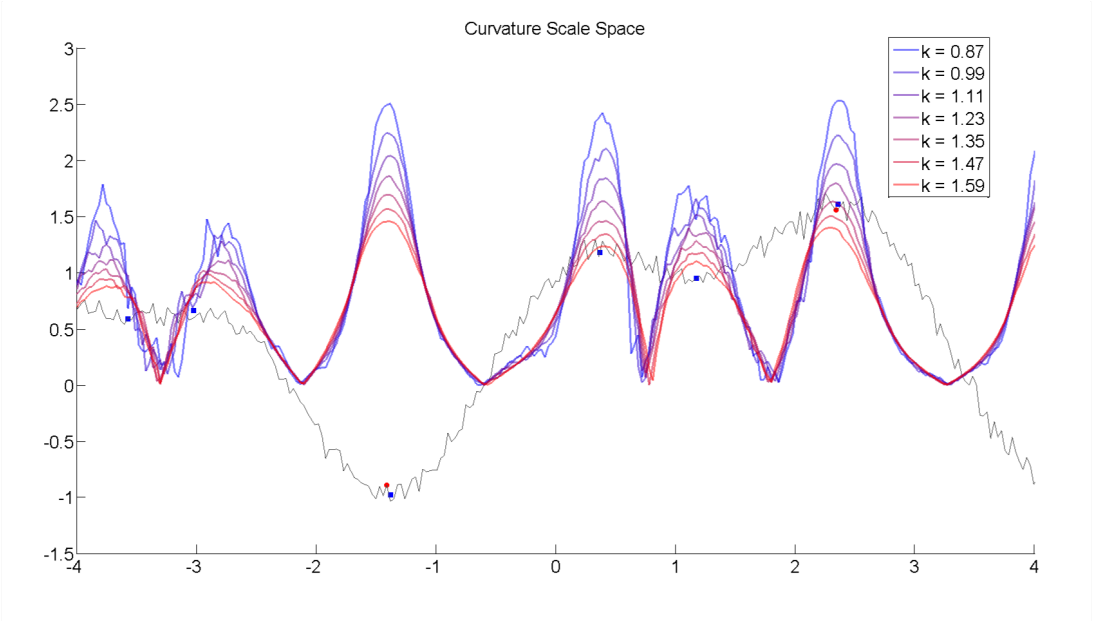


Figure 3.10: 2D visualization of feature detection based on stable local curvature scale-space maxima. Scale sizes k varied from 0.87 to 1.59, with corresponding graphs colored in blue to red respectively for better visual distinction. Ground-truth feature locations are marked as blue points, while features stable across 3 adjacent scales are marked as red points. Not only are only 2 of 6 detected but also features are detected at multiple locations.

proposed by Ho et al. [19]. The confidence in a curvature value at a pixel location p is defined as

$$\gamma(p) = \frac{|c(p) - \mu_c|}{\sigma_c}. \tag{3.20}$$

Feature locations are removed if the confidence lies below a *confidence threshold* γ_t with

$$\mu_c = \frac{1}{|N_f|} * \sum_{N_f} c(p_i) \tag{3.21}$$

$$\sigma_c = \sqrt{\frac{1}{(|N_f| - 1)} * \sum_{N_f} (c(p_i) - \mu_c)^2}. \tag{3.22}$$

Filtering feature location by confidence did not lead to good results, and instead of removing features solely based on noise and keeping features on curved surface patches it had the opposite effect. A 2D example of the effect of confidence filtering compared to the effect of filtering features that share shallow valleys can be seen in Figure 3.11. As can be seen in case of

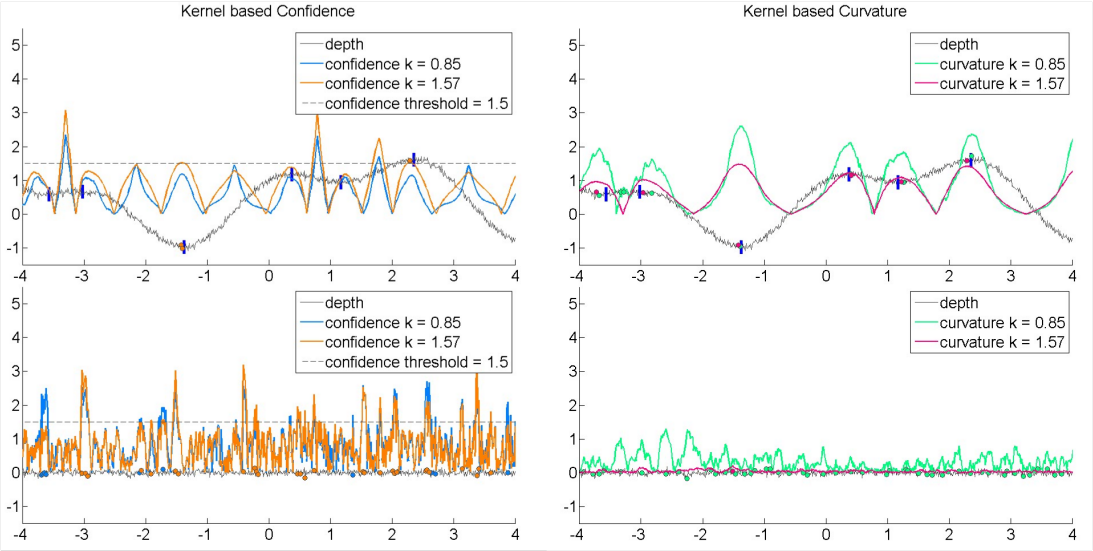


Figure 3.11: Comparison between filtering features based on curvature valleys as opposed to a confidence threshold for a 2D case. On the left side, a sinusoidal curve and a flat line, both of which have been perturbed by Gaussian noise, are shown with the corresponding curvature confidence, for two different fitting kernel sizes, $k = 0.85$ and $k = 1.57$. Features identified by confidences higher than the marked threshold are marked with dots. On the right side, the same noisy curves are shown with their corresponding curvature response for the same two kernel sizes. The identified feature locations for each kernel size using our approach are marked by dots. Ground-truth feature locations on both sides are marked with a vertical blue line. It is clear to see that features identified using a curvature valley threshold correspond much better to the ground truth and that no features were identified for the flat line. In case of the confidence threshold for the sinusoidal curve, too many features were filtered, and the flat line feature identification seems to exhibit random behavior.

zero ground-truth curvature, increasing the neighborhood size reduces the absolute response of curvature estimates but still exhibits random variation that leads to unreliable confidence peaks.

3.8.3 Filtering features based on curvature saliency – satisfactory

To get stable features, locations identified in Section 3.7 are filtered taking the curvature at $p_i \in M$ and looking for other feature locations $q \in M$ that exhibit higher local curvature, $c(q) > c(p_i)$. If q and p_i share an adjacent local curvature minimum at location p_v, with $c(p_v) > c(p_i) - \epsilon$, we call p_v a curvature valley connecting q and p_i and discard p_i from M. In this way, we focus on more salient feature locations and discard those based on minor surface perturbations and reduce the feature density around local maxima. ϵ was set empirically to 0.0001.

Additionally, a global absolute curvature threshold κ_{min} is applied to filter local curvature maxima that do not satisfy $c(p) > \kappa_{min}$.

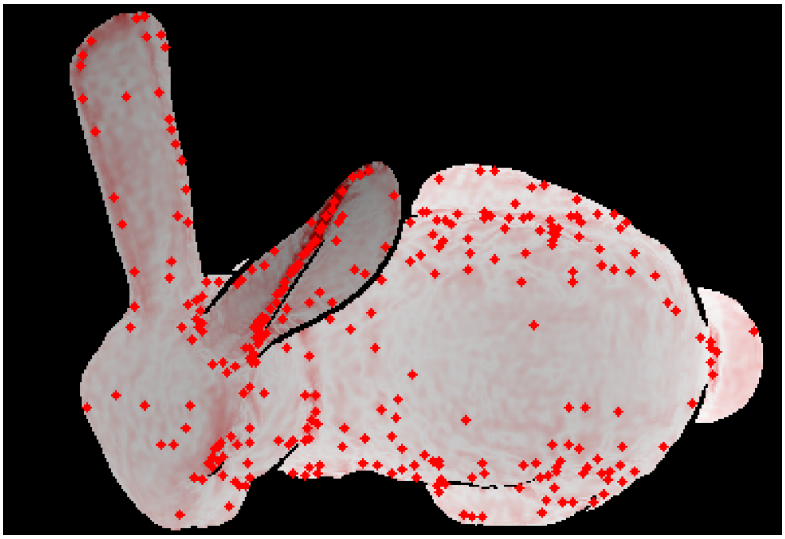


Figure 3.12: Range scan image with marked feature locations after filtering.

3.8.4 Considerations regarding the range scanner error model

Range scan images captured with Microsoft Kinect™ devices exhibit increasing sensor noise near image corners ([9, 14, 41]), so a second filtering step is performed that removes feature locations lying outside of an ellipse centered at the range image center and having its major axes of horizontal length $0.8 \times (\frac{w}{2})$ and vertical length $0.8 \times (\frac{h}{2})$, w and h being the number of columns and rows of the range scan image.

The resulting set of filtered feature locations M_f for the Stanford bunny range scan is shown in Figure 3.12.

3.9 Global Descriptor

In this step we will set up the global descriptor of point triples using the remaining feature locations in the previous step. Our algorithm produces a set of point triples which will later be used to find correspondences.

To be able to calculate a transformation that maps the points of the target range scan S_T to points of the reference range scan S_R, it is necessary to find correspondences between points of either range scan. In most cases, this is done by pointwise matching of feature descriptors like Gelfand et al. [15] or Zhong et al. [42]. Our approach however is similar to Chen et al. [7] and Aiger et al. [1] and builds a *global descriptor* which holds information of the global geometrical semantics of the range scan. It is essentially a set of *point triples* that store *minimum* and *maximum* lengths of the edges connecting each point pair of the *point triple*. We additionally take into account the error radius of the triple points. Similar to Aiger et al. [1] we will refer to a point triple used to calculate an aligning transformation as a *base*. They use more than three points to be able to calculate length ratios between two point pairs. Other than them, we follow

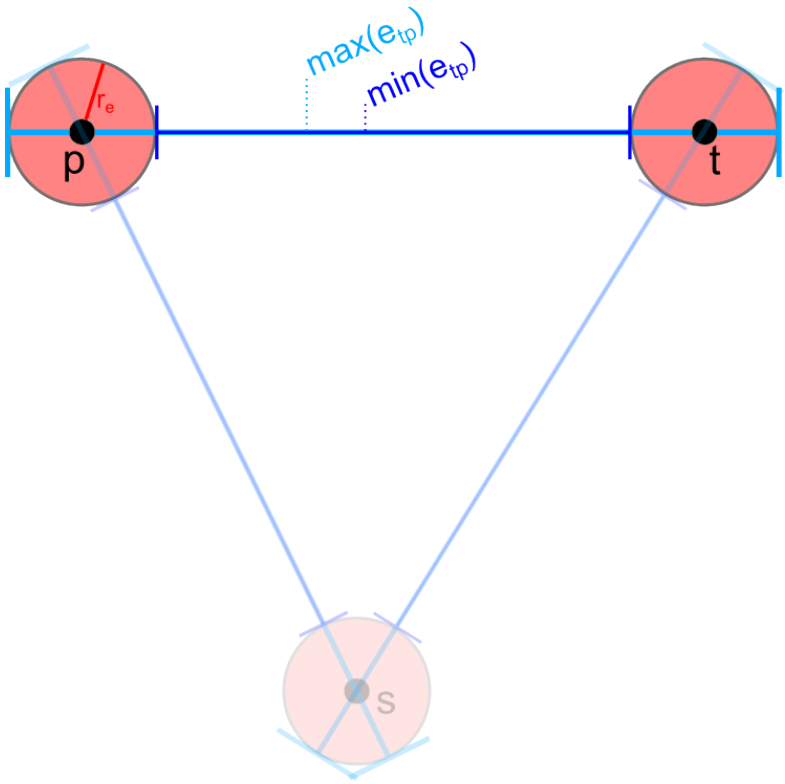


Figure 3.13: Point triple setup for points a, b and c, taking into account point errors r_e, For each edge e_{ij}, store $\|e_{ij} - (r_e(i) + r_e(j))\|$ and $\|e_{ij} + (r_e(i) + r_e(j))\|$.

the idea of Chen et al. [7] and use three points for our base. A 2D schematic of an example triple formation is depicted in Figure 3.13.

Given an estimate of the value of the diameter of the surface overlap λ_{max} of the range scans (see Section 3.3), a set of *point triples* is set up in the following way. For each point p of the set of salient features M_f, a set of pairs of its nearest neighbors is formed,

$$\{(s,t) \in N_{\lambda_{max}} \times N_{\lambda_{max}} | s \neq t\}. \tag{3.23}$$

with

$$N_{\lambda_{max}}(p) = \{p_n | \lambda_{min} <= \|p_n - p\| <= \lambda_{max}\}. \tag{3.24}$$

For efficiency, ANN (Approximate Nearest Neighbor) [3] was used for neighborhood queries in $\mathbb{R}^3$. λ_{min} is a user-defined parameter mentioned in Section 3.3 for the minimum distance between point p and its neighbors. It serves as a means of limiting the number of resulting *point triples*. It can also be dynamically set to the distance of p to its the nearest neighbor. A *point triple* t is formed of points a, b and c and edges e_{ab}, e_{bc} and e_{ca} connecting them. Additionally, for each edge we store its minimum and maximum lengths $\check{e}_{ij}$ and $\hat{e}_{ij}$, which depend on the point error of the edge endpoints $i, j \in \{a, b, c\}$. Edge-length intervals are hence calculated

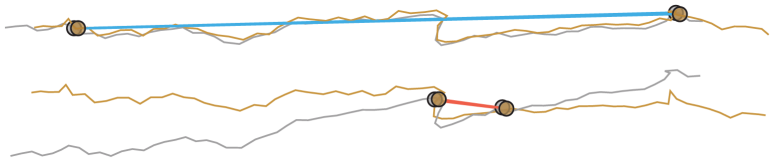


Figure 3.14: Effect on alignment stability if using a wide base for transform calculation. Dots mark the bases used on both surfaces (grey and brown). Top: a wide base is used, which leads to globally better alignment. Bottom: small bases lead to sub-optimal alignment. Image courtesy of Aiger et al. [1].

with

$$r_{ij} = r_e(i) + r_e(j) \tag{3.25}$$
$$\check{e}_{ij} = \|e_{ij}\| - r_{ij} \tag{3.26}$$
$$\hat{e}_{ij} = \|e_{ij}\| + r_{ij}. \tag{3.27}$$

As described in Goodrich et al. [17], a wider base is more stable with regard to the resulting alignment transformation as depicted in Figure 3.14. However, point pairs found with 3.23 can still lie very close to each other. It is therefore necessary to only consider those point triples which satisfy a *balanced edge length condition*. With the edge e_{ab} of the triple that exhibits the highest $\hat{e}_{ij}$, i.e. $e_{ab} = argmax_{ij}(\hat{e}_{ij})$, in a point triple, the remaining edges of the same triple e_{bc} and e_{ca} have to satisfy

$$min(|e_{bc}|, |e_{bc}|) >= |e_{ab}| * \beta \tag{3.28}$$

so that the point triple can be considered a stable base. β can either be specified by the user or be estimated from λ_{max}.

3.9.1 Global Descriptor matching

To identify global descriptor pairs that can be used to calculate an aligning transformation, the global descriptors D_T of S_T and D_R of S_R are matched by evaluating triples t_r of D_R and triples t_t of D_T with the following relaxed isometry condition

$$\sum_{t_t} \hat{e}_{ij} \geq \sum_{t_r} \check{e}_{ij}, \tag{3.29}$$
$$\sum_{t_t} \check{e}_{ij} \leq \sum_{t_r} \hat{e}_{ij} \tag{3.30}$$

which gives only a candidate. It approximately states, together with 3.28, whether the intervalls $[\check{e}_{ij}, \hat{e}_{ij}]$ of all edges of triples t_t and t_r intersect. A candidate transformation $\hat{T}$ will be evaluated using the remaining points of S_T and S_R after it is estimated.

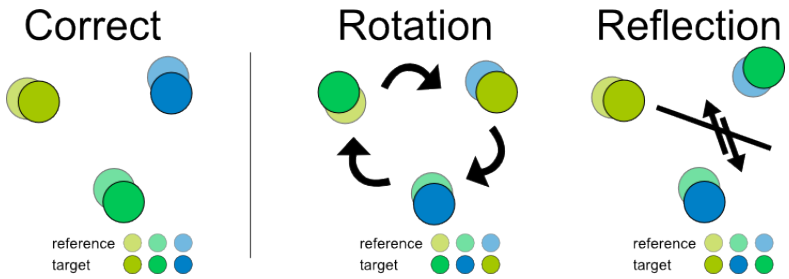


Figure 3.15: Effect of wrong triple point correspondence order. With correct correspondence shown left. Examples for a wrong correspondence order and resulting displacements are shown in the middle and right.

If a target triple t_t matches a reference triple t_r with respect to conditions 3.29 and 3.30, it is not guaranteed that points are in the same order. Assuming that the triples consist of corresponding points, a wrong order of points would still lead to a wrong transformation, perturbed by a reflection, a rotation, or both. Therefore, to ensure that the correct globally aligning transformation will be computed, transformations have to be calculated for a triple of every permutation of points a, b and c of t_t as well. Figure 3.15 shows effects of wrong point order.

Even though point triples might match based on edge lengths, we cannot tell whether they are positioned at the same location of the surface before we evaluate their alignment transformation candidate using all other points of the surface. This will be described in Section 3.11.

3.10 Transformation estimate calculation

Eventually, the aim of our algorithm is to find an aligning transformation between two point clouds. The process of estimating a candidate transformation using a pair of matching point triples is described here. The output is a 3×3 transformation matrix consisting of a rotation part and a translation part.

The extracted point correspondences between point triples t_r and t_t, as described in Section 3.9, are now used to calculate a transformation candidate T. Similar to Chen et al. [7], we follow a RANSAC-based approach by randomly selecting a triple t_r of D_R and a matching triple t_t of D_T. T has to minimize the error between triple point positions, $argmin_{\hat{T}}(\sum_t |q_t - (\hat{T} * p_t)|)$, with $q \in t_r$ and $p \in t_t$. Since range-scan data records absolute distance information, the transformation to be calculated is a combination of a rotation R and a translation T ignoring any scaling.

The optimal rotation is calculated using *Singular Value Decomposition (SVD)* of the covariance matrix of points $q_{1,2,3}$ of triple t_r and points $p_{1,2,3}$ of triple t_t. With the centroids of the triples

$$c_q = \frac{q_1 + q_2 + q_3}{3}, \quad c_p = \frac{p_1 + p_2 + p_3}{3} \tag{3.31}$$

and both triples positioned around the origin with

$$t'_r = t_r - c_q, \quad t'_t = t_t - c_p. \tag{3.32}$$

the translation component is calculated as

$$T = t'_t - R * t'_r \tag{3.33}$$

and the rotation component as

$$R = U \begin{pmatrix} 1 & & \\ & 1 & \\ & & det(UV^T) \end{pmatrix} V^T. \tag{3.34}$$

There are several other methods for approximately aligning 3D points sets. SVD was chosen because it is built into the *OpenCV* C++ libray, efficient, and in a comparison conducted in Eggert et al. [13], it proved to be the most stable of all the techniques presented in the report.

We evaluate every calculated transformation $\hat{T}$, such that points q_i of transformed triple $\hat{t}_t$, i.e. $\hat{t}_t = \hat{T} * t_t$, are at most $r_{q,i} + r_{p,i}$ away from their correspondence points p_i. So, a transformation is only considered further if the point error spheres with radii r_q and r_p intersect:

$$|q_i - p_i| < r_{q,i} + r_{p,i}. \tag{3.35}$$

If no such transformation can be found, the triple candidate t_t is discarded.

3.11 Match Verification

The last step in our algorithm verifies whether the previously calculated transformation $\hat{T}$ globally aligns the target range scan S_T to the reference scan S_R. Other than usual RANSAC approaches, we don't count the number of inliers for each candidate $\hat{T}$ and choose the one which leads to the highest number of inliers, because this would lead us to a "best fit" solution. Our output is a Boolean specifying whether a match was found or not.

We consider a transformation as globally aligning if it satisfies the relaxed *geometric consistency* criteria defined in Weise et al. [40], i.e., does not result in **Invalid Occupied Space violation** or **Invalid Free Space violation**. We test the criteria taking into account the pointwise error model extracted in Section 3.4 by evaluating every point p_t of the target range scan against the reference surface S_R and vice versa, and accept $\hat{T}$ only if no disqualified point was found. For every evaluated point we extract its *transform tolerance* and then, using it during an *surface overlay test*, evaluate for *geometric consistency*.

The transformation $\hat{T}$ calculated in the previous step is only guaranteed to optimally align the triple points which were used to estimate it. This means that it is locally optimal and the respective pointwise spatial error has high impact on the transformation. Slightly different point positions can lead to a significantly different transformation. For a globally optimal transformation, we would have had to have the solution to the correspondence problem already available,

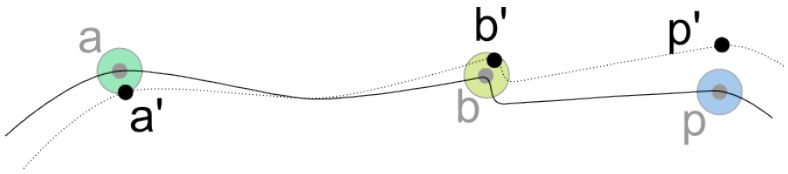


Figure 3.16: Effect of slightly different positions of points used to calculate transformation candidate $\hat{T}$. For simplicity we only show the 2D case with two points a and b used. As can be seen, point p would be displaced out of its error volume. Also the effect increases with distance to a, b.

so we could use all points to calculate the transformation. Figure 3.16 shows the possible impact of slightly different point positions on the resulting alignment.

Since $\hat{T}$ is now used to transform the target points, $p'_i = \hat{T} * p_i$, the error of transformed points p'_i is *relative* to the error of the triple points. In other words, if the triple points t_i used to calculate the transformation actually lay somewhere else within the boundaries of their error spheres, perturbed by an error vector e_i, then the transformation would be slightly different, i.e. $\hat{T}$ plus a secondary *Tilt*-transformation $\hat{T_{tilt}}$,

$$t_i \rightarrow \hat{T},$$
$$t_i + e_i \rightarrow \hat{T_{tilt}} * \hat{T}.$$

3.11.1 Transform Tolerance

As mentioned above, the estimated transformation $\hat{T}$ only minimizes the least squares error between corresponding triple points of both scans. This has to be taken into account during match evaluation. This is why, when evaluating a transformed target point p_t for correspondence, we model its error model as a tolerance volume. We chose the sphere as a simplified assumption of this tolerance volume. The concept is depicted in Figure 3.17.

In 3.1 we defined the *transform tolerance* of a point as our means to model the inaccuracy of a transformation estimate which is based on measurements which exhibit spatial error. We calculate it by tilting the point triple used for transformation estimation in three different directions and apply the resulting transformations to the evaluated target point. In this way, we find a volume in which the target point can lie if the triple position measurements differ within their error bounds.

To calculate the *transform tolerance* of p_t as the radius $\hat{r}_{tilt}$ of the transform tolerance sphere centered at p_t, we assume $\hat{T}_{tilt}$ as a rotation R that transforms the plane with normal n spun by the triangle $\{a, b, c\}$ such that the following points lie on it:

$$\begin{aligned} \hat{p_k} &= p_k + n * r_k, \\ \hat{p_l} &= p_l - n * max(r_l, r_i), \\ \hat{p_i} &= p_i - n * max(r_l, r_i) \end{aligned} \tag{3.36}$$

where $p_k \in \{a, b, c\}$, $p_l, p_i \in \{a, b, c\} \setminus p_k$ and $p_l \neq p_i$. r_i corresponds to the radius of the error sphere of the respective triple point. We calculate three such transformations R_j by assigning

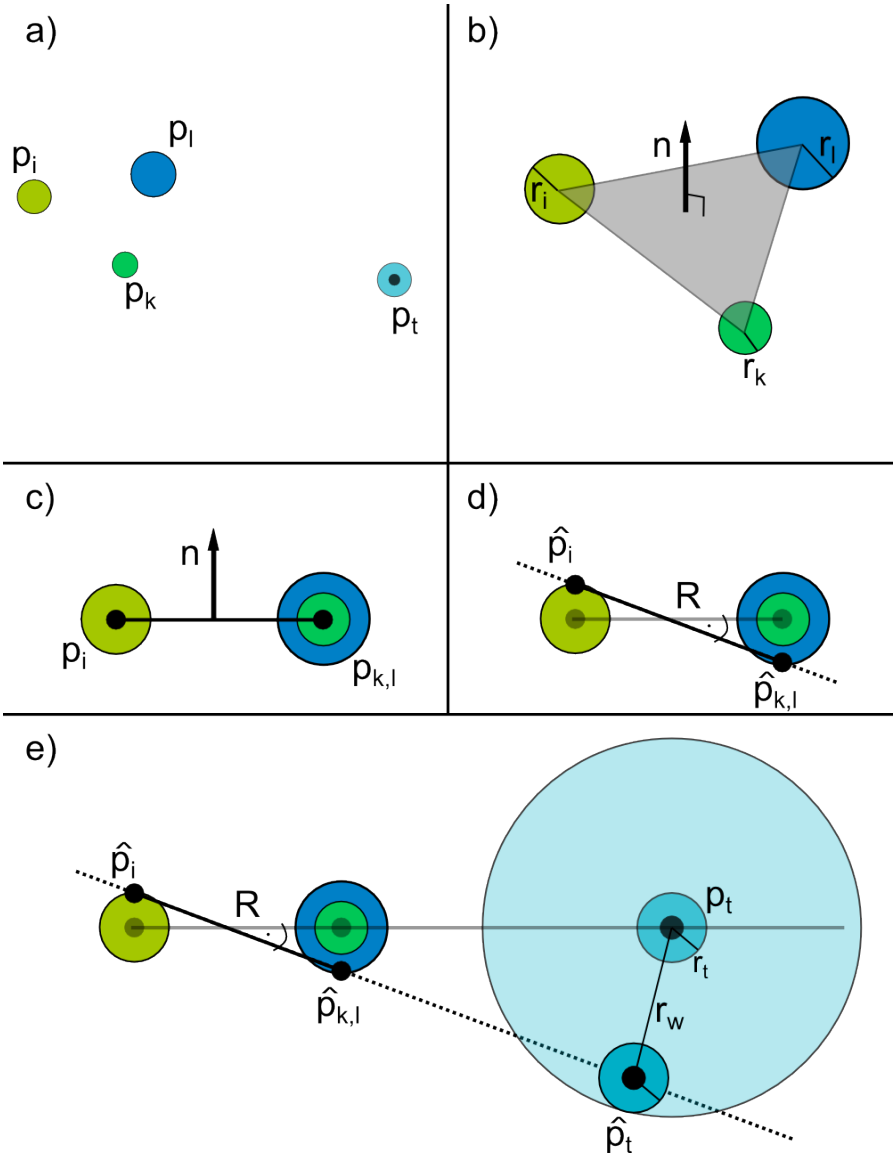


Figure 3.17: Calculating the transform tolerance from triple points $p_{\{i,k,l\}}$ for an evaluated point p_t. a) shows the three triple points used to calculate the candidate transform with circles the size of the error radii r_i, r_k and r_l together with the evaluated point p_t. b) shows the triangle with normal n spanned by the triple points. c) shows the triangle from the side to get a better understanding. d) shows how rotation R is set up such that the points $\hat{p_i} = p_i + n * r_i$, $\hat{p_k} = p_k - n * max(r_k, r_l)$ and $\hat{p_l} = p_l - n * max(r_k, r_l)$ lie on the plane of the rotated triangle. e) shows how to calculate the displacement radius r_w by applying R to p_t. Repeating this procedure for the other two triple points and their opposite edges results in transform tolerance $\hat{r}_w$ as given by $max(\{r_w\}) + r_t$.

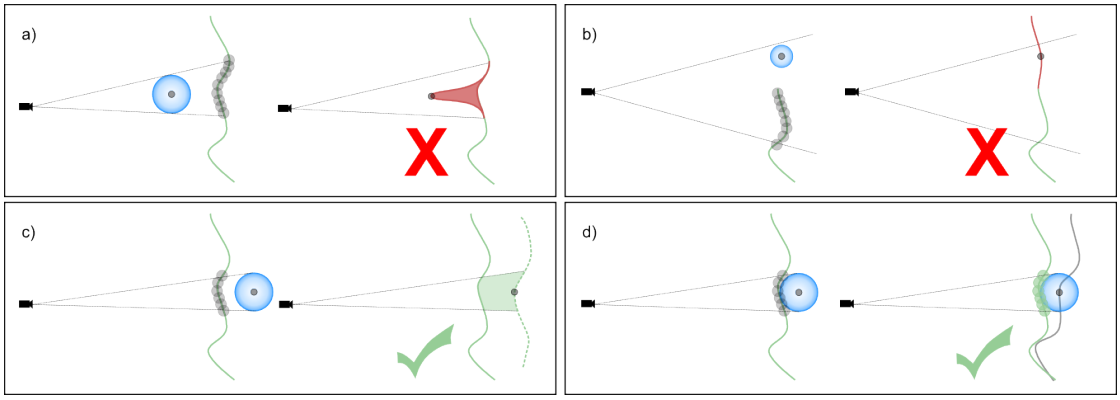


Figure 3.18: When checking for correspondences of a transformed target point $\hat{p}_t$, there are several possible situations. The *transform tolerance sphere* of the target point is shown in blue, the reference scan surface is green with points corresponding to pixels covered by the projected *tolerance sphere* with their respective error spheres shown in grey. Each sub-figure shows the observed situation during overlay testing on the left side and its interpretation on the right side. In negative cases (a, b) the incorrectly registered target surface is marked red. **(a)** shows an *Invalid Occupied Space Violation*, which basically says that a target point occludes a patch of the reference scan but its error sphere does not intersect the error sphere of any reference point. This means that if it were part of the reference surface, it would have been captured. **(b)** shows the *Invalid Free Space Violation* when a target point lies isolated within the reference scan's image space and should have therefore been captured by the scanner. In **(c)** the test cannot tell whether $\hat{p}_t$ is transformed correctly or not because it can be a point that was occluded when capturing the reference scan. Similarly, when the target point does not project into the image domain of the reference scan at all, it might simply not be part of the overlap. **(d)** shows the case in which $\hat{p}_t$ is assumed to spatially correspond to one of the reference points.

p_k to a, b and c respectively for each calculation. We set the transform tolerance radius $\hat{r}_{tilt}$ of p_t to be:

$$\hat{r}_{tilt} = argmax_j(|p_t - R_j * p_t|). \tag{3.37}$$

3.11.2 Surface Overlay Test

After the transform tolerance radius $\hat{r}_{tilt}$ has been calculated for each point p_t of the target range scan, we can use that information when verifying transformation $\hat{T}$. We evaluate every target point p_t by applying $\hat{T}$ to transform it into the reference scan's coordinate system. We disqualify $\hat{T}$ if one of the following *geometric consistency violations* can be found. See possible overlay conditions depicted in Figure 3.18.

- **Invalid Occupied Space Violation:** As depicted in Figure 3.18 a): if p_t lies perspectively in front of the reference surface but no spatial correspondence can be found for p_t because the error sphere with radius $\hat{r}_{tilt}$ centered at p_t does not intersect the error sphere of any point p_r of the reference surface.

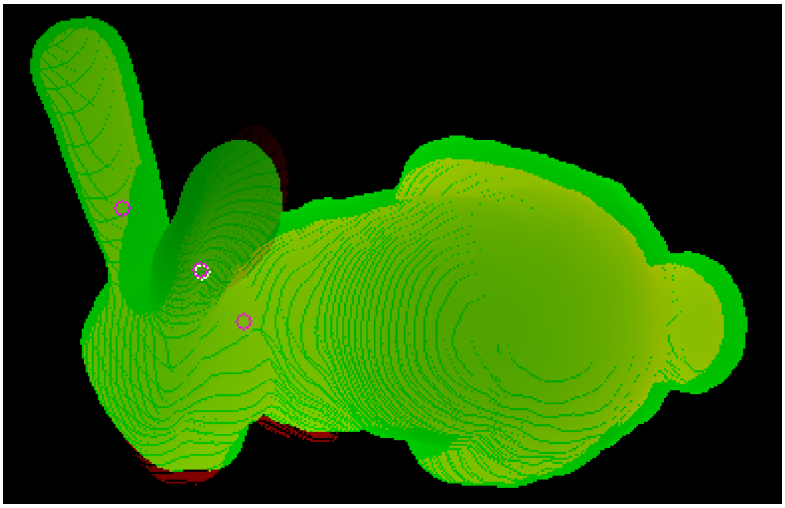


Figure 3.19: A screen-space view of the registered range scans of the Stanford bunny. The reference scan is shown in green and the target scan in red, both in the image space of the reference scan. The triple used to calculate the transformation is also marked. The alignment suffices as an initial state for fine registration with ICP and is therefore accepted.

- **Free Space Violation:** If p_t should have been captured by the scanner but again has no spatial correspondence. See Figure 3.18 b). It is important to note that for scans captured with a Microsoft Kinect^{TM} scanner, missing data in the reference scan would lead to this violation. Therefore, when dealing with Kinect scans, this check was omitted.

$\hat{T}$ cannot be disqualified if a spatial correspondence can be found for p_t, i.e., if its error sphere intersects the error sphere of at least one point p_r of the reference scan. $\hat{T}$ can also not be disqualified if the error sphere centered at p_t may not intersect any error sphere of the reference scan but is occluded be the reference scan.

If no target point met a disqualification criterion, we have to evaluate all points p_r of the reference scan with the inverse of $\hat{T}$ against the target scan. Only if we cannot find any disqualified point can $\hat{T}$ be considered an aligning transformation.

Evaluating all points $p_t \in S_T$ against all points $p_r \in S_R$ and vice versa would lead to $O(n^2)$ complexity. To reduce complexity we identify candidate points against which to evaluate by projecting the error sphere with radius $\hat{r}_{tilt}$ into the image-space of the reference scan. We then evaluate for overlay conditions only those points of the reference scan corresponding to pixels that are covered by the projected sphere.

A verified registration of the two scans for the Stanford bunny is shown in Figure 3.19 where the aligned target scan has been projected to the image-space of the reference scan.

A pseudo-implementation of the match verification step is given in Algorithm 3.1.

input : Transformed target points P',
reference points Q,
target triple t_p,
transformation $\hat{T}$
output: Boolean whether a valid match was found

```
1  for p' ← P' do
2      rw ← TransformTolerance(t_p, p')
3      SSp ← SSPixelsCovered(rw, p')
4      for q ∼ q' ← SSp do
5          if DisqualificationCheck(q, (p', rw)) then
6              return FALSE
7          end
8      end
9  end
10 return TRUE
```

Algorithm 3.1: Pseudo code for match verification

CHAPTER 4

Results

This chapter covers the evaluation of the presented algorithm. Section 4.1 presents a detailed analysis of registration capabilities of the presented algorithm. Section 4.2 describes a comparison conducted against a state-of-the-art algorithm in global registration of point clouds. In Section 4.3 the results of the evaluation will be critically discussed.

4.1 Evaluation

A special emphasis was put on the algorithm's tolerance towards *geometrical variations* between captures of the same object, as well as its matching capabilities with respect to *increasing angle difference* of capture viewpoints. For the analysis of these criteria, synthetic range scans were made to provide for a controlled environment and ground-truth information.

The presented algorithm was also compared to the *Super4-PCS* approach presented by Mellado et al. [28], to see whether it can detect a scan mismatch that other algorithms would still interpret as a transformation candidate, i.e., a false positive.

Furthermore, we applied the algorithm to scans of real scenes captured with the Microsoft Kinect™ v2 to evaluate its real-world value.

The evaluation was conducted on a desktop PC with the following specifications:

- Operating System: Microsoft Windows™10 64bit
- CPU: Intel i7™950
  - 3GHz
  - 4 physical cores
  - 8 logical cores
- Main memory: 6GB

4.1.1 Synthetic range scans

To better control the scan environment and to have ground-truth information about the view-space pose of the scanned object available, we created synthetic range scans using a system implemented in *C++* using *OpenGL*[1]. The idea was loosely based on the framework presented by Berger et al. [4], but unlike them, we did not use implicit sufarces as input.

As input the system receives a file of a 3d mesh as well as parameters which control the view-space pose of the object. The virtual camera was modelled to have a similar field of view as the Microsoft Kinect™ v2 as described in Section 2.6. For maximum precision the *near-* and *far plane* of the view frustum were positioned near the object's boundaries. The object was *rasterized* and the depth buffer was used as the output range scan.

Together with the input view-space pose parameters as ground-truth data we could use the output scan as input for our registration algorithm.

We did not model properties of the sensor like noise or lens distortions as a full simulation of a physical sensor was out of scope for this thesis. Although, as future work a full simulation as in [4], controlling the sensor noise model and object material dependent sampling, mirroring the results of evaluation work like [31] or [25] would greatly benefit the comparison to results from using real range scans.

4.1.2 Tolerance to geometrical variation

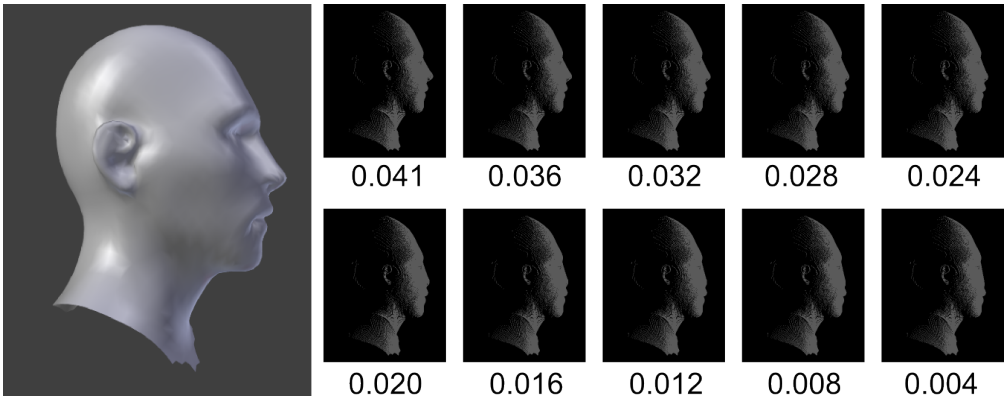


Figure 4.1: The human head model used for transform tolerance evaluation based on the Hausdorff distance of the nose tip is shown on the left. The resulting scans with shrinking nose are shown on the right. The Hausdorff distances $\hat{h}_{HD}$ of the nose tip, normalized to the extent of the model, are outlined below the respective capture image.

The *transform tolerance* concept presented in Section 3.11.1 has the potential to greatly impact the geometrical variability of the shapes being tolerated during matching. We applied our algorithm to synthetic range scans of the human head model shown in Figure 4.1. The aim was to determine what amount of geometrical differences would still be accepted as a match.

[1] `https://www.opengl.org/`

The geometrical difference was modelled using the well-known *Hausdorff distance* of a manually chosen feature of the human head model. The nose was identified as the model's most significant feature. Several versions of the human head model were created and synthetically scanned, as described above, from the same perspective. With each scan, the nose vertices were further pulled towards the nose's base plane, therefore decreasing the Hausdorff distance of the points of the nose to the base plane of the nose at each step. Figure 4.1 shows all versions of the human head model created together with their respective normalized Hausdorff distance of the nose tip. The Hausdorff distance h_{HD} was normalized against the model's extent δ_{xyz},

$$\hat{h}_{HD} = \frac{h_{HD}}{\delta_{xyz}}, \tag{4.1}$$

where $\hat{h}_{HD}$ denotes the normalized Hausdorff distance, h_{HD} the absolute Hausdorff distance and δ_{xyz} denotes the diameter of the model's axis aligned bounding box across all three axes x, y and z, defined by the two points min_{xyz} and max_{xyz},

$$\delta_{xyz} = \|max_{xyz} - min_{xyz}\| . \tag{4.2}$$

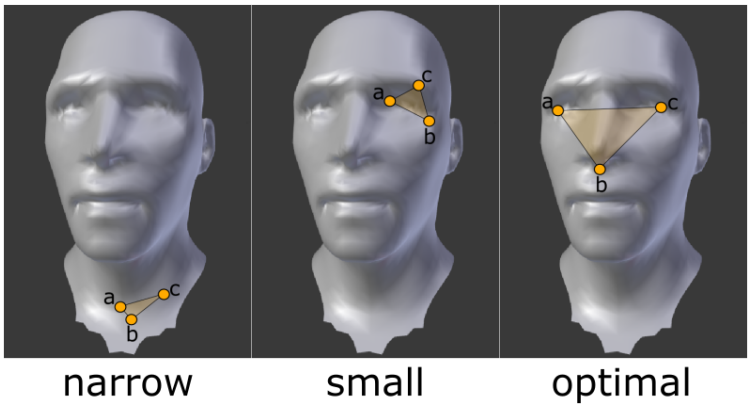


Figure 4.2: Selected feature triples in geometrical difference tolerance evaluation.

To focus the evaluation on the transformation verification step described in Section 3.11, for each perturbation of the nose, 3 fixed, manually selected, feature triples were used. This resulted in 30 runs, 10 degrees of normalized Hausdorff distance per feature triple, $\{0.041, 0.036, 0.032, 0.028, 0.024, 0.020, 0.016, 0.012, 0.008, 0.004\}$. The feature triples were chosen to differ in shape and size. They are shown in Figure 4.2.

The *narrow* triple was chosen to evaluate triples which lead to a high amount of *tilt* during transform tolerance calculation and therefore lead to a higher displacement of the target point being evaluated for correspondence. The furthest point from the feature triple being evaluated exhibited a radius of 0.083, normalized against the objects diameter.

The case that a balanced (roughly equilateral) triple is used for transform estimation is shown with the *small* triple. The *small* case also describes the situation in which a smaller overlap is

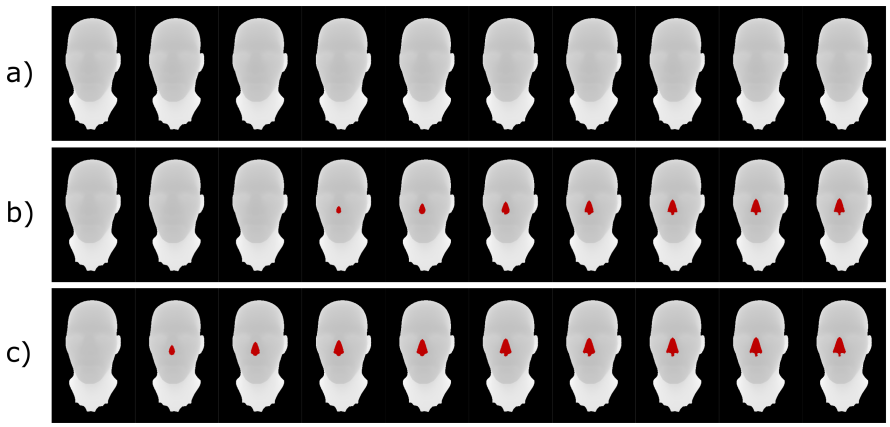


Figure 4.3: a) shows mismatching pixels in red for the *narrow* case. b) shows mismatches for the *small* case, c) for the *optimal* case.

to be expected and smaller triples have to be chosen. Here, the highest calculated normalized transform tolerance radius was 0.032.

As stated in Section 3.9, it is advisable to choose triples for transform estimation that approximately cover the diameter of the overlap area of the reference and target scan. In the case of our evaluation, the capture perspective was equal for both scans, which leads to full overlap. We therefore also chose a triple which in size is close to the diameter of the visible surface area. The *optimal* case shows a triple that was also selected by our algorithm. Here the maximum transform tolerance was 0.012.

For each triple the base scan with the flat nose was registered against all altered versions.

Results

Figure 4.3 shows mismatched points for each run in red.

Choosing the *narrow* triple led to a verified match for all scans. In the *small* case the scan with Hausdorff distance ratio of 0.016 could not be verified as a match anymore.

The only scan which could be verified as a match in the *optimal* case was the one with Hausdorff distance ratio 0.004. Starting from 0.008, no more scans could be verified.

The bigger the overlap the less tolerant our algorithm is toward geometrical variation. This means that if the amount of overlap between surfaces is known we can optimize the size of point triples and therefore the tolerance of our algorithm toward more accurate decisions regarding surface matches.

4.1.3 Robustness to angle difference

An important metric for surface registration is an algorithm's ability to match captures from arbitrary viewing angles. We analysed our algorithm regarding its registration capability on 2 scans of the same object. To be able to control the capture conditions and have ground-truth

pose data available, synthetic range scans of digital 3D models were used. Together with each scan, we stored the ground-truth information about the camera pose.

For each object, we created 25 captures c_{0-24} by rotating the camera in steps of 5 degrees around the object's origin. This resulted in a total angle coverage of 120 degrees. The distance between the virtual camera and the center of the objects was kept constant between captures at 1.7. Objects were captured with a field of view of 55 degrees, which resulted in angle coverage of the objects of 147 degrees per capture. This results in 147 degrees overlap at 0 degrees capture angle difference and leaves 27 degrees overlap remaining at 120 degrees capture angle difference.

We posed the question for how many (and which) of the 5 given objects our algorithm was able to verify a given match at increasing view angle difference between two captures of the same object.

The objects used for the evaluation are shown in Figure 4.4. The goal was to use manmade as well as non-manmade objects to see how the algorithm performs for different geometrical properties. Manmade objects often exhibit a high amount of flat surfaces, sharp edges and straight lines, whereas non-manmade objects usually do not and exhibit bumps and dents at various scales.

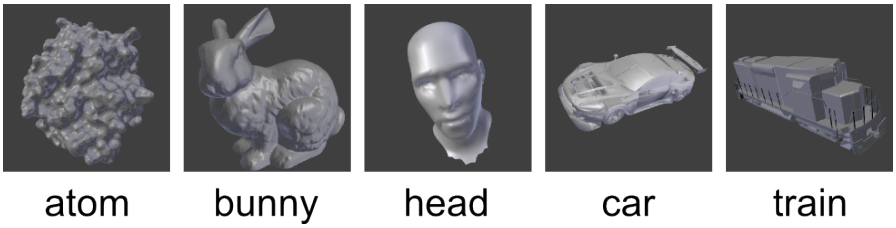


Figure 4.4: The 5 virtual objects used for view angle difference evaluation.

Objects

The *atom* object is a synthetic object and exhibits a high variation in curvature from every angle, without symmetries. Due to the overall spherical shape of its convex hull, the amount of surface area covered at captures at each angle is roughly the same, and a linear decrease in surface overlap between two captures was expected. The curvature distribution exhibited by the atom model also leads to approximately uniformly distributed salient points being detected. Figure 4.5 shows a closer look at the atom model's surface geometry, exhibiting geometric perturbations at smaller scales that are not due to noise.

The *bunny* represents a non-manmade object with curvature features at different scales, e.g., the paws on a wide scale and the small details of the fur. Geometrical variation is bigger at areas of the torso and legs, whereas the head exhibits mid-scale curvature features at the eyes and snout. Taking a close look at the furry areas, as shown in Figure 4.6 of the bunny model, one can observe the high amount of curvature of the fur.

The *human head* model has few, but significant, curvature-based features compared to the rest of the objects. Also, the features are mainly apparent on the face.

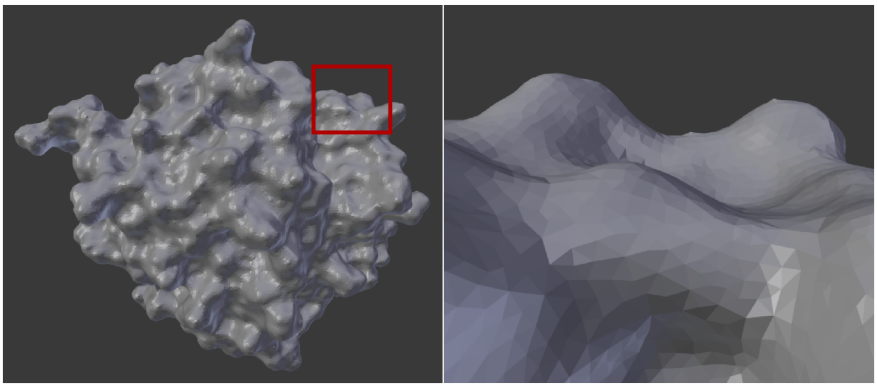


Figure 4.5: A closer look at the *atom* model used during evaluation.

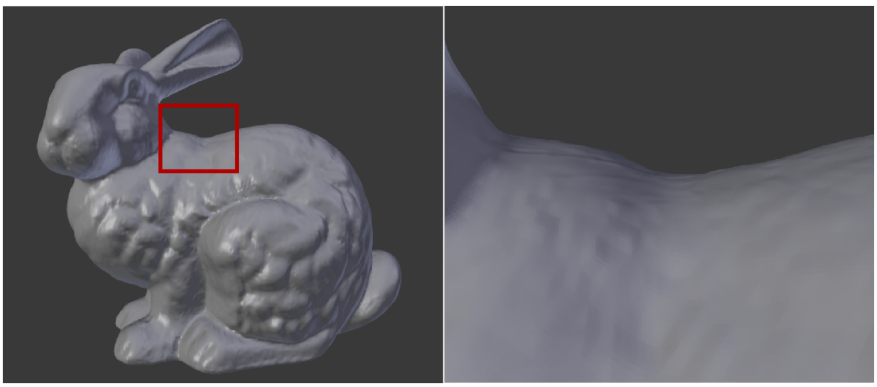


Figure 4.6: A closer look at the *bunny* model used during evaluation.

The *car* and the *train* represent the manmade objects among the used models, and exhibit many sharp edges and small details, with the train also consisting of many flat surfaces.

For each object, we made 25 runs, matching captures c_0 and c_{0-24}. Each run was limited to 2 minutes execution time on the given hardware. If it took the algorithm longer to detect a match, it was interpreted as the equivalent to having detected a mismatch.

Table 4.1 lists the parameters used during matching of scans for all models. A description of the parameters can be found in Section 3.3. The parameter values were found by experimentation and adjusted taking into account the geometrical properties of the respective object as described above.

As can be seen from Figure 4.7, from 0 to 20 degrees, which corresponds to 147 and 127 angle overlap, for all objects, a match could be verified. Starting from 35 degrees (112 degrees overlap), the match detection rate decreases with 80 degrees (67 degrees overlap) marking the first run in which no match could be verified, for any object. This results in 16 runs out of 25 yielding successful matches. Out of 16 runs, the *atom* could be verified 14 times. On average, 92 triples could be evaluated before the evaluation timed out. The *bunny* could be verified 10

-	Atom	Bunny	Human Head	Car	Train
Occlusion Threshold(mm)	0.05	0.1	0.05	0.08	0.06
Curvature absolute threshold	13.0	8.0	10.0	5.0	8.0
Maximum triple edge length (mm)	0.25	0.25	0.25	0.25	0.2
Minimum triple edge length (mm)	0.05	0.05	0.05	0.05	0.05

Table 4.1: Parameters used for each model

times. Here, 136 triples could be evaluated on average before the timeout was reached. The *head* was successfully verified 13 times, with 63 triples having been evaluated on average in cases of evaluation timeout. The *car* and *train* could be verified 8 times. For the car, 711 triples were evaluated on average before timing out. In case of the train, an average of 1444 triples could be evaluated before the timeout limit. The first expected mismatch was reported at 25 degrees difference (112 degrees overlap) for the *car* and 40 degrees difference (107 degrees overlap) for the *train*. Other than the *train*, the *car* could be matched at 55 degrees angle difference, which corresponds to 92 degrees overlap.

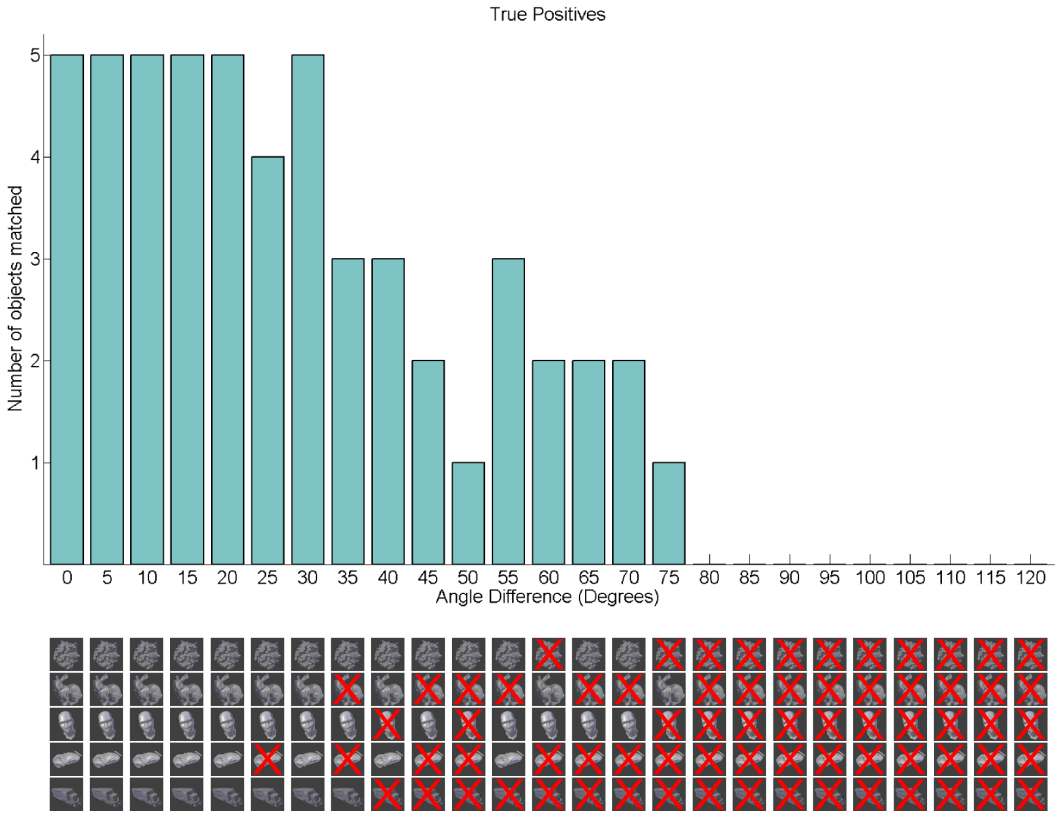


Figure 4.7: Evaluation of angle difference between viewpoints. Shows for which objects the algorithm was able to detect a match at a given viewing angle difference.

To put the above results into context, Figures 4.8 through 4.12 show feature count metrics for each scan.

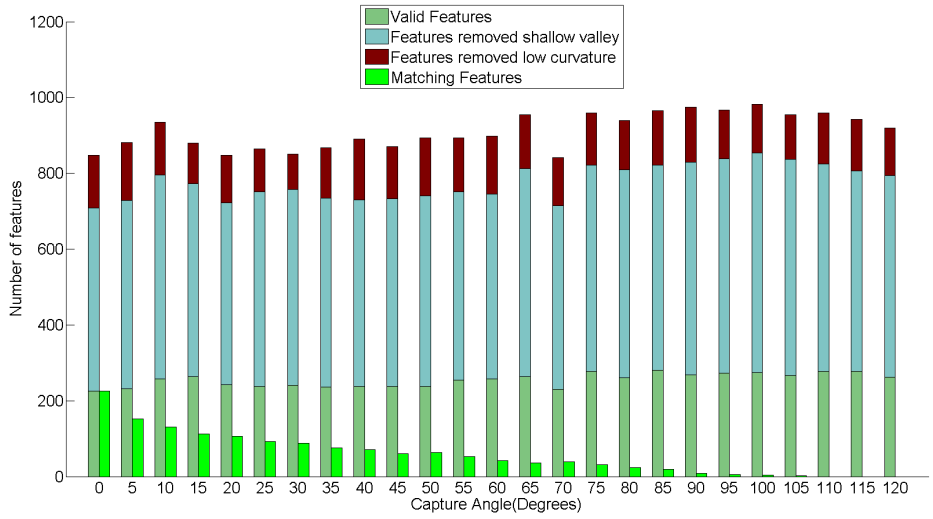


Figure 4.8: Feature counts for the *atom* model. The total amount of salient points detected averages to 911.04 over all viewing-angles. From the total amount of salient points, on average 133.8 were removed because of a curvature response below 13. From the remaining points another 522.36 were discarded due to having a point in their close vicinity that had a higher curvature response. As can be seen from the depicted ground-truth feature matches, 74 feature matches get lost when rotating the capture angle by the first 5 degrees, but after that, the amount of feature matches deteriorates slowly, and only at an angle difference of 110 degrees, no features could be matched anymore. On average 6.61 feature matches are lost at each capture step.

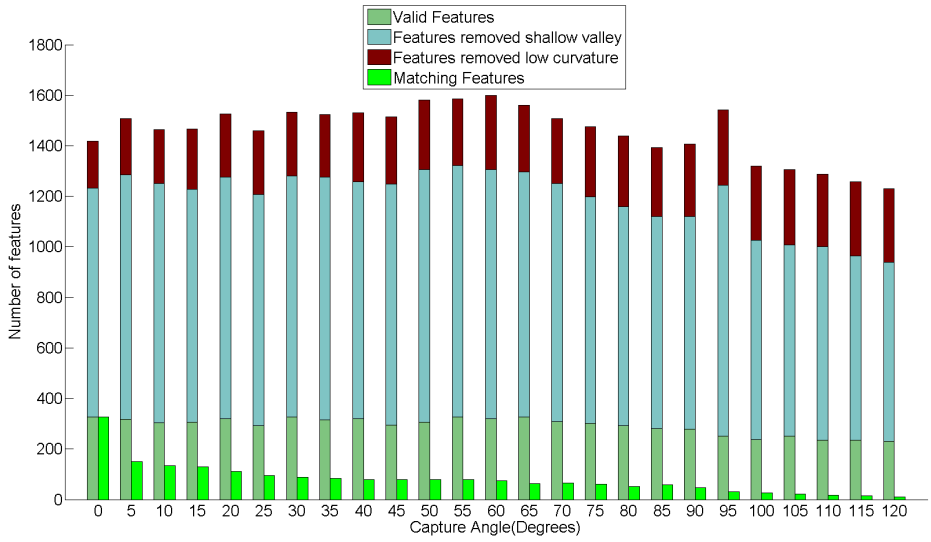


Figure 4.9: Detected features for the *bunny* model per angle difference. On average 1457.32 salient points were detected, with more being detected as long as the facial features are in view. On average 899.88 points were removed from the feature list because they had a close point with a higher curvature response, leaving between 500 and 600 salient points remaining. Points removed from consideration because of a curvature response below 8.0 were on average 265.16. At viewing-angle difference of 5 degrees, 151 features matched. At increasing angle differences, on average 6.13 feature matches were lost, however, even at an angle difference of 120 degrees, still 10 features matched under ground-truth transformation.

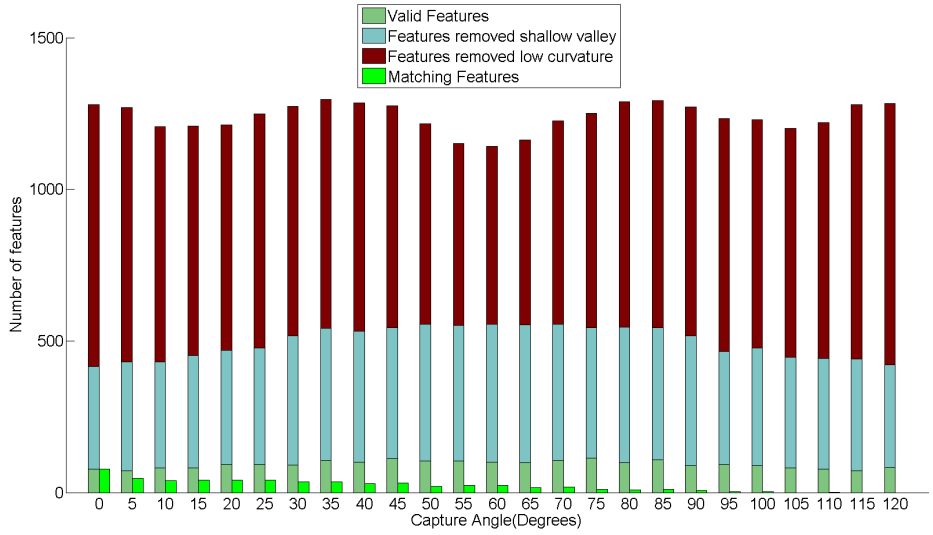


Figure 4.10: For the *human head* an aspect that is easily detectable from this plot is the symmetry of the model. On average 1240.24 salient points were detected. 403.64 points were removed from the feature list because they had a close point with a higher curvature response. Points not picked as features because of a curvature response below 10 were on average 743.56. At viewing-angle difference of 5 degrees, 46 feature locations matched. At incresing angle differences, on average 2 feature matches were lost. The first viewing-angle difference step at which no more feature locations could be matched when applying the ground-truth transformation was at 105 degrees.

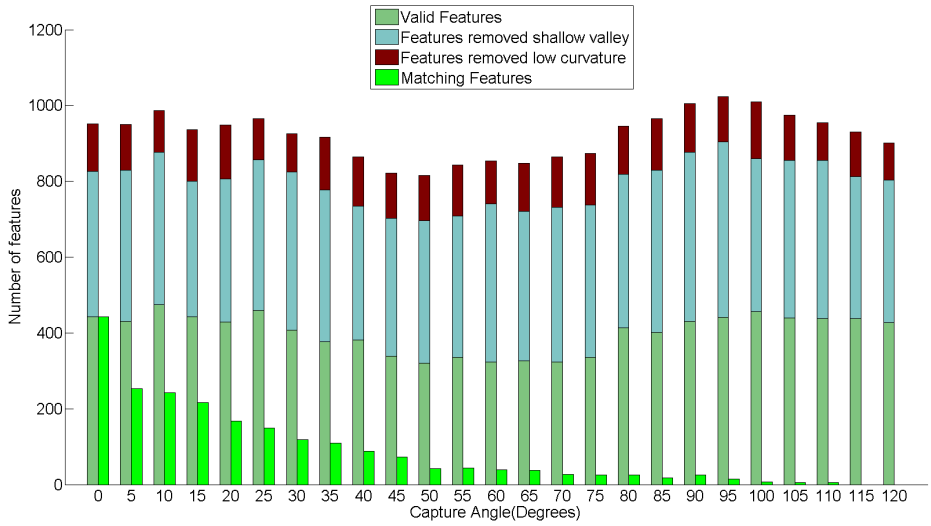


Figure 4.11: The *car* model had 922.92 salient points identified on average with a dip in the amount of salient points extracted when capturing from frontal perspectives. An average of 398.04 features were not selected because of a point of higher curvature response near. 123.52 of the average salient points were discarded based on their low curvature response of < 5. Ground-truth feature matches lost with every angle step were 11, although feature matches deteriorated faster between 5 and 50 degrees angle difference. Only at 115 degrees of capture-angle difference, no more feature locations matched.

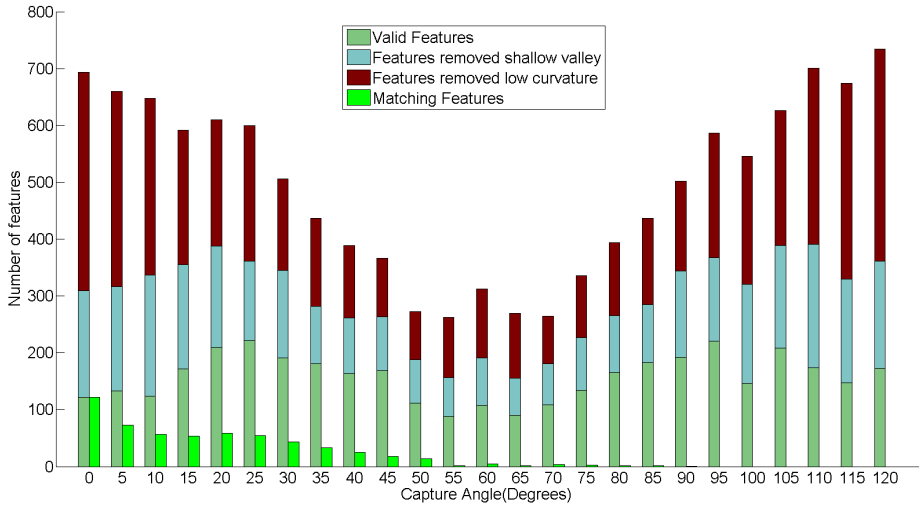


Figure 4.12: 496.76 salient points were identified for the train model, averaged over all capture poses, even though numbers varied between a maximum of 734 at 120 degrees of angle difference and a minimum of 270 salient points at 55 degrees. On average, 137 feature points were discarded because of a close point with higher curvature response. 202 salient points were taken from consideration because they had a curvature response below 8. An average of 3.17 feature matches were lost with each angle step with 95 degrees marking the spot at which no more feature could be matched.

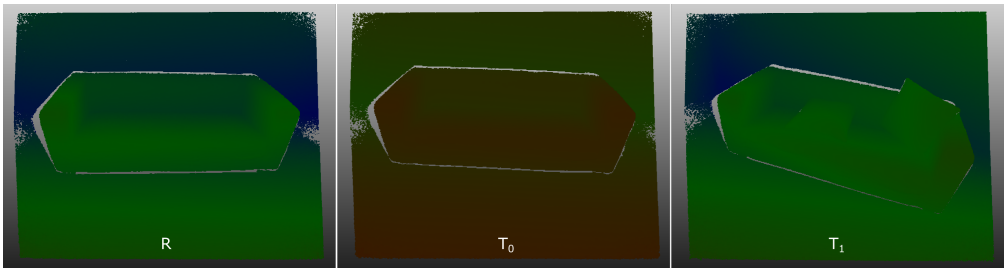


Figure 4.13: The range scans made with the Microsoft Kinect™ v2 commodity range scanner.

4.1.4 Registration of Kinect v2 scans

To show the algorithm's real-world applicability, it was applied to range scans made with the Microsoft Kinect™ v2 commodity range scanner. The main focus was on whether the algorithm was able to detect a match (*true positive*) as well as a mismatch (*true negative*). The scans used were made from a static scene from several perspectives.

The scene consisted of a couch standing close to a wall. 3 scans, R, T_0 and T_1, were made from $1,5m$ distance, as depicted in Figure 4.13. R served as the *reference* scan which both T_0 and T_1 were registered against. T_0 is a scan of the same scene from a slightly different perspective. T_1 represents a different scene in which pillows were placed upon the couch. Two questions were asked:

- Is the algorithm able to detect a match between R and T_0
- Is the algorithm able to detect a *mis*-match between R and T_1

Scene

Figure 4.14: An RGB image of the scene captured for the Microsoft Kinect™ v2 evaluation.

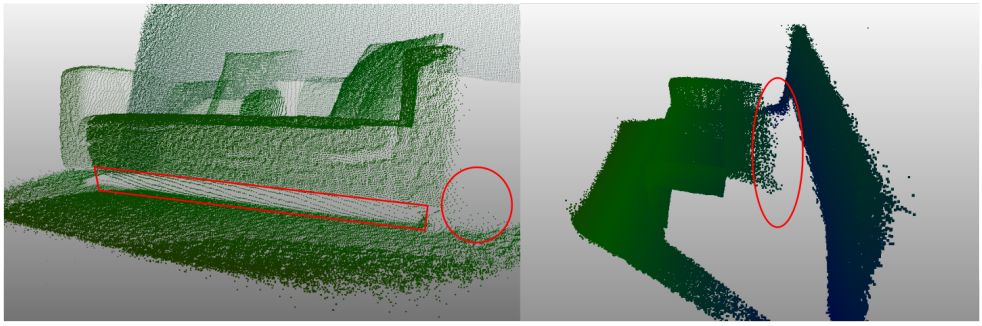


Figure 4.15: The *smear* artefact, which can happen if two objects are captured close to each other, between the floor and the lower front edge of the couch or the wall and the back rest of the couch. As can be seen, also points of the wall are displaced toward the couch.

Scans were made in artificial lighting conditions such that there could not be any sunlight which could interfere with the infrared emitted from the Kinect device. The couch measures $180 \times 85 \times 65$ centimeters. The armrests are 18 centimeters wide. The couch is completely coated with blue cotton. The coating exhibits small wrinkles and dents, which can occasionally be set off the main surface by roughly 1 centimeter. The pillows are covered in cotton as well and are colored with stripes in different shades of grey and different width. The floor consists of a slightly reflective laminate flooring. Figure 4.14 shows an RGB image of the scene.

The surface conditions are explained here because, as mentioned in Section 2.6, the quality of depth capture depends on surface characteristics such as color and reflectivity.

Scan cleanup

Scans made with the Kinect v2 scanner exhibit certain artifacts. Implementing their removal in an automated way was out of the scope of this thesis, therefore the artefacts described below were removed by hand prior to registration. The range scans were transformed into 3D point clouds. The open source software package *MeshLab*[2] was used to manipulate the point clouds and remove points based on the below artefacts. The cleaned point clouds were then tranformed back into image space to form a cleaned range scan.

The first artefact to be removed is the *smear* of depth values appearing at shallow capture angles and when two disjoint objects are only centimeters away from each other. Its effect on a range scan is depicted in Figure 4.15.

The infrared-emitter is not at the same location on the Kinect device as the infrared sensor, but is slightly displaced. This leads to a *shadowing* effect. This *depth occlusion*, at areas which the sensor can see but the emitter cannot, happens on the right side of an object of interest from the sensor's perspective, between the object's edges and the background surface (in case of the scans made, the wall). Figure 4.16 shows this effect.

[2] `http://meshlab.net`

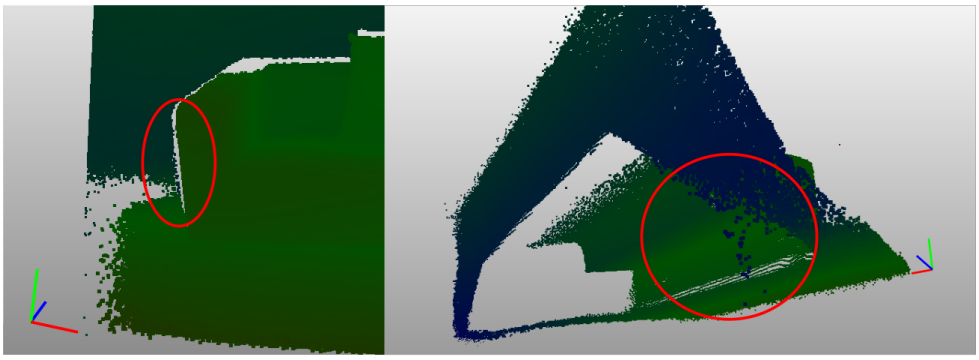


Figure 4.16: The *shadow* artefact exhibited on the right side of objects being scanned with the Microsoft Kinect v2 sensor. They come from the fact that the infrared emitter and the receiving sensor are at different locations on the device. This leads to measurements behind the object of interest, which exhibit wrong depth information in areas of infrared shadows.

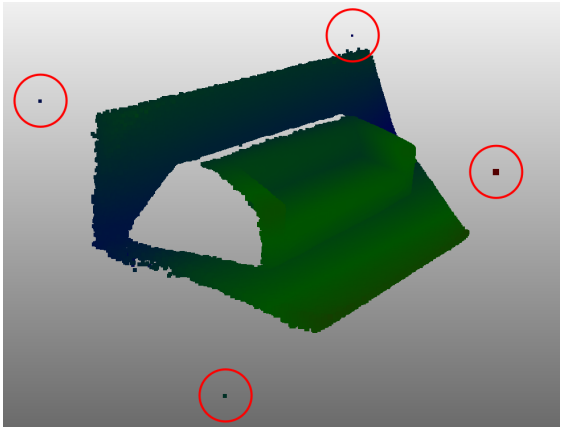


Figure 4.17: *Outliers* that represent measurement errors.

Outliers are due to measurement errors near object edges, and especially near the corners of the scan image, as shown in Figure 4.17.

Match verification results

Registering R against T_0 yielded a successful match and aligned point clouds. The coarsely aligned point clouds are shown on the left of Figure 4.18. 218 triples were selected from R and 313 were selected from T_0. The average transform tolerance when aligning T_0 to R using the verified transformation was $86.8486mm$. The average transform tolerance when verifying the reverse transformation, aligning R to T_0, was $84.148mm$. The match could be verified in $1m29s$.

Mismatch verification results

Scans R and T_1 were classified as a mismatch. Figure 4.18, on the right side, shows the points indicating the mismatch marked in red in the range-scan image of R.

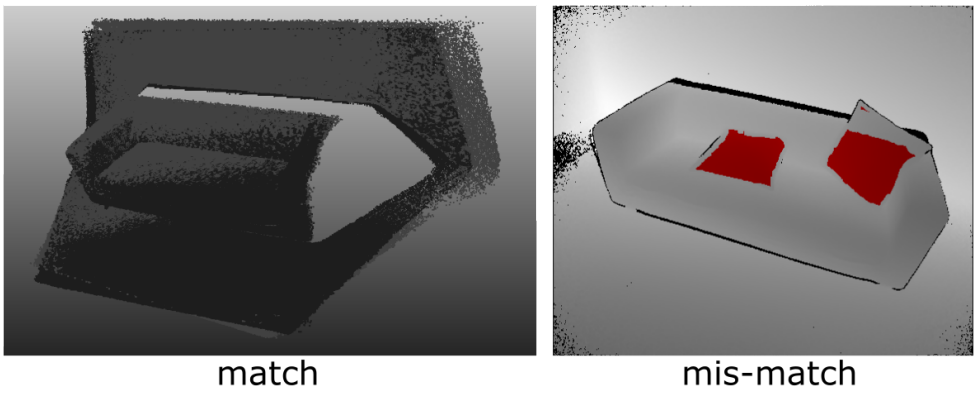


Figure 4.18: left: The coarsely aligned point clouds R (white) and T_0 (dark grey). **right:** The points indicating a mismatch between R and T_1 for a candidate transformation $\hat{T}$.

4.2 Comparison to state-of-the-art

The algorithm was also compared to the approach by Mellado et al. [28] to evaluate point-cloud matching capabilities and precision. As described in Section 2.5, their algorithm calculates multiple aligning transformation candidates and evaluates each by the *LCP* metric, based on a user-provided *delta* parameter used for pointwise correspondence search.

The conducted evaluation aimed to register synthetic range scans of the *bunny* model with geometric distortion of increasing degree against a reference scan. Synthetic scans were made as described in Section 4.1.1. The models were distorted by scaling the features of the fur, the face and in the maximum case by pulling out a "blob" of the left leg. The maximum case was considered a mis-matching surface in this evaluation. The goal was to evaluate the algorithm's capability of discerning a mismatch between two scans where other state-of-the-art algorithms report an aligning transformation as a "best fit", even though the two scans are not of the same object. Additionally, the registration quality was compared using a *root mean squared error (RMSE)* metric for the point correspondences, axis-wise Euler angle differences for the rotation part of the resulting transformations, and the *L2*-norm between translation vectors of the ground-truth transformation and the reported result.

Four scans of the *bunny*, scaled to fit into the unit cube, were made using rasterization with increasing degree of geometric distortions as described above. One scan of the undistorted model was made from a different perspective. This served as the reference scan during registration. The capture perspectives are shown in Figure 4.19. The distorted scans are shown with their color-coded *Hausdorff distance* with respect to the undistorted version of the model in Figure 4.20.

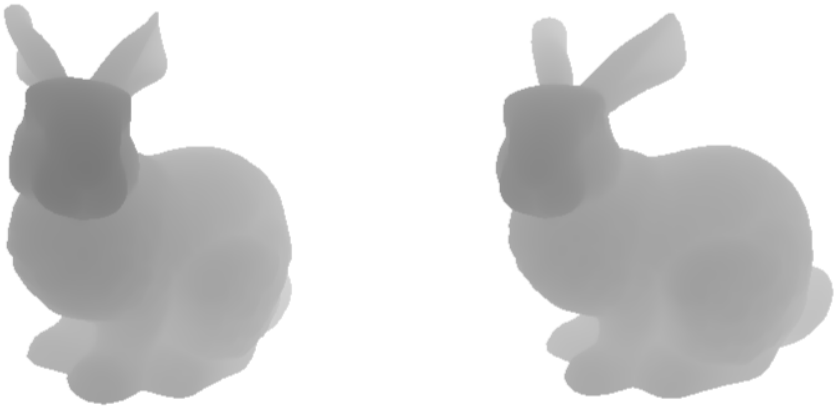


Figure 4.19: The capture perspectives used during comparison. The camera's distance to the object's center of mass was kept constant, whereas the viewing angle was altered by 15 degrees.

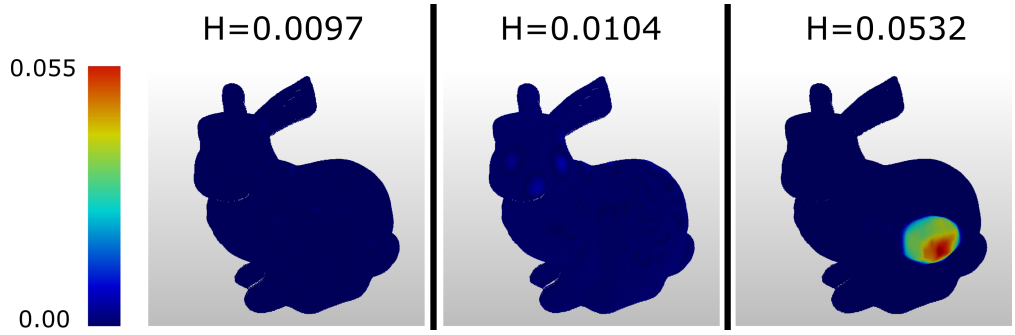


Figure 4.20: The geometrically distorted scans used for comparing the presented approach to a state-of-the-art algorithm. Color scale encodes the Hausdorff distance with respect to the undistorted scan in absolute values with blue indicating no displacement and red indicating the maximum displacement. The maximally exhibited Hausdorff distance for the respective scan is shown above as *H*.

For each of the distorted versions of the scan, both algorithms were applied to calculate an aligning transformation, if possible. Since synthetic range scans were used, the ground-truth transformation was known and could be used to measure the registration quality. The results of both algorithms for all scans used with the resulting *RMSE* color-coded are shown in Figure 4.21.

The reported transformations were also compared to the ground-truth transformation in terms of the difference in Euler rotation angles per axis and the *L2*-norm between translation vectors. The results are shown in Figure 4.22 and 4.23, for an increasing amount of distortion.

As can be seen, the resulting transformation reported by our algorithm resulted in a rotation angle difference to the ground-truth transformation of no more than 0.5 degrees, whereas Su-

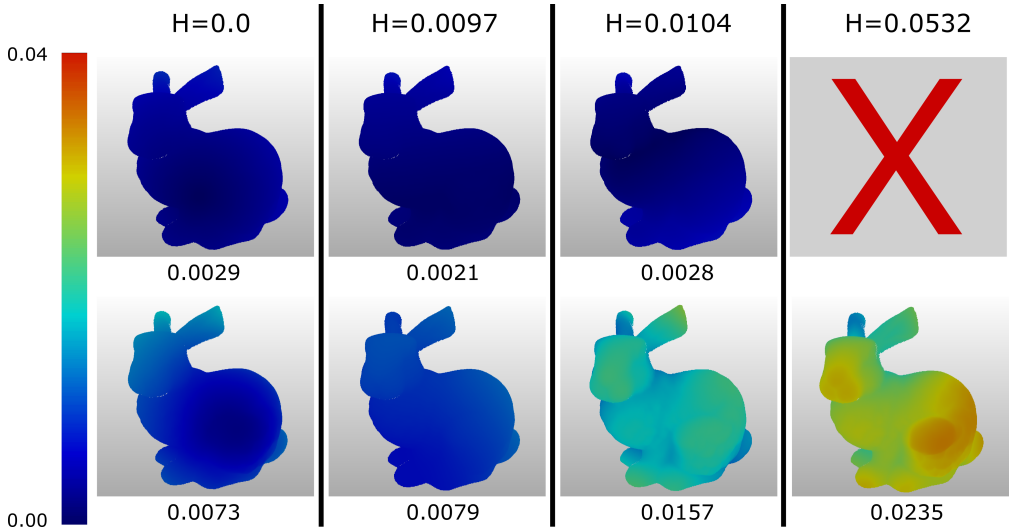


Figure 4.21: Aligned point clouds using both algorithms. The top row shows scans aligned using our algorithm with distortion degree increasing from left to right. The bottom row shows scans aligned using Super 4-PCS. We provide the maximum Hausdorff distance to the ground-truth result below the images. As can be seen, we cannot provide a result for the maximum displacement case since our algorithm detected a mismatch.

per 4-PCS produced transformations which were off by 2.182 degrees in case of the distortion of Hausdorff distance 0.532. The error in the translation for transformations reported by our algorithm lay at 0.0124 at maximum.

The error in translation in case of Super 4-PCS increased from 0.0191 in case of no distortions to 0.0805, which is almost 10% of the model's longest side, in case of Hausdorff distance 0.532.

The transformation reported by our algorithm led to higher registration quality for all experiments. *Super 4-PCS* reported an aligning transformation even though the extruded part of the bunny model exhibits a Hausdorff distance out of the tolerance interval. As depicted in Figure 4.21 by the red "X", our algorithm detected this and did not report a transformation, labeling the candidate scan a mismatch. Also no transformation differences could be reported for our algorithm, as shown in Figures 4.22 and 4.23.

It is important to note that, other than *Super 4-PCS*, our algorithm's main goal is not to report a "best match" but to verify if the two surfaces match at all at their overlap within a tolerance, with the aligning transformation as a byproduct.

Another important aspect is the runtime behavior of the compared algorithms. Other than for the evaluation reported in Section 4.1.3, we did not limit the runtime of the algorithms this time. Table 4.2 summarizes the runtimes of both algorithms for all distortion cases. *Super 4-PCS* reported an aligning transformation within 7 seconds in case of no distortion, whereas it took our algorithm 54 seconds to verify a match. For the maximum distortion case it took *Super 4-PCS* almost fourty minutes to converge at a transformation estimate. It took our algorithm more than

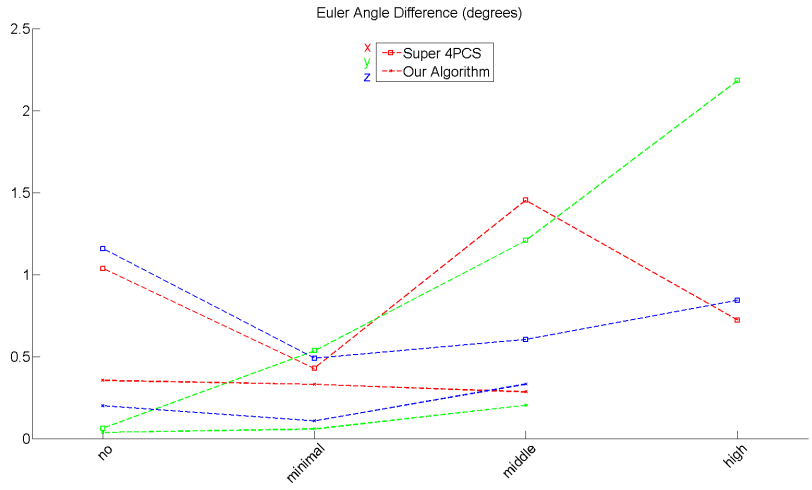


Figure 4.22: Error in rotation between our algorithm and Super 4-PCS relative to the ground-truth transformation. Errors are expressed in Euler angles around each axis, x in red, y in green and z in blue.

	Our algorithm	**Super 4-PCS**
no	54	7
$minimal$	74	7
$middle$	934	57
$high$	13245	2395

Table 4.2: Runtime of both algorithms in seconds for all distortion cases.

two hours to evaluate all transformation cadidates and not find an aligning transformation. The times for our algorithm include all steps described in Chapter 3. This includes pre-processing steps not needed by *Super 4-PCS*.

For our algorithm the most time-consuming step is *match verification* in case the surfaces match and the true aligning transformation is being evaluated. In this case, every point of the target surface is evaluated against the reference surface and vice versa. This means that, assuming that the first transformation being evaluated is in fact the true one, as was the case for no distortion, our algorithm, on the evaluation PC, would take at least 54 seconds in order to verify the match.

Since *Super 4-PCS* keeps the best aligning transformation, limitting the runtime for both algorithms to any time between 54 and 13245 seconds would give *Super 4-PCS* time to improve upon its transform estimate but limit the time for our algorithm to find *the* aligning tranformation. If this time limit runs out *Super 4-PCS* reports its best estimate whereas our algorithm reports the last transformation evaluated.

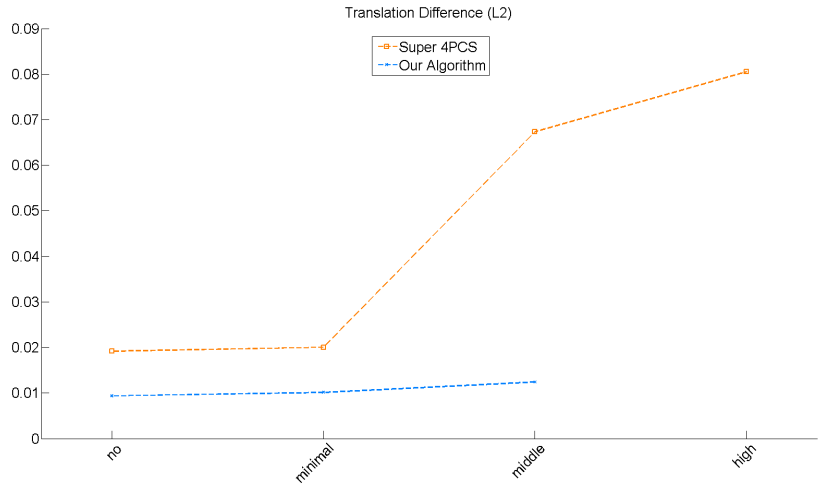


Figure 4.23: Errors in translation of aligned scans using our algorithm and Super 4-PCS compared to the ground truth. Errors are expressed as $L2$-norm between translation vectors of the transformations.

4.3 Discussion

Evaluating our algorithm's capability of tolerating transformations based on inaccurate correspondences showed its ability to provide detection of scan mismatches if the amount of geometric distortion is high. The amount of tolerance is, however, dependent on the size of the base triple used to estimate the transformation. An optimal tolerance could be achieved by finding the point triple that best approximates the surface overlap between the reference and target scan and is therefore dependent on the estimate of surface overlap by the user. In Section 4.1.2 we saw that the bigger the overlap the more accurate our algorithm's decision on a match can be since its tolerance decreases.

A big challenge is finding the right parameter values. The choice of curvature absolute minimum threshold (κ_{min}), for example, has a high impact on the number of features being used for point triple creation and therefore, on how many triples will be available to choose from for match verification. On the other hand, choosing a too high value for κ_{min} may lead to too few features being selected, in which case no correspondences can be found. Another parameter with high impact on the quality of the aligning transformation, as well as the computation time needed to find it, is the maximum triple edge length (λ_{max}). Choosing a too high value, bigger than the actual overlap between scans, will lead to a mis-match, whereas a too small number would lead to too many triples and very inaccurate alignment, as discussed in Section 3.9. An optimisation would be to start with a very high value for λ_{max} and, if no match can be found, decreasing it iteratively until either λ_{min} or a different minimum value is reached. Aiger et al. propose a similar approach in [1].

Following Chen et al. [7], leveraging the spatial relation between base points forming a triple proved to be sufficient for correspondence search. However, extracting local surface descriptors

like Gelfand et al. [15] or Zong et al. [42] could further improve the process of finding correspondences if the additional computation cost for extracting descriptors can be invested.

By incorporating the uncertainty into the accuracy of the transformation, a better model of the alignment error can be applied during transformation verification.

Section 4.1.3 showed the algorithm's weakness in case of man-made objects that exhibit a high amount of flat surfaces, straight lines and sharp edges. Here, the curvature-based feature identification struggles with finding reliable salient points. In the case of predominantly flat surfaces, an approach like Mellado et al. [28], without prior surface characteristic extraction, could lead to better results. It would apply specifically well to flat surfaces since it uses coplanar 4-points as a base.

In case of scans made with commodity hardware, like the *Microsoft Kinect v2*, many artefacts, as shown in Section 4.1.4, are introduced that cannot be modelled with a non-stochastic alignment evaluation as the one presented in this thesis. Without prior removal of outliers, however, a stochastic analysis like *LCP* is better suited to model the quality of an aligning transformation. When considering only noise, the error model implementation as described in Section 3.4 provides more information on the pointwise inaccuracy such that the tolerance can be calculated in a contextually aware way.

As shown in the presented comparison to the approach by Mellado et al. [28], our algorithm not only provides an aligning transformation in cases where this is possible, but it also evaluates the match of the scan pair such that scans captured of different objects can be discarded.

A weakness of the presented algorithm is the RANSAC-based approach of exhaustively searching all transformation candidates needed to conclude with a result in case a scan pair is a mismatch. In the conducted experiments, a timeout parameter was needed to stop further evaluation of a scan pair and interpret a timeout as a mismatch. This is due to the fact that the presented algorithm is still primarily aimed at finding an aligning transformation but using the match verification as a possibility of aborting the evaluation of one transformation candidate early. Additionally, no computation time has to be invested in fine registration if two scans are not matching.

CHAPTER 5

Conclusion

Starting with a summary of the main topics covered in Section 5.1, we provide a short coverage of our global registration algorithm that highlights the lessons learned during implementation. In Section 5.2 we provide an outlook on possible future work which can be conducted in order to improve the algorithm.

5.1 Synopsis

The aim of our algorithm is to provide global registration of range images made with a commodity range scanner, e.g., Microsoft Kinect^{TM}. Other than previous work, we additionally wanted to provide a decision about whether or not a scan is considered a match. We introduced the concept of *transform tolerance*, which incorporates the inaccuracy of an estimated transformation into the transformation verification step. The output of our algorithm is then either a roughly aligning transformation or an indication that the geometric differences between the registered scans exceed a tolerance interval that is based on the scanner's measurement error model.

We first reproject the pixels of the range scan back into the 3D domain and introduce the sensor's error model per 3D point. This gives us the possibility to introduce the error model also in our global descriptor.

The extraction of the *occlusion boundaries* is a step that leverages the 2D nature of range scan images. We used an edge-detection kernel to identify the points on occlusion boundaries. These points are then considered in the step of *local curvature estimation* to discard points in our quadratic surface fitting approach that lie across an occlusion boundary and would lead to unreliable curvature estimates, as noted in Section 3.6.

We build our global descriptor from *local curvature maxima*, which are filtered using a *high-pass filter*. We build *point triples* that incorporate the sensor error model. Using the triples on both scans, we find correspondences using isometry and rigidity constraints. Matching triples are used to estimate a 6DOF transformation using SVD.

Transformation verification is done by approximating the effect of the inaccurate transformation as a *transform tolerance* for all evaluated points of the aligned point clouds, therefore

leveraging the sensor error model during matching. All points are evaluated against *Occupied and Free Space violation* criteria, and if a point not satisfying these criteria is found, we discard the transformation and proceed with the next candidate.

The output of our algorithm is either an aligning transformation or the conclusion that the two range scans do not depict the same object.

5.2 Limitations and Future Work

Although promising results were produced, as presented in Chapter 4, many improvements to better handle different scenarios can be made to our algorithm. An important point to note is the algorithm's inability to handle artefacts apparent in range scans made with commodity hardware, e.g., as outlined in Section 4.1.4, *outliers*, *depth smear* and *depth shadows*. A fully automatic approach would greatly benefit the algorithm's usage in real-world scenarios.

The detection of a mismatch is currently done by exhaustively evaluating the estimated transformation candidates and concluding with a mismatch only if no aligning transformation could be found. We applied an execution time limit empirically, but for a decisive evaluation, all transformation candidates would need to be evaluated.

The spherical regression approach for curvature estimation leads to good results for non-manmade objects, but objects that exhibit a high amount of corners, sharp edges and flat surfaces pose a problem to our algorithm. Here, approaches like the one by Sipiran et al. [36] lead to more reliable features.

A possibility to completely rid our algorithm of the necessity to identify salient points and focus on the transformation evaluation would be to combine our error model with the 4-point correspondence search in Mellado et al. [28] and apply an adapted approach for computing the *transform tolerance* to 4-point bases during match evaluation. In this way, our proposed method of match evaluation can improve upon the output of Mellado et al. and can provide a decision of whether their best match is a match at all.

If performance is a focus, parts of our algorithm could leverage the highly parallel computing capabilities of today's GPUs. The curvature estimation would be one of the candidates for implementing such a general purpose GPU (GPGPU) approach.

Bibliography

[1] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM Transactions on Graphics (TOG)*, volume 27, page 85, 2008.

[2] Ali Al-Sharadqah, Nikolai Chernov, et al. Error analysis for circle fitting algorithms. *Electronic Journal of Statistics*, 3:886–911, 2009.

[3] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[4] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):20, 2013.

[5] Paul J Besl, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[6] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005.

[7] Chu-Song Chen, Yi-Ping Hung, and Jen-Bo Cheng. Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(11):1229–1234, 1999.

[8] Xiaobai Chen, Abulhair Saparov, Bill Pang, and Thomas Funkhouser. Schelling points on 3d surface meshes. *ACM Transactions on Graphics (TOG)*, 31(4):29, 2012.

[9] Benjamin Choo, Michael Landau, Michael DeVore, and Peter A Beling. Statistical analysis-based error models for the microsoft kinecttm depth sensor. *Sensors*, 14(9):17430–17450, 2014.

[10] Yago Díez, Ferran Roure, Xavier Lladó, and Joaquim Salvi. A qualitative review on 3d coarse registration methods. *ACM Computing Surveys (CSUR)*, 47(3):45, 2015.

[11] Santiago Díez Donoso, Joan Martí Bonmatí, and Joaquim Salvi. Hierarchical normal space sampling to speed up point cloud coarse matching. © *Pattern Recognition Letters, 2012, vol. 33, núm. 16, p. 2127-2133*, 2012.

[12] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[13] D.W. Eggert, A. Lorusso, and R.B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5):272–290, 1997.

[14] Péter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 388–394. IEEE, 2015.

[15] Natasha Gelfand, Niloy J Mitra, Leonidas J Guibas, and Helmut Pottmann. Robust global registration. In *Symposium on geometry processing*, volume 2, page 5, 2005.

[16] Guy Godin, Denis Laurendeau, and Robert Bergevin. A method for the registration of attributed range images. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 179–186. IEEE, 2001.

[17] Michael T Goodrich, Joseph SB Mitchell, and Mark W Orletsky. Practical methods for approximate geometric pattern matching under rigid motions:(preliminary version). In *Proceedings of the tenth annual symposium on Computational geometry*, pages 103–112. ACM, 1994.

[18] Michael Greenspan and Guy Godin. A nearest neighbor method for efficient icp. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 161–168. IEEE, 2001.

[19] Huy Tho Ho and Danny Gibbins. Curvature-based approach for multi-scale feature extraction from 3d meshes and unstructured point clouds. *IET computer vision*, 3:201–212, 2009.

[20] Berthold Horn. *Robot vision*. MIT press, 1986.

[21] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Transactions on Graphics (TOG)*, 25(3):569–578, 2006.

[22] Yani Ioannou, Babak Taati, Robin Harrap, and Michael Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 501–508. IEEE, 2012.

[23] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[24] Timothée Jost and Heinz Hugli. A multi-resolution scheme icp algorithm for fast shape registration. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 540–543. IEEE, 2002.

[25] Kourosh Khoshelham. Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, page W12, 2011.

[26] PLC Point Cloud Library. https://pointclouds.org. `https://pointclouds.org`. Accessed: 2016-09-30.

[27] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, nov 2004.

[28] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library, 2014.

[29] MeshLab. https://www.meshlab.net. `https://www.meshlab.net`. Accessed: 2017-05-30.

[30] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[31] Diana Pagliari and Livio Pinto. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. *Sensors*, 15(11):27569–27589, 2015.

[32] D Prieler. Local reconstruction using isotropically fair neighborhoods. diploma thesis, University of Technology Vienna, 2016.

[33] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference o*, pages 145–152, 2001.

[34] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, 2007.

[35] John Sell and Patrick O'Connor. The xbox one system on a chip and kinect sensor. *IEEE Micro*, 34(2):44–53, 2014.

[36] Ivan Sipiran and Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963–976, 2011.

[37] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[38] Georgios Stavropoulos, Panagiotis Moschonas, Konstantinos Moustakas, Dimitrios Tzovaras, and Michael Gerassimos Strintzis. 3-d model search and retrieval from range images using salient features. *Multimedia, IEEE Transactions on*, 12(7):692–704, 2010.

[39] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[40] Thibaut Weise, Bastian Leibe, and Luc Van Gool. Accurate and robust registration for in-hand modeling. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[41] Lin Yang, Longyu Zhang, Haiwei Dong, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. Evaluating and improving the depth accuracy of kinect for windows v2. *Sensors Journal, IEEE*, 15(8):4275–4285, 2015.

[42] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 689–696. IEEE, 2009.