Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Unified Pipeline for 3D Reconstruction from RGB-D Images using Coloured Truncated Signed Distance Fields

Miroslava Slavcheva

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Unified Pipeline for 3D Reconstruction from RGB-D Images using Coloured Truncated Signed Distance Fields

Einheitliche Methode zur 3D-Rekonstruktion von RGB-D Bildern mittels Farbiger Gestutzter Vorzeichenbehafteter Distanzfelder

Author:	Miroslava Slavcheva
1st examiner:	PD Dr. Slobodan Ilic
2nd examiner:	Prof. Dr. Nassir Navab
Assistant advisor(s):	Wadim Kehl
Thesis handed in on:	Friday, February 13th, 2015

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

February 13th, 2015 Miroslava Slavcheva

Acknowledgments

This work has been a memorable journey, which would not have been possible without the help of a number of remarkable people.

I would first like to thank Prof. Dr. Nassir Navab for providing me with the opportunity to accomplish my thesis at the Chair of Computer Aided Medical Procedures at TUM, as well as the industrial collaborators from Siemens CT RTC SET INT-DE for granting access to their facilities.

Next is a tribute to the two figures who never ceased to inspire and motivate me during the past months. My most sincere gratitude goes to Slobodan for all the support, care, guidance and easy-going discussions. Above all, my infinite thankfulness is devoted to Wadim for all his knowledge, advice, patience, dedicated time, provocative ways to stimulate pushing the boundaries of research, and simply everything.

Thanks also to all the people at the CAMP chair for being such an aspiration through their great work. Special credit to David for recording the sequences, to Adrian for his assistance in the visualisations and for sharing the struggle towards thesis perfection, to Tolga for the influential personality and encouraging pep talks, and to Fausto for being a creative catalyst for ideas at all times.

Last but not least, I would like to mention my family, my friends, all the people who dropped by for discussing the insignificance of life over a cup of tea and my rowing buddies, who all motivated me with their smiling faces and wise words.

Abstract

A complete pipeline for 3D object reconstruction is presented, including initial camera tracking and subsequent refinement via global pose optimisation. Both stages are developed via minimisation of an objective function based on signed distance fields, generated from RGB-D image pairs. The energy is robustified via the incorporation of geometric, photometric and surface orientation constraints. Moreover, registration is done directly, circumventing any correspondence search, which is a major shortcoming of ICP-like approaches. The final outcome of the pipeline is a detailed smooth 3D model.

An extensive evaluation is included, which demonstrates more accurate tracking than state-of-the-art approaches, such as visual odometry and ICP in unconstrained set-ups. The introduction of a multiview pose optimisation routine after the initial trajectory estimation is shown to bring advantages over methods which lack posterior refinement, such as KinectFusion. An assessment of the final meshes confirms precision limited only by the 3D grid resolution.

List of Abbreviations

ADF	Adaptively-sampled Distance Field
AUQ	Augmented Unit Quaternion
BA	Bundle Adjustment
CTSDF	Coloured Truncated Signed Distance Field/ Function
DoF	Degree(s) of Freedom
DVO	Dense Visual Odometry
GICP	Generalized-ICP
ICP	Iterative Closest Point
RPE	Relative Pose Error
SDF	Signed Distance Field/ Function
SLAM	Simultaneous Localisation and Mapping
TSDF	Truncated Signed Distance Field/ Function
TV	Total Variation
VIS	Variational Implicit Surface

Contents

Part I.

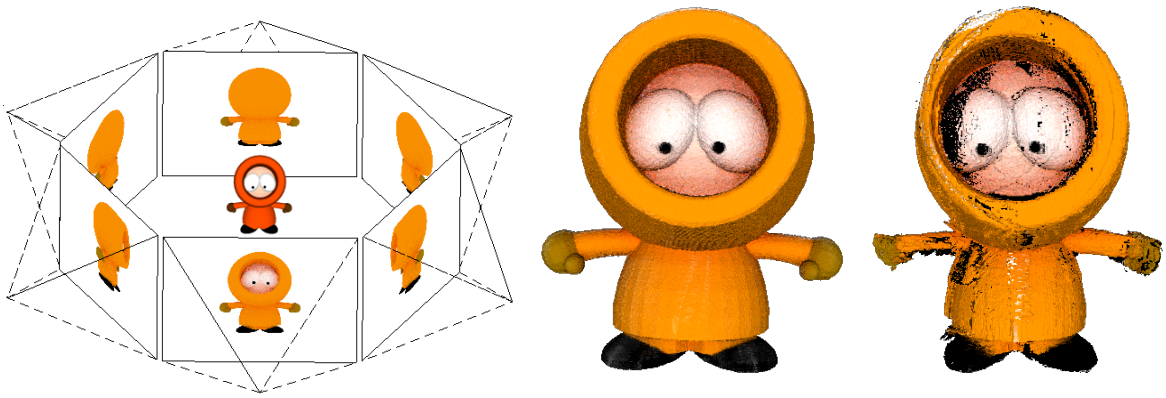
Introduction and Theory

1. Introduction

Three-dimensional digitised models of real-world objects are required in many domains, ranging from non-destructive industrial inspection and evaluation [96], robotic reasoning, navigation and manipulation [66], interior and infrastructure design [79], surgical planning [22], art history and digital museums [49], 3D computer games and infotainment, interactive retail, to many more. Such a broad scope of applications has naturally lead to intensive research in the area of 3D reconstruction. Moreover, the recent advancements in range sensing technology have facilitated a dramatic boost in the quality and performance of these methods.

1.1. Motivation

The increased availability of inexpensive RGB-D sensors in recent years has prompted exceptional focus on 3D object and environment modeling. Given a static scene and depth data acquired from multiple views upon a rigid object, the camera poses (the 6 degrees of freedom rotation and translation) in the world need to be accurately known at every instance in order to generate the respective model. Figure 1.1 illustrates the destructive effect that even slightly inexact poses can have on fused meshes.



Multiview setting around an object

Meshed model from accurate poses

Meshed model from inaccurate poses

Figure 1.1.: Effect of pose accuracy on object reconstruction

Thus, the reconstruction process is usually split in several stages after the (RGB-D) data acquisition: initial camera trajectory estimation, leading to a partial rough object reconstruction, followed by a multiview pose optimisation, finalised by integration of the registered scans into a 3D model [4, 12]. Figure 1.2 is a schematic diagram of this process. Each of its stages can be done in a variety of ways, which have been subject to research in the literature for many years.

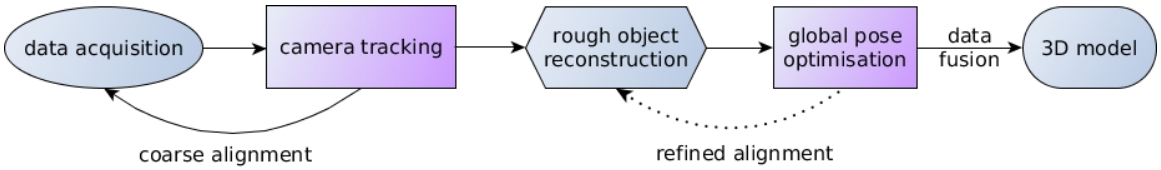


Figure 1.2.: General workflow for 3D object reconstruction

Both the tracking and multiview refinement tasks can be considered as varieties of the 3D registration problem, whereby the relative camera transformation between two frames is searched on the basis of best alignment of the 3D points. Among the most common techniques is the Iterative Closest Points algorithm (ICP) [5, 10]. Although it is very effective, provided good initial alignment, and simple to implement, it suffers from a number of drawbacks. While guaranteed to converge, it may often miss the global minimum, and thus provide a sub-optimal solution. Moreover, it may fail when there is only partial overlap between the two 3D point clouds or when there is significant noise in the data. In addition, ICP is usually very computationally expensive, as in each iteration the point correspondences between the two data sets need to be determined. This matching operation is, furthermore, another source of failures. To the present day, there are research efforts concentrated on robustifying and improving the ICP algorithm [70], however, it still has disadvantages.

Another common approach for the estimation of camera motion is visual odometry [61]. In its classical form, it identifies corresponding feature points in two colour images and uses them to determine the relative movement. In addition to the inherent correspondence problem, the method has the disadvantage that a large portion of the available data is discarded, and thus not utilised further, in the keypoint selection process. Moreover, feature detection is in itself prone to errors. This technique has been leveraged for use in association with depth data, which achieves impressive tracking results in real time [76, 41]. While all available depth data is used, the method still relies on the photoconsistency error, which is sensitive to the quality of the colour camera, to illumination changes, to non-Lambertian surface characteristics, and to proper alignment of the RGB and depth sensors. Furthermore, whereas the geometric error decreases as camera parameters improve, the photometric error tends to oscillate, leading to inferior convergence properties [3].

A variety of approaches have targeted the problem from an entirely different perspective, using implicit surface representations to model the object and its environment. However, a unified pipeline, which uses a common representation for all steps of the process has been a goal for years [13]. Claes *et al.* use an implicit surface model for every task, called the VIS (variational implicit surface), however, its generation and handling seem excessively complex. Thus, an elegant unified pipeline does not yet exist to the best of our knowledge.

The seminal KinectFusion work [59, 36] popularised the use of signed distance fields for the registration task. There the object is represented as a truncated signed distance field (TSDF), which is an implicit function assigning to every point in space the distance to its closest surface point. A global TSDF is continuously updated as new data is acquired

and registered. The point clouds of incoming scans are aligned with it via a variant of ICP. Hence, it suffers from all the characteristic ICP problems, which is a weakness of the approach. Our goal will be to avoid any ICP-like steps in the reconstruction pipeline.

The use of SDFs for registration has been pursued by many other authors [7, 9, 46, 53]. In all of these works, the partially generated model is represented as a continuously growing TSDF (as in KinectFusion), and the points from new depth frames are registered to it in order to determine the camera motion. Since tracking drift accumulates over time, the global model might be too erroneous and lead to even bigger error in the pose estimation. Aiming to overcome this issue, we will use frame-to-frame registration when the pose is completely unknown.

Interestingly, the approaches by Bylow *et al.* [7] and Canelhas *et al.* [9] do not employ ICP in the registration, but directly project the points of the incoming depth map onto the global TSDF. This is the pursued avoidance of the matching problem. However, we will seek a representation different than explicit point coordinates, so that the registration can be further robustified. Moreover, these two methods still apply frame-to-model registration, which, as pointed out before, is prone to accumulation of errors.

Frame-to-frame alignment can, however, be inaccurate when there is significant noise in the data. To compensate for such issues, various techniques for improving the pose estimate from SDF registration have been proposed. Most commonly, the error function being minimised is enhanced via additional terms. Masuda [54] incorporates surface normal information, which can be estimated directly through the SDF itself, while Kehl *et al.* [39] propose to use colour information associated with the voxels of the SDF grid. Both of these ideas have been successfully applied in feature-based approaches as well [72], and have demonstrated their power for improving registration.

After an initial trajectory estimation, usually a set of keyframes is selected, which are used for range data fusion into the finalised model. Prior to this, a global refinement of the registration between them might be performed. This can be done via more accurate alignment, which makes use of the already determined poses as initialisation. Another often employed technique is the representation of the multiview registration problem as a graph optimisation task. To this end, the set of keyframes is converted into a graph in which each pose is a node, while the transformations between the poses are the connecting edges [17, 39]. The pose optimisation problem can then be solved via existing approaches such as the g^{2}o general framework for graph optimisation [47]. This, however, is an extremely expensive procedure. Our target is to avoid it by combining the model generation by scan fusion and the multiview refinement into a single step.

To sum up, signed distance fields have proven their practicality for solving the registration problem in comparison to feature-based and explicit models, and will be the representation of choice for this thesis. Escaping drift errors will be achieved through initial alignment of single-frame SDFs. Subsequent refinement steps and data fusion will be combined into one, leading to a multiview optimisation in which every pose is refined against a global TSDF model. ICP will be escaped in every step, whereby the registration will always be done directly between the voxel grids of two SDFs. Last but not least, means for robustifying the geometric error term will be sought after, such as the addition of surface orientation and photometric information. The 3D model acquisition pipeline will, furthermore, be designed in such a way that its camera tracking and pose optimisation components can be used as stand-alone methods.

1.2. Problem Statement

Having identified the advantages of previous approaches, we set out the following goals for the 3D model acquisition pipeline developed through this thesis:

- A **complete pipeline**, starting from the images acquired by the depth sensor, until the rendered 3D model.
- A **unified pipeline**, which uses the same objective function for each step of the reconstruction process, i.e. both in camera tracking and in global pose optimisation.
- A **direct method**, which registers data avoiding any explicit correspondence search.
- A **dense method**, which uses all available depth data.
- A method which also makes use of other data readily available from RGB-D images, i.e. **colour and surface normal information**.

1.3. Outline of the Thesis

The remainder of this thesis is organised as follows:

- **Chapter 2** (*Related Work*) analyses previous work on camera tracking and 3D reconstruction, identifying certain strengths and weaknesses which lead to the formation of the proposed approach.
- **Chapter 3** (*Background*) provides an overview of the required mathematical background, with a special emphasis on the theory about signed distance fields and their properties.
- **Chapter 4** (*Signed Distance Field Registration*) contains the main derivation of the proposed unified energy and its use for registration of RGB-D images.
- **Chapter 5** (*3D Reconstruction Pipeline*) describes the complete pipeline including optional preprocessing, frame-to-frame camera tracking, global multiview pose optimisation and generation of a final model via TV-L^1 variational minimisation.
- **Chapter 6** (*Evaluation Methodology*) summarises the metrics used for evaluating the proposed approach. These are applied separately on the trajectories from camera tracking and on the 3D model produced at the end of the pipeline.
- **Chapter 7** (*Experimental Results and Assessment*) contains the results of the proposed method, evaluated according to the above metrics. In addition, there are comparisons to a variety of state-of-the-art techniques, including KinectFusion [59], visual odometry [41] and ICP [73].
- **Chapter 8** (*Summary*) extracts the achievements and lessons learned from this work, and highlights its contributions.
- **Chapter 9** (*Future Work*) suggests directions for future work with some concrete ideas about the next steps for improving the memory footprint and speed of the pipeline.

2. Related Work

In this chapter related work on camera tracking and pose optimisation will be discussed. Key weaknesses and advantages of state-of-the-art methods will be identified, aiding the specific choice of methodology for the proposed approach.

2.1. Registration

Given point clouds obtained from different points of view upon the surface of an object, the *registration task* is to place them into a common reference frame by estimating the relative rigid-body transformation between them [25, 10]. The problem is further classified into *pair-wise registration*, when only two datasets are involved, and *multi-view registration*, when more than two views are given. Depending on the availability of prior information, there is *registration refinement*, which assumes an initial guess about the transformation, and *unconstrained registration*, in which no knowledge about the setting is presumed [34]. All of these scenarios have long been subject to research in the computer vision community.

Iterative Closest Points

Arguably the most widely spread technique for point cloud registration is the Iterative Closest Points (ICP) method, which was developed in the early 1990s, but is being extended and improved to the present day. Almost simultaneously Besl and McKay [5] and Chen and Medioni [10] outlined a method for matching 3D shapes, which handles all 6 DoF of the motion, does not require processing the 3D data, is independent of the shape representation and can cope with moderate amounts of normally distributed noise [5]. However, their algorithms could not perform well in the presence of gross statistical outliers and were quite costly. In addition to providing a detailed overview of existing implementations and extensions of the original ICP, Rusinkiewicz and Levoy [70] identify the general steps of the procedure as follows:

- point selection in one or both datasets,
- matching between the selected points,
- weighting the corresponding pairs,
- rejection of unreliable pairs,
- choice of error metric based on the point pairs,
- minimisation of the chosen error metric.

Besl and McKay's proposal used a point-to-point error metric between the datasets, while Chen and Medioni used a point-to-plane measure, which is usually more robust. Extensions to multiview settings were already outlined by Chen and Medioni as a global view-to-model process, and followed shortly after [58, 65, 52].

Dealing with the above-mentioned problems was attempted in various ways by many authors. Zhang [94] analyzed the distance distribution in order to derive a statistical method for outlier rejection, making the method more robust with respect to appearing and disappearing points, outliers, and occlusion. Fitzgibbon [25] suggested a speedup, achieved through employing the distance transform representation.

It is important to mention that the classical ICP suggested by Besl and McKay is guaranteed to monotonously converge to a local minimum from any initial setting, although this might not be the global minimum [5]. Extensions of the method, on the other hand, often do not even have a proof of convergence to a local minimum [94]. Therefore, there is a trade-off between the robustness and convergence properties of any ICP variant.

Notably, Johnson and Kang [37] proposed to solve the multiview registration problem via an ICP modification, termed *colour ICP*, which considers not only 3D information, but also colour. They showed that this extension makes the registration error decrease significantly, which is why we will also include photometric information in the error metrics for the proposed approach. The last step of Johnson and Kang's method is view integration based on a 3D occupancy grid, which stores likelihoods for the spatial distribution of points, i.e. a representation resembling SDFs. Henry *et al.* [32] proposed a similar approach for RGB-D cameras, called *RGBD-ICP*, which additionally uses the point-to-plane metric. Using colour has also been backed through multiple examples in Bernardini and Rushmeier's overview [4] of the 3D model acquisition pipeline that uses range data registration.

Similarly, Schütz *et al.* [72] proposed another extension of ICP, which adds not only the surface colour as a constraint, but also the surface orientation, i.e. the consistency between surface normal vectors. This *multi-feature ICP* manages to avoid errors in ambiguous cases, in which the surface geometry is not distinguishable enough for successful application of classical ICP. The authors show that colour and normal constraints on their own aid the geometric error term, while all three components together give even better results.

One of the most robust ICP implementations is Segal *et al.* 's *Generalized-ICP* [73]. It builds on the evidenced improvement brought by the point-to-plane metric, and represents both surfaces as locally planar, thereby deriving a so called plane-to-plane metric. The approach is allegedly more robust to incorrect correspondences, and allows for the inclusion of probabilistic models, such as measurement noise. Since GICP is found to be so powerful and an implementation of it is freely available in the Point Cloud Library[1], we used it extensively for comparison with our approach.

In conclusion, the ICP algorithm is general, simple and extensible. However, it has numerous failure cases, whose handling interferes with its convergence properties. Besl and McKay also put focus on the computational issues, whereby brute-force comparisons between points might take up several universe lifetimes [5]. Therefore, we are aiming for an approach which is completely ICP-free, so that it would not get stuck in local minima and will not be excessively costly because of the computation of all explicit point matches.

[1] Point Cloud Library (PCL), last accessed: 2 February 2015.

Dense Registration

Tracking for simultaneous localisation and mapping (SLAM) applications is often done on the basis of a limited amount of sparse keypoints. The advantage thereof is the small memory footprint for storing this data. However, this approach is limited to small-scale scenarios, as for example in *Parallel Tracking and Mapping (PTAM)* [43] or RGB-D SLAM [21]. The performance of such approaches suffers in textureless scenes, which is why every-pixel techniques have been proposed, like *Dense Tracking and Mapping (DTAM)* [60]. In it the whole image is taken into account for alignment, which stabilises the tracking in extreme scenarios, such as when rapid motion occurs.

Visual Odometry

Visual odometry is another commonly used approach for tracking. Originally it was based on matching and tracking feature points over frames [61]. It owes its name to the fact that only visual input is utilised. With the increased availability of inexpensive RGB-D sensors, this trend changed, and the method became a dense energy-based technique, aiming to minimise the photometric error between frames [76, 41]. In these approaches, the depth information is utilised to determine the 3D point locations, which are then used to estimate the rigid body motion which brings the two clouds into best alignment, evaluated through their photoconsistency. There is a further extension which robustifies the method by including loop closure detection and selects keyframes for global graph-based optimisation [40]. This has been extended by Dimashova *et al.* [17], where the optimised energy itself includes not only the RGB-D odometry metric, but also a point-to-plane ICP term maximising shape consistency, and a term compensating for accumulated drift.

KinectFusion and Its Extensions

KinectFusion [59, 36] is a revolutionary approach for building smooth, dense surface models in real time using an RGB-D sensor. In it only the depth data is used for registration, while the colour information is only utilised for display to the user. There is a global volumetric model (a TSDF), which is incrementally updated, and which is used for the registration of every incoming depth frame. As a new depth map is acquired, it is converted into a point cloud, which is then projected onto the TSDF volume. Afterwards the camera motion is determined via a variant of ICP. The range data is subsequently integrated into the global model using the determined pose. In the meantime, the TSDF is rendered via ray-casting in order to obtain a noise-free vertex map, which is used for better registration and for display to the user.

Although the approach is extremely powerful and achieves impressive results, it suffers from a number of disadvantages. Firstly, the registration and integration is done in a frame-to-model fashion and, therefore, if a wrong measurement is fused into the global model, the tracking will be inaccurate in all subsequent frames. Moreover, a variant of ICP is used for the point cloud to model registration, which carries all the issues described in the preceding sections. Last but not least, there is no global refinement of the determined poses, so the drift accumulated during tracking cannot be compensated and the final model cannot be further improved. These are the weaknesses that our approach will

target: we will favour frame-to-frame tracking, we will not use ICP and we will employ a refinement step after tracking.

A drawback that we will not focus on is the fact that the dense volumetric representation occupies a lot of memory and is slow to process. This has been handled by *KinectFusion* extensions which employ space partitioning approaches, such as octree representations [91, 92]. The reduced amount of memory entails a decreased computational effort for its handling. The original approach, as well as its modification *Kintinuous* [86] which targets large-scale envoronments, execute the processing on the GPU, thereby achieving real-time results. Due to the existence of such techniques, we will not be concerned with the memory requirements or speed of the approach, since they can be straightforwardly incorporated into the existing pipeline.

SDF Registration

Representing surfaces captured in range images can be done via their point clouds, as in the previously described methods, or via implicit representations. Many authors have demonstrated the aptness of various such models, both for solving the pairwise and the multiview registration problems [6, 82, 34, 32, 29, 83, 31]. The representation of choice for this thesis is, in the nature of *KinectFusion*, an SDF.

The theory of signed distance fields (SDFs) will be presented in a subsequent chapter (*cf.* Section 3.4), while here it is sufficient to know that they are functions associating each point in space with the signed distance to its closest surface location. Rouhani and Sappa [68] have proven that using implicit surfaces allows for avoiding the expensive correspondence search that is a drawback of ICP. They thus represent the error metric as *point-to-implicit*. In this work we will further extend this notion to **implicit-to-implicit** registration, where both surfaces being aligned are represented through their SDFs.

There exist two major works on the topic of registration using the SDF model: the thesis of Kubacki [45] and its accompanying publication [46]; and the thesis of Canelhas [8], and its associated *SDF Tracker* [9]. Both authors minimise the point-to-implicit error, in a *KinectFusion*-like manner. A global SDF is used for registration of incoming point clouds. While Kubacki proposes a novel ICP matching criterion and error metric based on the properties of implicit representations, Canelhas uses the direct projection of points into the SDF volume, thus minimising the sum of squared point-to-model distances. Due to the properties of the volumetric representation, when there is perfect alignment, the readings of the points in the volume will all be zero. This fact is used for the formation of a frame-to-model error metric, which is iteratively minimised, until an estimate for the 6 DoF camera pose is found.

The same technique is also applied by Bylow *et al.* [7]. The optimal camera transformation is determined via a Taylor linearisation of the objective function, leading to the solution of an inexpensive 6×6 system, refined iteratively. Substituting the costly ICP matching with this approach is therefore very advantageous, and will be employed by us as well. However, like *KinectFusion*, all these methods lack a pose refinement step, which will be tackled by us.

Paragios *et al.* [63] have suggested an alternative to the Taylor approximation. Instead, the Jacobian of the registration energy with respect to the camera pose is determined, so that a simple gradient descent approach can be carried out. Due to the fact that in the

rigid body case the same motion is applied on every point in space, all voxels in the SDF grid have the same voting weight, so they can be summed up. Thus, this is also a very computationally cheap procedure. We will attempt both the Taylor approximation and gradient descent minimisation for solving the unconstrained tracking problem, as well as the pose refinement, and determine the cases when one of the methods outperforms the other.

As mentioned in the introductory chapter, various authors have proposed the addition of error terms in the SDF-based objective function in order to improve registration. Masuda uses constraints from surface normal orientations, which is a viable improvement provided a rough alignment of the scans [53, 54], and is therefore more suited for a pose optimisation step. On the other hand, Kehl *et al.* [39] have shown the boost of multiview fusion quality achieved by the association of colour information with the SDF. We will aim to use both of these improvements in tracking and in pose refinement, and look for conclusions about their applicability in the different scenarios.

2.2. Pose Optimisation

Once tracking has been accomplished, a pose optimisation step can be undertaken in order to further improve the estimated trajectory. This is usually done in a multiview setting, so that more information is integrated globally. If the initial error is, however, too large, all techniques would get stuck into local minima [74].

Graph Optimisation

One possibility for the global refinement is to select a set of keyframes and convert them into a graph structure [47]. Each keyframe camera pose is a node, while the transformations between poses are the connecting edges. The goal is then to find the tranformations which lead to the smallest global misalignment, expressed by the strongest multiview geometry consensus. This technique has been successfully applied by many authors [74, 69, 17, 39]. The problem is that with an increasing number of keyframes, the number of connections in the graph structure increases exponentially, and thus requires more time to be optimised.

Global Optimisation

There is another direction for pose refinement, usually taken by unified approaches. They are unified in the sense that every component of the process is based on the same energy minimisation, however, it can be handled differently in the various stages. Ren and Reid [67] have successfully done this with tracking and calibration, using the Jacobian energy derivation of an objective function based on the truncated distance transform.

Claes *et al.* [13] have achieved this in the setting we are targeting: initial crude alignment, followed by multiview refinement, using the same formulation for both. They, however, use a so-called variational implicit surface (VIS) representation. Although it shares many properties with SDFs, it requires more computations and more expensive handling. As approaches of this nature are expected to be computationally significantly cheaper than graph-based optimisations, we will implement such a unified pipeline based on SDFs.

3. Background

This chapter summarises the required mathematical background. Starting from the basic properties of depth images delivered by RGB-D sensors, it is then explained how they can be projected into 3D point clouds. Next, the utilised models for representing rigid body motion are described. Most notably, the theory of coloured truncated signed distance fields is presented, together with two techniques for model generation by SDF fusion: weighted averaging and variational TV-L^1 energy minimisation.

3.1. RGB-D Sensors

RGB-D cameras are active depth sensors, which project a known light pattern onto a scene. The camera observes the pattern and uses the differences to corresponding points in the original structure, together with the known projector-to-camera transformation, to determine depth. In addition, a pre-registered RGB image is delivered. This section will, however, focus solely on depth sensing technologies and not discuss the properties of the colour cameras.

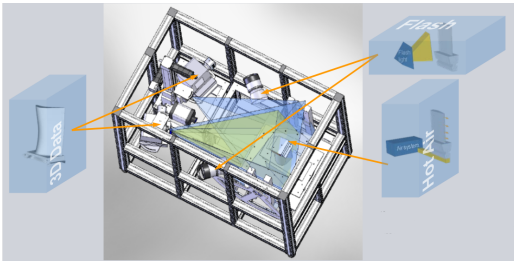
There are two important metrics for the quality of depth sensors. First, the **depth resolution** is the minimum measurable depth difference, i.e. the smallest distance between two depth values which can still be discerned. It is determined by the bits per pixel used for storing the measurements of disparity of the speckles of the projected pattern [42]. As such, the depth resolution degrades with increasing distance from the camera to the measured surface. The other characteristic is **depth accuracy**, which refers to the imprecision in measuring disparity.

Two types of depth sensors have been used for the generation of evaluation sequences for the proposed approach. Although both of them employ structured light to estimate depth, there is a considerable difference in the fidelity of the depth maps they produce.

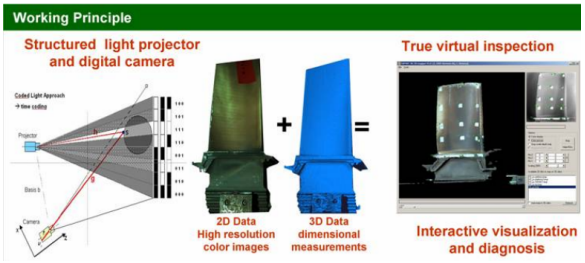
On the one hand, an industrial scanner, the Siemens Global Inspection System (GIS), has been used. This is a high-end non-destructive evaluation technology, which provides a wide variety of testing possibilities in addition to depth sensing: crack detection via ultrasonic vibrations and infrared imaging, hot-air thermography, flash thermography, and resistance testing via pressure exertion [96, 23]. Its basic operational principle is demonstrated in Figure 3.1.

As the main application of this sensor is industrial metrology, it is extremely precise. The measurements are with a depth resolution of 0.075 mm [23], which is comparable to what is achieved by triangulation laser rangefinders for computer vision applications[1]. Moreover, both the colour and depth maps are of resolution 10.7 megapixels, which makes up a vast amount of precise measurements. Thus, the depth data provided by this sensor has extremely high fidelity and can be used as-is.

[1] 3-D Modeling with Laser Rangefinder, Wikipedia, last accessed: 29 January 2015.



Component diagram [23]

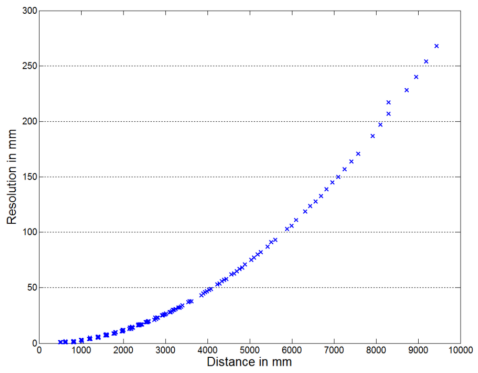


Principle of operation [96]

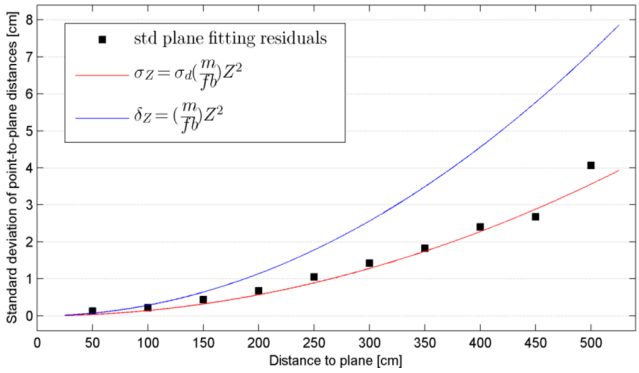
Figure 3.1.: Siemens Global Inspection System overview

On the other hand, a Kinect-like sensor[2] was used for acquiring other test sequences. Such inexpensive, consumer-level depth cameras have a much lower accuracy, which decreases quadratically with distance. The manifested errors are both systematic and random. Systematic errors are accounted for during the alignment of RGB and depth data, done in middleware using the known mathematical principle for depth measurement and its parameters. They can be further corrected by careful calibration techniques. Random errors, on the other hand, should be modeled and compensated for algorithmically [42].

The Kinect has two operating range modes: default (0.8 - 4.0 m) and near (0.4 - 3.0 m, used for the evaluation sequences in this work) [1]. It has been experimentally found that the measurement error is already a few millimeters at the start of the range, increasing quadratically up to about 4 cm at the maximum range [42]. More specifically, at 2.0 m displacement, the precision is about 4.0 cm, while the depth resolution is approximately 1.2 cm [2]. Figure 3.2 displays the sensor resolution and error as a function of distance.



Depth resolution as a function of distance [2]



Measurement error as a function of distance [42]

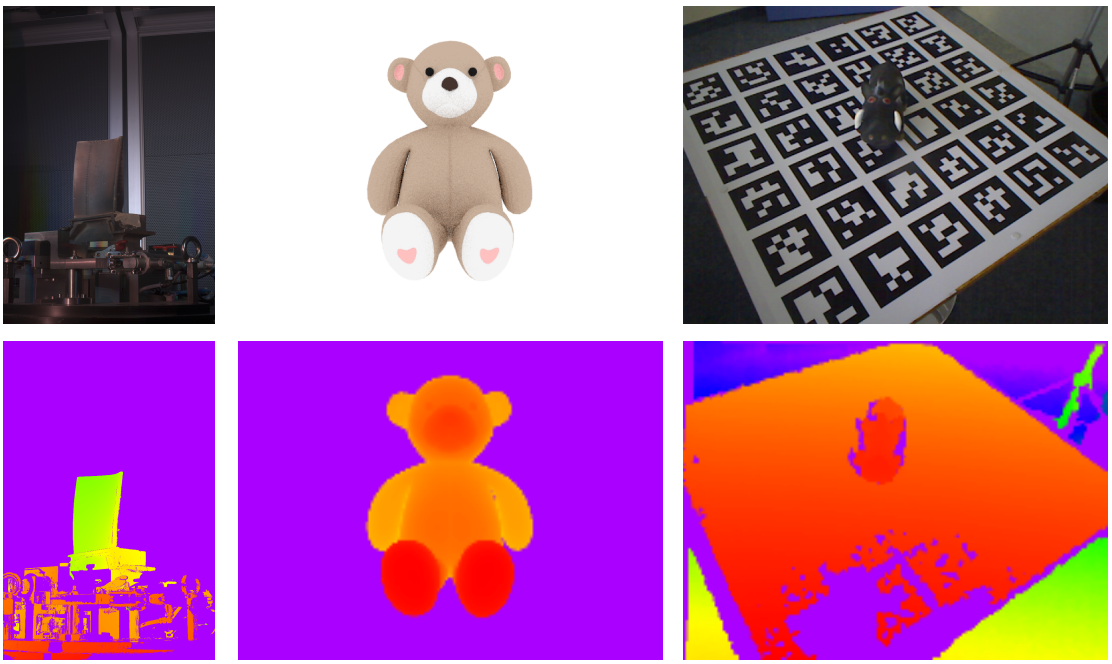
Figure 3.2.: Measurement characteristics of a Kinect-like sensor

In addition to the inferior measurement capabilities, the delivered RGB-D pairs are of size 640×480 pixels, which is a 35 times lower resolution than that of the industrial camera. Therefore, it is vital that the noise in the depth maps is reduced in a preprocessing step in order to achieve reliable results. Usual smoothing techniques need to be adapted for this

[2] Kinect for Windows Sensor Components and Specifications, last accessed: 29 January 2015.

task, since range images exhibit different properties than colour images. The algorithmic details of the depth refinement procedure will be explained in Section 5.2.

The differences in depth map quality are visualised in Figure 3.3. The range images are colour-coded according to distance, with red being the closest to the camera and blue - farthest. Purple denotes missing depth data.



GIS Synthetic Kinect

Figure 3.3.: Sensor quality comparison (top row: RGB images, bottom row: depth maps)

On the left an RGB-D image pair obtained with the Siemens Global Inspection System is displayed. Data is acquired everywhere on the scanned object, including edges. Furthermore, the transitions between neighbouring values are smooth, with slightly more noise noticeable along edges.

The middle images are representative of a synthetically generated dataset, in which the image resolution is 640×480, i.e. the same as for the Kinect. The depth map is dense and smooth. One can observe the similarity between the range data acquired with the industrial sensor and the simulated sequence.

The difference with images obtained by Kinect-like sensors is striking, as shown on the right. First, the data is very noisy everywhere and fuzzy along edges. Moreover, measurements are completely missing along depth discontinuities, such as the edges of the table and the horns of the object. The inability to acquire such fine details is a major drawback of these devices.

To sum up, there can be significant differences between depth sensors. In order to compensate for them, preprocessing will be required for the inferior devices. In addition, using non-depth dependent terms in the objective function will be beneficial for the robustness of registration of depth maps produced by such cameras.

3.2. Pinhole Camera Model

The projection of 3D scene points onto an image plane is described by the camera model. In this thesis, the standard pinhole camera model [30, pp. 153–158] was employed, whereby the camera is represented as an infinitesimally small hole. Only light rays which go through the hole and intersect the image plane get a projected point in the image. Figure 3.4 depicts this process.

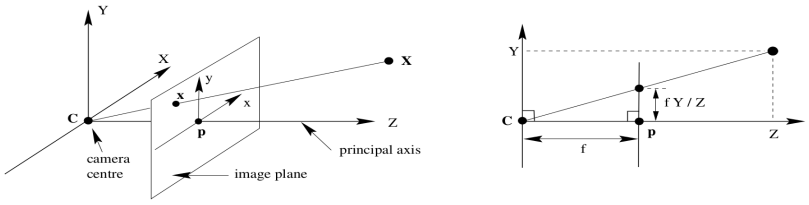


Figure 3.4.: Pinhole camera model [30]

In the case of CCD cameras, the pixels might be non-square. Thus, there are two scaling factors of the focal length f, in x- and y-direction respectively, leading to the quantities f_x and f_y, the *focal lengths* in pixels. Further, the intersection of the principal axis with the image plane, called the *principal point*, is denoted by (o_x, o_y) in pixel coordinates. These characteristic features are combined into the *internal camera calibration matrix* $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.1}$$

Let a 3D point be denoted as $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ and its corresponding point in the image plane be $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$. Then, when represented in homogeneous coordinates[3], they are related by the equation:

$$\widetilde{\mathbf{x}} = \mathbf{K}\,(\mathbf{I_{3\times 3}}\ \ \mathbf{0_{3\times 1}})\,\widetilde{\mathbf{X}}. \tag{3.2}$$

This is usually written as the projection relation $\pi : \mathbb{R}^3 \longrightarrow \mathbb{R}^2, \quad (X, Y, Z)^\top \mapsto (x, y)^\top$. Therefore, to deduce 2D image coordinates given a 3D point, one has to compute:

$$x = \frac{X}{Z} f_x + o_x, \qquad y = \frac{Y}{Z} f_y + o_y. \tag{3.3}$$

The projection can be inverted when the depth is known, leading to the inverse relation $\overline{\pi}^{-1} : \mathbb{R}^3 \longrightarrow \mathbb{R}^3, \quad (x, y, Z)^\top \mapsto (X, Y, Z)^\top$. It is used for calculating a point cloud out of a depth map, which associates a depth value Z to every pixel (x, y). The explicit formulas for the 3D coordinates are then as follows:

$$X = \frac{x - o_x}{f_x} Z, \qquad Y = \frac{y - o_y}{f_y} Z, \qquad Z = Z. \tag{3.4}$$

The focal length and principal point can be determined via standard calibration techniques [95]. It is further important to note that additional distortion parameters might be influencing the projection, which also have to be determined [95] and corrected for before proceeding with the images.

[3] From here onwards, the notation $\widetilde{\mathbf{p}}$ will be used to denote the homogeneous representation of any $\mathbf{p}$, and $\bar{\mathbf{p}}$ for obtaining the inhomogeneous coordinates of $\widetilde{\mathbf{p}}$.

3.3. Rigid Body Motion

The target objects for reconstruction are rigid bodies placed in a static scene around which a camera is moved. Alternatively, an object can be placed on a rotating turntable in front of a fixed camera, which is an equivalent motion, provided a segmentation of the background. Since these two types of motion differ only by the selected reference frame, estimating either one determines the other. In the current case the goal in registration has been chosen to be finding the rigid body transformation of the camera between two consecutive frames, so that a trajectory can be calculated.

Characteristic for this kind of motion is that it preserves the relative distance and orientation between any pair of points on the object, thus the term *rigid*. All such transformations in 3D Euclidean space form the special Euclidean group $SE(3)$ [35]. A rigid body motion consists of a rotation and a translation and, thus, has six degrees of freedom: three for its orientation in space, and three for its displacement from the center of the reference coordinate system. One way to represent such a transformation is as a 4×4 matrix,

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{3.5}$$

where $\mathbf{R} \in SO(3)$ is a 3×3 orthogonal matrix representing the rotation, and $\mathbf{t} \in \mathbb{R}^3$ is a vector corresponding to the translation. Because of the orthogonality of the rotational component, the inverse of a rigid body transformation can be computed as:

$$\mathbf{T}^{-1} = \begin{pmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.6}$$

The issue of this notation is that the matrix contains 16 elements, which must express only 6 DoF. Therefore, other parameterisations have been sought after, with [16] giving a detailed overview on rotation representations. Two versions have been employed for this thesis: one with twist coordinates, and another one using unit quaternions for the rotation and 3-element vectors for the translational component. In later derivations, Jacobians of points in space with respect to camera pose will be needed, so the following sections explain the properties of the chosen representations and calculate these Jacobians.

3.3.1. Twist Coordinates

A minimal way of representing rigid body motion is through the Lie algebra $se(3)$ of the $SE(3)$ group, which has only 6 DoF [51]. A rigid body transformation is written in twist coordinates

$$\xi = (\mathbf{u} \quad \boldsymbol{\omega})^\top = (u_1, u_2, u_3, \omega_1, \omega_2, \omega_3)^\top, \tag{3.7}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the part representing the rotational component of the transformation, while $\mathbf{u} \in \mathbb{R}^3$ corresponds to the translation.

Next, using the generators of the algebra [18]

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{G}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{G}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{G}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.8}$$

an element of $se(3)$ has the form

$$u_1\mathbf{G}_1 + u_2\mathbf{G}_2 + u_3\mathbf{G}_3 + \omega_1\mathbf{G}_4 + \omega_2\mathbf{G}_5 + \omega_3\mathbf{G}_6. \tag{3.9}$$

The respective transformation can be obtained as a 4×4 matrix by exponentiation:

$$\mathbf{T}(\xi) = \exp\left(\hat{\xi}\right) = \exp\begin{pmatrix}\boldsymbol{\omega}_\times & \mathbf{u} \\ \mathbf{0} & 0\end{pmatrix}, \tag{3.10}$$

where $\boldsymbol{\omega}_\times$ is the skew-symmetric matrix corresponding to $\boldsymbol{\omega}$:

$$\boldsymbol{\omega}_\times = \begin{pmatrix}0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0\end{pmatrix} \tag{3.11}$$

and

$$\exp(\boldsymbol{\omega}_\times) = \mathbf{I} + \boldsymbol{\omega}_\times + \frac{1}{2!}\boldsymbol{\omega}_\times^2 + \frac{1}{3!}\boldsymbol{\omega}_\times^3 + \cdots = \mathbf{I} + \frac{\sin\theta}{\theta}\boldsymbol{\omega}_\times + \frac{1-\cos\theta}{\theta^2}\boldsymbol{\omega}_\times^2. \tag{3.12}$$

Substituting this back into equation 3.10 provides a closed-form expression for converting a twist into a transformation matrix [18]:

$$\begin{aligned} \theta &= \sqrt{\boldsymbol{\omega}^\top\boldsymbol{\omega}} \\ A = \frac{\sin\theta}{\theta}, \quad B &= \frac{1-\cos\theta}{\theta^2}, \quad C = \frac{1-A}{\theta^2} \\ \mathbf{R} &= \mathbf{I} + A\boldsymbol{\omega}_\times + B\boldsymbol{\omega}_\times^2 \\ \mathbf{V} &= \mathbf{I} + B\boldsymbol{\omega}_\times + C\boldsymbol{\omega}_\times^2 \\ \mathbf{T}(\xi) = \exp\left(\hat{\xi}\right) &= \begin{pmatrix}\mathbf{R} & \mathbf{V}\mathbf{u} \\ \mathbf{0} & 1\end{pmatrix} = \begin{pmatrix}\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1\end{pmatrix}. \end{aligned} \tag{3.13}$$

Calculating the twist coordinates given a transformation matrix is the inverse process, whereby the logarithm is taken instead of the exponential [18]:

$$\begin{aligned} \theta &= \arccos\left(\frac{\operatorname{tr}(\mathbf{R}) - 1}{2}\right) \\ \omega &= \ln(\mathbf{R}) = \frac{\theta}{2\sin\theta}\left(\mathbf{R} - \mathbf{R}^\top\right) \\ \mathbf{u} &= \mathbf{V}^{-1}\mathbf{u} = \left(\mathbf{I} - \frac{1}{2}\boldsymbol{\omega}_\times + \frac{1}{\theta^2}\left(1 - \frac{A}{2B}\right)\boldsymbol{\omega}_\times^2\right)\mathbf{t}. \end{aligned} \tag{3.14}$$

Having defined the conversion from twists to transformation matrices and vice versa, the Jacobian of a point in 3D Euclidean space with respect to the twist which generated it can be calculated. Let $\mathbf{x} \in \mathbb{R}^3$ be a 3D point to which a rigid body transformation $\mathbf{T}$ is applied, moving it to point $\mathbf{y} \in \mathbb{R}^3$. In homogeneous coordinates:

$$\tilde{\mathbf{y}} = \mathbf{T}\,\tilde{\mathbf{x}} = \begin{pmatrix}\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1\end{pmatrix}\tilde{\mathbf{x}}. \tag{3.15}$$

Using the generators of the Lie algebra and the representation from equation 3.9, the above transformation can be rewritten as:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = (u_1\mathbf{G}_1 + u_2\mathbf{G}_2 + u_3\mathbf{G}_3 + \omega_1\mathbf{G}_4 + \omega_2\mathbf{G}_5 + \omega_3\mathbf{G}_6) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} =$$
$$= \begin{pmatrix} u_1 & & & + & \omega_2\, x_3 & - & \omega_3\, x_2 \\ u_2 & - & \omega_1\, x_3 & & & + & \omega_3\, x_1 \\ u_3 & + & \omega_1\, x_2 & - & \omega_2\, x_1 & & \\ & & & & 0 & & \end{pmatrix}. \tag{3.16}$$

It is now straightforward to obtain the Jacobian by deriving with respect to each of the six twist coordinates:

$$\frac{\partial \mathbf{y}}{\partial \xi} = \begin{pmatrix} \frac{\partial y_1}{\partial u_1} & \frac{\partial y_1}{\partial u_2} & \frac{\partial y_1}{\partial u_3} & \frac{\partial y_1}{\partial \omega_1} & \frac{\partial y_1}{\partial \omega_2} & \frac{\partial y_1}{\partial \omega_3} \\ \frac{\partial y_2}{\partial u_1} & \frac{\partial y_2}{\partial u_2} & \frac{\partial y_2}{\partial u_3} & \frac{\partial y_2}{\partial \omega_1} & \frac{\partial y_2}{\partial \omega_2} & \frac{\partial y_2}{\partial \omega_3} \\ \frac{\partial y_3}{\partial u_1} & \frac{\partial y_3}{\partial u_2} & \frac{\partial y_3}{\partial u_3} & \frac{\partial y_3}{\partial \omega_1} & \frac{\partial y_3}{\partial \omega_2} & \frac{\partial y_3}{\partial \omega_3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & x_3 & -x_2 \\ 0 & 1 & 0 & -x_3 & 0 & x_1 \\ 0 & 0 & 1 & x_2 & -x_1 & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\mathbf{x}_\times \end{pmatrix}. \tag{3.17}$$

This is an important result, which will be used in the derivation of the energy minimisation scheme in later sections. To sum up, given a rigid body transformation $\mathbf{T}$ and a 3D point $\mathbf{x}$, if $\tilde{\mathbf{y}} = \mathbf{T}\,\tilde{\mathbf{x}}$, then the Jacobian is $\mathbf{J}_{twist}(\mathbf{y}, \xi) = \frac{\partial \mathbf{y}}{\partial \xi} = \begin{pmatrix} \mathbf{I} & -\mathbf{x}_\times \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -(\overline{\mathbf{T}^{-1}\tilde{\mathbf{y}}})_\times \end{pmatrix}$.

Another benefit of the Lie algebra representation is the fact that by definition it is a vector space (with a special operation called the "Lie bracket") [28], and as such it is closed under scalar multiplication. This property is essential for numerical approaches, such as gradient descent, as it permits the multiplication with a scalar step size, guaranteeing that a rigid body motion will be obtained, contrary to the case of general rigid body transformation matrices where conserving the orthogonality of the rotational matrix is not promised.

3.3.2. Augmented Unit Quaternions

A general quaternion is a 4-element vector $\mathbf{q} = (q_w, q_x, q_y, q_z)^\top$ with

$$\begin{aligned} \text{adjoint:} \quad & \bar{\mathbf{q}} = (q_w, -q_x, -q_y, -q_z)^\top, \\ \text{norm:} \quad & \|\mathbf{q}\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}, \\ \text{inverse:} \quad & \mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{\|\mathbf{q}\|}. \end{aligned} \tag{3.18}$$

A unit quaternion is a special case of a quaternion with unity norm [16]. It is a popular way of representing rotations in 3D Euclidean space using the following formula [19]:

$$\mathbf{R}(\mathbf{q}) = \begin{pmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_zq_w & 2q_xq_z + 2q_yq_w \\ 2q_xq_y + 2q_zq_w & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_xq_w \\ 2q_xq_z - 2q_yq_w & 2q_yq_z + 2q_xq_w & 1 - 2q_x^2 - 2q_y^2 \end{pmatrix}. \tag{3.19}$$

More specifically, the unit quaternion is a modification of the axis-angle representation. q_w corresponds to the angle of rotation θ: $q_w = \cos\frac{\theta}{2}$, while the remaining elements represent the normalised rotation axis $\hat{\mathbf{r}}$: $(q_x, q_y, q_z)^\top = \hat{\mathbf{r}}\ \sin\frac{\theta}{2}$ [84].

Transforming a rotation matrix back to a quaternion is slightly more complex. It is done by first deriving a solution for one of the four quaternion elements, using the fact that the quaternion norm is unity and the rotational matrix is orthogonal. Then it is straightforward to determine the remaining three elements. Here the version where first q_w is found will be presented, while the reader is referred to [24] for the remaining options:

$$q_w = \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}}, \quad \mathbf{q} = \begin{pmatrix} q_w \\ \frac{R_{32} - R_{23}}{4q_w} \\ \frac{R_{13} - R_{31}}{4q_w} \\ \frac{R_{21} - R_{12}}{4q_w} \end{pmatrix}. \tag{3.20}$$

It is essential to normalise back to unit quaternions when performing any kind of operations on rotations.

For representing rigid body motion, the translational part has to be accounted for as well. In this way, a 7-element vector $\boldsymbol{\lambda} \in \mathbb{R}^7$ is formed, which will be termed the **augmented unit quaternion** hereafter:

$$\boldsymbol{\lambda} = \begin{pmatrix} \mathbf{t} & \mathbf{q} \end{pmatrix}^\top = \begin{pmatrix} t_x & t_y & t_z & q_w & q_x & q_y & q_z \end{pmatrix}^\top, \tag{3.21}$$

where $\mathbf{t} \in \mathbb{R}^3$ is the translational part of the rigid body motion, while $\mathbf{q}$ is the unit quaternion corresponding to the rotation, as explained above.

Having already defined the conversions between unit quaternions and rotation matrices, the conversions between augmented unit quaternions and 4×4 transformation matrices are done in the same way, with additionally including the translation vector.

In the next step, the Jacobian of a point with respect to the augmented unit quaternion that generated it will be calculated, as was done for twists. Again, let $\mathbf{x} \in \mathbb{R}^3$ be a 3D point to which a rigid body motion $\mathbf{T}$ is applied, transforming it into point $\mathbf{y} \in \mathbb{R}^3$. Representing $\mathbf{T}$ through its augmented unit quaternion expansion, as shown in Equation 3.19, leads to the following relation in homogeneous coordinates:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{pmatrix} = \widetilde{\mathbf{y}} = \mathbf{T}(\boldsymbol{\lambda})\, \widetilde{\mathbf{x}} = \begin{pmatrix} \mathbf{R}(\mathbf{q}) & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} (1 - 2q_y^2 - 2q_z^2)x_1 + (2q_xq_y - 2q_zq_w)x_2 + (2q_xq_z + 2q_yq_w)x_3 + t_x \\ (2q_xq_y + 2q_zq_w)x_1 + (1 - 2q_x^2 - 2q_z^2)x_2 + (2q_yq_z - 2q_xq_w)x_3 + t_y \\ (2q_xq_z - 2q_yq_w)x_1 + (2q_yq_z + 2q_xq_w)x_2 + (1 - 2q_x^2 - 2q_y^2)x_3 + t_z \\ 1 \end{pmatrix}. \tag{3.22}$$

It is now easy to derive the Jacobian of a point with respect to an augmented quaternion[4]:

$$\begin{aligned} \mathbf{J}_{AUQ}(\mathbf{y}, \boldsymbol{\lambda}) = \frac{\partial \mathbf{y}}{\partial \boldsymbol{\lambda}} &= \begin{pmatrix} \frac{\partial y_1}{\partial t_x} & \frac{\partial y_1}{\partial t_y} & \frac{\partial y_1}{\partial t_z} & \frac{\partial y_1}{\partial q_w} & \frac{\partial y_1}{\partial q_x} & \frac{\partial y_1}{\partial q_y} & \frac{\partial y_1}{\partial q_z} \\ \frac{\partial y_2}{\partial t_x} & \frac{\partial y_2}{\partial t_y} & \frac{\partial y_2}{\partial t_z} & \frac{\partial y_2}{\partial q_w} & \frac{\partial y_2}{\partial q_x} & \frac{\partial y_2}{\partial q_y} & \frac{\partial y_2}{\partial q_z} \\ \frac{\partial y_3}{\partial t_x} & \frac{\partial y_3}{\partial t_y} & \frac{\partial y_3}{\partial t_z} & \frac{\partial y_3}{\partial q_w} & \frac{\partial y_3}{\partial q_x} & \frac{\partial y_3}{\partial q_y} & \frac{\partial y_3}{\partial q_z} \end{pmatrix} = \\ &= \begin{pmatrix} 1 & 0 & 0 & -2q_zx_2 + 2q_yx_3 & 2q_yx_2 + 2q_zx_3 & -4q_yx_1 + 2q_xx_2 + 2q_wx_3 & -4q_zx_1 - 2q_wx_2 + 2q_xx_3 \\ 0 & 1 & 0 & 2q_zx_1 - 2q_xx_3 & 2q_yx_1 - 4q_xx_2 - 2q_wx_3 & 2q_xx_1 + 2q_zx_3 & 2q_wx_1 - 4q_zx_2 + 2q_yx_3 \\ 0 & 0 & 1 & -2q_yx_1 + 2q_xx_2 & 2q_zx_1 + 2q_wx_2 - 4q_xx_3 & -2q_wx_1 + 2q_zx_2 - 4q_yx_3 & 2q_xx_1 + 2q_yx_2 \end{pmatrix}. \end{aligned} \tag{3.23}$$

[4] A calculation of this Jacobian has been presented in [64], Table 1. However, note the difference in the $\frac{\partial y_2}{\partial q_z}$ component: the middle value should be $-4q_zx_2$ instead of $-4q_xx_2$.

There is one other notable difference between the Lie algebra representation and augmented unit quaternions, namely the scalar multiplication. The translational part of a AUQ can, naturally, be scaled by simple multiplication. The rotation, however, is not linear and has to be interpolated. Implementationally, this is done by interpolating between the identity quaternion $\mathbf{q}_{id} = (1, 0, 0, 0)^\top$ and the unit quaternion that is being scaled.

There are two ways of interpolating by a factor of μ between two unit quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$: *lerp* (linear interpolation) and *slerp* (spherical linear interpolation) [84]:

$$\begin{aligned} lerp(\mathbf{q}_1, \mathbf{q}_2, \mu) &= (1-\mu)\,\mathbf{q}_1 + \mu\,\mathbf{q}_2 \\ slerp(\mathbf{q}_1, \mathbf{q}_2, \mu) &= \frac{\sin\left((1-t)\alpha\right)}{\sin\alpha}\,\mathbf{q}_1 + \frac{\sin\left(\mu\alpha\right)}{\sin\alpha}\,\mathbf{q}_2, \quad \alpha = \arccos \mathbf{q}_1 \cdot \mathbf{q}_2. \end{aligned} \tag{3.24}$$

The disadvantage of *lerp* is that it results in faster movement between 20° and 160°, while *slerp* provides smoother motion, but is more expensive to compute. However, the pose updates that are handled in this work are usually less than a degree, thus, using *lerp* is a sufficiently good approximation. Additionally, it is important to normalise the resulting interpolated quaternion back to unit norm.

3.4. Signed Distance Fields

Signed distance fields (SDFs) have a broad range of applications in graphics, computer vision and medical imaging, such as curve smoothing, detection of dominant points on digital curves, finding convex hulls, determining object skeletons, centerlines and medial axes, computing Dirichlet tessellations, morphing, hypertexture, scene motion, collision detection, obstacle avoidance, and many others [88, 57, 38]. They are also useful in efficiently combining information from sensors of different nature, such as sonar and stereo [20]. In this section the concept of a signed distance field, also referred to as signed distance function[5], will be presented. Moreover, various properties and approaches for fusing SDFs generated from accurate poses will be described.

3.4.1. Definition

A signed distance function is an n-dimensional implicit function, which associates a scalar value with each point of its n-dimensional domain [62, pp. 32–37]. As the subject is 3D reconstruction, the focus will be on 3-dimensional SDFs, which will not be explicitly specified hereafter. Formally, the function

$$\phi\colon \Omega \subseteq \mathbb{R}^3 \to \mathbb{R} \tag{3.25}$$

assigns to each point $\mathbf{x} \in \mathbb{R}^3$ its signed distance to the closest object boundary point. Points located within the object bounds have negative signed distance values, while points outside are assigned positive values. The interface between them is the object surface, where the signed distance is zero. Therefore, the surface can be extracted out of such a field as its zeroth level-set crossing via methods like marching cubes or ray-tracing [27].

[5] The term *signed distance function* will be avoided when referring to the digital implementation, because there the representation is discrete and the actual function is not known. What is being dealt with is a 3D voxel grid of scalar values, which is better reflected by the term *field*.

SDFs have numerous advantages over other representations. They not only represent the object boundaries, like boundary representations, but also its interior and its surrounding volume. Moreover, they provide a means to inexpensively compute any offset surface by simply changing the extracted iso-surface value [27].

Unfortunately, implicit functions are very costly to compute [27]. This is why they are usually represented as discrete volumes, subdivided into voxels. For this thesis, the voxels were cubical, but any other shape can be chosen. As the voxel size approaches zero, it approximates a point better and better, and therefore becomes closer to the actual distance function. However, the available memory is a limiting factor for the choice of voxel dimensions and, consequently, for the quality of the approximation that the discrete grid provides for the underlying SDF.

3.4.2. Generation

There are two common ways in which the voxel grid of an SDF can be generated:

- Cast a ray from the camera center through each pixel of the range image, and calculate the difference between the surface point (whose coordinates are obtained via Equation 3.4) and the value at the depth map pixel (obtained via Equation 3.3) [14].

- Project each voxel center onto the depth map (using Equation 3.3), then convert the calculated pixel coordinates into a 3D point (using Equation 3.4) and find its distance to the voxel center [90].

The advantage of the ray-casting approach is that each pixel is accessed only once. However, the obtained values then have to be assigned to voxels discretised according to the projected 3D points. Depending on the volume size, this might be excessively expensive. Therefore, we chose the second option for our implementation. In it every voxel is accessed only once and we have control over the memory consumption by choosing an appropriate voxel resolution. In addition, in this way a colour value can be assigned to each voxel, by simply reading the RGB value of the projected pixel from the colour image. Notably, this value is accurate only for voxels which coincide with the object surface, while for the other voxels this is the colour on the ray to the camera center.

Additionally to the signed distances, a binary weight is assigned to each voxel, signifying the level of confidence about its value. In this sense, voxels which have not been observed receive a weight of 0, whereas the rest have weight 1. In addition, a thin layer of thickness $\eta > 0$ behind the surface, denoting how solid the object is expected to be, also gets confidence 1.

For numerical stability, i.e. avoiding mixtures of very far and very near values, and since we are interested in the exact values only near the surface, the signed distances are scaled by a factor $\delta > 0$, which controls the dimension of the near-surface region, and truncated into the interval $[-1, 1]$.

Thus in addition to the distance field ϕ, a coloured truncated signed distance field has an associated weight field ω and an associated colour grid ζ. The equations below summarise the whole CTSDF generation process for any voxel center $\mathbf{x} = (x, y, z)^\top \in \mathbb{R}^3$ (D denotes a range image, while I is the corresponding colour image), while Figure 3.5 visualises the

truncation and weighting steps. Note that for brevity from here onwards, the term SDF will be used for referring to a CTSDF, unless otherwise specified in the context.

$$
\begin{aligned}
\phi_{non-truncated}(\mathbf{x}) &= D(\pi(\mathbf{x})) - z \\
\phi(\mathbf{x}) &= \begin{cases} sgn(\phi(\mathbf{x})) & \text{, if } |\phi(\mathbf{x})| \geq \delta \\ \dfrac{\phi(\mathbf{x})}{\delta} & \text{, otherwise} \end{cases} \\
\omega(\mathbf{x}) &= \begin{cases} 1 & \text{, if } \phi_{non-truncated}(\mathbf{x}) > -\eta \\ 0 & \text{, otherwise} \end{cases} \\
\zeta(\mathbf{x}) &= I(\pi(\mathbf{x}))
\end{aligned}
\tag{3.26}
$$

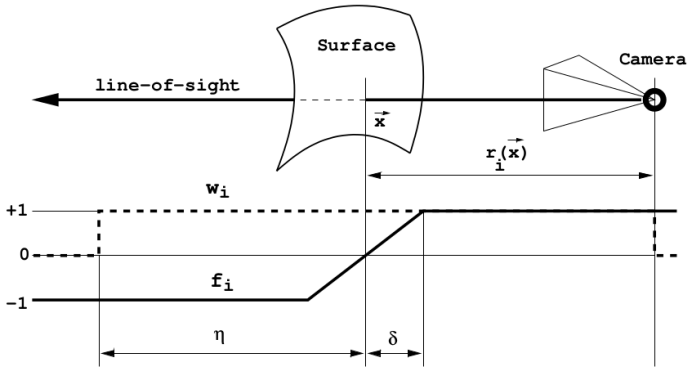


Figure 3.5.: TSDF generation [90]

Figure 3.6 shows several vertical cross-sections through the distance field of a synthetic depth map. The teddy has dimensions approximately $25 \times 30 \times 35$ cm. Therefore the following parameters were selected: voxel size of 2 mm, δ: 2 mm and η: 1 cm. Blue corresponds to a signed distance of -1, i.e. inside the object, red - to distance 1, i.e. outside the object. The remaining colours are interpolated between these values. There is always only a very thin band around the object boundary where the values are different from ± 1.

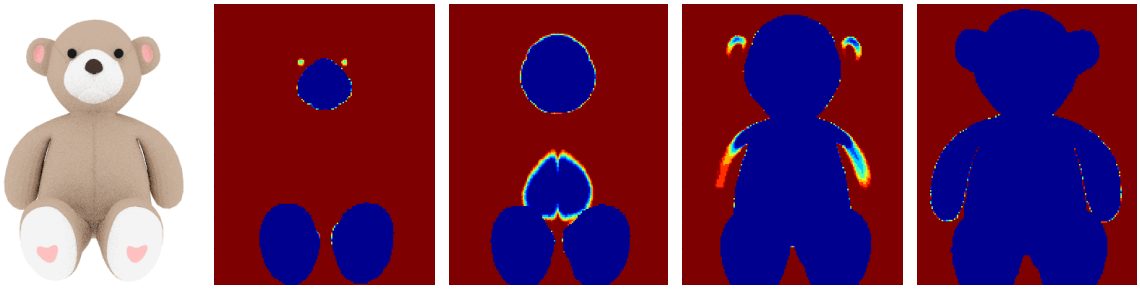


Figure 3.6.: Cross-sections through a CTSDF

3.4.3. Properties

The most important SDF property that we will make use of is the fact that its gradient at any surface location equals the unit surface normal at that point [62, 38]:

$$\|\nabla_{\mathbf{x}}\phi(\mathbf{x})\| = \|\overline{\mathbf{n}}(\mathbf{x})\| = 1, \qquad \forall \mathbf{x} \in \partial\Omega. \tag{3.27}$$

Intuitively, since the distance is a Euclidean measure, moving twice as close to the surface from a given point in space gives a signed distance value which is two times smaller, and thus the unity norm is preserved [62]. This relation is, however, not defined at points which are equally distant from more than one surface location, such as the center of a sphere. In such cases the gradient is undefined [38].

When handling this equation in a discrete setting, the gradient is calculated using a numerical scheme, such as central differences, and is, therefore, defined everywhere. However, because of loss of accuracy in the initial discretisation step, the norm might not be unity any more [62]. Thus, it has to be normalised in numerical implementations. Let $\mathbf{x} = (x, y, z)^\top$ be a 3D point representing the center of a voxel in the discretised grid. Then, the normal is calculated via the formula

$$\begin{aligned} \mathbf{n}(\mathbf{x}) &= \left(\frac{\partial\phi(\mathbf{x})}{\partial x}, \frac{\partial\phi(\mathbf{x})}{\partial y}, \frac{\partial\phi(\mathbf{x})}{\partial z} \right)^\top = \\ &= \left(\frac{\phi(x+1,y,z) - \phi(x-1,y,z)}{\Delta x}, \frac{\phi(x,y+1,z) - \phi(x,y-1,z)}{\Delta y}, \frac{\phi(x,y,z+1) - \phi(x,y,z-1)}{\Delta z} \right)^\top \end{aligned} \tag{3.28}$$

and the SDF spatial gradient becomes

$$\nabla_{\mathbf{x}}\phi(\mathbf{x}) = \overline{\mathbf{n}}(\mathbf{x}) = \frac{\mathbf{n}(\mathbf{x})}{\|\mathbf{n}(\mathbf{x})\|}. \tag{3.29}$$

Another important SDF property is its viewpoint independence. However, this is the case only for SDFs representing the whole object, i.e. when the multiple viewpoints are fused. When dealing with single-frame SDFs the voxels behind the surface are never seen, leading to the formation of viewpoint-dependent protrusions, as shown in Figure 3.7. Thus, these artefacts must be avoided in frame-to-frame registration. This is easily achieved thanks to the binary weights: since voxels behind the object are not observed, they have a weight of 0. So in registration, only voxels with non-zero weight can be taken into account.

In addition, the gradient at these locations is faulty and must not be used. Since the protrusions are always transitions between signed distance values of 1 and -1, an implementation that uses central differences for the gradient must simply discard voxels where any element of the gradient has an absolute value of 1.

Figure 3.7.: Formation of view-dependent protrusions in the generation of single-frame SDFs

3.4.4. Fusion

Once the camera poses have been accurately estimated, the separate depth measurements can be fused through their SDFs. The result is a view-independent implicit surface representation in a voxel grid, which can be meshed and rendered using the marching cubes method [50]. Associated colour is assigned from the RGB grid of the CTSDF.

Coloured Weighted Average

The most commonly used SDF fusion method was proposed by Curless and Levoy [14]. It is a simple weighted average, which can be formed incrementally as new measurements are integrated into the model. Let us denote this rolling weighted average at time step t by Φ_t and its associated weight field by W_t. Then the following formulas describe the update:

$$\begin{aligned} \Phi_{t+1}(\mathbf{x}) &= \frac{W_t(\mathbf{x})\Phi_t(\mathbf{x}) + \omega_{t+1}(\mathbf{x})\phi_{t+1}(\mathbf{x})}{W_t(\mathbf{x}) + \omega_{t+1}(\mathbf{x})} \\ W_{t+1}(\mathbf{x}) &= W_t(\mathbf{x}) + \omega_{t+1}(\mathbf{x}). \end{aligned} \tag{3.30}$$

If all the SDFs are available, the weighted average can be calculated as:

$$\phi(\mathbf{x}) = \frac{\sum\limits_{i\in\{1,\dots,\#SDFs\}} \omega_i(\mathbf{x})\phi_i(\mathbf{x})}{\sum\limits_{i\in\{1,\dots,\#SDFs\}} \omega_i(\mathbf{x})}, \qquad \omega(\mathbf{x}) = \frac{\sum\limits_{i\in\{1,\dots,\#SDFs\}} \omega_i(\mathbf{x})}{\#SDFs}. \tag{3.31}$$

It may turn out that certain voxels have not been seen from any viewpoint, so their cumulative weight is zero. This is likely to happen only with voxels which are inside the object. Therefore, for implementational purposes their signed distance is directly set to -1 and division by zero is avoided.

The result so far is a TSDF, which can be meshed and rendered. Colour information can be easily added using the RGB field, as suggested by Bylow *et al.* [7]. There is a rolling average for each of the three colour channels. The weights are, however, assigned according to the cosine of the angle θ between the line of sight and the surface normal at the given location, in a combination with the weight of the voxel itself. In this way larger certainty is given to colours of points whose normal is pointing towards the camera. The equations below summarise this procedure, with R, G and B representing the three colour voxel grids, and W^C - the associated colour weight grid. The equivalent formula given all SDFs is straightforward to derive in the same way as Equation 3.31.

$$\begin{aligned} w^c_{t+1}(\mathbf{x}) &= w_{t+1}(\mathbf{x})\cos(\theta_{t+1}(\mathbf{x})) \\ R_{t+1}(\mathbf{x}) &= \frac{W^C_t(\mathbf{x})R_t(\mathbf{x}) + w^c_{t+1}(\mathbf{x})r_{t+1}(\mathbf{x})}{W^C_t(\mathbf{x}) + w^c_{t+1}(\mathbf{x})} \\ G_{t+1}(\mathbf{x}) &= \frac{W^C_t(\mathbf{x})G_t(\mathbf{x}) + w^c_{t+1}(\mathbf{x})g_{t+1}(\mathbf{x})}{W^C_t(\mathbf{x}) + w^c_{t+1}(\mathbf{x})} \\ B_{t+1}(\mathbf{x}) &= \frac{W^C_t(\mathbf{x})B_t(\mathbf{x}) + w^c_{t+1}(\mathbf{x})b_{t+1}(\mathbf{x})}{W^C_t(\mathbf{x}) + w^c_{t+1}(\mathbf{x})}. \end{aligned} \tag{3.32}$$

TV-L^1 Minimisation

Although the weighted average is simple and cheap to compute, it does not always produce sufficiently smooth models. To overcome this issue Zach *et al.* [90] proposed a TV-L^1 variational energy optimisation framework. It combines an L^1-norm geometrical data fidelity term, minimising differences between per-voxel signed distances, and a total variation spatial regularisation term, which ensures that the resulting model is smooth. Schroers *et al.* [71] improve this method by introducing a normalisation factor ε, which prevents over-smoothing. Further, Kehl *et al.* [39] achieved even better geometrical fidelity by including a regulariser based on colour consistency of the CTSDFs. This is the method of choice for our final model generation. It is speeded-up by using the weighted average, explained in the previous section, as an initialisation.

Given n CTSDFs, the energy being minimised is as follows:

$$\begin{aligned} E(\phi,\zeta) &= \sum_{\mathbf{x}\in voxels} (D(\phi_{1..n},\omega_{1..n},\zeta_{1..n},\phi,\zeta) + w_{geom}S(\nabla\phi) + w_{colour}S(\nabla\zeta)) \\ S(\nabla u) &= |\nabla u| \\ D(\phi_{1..n},\omega_{1..n},\zeta_{1..n},\phi,\zeta) &= \frac{1}{\varepsilon + \sum_{i\in SDFs}\omega_i} \left(\sum_{i\in SDFs} \omega_i(|\phi-\phi_i| + |\zeta-\zeta_i|) \right), \end{aligned} \tag{3.33}$$

and the solutions for the optimal signed distance field ϕ and colour field ζ are the steady states of the following gradient descent equations [39]:

$$\begin{aligned} \partial\phi &= w_{geom} div(S'(\nabla\phi)) - \frac{\partial D}{\partial\phi}(\phi_{1..n},\omega_{1..n},\zeta_{1..n},\phi,\zeta) \\ \partial\zeta &= w_{colour} div(S'(\nabla\zeta)) - \frac{\partial D}{\partial\zeta}(\phi_{1..n},\omega_{1..n},\zeta_{1..n},\phi,\zeta). \end{aligned} \tag{3.34}$$

The weight factors of the regularisers w_{geom} and w_{colour} have to be carefully chosen, so that fine geometrical details are not degraded by over-smoothing. At the end of this step, a highly detailed, finely coloured 3D model is obtained. Figure 3.8 shows the difference between a coloured weighted average from 12 views and its corresponding smooth model after 20 iterations of TV-L^1 minimisation. Note the improvement on fine details, such as the horns.

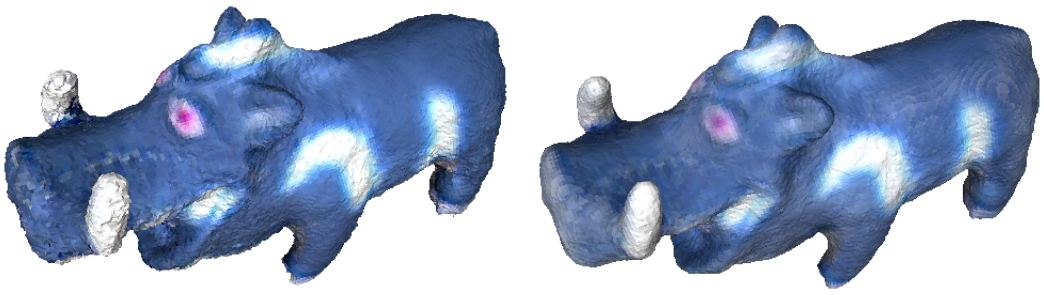


Figure 3.8.: Coloured weighted average (left) versus its corresponding TV-L^1 minimised model (right)

Part II.

Proposed Method

4. Signed Distance Field Registration

This chapter presents the CTSDF-based registration. First, the objective function including geometric, colour and surface orientation constraints is stated. Then, its application for the two pipeline stages, camera tracking and pose optimisation, is derived. Certain important implementational aspects are also highlighted.

4.1. Objective Function

The representation of choice for the proposed method is the coloured truncated signed distance field, as described in the preceding chapters. As our goal is a direct dense approach, we will always estimate motion directly using the per-voxel differences of the 3D grids of two SDFs. We will additionally incorporate colour and normal information for surface locations. The equations below summarise the separate energy components:

$$E_{geometric}(\xi) = \frac{1}{2} \sum_{\mathbf{v} \in voxels} \left(\phi_{reference}(\mathbf{v}) - (\phi_{current}(\xi))(\mathbf{v}) \right)^2 \tag{4.1}$$

$$E_{colour}(\xi) = \frac{1}{2} \sum_{\substack{\mathbf{v} \in surface \\ voxels}} \left(\zeta_{reference}(\mathbf{v}) - (\zeta_{current}(\xi))(\mathbf{v}) \right)^2 \tag{4.2}$$

$$E_{normals}(\xi) = \sum_{\substack{\mathbf{v} \in surface \\ voxels}} \left(1 - \overline{\mathbf{n}}_{reference}(\mathbf{v}) \cdot (\overline{\mathbf{n}}_{current}(\xi))(\mathbf{v}) \right) \tag{4.3}$$

Here ξ is the 6 DoF pose that is being estimated, and denotes either the twist or the augmented unit quaternion representation, as described in the section on rigid body transformations (*cf.* Section 3.3). ϕ is the signed distance field, ζ is its associated colour field, and $\overline{\mathbf{n}}$ is the field of unit normals, which equals the spatial gradient of the SDF (*cf.* Section 3.4.3). When ξ is applied to these grids as $\phi(\xi)$, $\zeta(\xi)$ and $\overline{\mathbf{n}}(\xi)$, it represents the *current* CTSDF generated from the respective pose. When not specified, this is zero translation and identity rotation, i.e. ϕ,ζ and $\overline{\mathbf{n}}$ denote a fixed *reference* CTSDF, which is the model to which the *current* one is aligned. Summation is done over voxels because the same transformation is applied to the whole grid.

The **geometric component** stems from the fact that given perfect alignment of the grids, they will have the same values at every voxel. The **photometric term** enforces that the surface colours of overlapping voxels must be the same when they are properly aligned. Therefore minimising the per-voxel sum of squared differences is optimal.

Similarly, the **normal vector component** reflects the fact that registered surfaces have the same orientation. There is one notable difference. While the distance and colour fields are scalar (when colour is represented as intensity or is split into its red, green and blue channels), the normals are vector quantities. Apart from the entailed implementational adjustments, the energy is also modified. The dot product of two unit vectors is a measure

of the cosine of the angle between them. It is one for vectors pointing in the same direction, and zero for perpendicular vectors. Therefore, when the SDFs are aligned, the dot product will be maximised. In order to include the term in the minimisation schemes, one (the largest possible unit vector dot product value) minus the dot product is taken. This is a more complex expression than the straightforward differences in Equations 4.1 and 4.2, however, it better reflects the properties of vector fields. It is possible to use a simple vector difference instead of the dot product, as was done by Masuda [53], but we expect that the dot product is a more robust measure.

Combining these terms yields the complete objective function definition:

$$E_{CTSDF}(\xi) = E_{geometric}(\xi) + w_{colour} E_{colour}(\xi) + w_{normals} E_{normals}(\xi), \tag{4.4}$$

where w_{colour} and $w_{normals}$ are weights controlling the influence attributed to the respective error terms. The optimal solution ξ^* is the rigid-body transformation that best aligns the two CTSDFs, i.e. which minimises their per-voxel difference:

$$\xi^* = \underset{\xi}{\operatorname{argmin}}\, E_{CTSDF}(\xi). \tag{4.5}$$

The numerical scheme for solving this energy will be derived in the following sections. Inspired from the related work, two optimisation schemes will be used: first-order Taylor linearisation and gradient descent. Each of them can be applied for solving the pairwise registration in the camera tracking setting and the multiview registration in the global pose optimisation scenario. We have experimentally found that the Taylor approach is more suitable in the absence of an initial pose estimate, as in frame-to-frame tracking, while the gradient descent scheme is more efficient in the multiview global optimisation, where approximate poses are already known. This is why each of the derivations will be described in its more relevant scenario. However, both schemes are generally applicable for both registration cases.

Implementation Note

In the case when single-frame CTSDFs are used, they exhibit the view-dependent protrusions that were described earlier (*cf.* Figure 3.7). This is why the binary weights need to be taken into account when working with this energy, modifying the formulas in the following way:

$$E_{weighted\_geometric}(\xi) = \frac{1}{2} \sum_{\mathbf{v} \in voxels} \left(\phi_{reference}(\mathbf{v})\, \omega(\mathbf{v}) - (\phi_{current}(\xi))(\mathbf{v})\, (\omega(\xi))(\mathbf{v}) \right)^2 \tag{4.6}$$

$$E_{weighted\_colour}(\xi) = \frac{1}{2} \sum_{\substack{\mathbf{v} \in surface \\ voxels}} \left(\zeta_{reference}(\mathbf{v})\, \omega(\mathbf{v}) - (\zeta_{current}(\xi))(\mathbf{v})\, (\omega(\xi))(\mathbf{v}) \right)^2 \tag{4.7}$$

$$E_{weighted\_normals}(\xi) = \sum_{\substack{\mathbf{v} \in surface \\ voxels}} \left(1 - (\overline{\mathbf{n}}_{reference}(\mathbf{v})\, \omega(\mathbf{v})) \cdot ((\overline{\mathbf{n}}_{current}(\xi))(\mathbf{v})\, (\omega(\xi))(\mathbf{v})) \right) \tag{4.8}$$

This, however, is an overload of the notation, which is very simple to implement. All that needs to be done is a straightforward multiplication of the distance, colour and normal fields with the grid of weights. Therefore, we will omit the binary weight factors in the following derivations, so that the focus is on the more essential aspects. We will also eliminate the voxel $\mathbf{v}$ from the formulas, because all operations are applied on all voxels, and thus mentioning $\mathbf{v}$ does not bring additional information.

4.2. Registration Derivation for Camera Tracking

For simplicity of notation, we will derive each of the three energy components separately. They are then combined straightforwardly via the weights w_{colour} and $w_{normals}$.

Geometric Component

Let us view the signed distance field ϕ as a function of the pose ξ from which it was generated. Let $\nabla_{\mathbf{x}}\phi$ denote the spatial gradient of a signed distance field, and $\nabla_{\xi}\phi$ be its Jacobian with respect to the pose ξ, which can be represented by a twist or by an AUQ. It can be derived as follows using the chain rule:

$$\nabla_{\xi}\phi = \frac{\mathrm{d}\phi}{\mathrm{d}\xi} = \frac{\mathrm{d}\phi}{\mathrm{d}\mathbf{v}}\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\xi} = \nabla_{\mathbf{x}}\phi\, J(\mathbf{v},\xi). \tag{4.9}$$

Depending on the choice of pose representation $\nabla_{\xi}\phi \in \mathbb{R}^{1\times 6}$ for twists or $\nabla_{\xi}\phi \in \mathbb{R}^{1\times 7}$ for AUQs.

We represent the distance field by its first-order Taylor approximation around the current pose estimate $\xi^{(k)}$:

$$\phi(\xi) = \phi\left(\xi^{(k)}\right) + \nabla_{\xi}\phi\left(\xi^{(k)}\right)\left(\xi - \xi^{(k)}\right) = \phi\left(\xi^{(k)}\right) - \nabla_{\xi}\phi\left(\xi^{(k)}\right)\xi^{(k)} + \nabla_{\xi}\phi\left(\xi^{(k)}\right)\xi. \tag{4.10}$$

Substituting this into Equation 4.1 leads to the following expression, in which the terms dependent on ξ are written in a different colour than the free ones:

$$\begin{aligned}
E_{geometric}(\xi) &= \frac{1}{2}\sum_{voxels}\left(\phi_{ref} - \phi_{curr}\left(\xi\right)\right)^2 = \\
&= \frac{1}{2}\sum_{voxels}\left(\phi_{ref} - \phi_{curr}\left(\xi^{(k)}\right) + \nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi^{(k)} - \nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi\right)^2 = \\
&= \frac{1}{2}\sum_{voxels}\Bigg(\left(\phi_{ref} - \phi_{curr}\left(\xi^{(k)}\right) + \nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi^{(k)}\right)^2 + \\
&\qquad + \xi^{\top}\nabla_{\xi}^{\top}\phi_{curr}\left(\xi^{(k)}\right)\nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi - \\
&\qquad - 2\left(\phi_{ref} - \phi_{curr}\left(\xi^{(k)}\right) + \nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi^{(k)}\right)\nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi\Bigg).
\end{aligned} \tag{4.11}$$

We are looking for the value of ξ that minimises the energy, therefore we need the solution for which the derivative with respect to pose is zero. Deriving the energy with respect to the pose is done as follows:

$$\begin{aligned}
\frac{\mathrm{d}E_{geometric}}{\mathrm{d}\xi} &= \frac{1}{2}\sum_{voxels}\Bigg(0 + 2\nabla_{\xi}^{\top}\phi_{curr}\left(\xi^{(k)}\right)\nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi - \\
&\qquad - 2\left(\phi_{ref} - \phi_{curr}\left(\xi^{(k)}\right) + \nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi^{(k)}\right)\nabla_{\xi}^{\top}\phi_{curr}\left(\xi^{(k)}\right)\Bigg) = \\
&= \sum_{voxels}\Bigg(\nabla_{\xi}^{\top}\phi_{curr}\left(\xi^{(k)}\right)\nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi - \\
&\qquad - \left(\phi_{ref} - \phi_{curr}\left(\xi^{(k)}\right) + \nabla_{\xi}\phi_{curr}\left(\xi^{(k)}\right)\xi^{(k)}\right)\nabla_{\xi}^{\top}\phi_{curr}\left(\xi^{(k)}\right)\Bigg).
\end{aligned} \tag{4.12}$$

This is a very convenient representation, because it can be easily transformed into a linear system:

$$\begin{aligned}
\mathbf{A} &= \sum_{voxels} \left(\nabla_\xi^\top \phi_{curr}\left(\xi^{(k)}\right) \; \nabla_\xi \phi_{curr}\left(\xi^{(k)}\right) \right) \\
\mathbf{b} &= \sum_{voxels} \left(\left(\phi_{ref} - \phi_{curr}\left(\xi^{(k)}\right) + \nabla_\xi \phi_{curr}\left(\xi^{(k)}\right) \xi^{(k)}\right) \nabla_\xi^\top \phi_{curr}\left(\xi^{(k)}\right) \right) \\
\frac{\mathrm{d}E_{geometric}}{\mathrm{d}\xi} &= \mathbf{A}\,\xi - \mathbf{b} \\
\Rightarrow \xi^* &= \mathbf{A}^{-1}\,\mathbf{b}
\end{aligned} \tag{4.13}$$

The optimal solution ξ^* is very cheap to compute, since $\mathbf{A} \in \mathbb{R}^{6\times6}$, $\mathbf{b} \in \mathbb{R}^{6\times1}$ for twists or $\mathbf{A} \in \mathbb{R}^{7\times7}$, $\mathbf{b} \in \mathbb{R}^{7\times1}$ for AUQs, and thus the matrix inverse is an inexpensive operation in this case.

We have found that for reasons of numerical stability, it is better to iteratively take steps towards the optimal solution ξ^* from the current one $\xi^{(k)}$ as follows:

$$\xi^{(k+1)} = \xi^k + \beta \left(\xi^* - \xi^{(k)}\right), \tag{4.14}$$

where β is the chosen step size. This neat formula concludes the derivation of the geometric component of signed distance field registration for camera tracking.

Colour Component

The colour grid ζ of a CTSDF can be split into three scalar fields, one for its red, green and blue channel. Exactly the same derivation as for the distance field can be applied to each of these grids. Thus, we will not explicitly write the formulas again. Because the complete objective function (Equation 4.4) is a simple weighted summation of the error terms, it is only needed to add the derived $\mathbf{A}_{colour}$ and $\mathbf{b}_{colour}$ to $\mathbf{A}$ and $\mathbf{b}$ respectively.

Normal Vector Component

As the grid of unit normals is a vector field, the above formulas need to be slightly modified in order to include the surface orientation constraints. First, we carry out a similar Taylor approximation step for the grid of surface normals:

$$\overline{\mathbf{n}}(\xi) = \overline{\mathbf{n}}\left(\xi^{(k)}\right) + \nabla_\xi \overline{\mathbf{n}}\left(\xi^{(k)}\right)\left(\xi - \xi^{(k)}\right) = \overline{\mathbf{n}}\left(\xi^{(k)}\right) - \nabla_\xi \overline{\mathbf{n}}\left(\xi^{(k)}\right)\xi^{(k)} + \nabla_\xi \overline{\mathbf{n}}\left(\xi^{(k)}\right)\xi. \tag{4.15}$$

Here $\nabla_\xi \overline{\mathbf{n}}\left(\xi^{(k)}\right)$ (of dimension 3×6 or 3×7, depending on the pose representation) is the Jacobian of a normal with respect to pose, which can again be found by an application of the chain rule:

$$\nabla_\xi \overline{\mathbf{n}} = \frac{\mathrm{d}\overline{\mathbf{n}}}{\mathrm{d}\xi} = \frac{\mathrm{d}\overline{\mathbf{n}}}{\mathrm{d}\mathbf{v}}\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\xi} = \nabla_{\mathbf{x}} \overline{\mathbf{n}}\; J(\mathbf{v}, \xi). \tag{4.16}$$

The term $\nabla_{\mathbf{x}} \overline{\mathbf{n}} \in \mathbb{R}^{3\times3}$ is the spatial gradient of a normal vector, which evaluates how the orientation changes with location, i.e. it is a measure of curvature:

$$\nabla_{\mathbf{x}} \overline{\mathbf{n}} = \begin{pmatrix} \partial\overline{\mathbf{n}}_x/\partial x & \partial\overline{\mathbf{n}}_x/\partial y & \partial\overline{\mathbf{n}}_x/\partial z \\ \partial\overline{\mathbf{n}}_y/\partial x & \partial\overline{\mathbf{n}}_y/\partial y & \partial\overline{\mathbf{n}}_y/\partial z \\ \partial\overline{\mathbf{n}}_z/\partial x & \partial\overline{\mathbf{n}}_z/\partial y & \partial\overline{\mathbf{n}}_z/\partial z \end{pmatrix} = \begin{pmatrix} \frac{\partial\overline{\mathbf{n}}}{\partial x} & \frac{\partial\overline{\mathbf{n}}}{\partial y} & \frac{\partial\overline{\mathbf{n}}}{\partial z} \end{pmatrix}. \tag{4.17}$$

Now the same procedure follows, whereby we substitute the Taylor approximation into the original energy component (Equation 4.3), and then derive with respect to the pose:

$$\begin{aligned} E_{normals}(\xi) &= \sum_{\substack{surface \\ voxels}} \left(1 - \overline{\mathbf{n}}_{ref} \cdot \overline{\mathbf{n}}_{curr}(\xi) \right) = \\ &= \sum_{\substack{surface \\ voxels}} \left(1 - \overline{\mathbf{n}}_{ref} \cdot \left(\overline{\mathbf{n}}_{curr}\left(\xi^{(k)}\right) - \nabla_{\xi}\overline{\mathbf{n}}_{curr}\left(\xi^{(k)}\right)\xi^{(k)} + \nabla_{\xi}\overline{\mathbf{n}}_{curr}\left(\xi^{(k)}\right)\xi \right) \right). \end{aligned} \tag{4.18}$$

Next, we apply the product rule in order to obtain the derivative of a dot product $(\mathbf{a} \cdot \mathbf{b})' = \mathbf{a}' \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{b}'$ with respect to each component j of the pose vector:

$$\begin{aligned} \frac{\mathrm{d}E_{normals}}{\mathrm{d}\xi_j} &= \sum_{\substack{surface \\ voxels}} \left(0 - \overline{\mathbf{n}}_{ref} \cdot \left(\nabla_{\xi}\overline{\mathbf{n}}_{curr}\left(\xi^{(k)}\right) \right) \delta_j \right) = \\ &= \sum_{\substack{surface \\ voxels}} - \overline{\mathbf{n}}_{ref} \cdot \left(\nabla_{\mathbf{x}}\overline{\mathbf{n}}_{curr}\left(\xi^{(k)}\right) \; J(\mathbf{v}, \xi^{(k)}) \; \delta_j \right), \end{aligned} \tag{4.19}$$

where δ_j is a 6×1 or 7×1 vector of zeros, with j$^{\text{th}}$ component equal to 1, which enforces that the dot product is between two 3-element vectors. The full derivative $\mathrm{d}E_{normals}/\mathrm{d}\xi$ is obtained by stacking the separate j-components.

We observe that the expression above does not depend on ξ, which is expected, since normals only influence through the change of their orientation and not of their location. Thus, the normal vector error term only contributes to the vector $\mathbf{b}$ in the linear system from Equation 4.13. A disadvantage of this formulation is that no equation system could be build only from normal vector constraints, since the matrix $\mathbf{A}$ would be undefined, which was not the case for the colour constraints. However, $E_{normals}$ can be used to stabilise any energy that can be derived into a linear system of the form of Equation 4.13. Moreover, since the normal vector field is the gradient of the signed distance field, the same derivation can be obtained via a second-order Taylor approximation of the geometric energy component, which would lead to a fusion of the geometric and surface orientation constraints.

With this the derivation of all objective function terms is concluded and they can be combined in order to yield a robust pose estimate given no prior knowledge of the transformation between frames.

4.3. Global Optimisation Derivation for 3D Reconstruction

Having derived many of the involved derivatives and Jacobians, it is now easy to present the gradient descent derivation based on a direct derivative with respect to pose:

$$\xi^{(k+1)} = \xi^{(k)} - \alpha \nabla E_{CTSDF}(\xi^{(k)}) = \xi^{(k)} - \alpha \frac{\mathrm{d}E_{CTSDF}}{\mathrm{d}\xi^{(k)}}, \tag{4.20}$$

where α is the iteration step size.

Geometric and Colour Components

As we have seen, the derivation concerning scalar fields is the same, so the geometric and colour terms will be explained together below, using the distance field as an example:

$$\begin{aligned}
\frac{\mathrm{d}E_{geometric}}{\mathrm{d}\xi} &= \frac{1}{2} \sum_{voxels} 2\left(\phi_{ref} - \phi_{curr}\left(\xi\right)\right) \frac{\mathrm{d}\left(-\phi_{curr}(\xi)\right)}{\mathrm{d}\xi} = \\
&= \sum_{voxels} \left(\phi_{curr}\left(\xi\right) - \phi_{ref}\right) \frac{\mathrm{d}\phi_{curr}(\xi)}{\mathrm{d}\xi} = \\
&= \sum_{voxels} \left(\phi_{curr}\left(\xi\right) - \phi_{ref}\right) \frac{\mathrm{d}\phi_{curr}(\xi)}{\mathrm{d}\mathbf{v}} \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\xi} = \\
&= \sum_{voxels} \left(\phi_{curr}\left(\xi\right) - \phi_{ref}\right) \nabla_{\mathbf{x}} \phi_{curr}(\xi)\ J(\mathbf{v}, \xi) = \\
&= \sum_{voxels} \left(\phi_{curr}\left(\xi\right) - \phi_{ref}\right) \nabla_{\xi} \phi_{curr}(\xi).
\end{aligned} \tag{4.21}$$

Hence, we obtain an elegant formula for the gradient descent step:

$$\xi^{(k+1)} = \xi^{(k)} - \alpha \sum_{voxels} \left(\phi_{curr}\left(\xi^{(k)}\right) - \phi_{ref}\right) \nabla_{\xi} \phi_{curr}\left(\xi^{(k)}\right) \tag{4.22}$$

Normal Vector Component

The derivative of the surface orientation error term yields exactly the same result as in the Taylor expansion case. Here it is easier to perceive the derivation, so we present it again:

$$\begin{aligned}
\frac{\mathrm{d}E_{normals}}{\mathrm{d}\xi_j} &= \sum_{\substack{surface \\ voxels}} - \frac{\mathrm{d}\overline{\mathbf{n}}_{ref}}{\mathrm{d}\xi_j} \cdot \overline{\mathbf{n}}_{curr}(\xi) - \overline{\mathbf{n}}_{ref} \cdot \frac{\mathrm{d}\overline{\mathbf{n}}_{curr}(\xi)}{\mathrm{d}\xi_j} = \\
&= \sum_{\substack{surface \\ voxels}} - \overline{\mathbf{n}}_{ref} \cdot \frac{\mathrm{d}\overline{\mathbf{n}}_{curr}(\xi)}{\mathrm{d}\xi_j} = \sum_{\substack{surface \\ voxels}} - \overline{\mathbf{n}}_{ref} \cdot \left(\frac{\mathrm{d}\overline{\mathbf{n}}_{curr}(\xi)}{\mathrm{d}\mathbf{v}} \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\xi_j} \right) = \\
&= \sum_{\substack{surface \\ voxels}} - \overline{\mathbf{n}}_{ref} \cdot \left(\frac{\mathrm{d}\overline{\mathbf{n}}_{curr}(\xi)}{\mathrm{d}\mathbf{v}} \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}\xi} \frac{\mathrm{d}\xi}{\mathrm{d}\xi_j} \right) = \sum_{\substack{surface \\ voxels}} - \overline{\mathbf{n}}_{ref} \cdot \left(\nabla_{\mathbf{x}} \overline{\mathbf{n}}_{curr}(\xi)\, J(\mathbf{v}, \xi)\ \delta_j \right).
\end{aligned} \tag{4.23}$$

Having determined all components, they can be combined by weighted summation in order to iteratively refine the camera pose.

5. 3D Reconstruction Pipeline

This chapter presents the way the two registration components, derived in Chapter 4, come together to produce a tracked camera trajectory and, subsequently, a refined 3D model. In addition, the preprocessing operations on colour and depth data are explained. The employed ways of model generation are also outlined.

5.1. Overview

The introductory chapter presented the general components of a 3D reconstruction system in Figure 1.2, identifying the two major stages: camera tracking via coarse registration, and global pose optimisation via refinement of the alignment. However, there exist numerous possibilities for the exact methodology of each one of them. Moreover, these components need to be connected to each other and to interact with the sensor and the output device. These details, together with the goals set out at the start of the thesis (*cf.* section 1.2), lead to the development of the pipeline depicted in Figure 5.1.

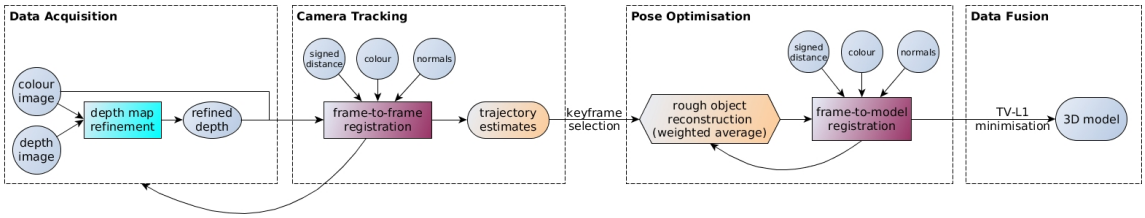


Figure 5.1.: Unified Pipeline for 3D Object Reconstruction

The proposed unified pipeline consists of four stages, which will be further explained in the remaining sections of this chapter:

- **Data Acquisition**: Interface between the images produced by the depth sensor and the associated colour camera, and the reconstruction system. Depending on the type of device, preprocessing for noise removal from the range data might be applied.
- **Camera Tracking**: Generating a camera trajectory using frame-to-frame registration based on the optimisation method described in Section 4.2. The pipeline so far can be used as a stand-alone component for camera tracking.
- **Pose Optimisation**: Refining the poses of a selected set of keyframes from the tracked trajectory using frame-to-model registration as outlined in Section 4.3. The global model is a weighted average of the keyframes' SDFs, which can be rendered as a 3D model, however, of a lower quality than the one obtained at the end of the pipeline.
- **Data Fusion**: Generation of a smooth model via TV-L^1 minimisation.

5.2. Depth Map Refinement

As mentioned in the background section on RGB-D cameras (*cf.* Section 3.1), although industrial structured light depth sensors, such as the Siemens Global Inspection System, provide extremely precise measurements, consumer-level devices, like the Microsoft Kinect, have a much lower accuracy, which decreases quadratically with distance.

Figure 3.2 testified that preprocessing and noise reduction will be beneficial when working with depth images produced by Kinect-like depth sensors. As the values become extremely unreliable with increasing distance, it is advisory to use a cutoff limit, after which the sensor readings are considered unreliable [77]. For the Kinect sequences presented in the evaluation section, this limit was set to 2 m. This, however, is not a major factor, since the objects of interest usually occupied a region of space closer than this threshold.

The cutoff procedure is, nonetheless, not sufficient, as there are random errors in the remaining depth map regions. One option to tackle this problem is classical noise removal using edge-preserving filters, such as the bilateral filter [80]. This technique is employed in the original *KinectFusion* approach [59]. Although it keeps depth discontinuities thanks to its range and low-pass domain components, which take into account the photometric and geometric similarity between pixels respectively, it does not account for the registration errors between colour and depth images outputted by RGB-D sensors. These errors mean that there is a slight misalignment between the edge positions in the colour and depth maps, which would result in incorrectly assigned RGB values on subsequently produced point clouds and 3D meshes. In addition, structured light sensors often fail to deliver a reading at depth discontinuities, such as edges, leading to holes in the depth maps (*cf.* the regions marked in purple in Figure 3.3). Consequently, research efforts have been targeting these problems by using joint bilateral filters, which apply the depth map smoothing steered by a weight field, calculated on basis of the colour image [55, 56, 11]. These methods lead to very high quality results, but usually consist of many complex stages, which take significant computational time. The work by Le *et al.* [48] produces one of the most compelling results and provides an extensive comparison with similar approaches.

The depth refinement method chosen for this thesis is also among the ones that make use of the colour image as support. It is a modification of the work by Vijayanagar *et al.* [85], which iteratively aligns the edges from the depth map with those from the colour map, using *anisotropic diffusion with conduction coefficients based on the RGB values*. It is based on a coarse-to-fine hierarchy, at the lowest resolution of which the missing data from the original depth map is recovered.

In short, their proposed approach creates a downsampled pyramid of the colour and depth image pair. An associated mask is determined at each level, indicating the depth map holes, edges and near-edge pixels, since these are unreliable values. Holes in the depth map are filled at the coarsest pyramid resolution only, using simple neighborhood averaging. Next, the pyramid is processed and upsampled in the reverse direction until a depth map of the original size is obtained. This processing is the actual anisotropic diffusion step, whereby conduction coefficients are calculated from the colour image, based on the similarity between neighboring pixels. The new depth value for each pixel is a sum of its old value and the values of its neighbors, weighted according to the conduction coefficients. Finally, the values are adjusted so that the newly produced quantities are from the set of initially existing depth values.

Algorithm 1: Depth Map Refinement Algorithm

Inputs : raw depth map D (size $w \times h$)
colour image I (size $w \times h$)
Output : refined depth map $\widetilde{D}$ (size $w \times h$)
Additional parameters: number of pyramid levels $pyramid\_levels$
number of anisotropic diffusion iterations per level ad_iters

// construct the pyramid
for $level \leftarrow 1$ **to** $pyramid\_levels$ **do**
  $D_{level} \leftarrow$ downsample $D_{level-1}$ by a factor of 2
  $M_{level} \leftarrow$ mask for the holes and Laplacian-detected edges of D_{level}
  $I_{level} \leftarrow$ downsample $I_{level-1}$ by a factor of 2
end

// optionally close holes
$\widetilde{D}_{pyramid\_levels} \leftarrow$ hole-filled $D_{pyramid\_levels}$

// anisotropic diffusion upsampling
for $level \leftarrow pyramid\_levels$ **to** 1 **do**
  compute conduction coefficients from I_{level}
  $\widetilde{D}_{level} \leftarrow$ execute ad_iters rounds of anisotropic diffusion
  $D_{level-1} \leftarrow$ joint bilateral upsampling by a factor of 2 on $\widetilde{D}_{level}$ and I_{level}
end

The reader is referred to the original paper for a detailed description of the separate stages, while Algorithm 1 gives an overview of the procedure implemented in this work. We introduced several notable modifications. Most importantly, although performing all the outlined steps leads to visually impressive results, there is a risk that the new values might cause more inaccuracies in subsequent handling of the depth maps for SDF creation. This is especially dangerous in the hole-filled regions. Thus, in order not to create any false data, we propose to either completely omit the hole filling step, or to execute only one iteration thereof, whereby only the outermost pixels at the contour of hole regions are filled. The other notable modification is the replacement of the upsampling followed by an adjustment of depth values to the previously existing set. Instead, we apply a joint bilateral upsampling step [44], which produces smoother depth maps without the expensive procedure of storing and looking up the set of possible values. Finally, again in order not to create false values, we tuned the parameters suggested in the original paper: the number of anisotropic diffusion iterations was decreased from 35 to 20 and the pyramid contained only 3 instead of 4 levels (mainly because the reconstructed objects were small enough not to require the extra level). Notably, we empirically found the best standard deviation of the Gaussian function that is used for calculating conduction coefficients to be 6.0, which is exactly the value suggested in the paper.

Although the original approach by Vijayanagar *et al.* is targeted for real-time processing using OpenCL and GPU parallelisation, the implemented modified version was running in 0.1 - 0.2 seconds entirely on the CPU. Figure 5.2 displays the effect of the depth refinement procedure on a sample Kinect RGB-D pair, taken from the *fr3/teddy* 3D object reconstruction sequence of the RGB-D Benchmark [78]. One can see the reduced amount

of missing data around the head, neck and feet of the teddy bear, in addition to the overall smoother transition between neighbouring pixels. With such improved input, the SDF registration can commence with boosted conditions for convergence, as outlined in the following sections.

RGB Image | Original Depth | Refined Depth

Figure 5.2.: Effect of the depth refinement algorithm

5.3. Camera Tracking

In the chapters so far we have reasoned in favour of frame-to-frame camera tracking in an unconstrained setting via implicit-to-implicit registration. Figure 5.3 depicts in detail the procedure we propose.

Alignment is always done between the CTSDF of two frames. The first camera pose is fixed to be identity rotation and zero translation, and therefore its voxel grid remains unchanged over iterations. Its depth map is projected into 3D space in order to determine the bounding volume, which is subsequently padded by a small margin (e.g. 2cm) on each side, in order to allow for movement of the other SDF.

The initial guess for the pose of the second frame is also identity. Over iterations it is gradually refined according to the energy minimisation scheme based on the linear system described in Section 4.2.

Convergence is determined based on a threshold for the translational part of a pose increment. This is sufficient to reflect the change to the whole pose, since the translation is influenced by the rotational change [78]. In this sense, if the translational component falls below a preset threshold (0.01 mm), the optimisation has converged, and the determined pose is returned. If this does not happen within a fixed maximum number of iterations, failure is detected and the depth map is discarded as unreliable. The next range image is then registered against the last reliable frame. However, in order to keep the number of poses in the trajectory consistent with the number of RGB-D pairs, an identity increment is added to the trajectory in failure cases. Algorithm 2 summarises the process.

Figure 5.4 displays a cross-section through the residual $E_{reference} - E_{current}$ over pose estimation iterations. We chose to show an extreme case of registration, where the initial difference was about 10°, so that the residual is clearly visible. In the images green denotes zero, which is the desired difference. Red corresponds to a difference in truncated signed distances of 2, while blue stands for -2. The sequence shows every 5^{th} iteration, with the last image corresponding to iteration 41, where convergence was detected. One can

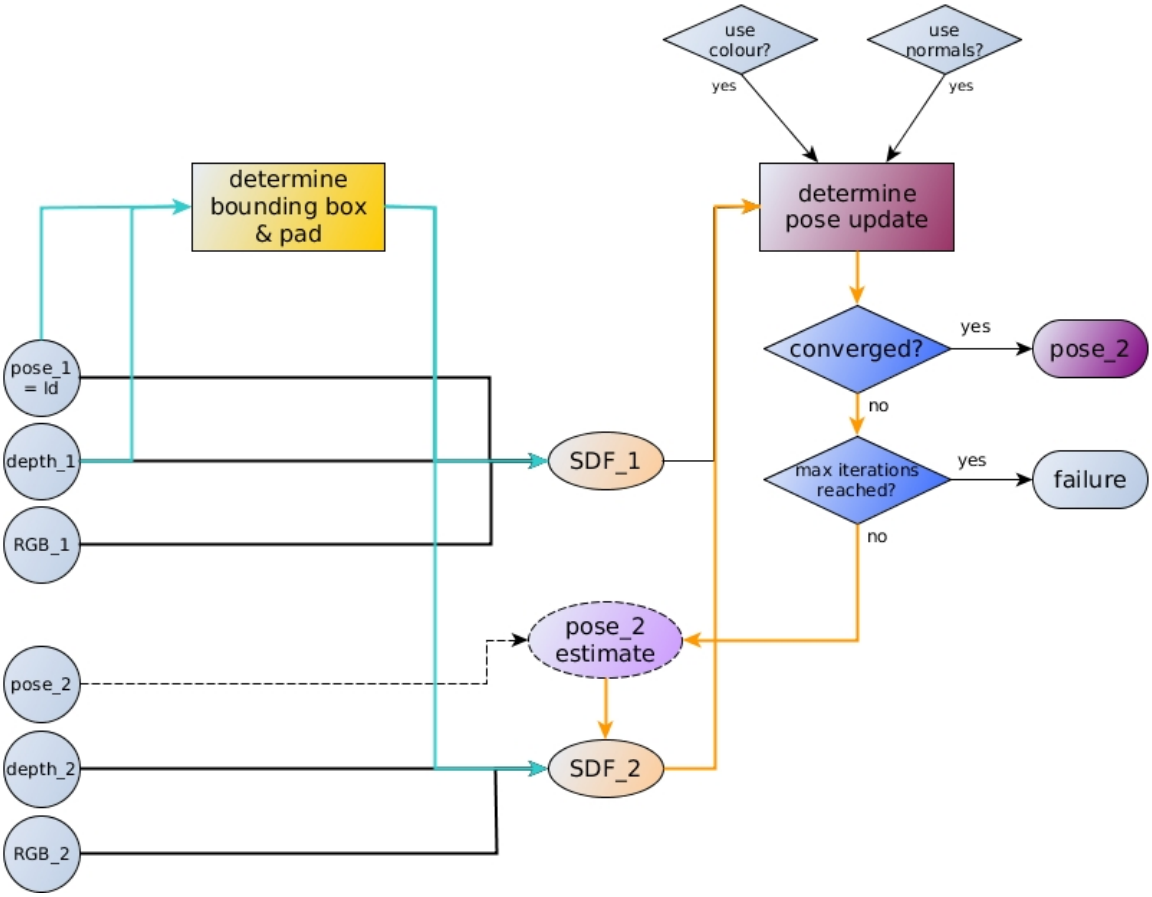


Figure 5.3.: Camera Tracking Flowchart

observe the large residual at the start of registration, which quickly decreases in the initial stages. From iteration 25 onwards only little difference is noticed in this cross-section, however, improvements occur in other regions of the volume, since no convergence has been detected yet. The final residual is not entirely perfect on the feet of the teddy bear, because the initial configuration was very extreme. In usual tracking cases, where the starting angular error is not more than 3-5°, the end error will be even smaller.

5.4. Global Pose Optimisation

The goal of multiview pose refinement is to improve the pose estimates of a selected set of keyframes from the previously determined trajectory. We propose to solve this problem via frame-to-model registration in the implicit-to-implicit CTSDF setting, following the gradient descent scheme outlined in Section 4.3. The global model is a coloured weighted average of the separate CTSDFs, generated as explained in Section 3.4.4. This is a convenient way to encompass all range data into one, aiming for the best consensus between all measurements.

Algorithm 2: Camera Tracking Algorithm (Frame-to-frame Registration)

Inputs : target RGB-D pair I_{target}, D_{target}
source RGB-D pair I_{source}, D_{source}
Output : pose of the source pair $\mathbf{P}$
Additional parameters: weight for the colour component of the energy w_{colour}
weight for the normals component of the energy $w_{normals}$
threshold for the translational update $convergence\_threshold$
maximum number of iterations max_iters

$volume \leftarrow determine\_padded\_bounding\_box(D_{target})$
$CTSDF_{target} \leftarrow generate\_ctsdf(D_{target}, I_{target}, volume, Identity)$

$\mathbf{P} \leftarrow Identity$
$iter \leftarrow 0$
while $iter < max\_iters$ **do**
  // generate the target CTSDF from the new pose estimate
  $CTSDF_{source} \leftarrow generate\_ctsdf(D_{source}, I_{source}, volume, P)$
  // execute an iteration of the linear-system based alignment
  $\mathbf{P}_{increment} \leftarrow lse\_alignment(CTSDF_{target}, CTSDF_{source}, w_{colour}, w_{normals})$
  // apply the increment
  $\mathbf{P} \leftarrow \mathbf{P}_{increment}\ \mathbf{P}$
  // check for convergence
  if $\|translational\_component(\mathbf{P}_{increment})\| < convergence\_threshold$ **then**
    return P
  end
  $iter \leftarrow iter + 1$
end
// if not converged: return identity, signifying failure
return $Identity$

Since global registration is extremely sensitive to the quality of the initial state [74], we employ a **coarse-to-fine scheme**. The optimisation starts with a large voxel size (4 or 2 mm). One alignment iteration is carried out for each single-frame CTSDF as source and the weighted average as destination. Once all pose updates are determined, they are simultaneously applied. The pose of the first camera remains fixed to identity (no rotation, zero translation) throughout the whole procedure, providing a common reference frame for the multiple views.

In the next iteration, the single-frame SDFs are generated from the new pose estimates, while the weighted average remains the same, since otherwise the objective function would be modified. However, in order to reflect the changes in the global setup, the weighted average is recalculated every several cycles (e.g. on every 10^{th} iteration). As soon as all translational pose updates fall below a preset threshold, convergence is detected. Then the next, finer level of the pyramid is taken, where the procedure is repeated.

Note that there is no need for a final voxel size much smaller than 1-2 mm, since the improvement is limited by the depth map resolution, and decreasing the voxel size would lead to accessing the same pixels repeatedly, thus not bringing additional information. Figure 5.5 visualises the process, while Algorithm 3 summarises it for implementation.

5.5. TV-L^1 Minimisation for Final Model Generation

Although the coloured weighted average used in the pose optimisation step is a model, which can be rendered via techniques such as marching cubes [50], it is often not sufficiently smooth, revealing even the smallest inaccuracies in pose estimation. Therefore, in a final step a variational TV-L^1 minimisation using geometric and colour regularisation is carried out, as described in Section 3.4.4.

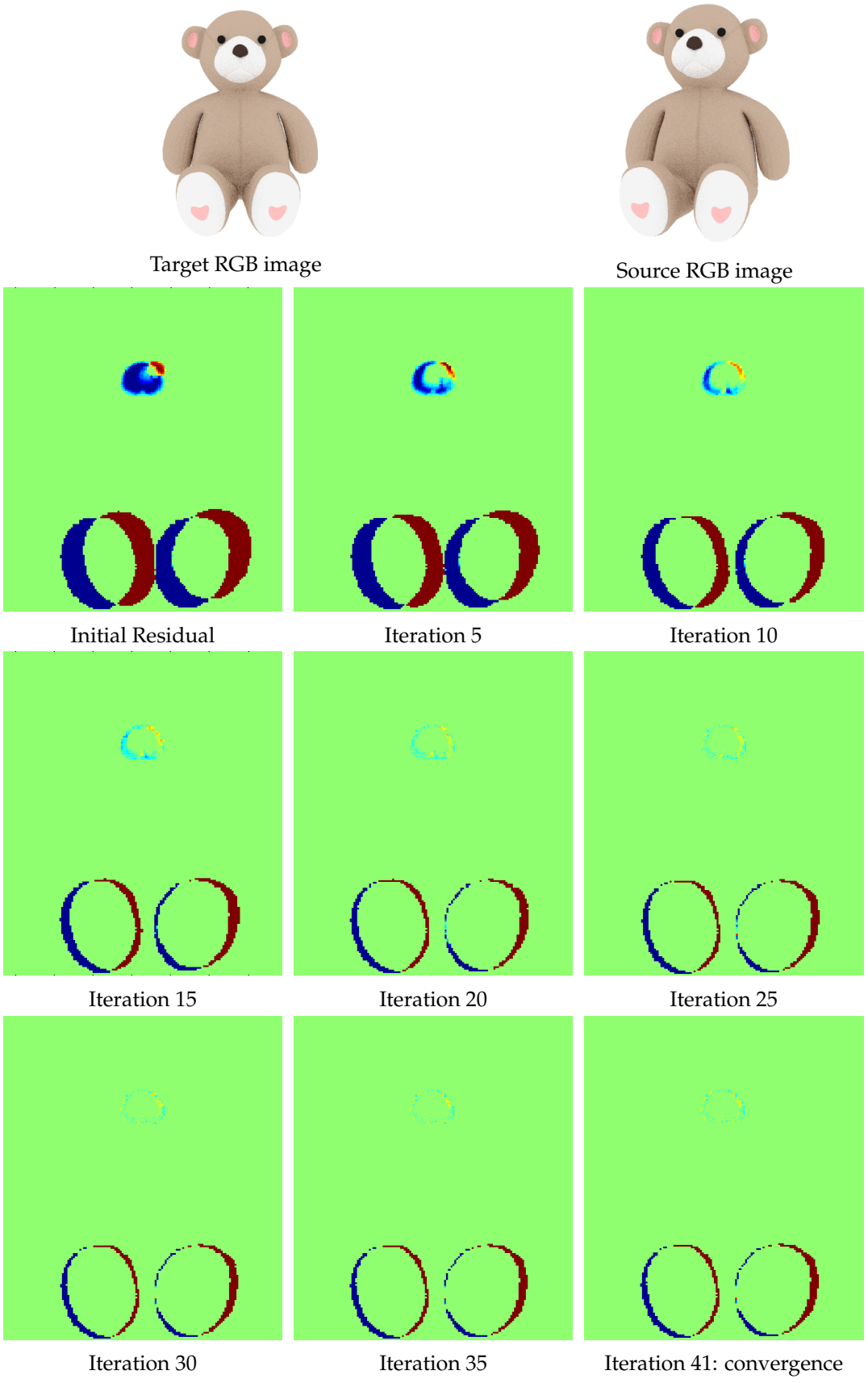


Figure 5.4.: Evolution of the residual during pose estimation

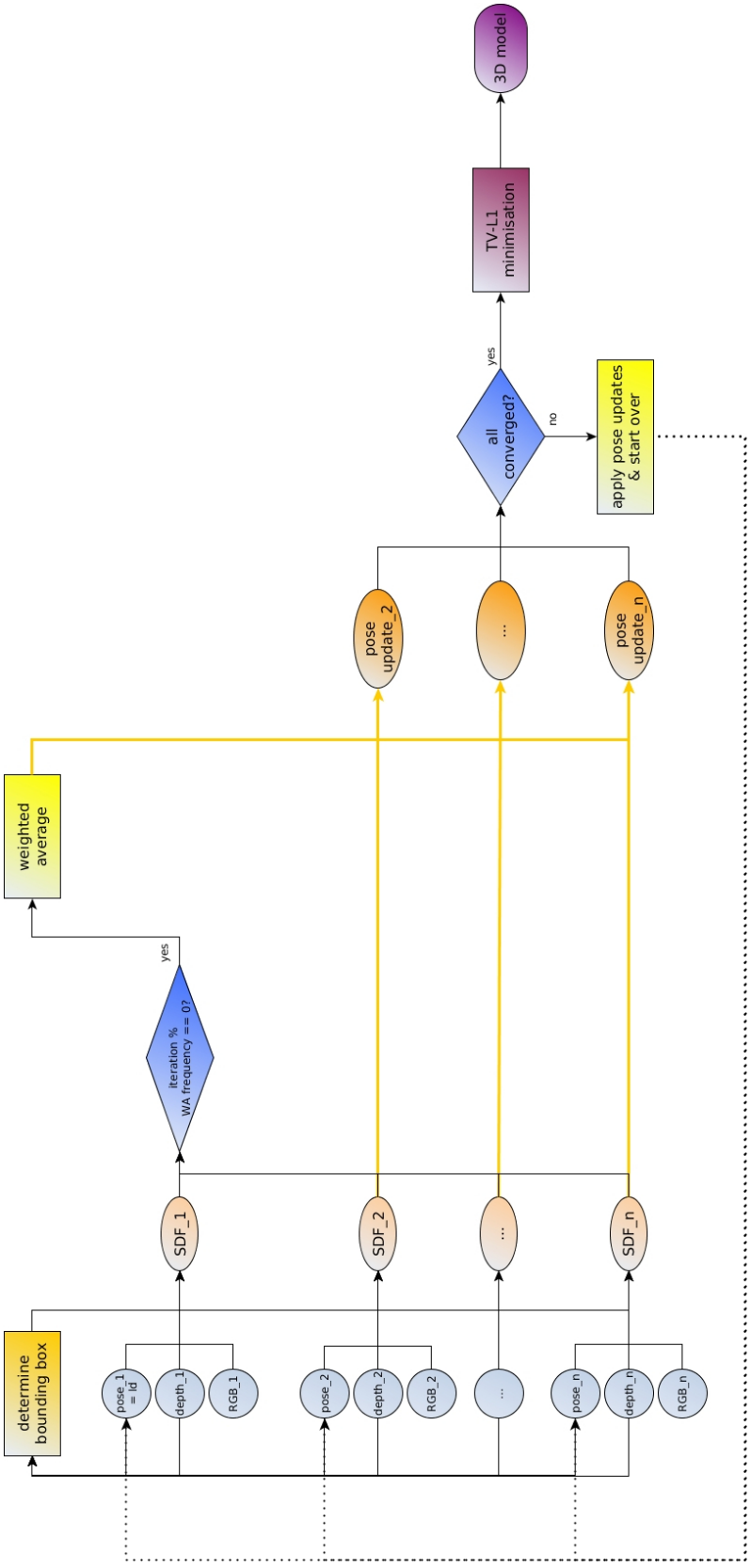


Figure 5.5.: Global Pose Optimisation Flowchart

Algorithm 3: Global Pose Optimisation (Multiview Frame-to-model Registration)

Inputs : set of RGB-D pairs $I_{1..n}, D_{1..n}$
rough pose estimates $\mathbf{P}_{1..n}^{init}$
Output : refined poses $\mathbf{P}_{1..n}$
Additional parameters: coarsest voxel size $init\_voxel\_size$
finest voxel size $final\_voxel\_size$
weight for the colour component of the energy w_{colour}
weight for the normals component of the energy $w_{normals}$
frequency of weighted average generation $wa\_frequency$
threshold for the translational updates $convergence\_threshold$
maximum number of iterations per pyramid level max_iters

$volume \leftarrow determine\_padded\_bounding\_box(D_{1..n}, \mathbf{P}_{1..n}^{init})$
$voxel\_size \leftarrow init\_voxel\_size$

$\mathbf{P}_{1..n} \leftarrow \mathbf{P}_{1..n}^{init}$
$iter \leftarrow 0$
while $voxel\_size \geq finest\_voxel\_size$ **do**
  $CTSDF_1 \leftarrow generate\_ctsdf(D_1, I_1, volume, voxel\_size, Identity)$
  while $level\_iter < max\_iters$ **do**
    // generate the remaining CTSDFs and optionally their weighted average
    $CTSDF_{2..n} \leftarrow generate\_ctsdf(D_{1..n}, I_{1..n}, volume, voxel\_size, \mathbf{P}_{1..n})$
    if $level\_iter$ *is a multiple of* $wa\_frequency$ **then**
      $weighted\_average \leftarrow generate\_coloured\_wa(CTSDF_{1..n})$
    end
    // execute an iteration of gradient descent alignment for each pose
    for *view* $p \leftarrow 2$ **to** n **do**
      $\mathbf{P}_p^{incr} \leftarrow grad\_descent\_alignment(weighted\_average, CTSDF_p, w_{colour}, w_{normals})$
    end
    // apply increments
    $\mathbf{P}_{1..n} \leftarrow \mathbf{P}_{1..n}^{incr}\, \mathbf{P}_{1..n}$
    // check for convergence
    if $\forall p \quad \|translational\_component(\mathbf{P}_p^{incr})\| < convergence\_threshold$ **then**
      convergence at this level has been reached, exit inner loop
    end
    $level\_iter \leftarrow level\_iter + 1$
  end
  $voxel\_size \leftarrow voxel\_size/2$
end

return $\mathbf{P}_{1..n}$

Part III.

Results and Evaluation

6. Evaluation Methodology

In this chapter the setups for testing the proposed approach are explained. First, the various types of range data are presented, together with the chosen data acquisition trajectories. Then the state-of-the-art methods for comparison are described. Finally, the exact metrics, on the grounds of which the evaluations are done, are explained.

6.1. Test Datasets

The types of data were already displayed in Figure 3.3, where they were used to demonstrate the difference in range sensor quality.

Synthetic Datasets and Trajectories

As a proof of concept, we tested the proposed method on noise-free synthetic datasets, acquired with various trajectories. Nine CAD object models[1] were selected for this purpose and resized to the range 20-40 cm per dimension, displayed in 6.1. They were chosen in a way reflecting the variety of shapes (for comparison with geometry-based methods) and textures (for comparison with visual odometry methods) encountered in everyday situations:

- uni-coloured objects: bunny, turbine blade;
- objects with limited texture: Kenny, teddy;
- objects with texture in patches or layers: cow, tram;
- objects with rich texture: leopard, juice box, tea box;
- objects with a single symmetry axis: Kenny, teddy, cow, tram, juice box;
- objects with multiple symmetry axes: tea box.

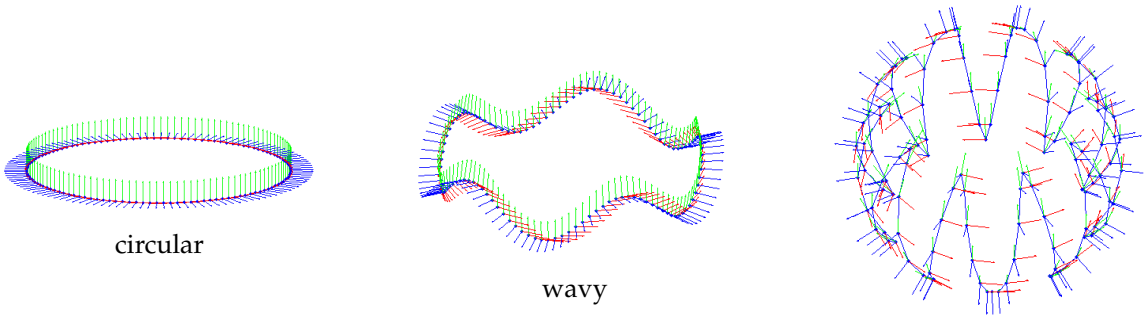
Next, a selection of trajectories was created for testing, as shown in Figure 6.2. Each of them contained 120 poses, generated from a sine wave propagated along a sphere with various frequency and amplitude settings.

- Circular trajectory: spacing 3° horizontally, no vertical displacement. Models a rotating turntable in front of a fixed camera.

[1] Model sources (last accessed: January 2015):
bunny: The Stanford 3D Scanning Repository,
turbine blade: Georgia Tech Large Geometric Models Archive,
Kenny, teddy, cow, tram, leopard, juice box, tea box: Archive3D

Figure 6.1.: Synthetic models: *bunny*, *Kenny*, *teddy*, *cow*, *tram*, *leopard*, *juice box*, *tea box*, *turbine*

- Wavy trajectory: sine wave frequency 5, amplitude 30 cm. Models hand-held sensors.
- Abrupt trajectory: sine wave frequency 10, amplitude 40 cm. Designed for identification of failure cases, since this is a quasi-unrealistic movement for a human.



circular wavy abrupt

Figure 6.2.: Synthetic trajectories

The datasets were rendered using Blender with the standard VGA resolution (640×480 pixels) of Kinect-like sensors. In this way we had 27 synthetic trajectories at our disposal, for each of which we have the original CAD model, the groundtruth trajectory, and noise-free RGB-D images. This is the complete set of information needed for any of the evaluation metrics, as explained later on in this chapter.

GIS Datasets and Trajectories

The Siemens Global Inspection System allows for manual selection of a trajectory, which is subsequently used for scanning an object. We acquired two sequences of 72 poses in a circular trajectory with spacing 5°. The models are shown in Figure 6.3.

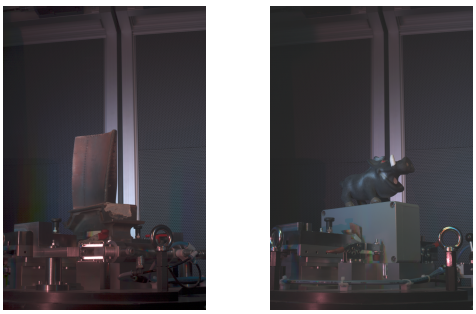


Figure 6.3.: GIS models: *turbine*, *Pumba*

In this setup we have the groundtruth trajectories at our disposal, together with high quality RGB-D pairs of resolution 4008×2672 pixels.

Kinect Datasets and Trajectories

We acquired sequences of 10 everyday objects using a Kinect-like device. The objects were placed on a turntable, which was rotated in front of a fixed sensor. In addition, we placed a markerboard on the turntable, which was used for groundtruth pose estimation, but not utilised in the reconstruction pipeline itself. The variety of objects can be seen in Figure 6.4. They were again chosen to represent a wide range of different shapes and textures:

- reflective: phone, muesli box;
- with a single symmetry axis: Pumba, cow, milk box, book;
- radially symmetric (cylindrical): muesli box, tape;
- geometry with fine structures: turbine blade, Pumba, bench vise.

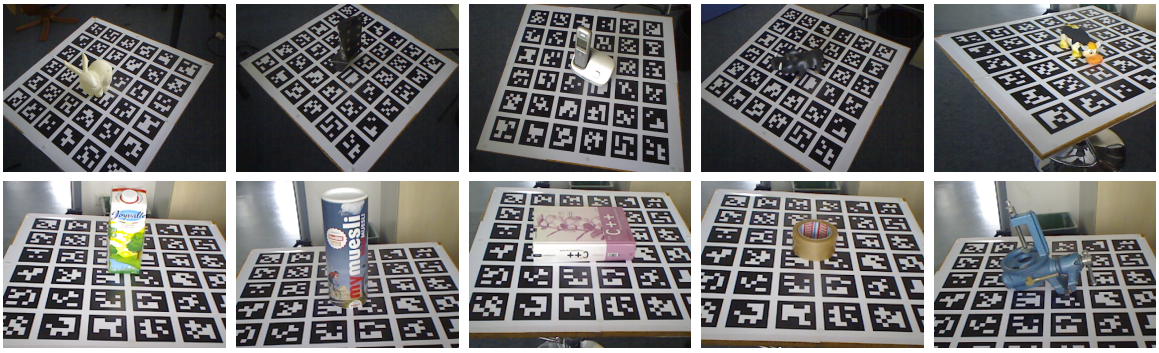


Figure 6.4.: Kinect-like models: *bunny*, *blade*, *phone*, *Pumba*, *cow*, *milk box*, *muesli box*, *book*, *tape*, *bench vise*

For this type of sequences we have near-groundtruth trajectory estimates from the marker-boards. In addition, the CAD models of the Stanford bunny and the turbine blade are available.

RGB-D Benchmark Datasets and Trajectories

The TUM RGB-D Benchmark [78] contains multiple datasets, the main application of which is SLAM. Although our target application is object reconstruction, we evaluated the proposed approach on some of the sequences, mainly from the *3D Object Reconstruction Category*. However, a large group of the provided datasets seemed inappropriate for our purposes, since they include moving people and large rooms, or since the object of interest is not visible in a high number of frames. The most suitable sequences for 3D object reconstruction seemed to be *fr3/teddy* and *fr1/plant*, shown in Figure 6.5.

Figure 6.5.: RGB-D benchmark [78] models: *fr3/teddy*, *fr1/plant*

Thanks to these datasets, we have measurements from another Kinect-like sensor. In addition, highly accurate groundtruth trajectories, acquired with an external measurement system, are provided with the benchmark.

6.2. Evaluation Metrics for Camera Tracking

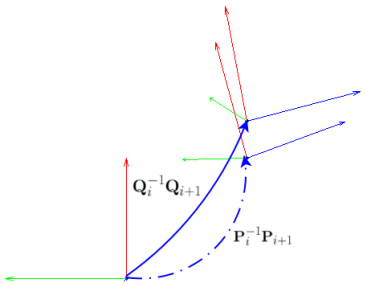
The most appropriate metric for camera tracking is the evaluation of the obtained trajectory against the groundtruth one, which we have available for every type of dataset. Since we carry out the tracking procedure frame-to-frame, we are interested in the frame-to-frame drift, and not drift per second, as can be the case for SLAM systems [41]. In this sense, we adapt the **relative pose error** (RPE) from the RGB-D benchmark [78], but we modify it for frame-to-frame evaluation. Thus, we estimate the RPE of every pose increment, and report a root-mean-squared, average, minumum and maximum RPE over the whole trajectory. In addition, we also evaluate the average, minimum and maximum angular error per frame-to-frame camera transformation.

To sum up, the metrics for evaluating the performance of camera tracking are the following:

- Given: groundtruth trajectory $\{\mathbf{Q_1}, \ldots, \mathbf{Q_n}\}$, and estimated trajectory $\{\mathbf{P_1}, \ldots, \mathbf{P_n}\}$.

- **Relative pose error** of the transformation from frame #i to #$i+1$:

$$RPE_{i\to i+1} = (\mathbf{P_i}^{-1}\mathbf{P_{i+1}})^{-1}(\mathbf{Q_i}^{-1}\mathbf{Q_{i+1}}). \tag{6.1}$$



- Root mean squared **drift per frame over the whole trajectory**:

$$RMS\_drift = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}\|transl(RPE_{i\to i+1})\|^2}. \tag{6.2}$$

- **Average, min, max drift** over frame-to-frame transformations.
- **Average, min, max angular error** over frame-to-frame transformations.

We chose to compare our tracking method against three state-of-the-art registration techniques:

- **ICP**: As the most widely used geometry-based registration method, the comparison with ICP is a must. The specific ICP variant we evaluated against was **Generalized-ICP** (GICP) [73]. It is allegedly very robust to outliers thanks to its probabilistic framework. GICP models both the data and target scans as locally planar structures, achieving a sort of plane-to-plane metric. This makes GICP one of the best-performing ICP adaptations. We used the open-source implementation from the Point Cloud Library. Note that we did not provide an initial guess for the transformation between registered scans, to reflect completely the camera tracking setting in an unknown environment. This can lead to problems with poor initialisation for ICP in the presence of significant noise and in lack of sufficient overlap.
- **KinectFusion**: Since this is the work that revived SDF-based approaches, we evaluated against it. For this we used **RecFusion**, which is an compellingly accurate and robust commercial implementation by ImFusion.
- **Dense Visual Odometry** (DVO): As a state-of-the-art camera tracking method, we evaluated our estimated trajectories versus the ones determined by the method of Kerl *et al.* [41], which is available online. Since the subject of these tests is camera tracking itself, we used only the frame-to-frame registration capabilities, without the loop closure detection and the subsequent refinement. Note, however, the different application scenarios: object reconstruction for our method and SLAM for the odometry method. It is, therefore, expected that the SDF method would perform better in small-scale limited-volume situations, while DVO would outperform in large-scale tracking scenarios. The comparisons are, nevertheless, insightful.

6.3. Evaluation Metrics for 3D Object Reconstruction

Whenever available, the original CAD model was compared to the final mesh, using CloudCompare. This software provides information about the distribution of cloud-to-cloud and cloud-to-mesh distance error, which is a direct measure of the quality of the obtained models. Since the end result of **RecFusion** is also a meshed model, we juxtaposed the error reported by **CloudCompare** about it and the error estimated on our results.

Notes

As mentioned in Chapter 5 explaining the full pipeline, when the frame-to-frame tracking does not converge, the image is deemed unreliable and is not used further in tracking, while a pose increment of identity is added to the trajectory for keeping its length equal to the number of frames. Our wrappers around **DVO** and **GICP** apply the same handling. However, in the error evaluation these identity transformations were not taken into account, because they would immensely increase the error even if tracking continued successfully afterwards.

We used a personalized version of **RecFusion**, which outputs the pose estimates in addition to the final model. However, when failure in registration was detected, no pose increment was stored. Therefore, the **RecFusion** trajectories usually contain less poses than the number of frames and the correspondence to the image indices is not known. Thus for evaluation we determined the closest groundtruth pose to each **RecFusion** pose and continued with the metrics from above in the usual way. Consequently, for this approach we evaluate only well determined poses and the assessment is slightly different. This can be a source of discrepancies between the reported error values, however, it was the most appropriate possibility to handle these results.

7. Experimental Results and Assessment

This chapter summarises the performed tests and discusses the significance of their results. The experiments are separated into sensor quality categories, ranging from noise-free synthetic data to inferior quality Kinect-like measurements. For each dataset, an evaluation of the estimated trajectory and of the refined 3D model is given. Comparisons with state-of-the-art methods for camera tracking and object reconstruction are also presented.

7.1. Synthetic Data

We tested our unified pipeline on the nine synthetic datasets presented in the previous chapter as a proof of concept for the power of implicit-to-implicit registration, using only the geometric component in the objective function. In addition, we compared the trajectory errors with those of **DVO**, **GICP** and **RecFusion**. Finally, we evaluated the accuracy of the final models outputted by our algorithm and by **RecFusion** against the original CAD models. The results are presented below.

Tracking Evaluation

Our estimates were extremely similar to the groundtruth for the circular and wavy trajectories, as can be seen from Figures 7.1 and 7.2. In the case of the circular trajectory **RecFusion** was nearly as accurate on the objects with articulate geometry. On the two models with multiple symmetry axes, the *juice box* and the *tea box*, it performed poorer than the rest of the approaches. **DVO** gave very good results on the objects with rich texture, and drifted consistently on the *bunny*, *Kenny*, *teddy* and *cow* sequences, which are either uniformly coloured or with very few distinctively coloured regions. In all cases **GICP** displayed the worst results. These trends were also visible for the wavy trajectories, where the differences were even more acute. Our approach reproduced the groundtruth trajectory very closely, followed by **DVO** and **RecFusion**, while **GICP** diverged severely in most cases.

The numerical comparisons corresponding to these results are summarised in Tables 7.1 and 7.2, and are visualised in Figures 7.3 and 7.4. The tables present the best performing result in bold font. We have the best metrics everywhere, apart from the minimum drift per pose increment, which was achieved by **RecFusion** with absolutely no error. Our biggest average angular error was 0.09°, while it never exceeded 0.46°. The root-mean-squared drift was never bigger than 1 mm, and was usually smaller, while the biggest recorded value was 3.15 mm. These values testify that our approach is limited by the voxel size, which was set to 2 mm for the tracking component. Interestingly, the implicit-to-implicit registration achieves smaller error on the wavy trajectory than on the circular one. In conclusion, we have shown that for these types of trajectories our method is better than state-of-the-art approaches, which usually have at least an order of magnitude higher drift

and angular error. This is because **DVO** is highly dependent on texture, while our pipeline can handle any type of colouring, while the ICP-based registration of **GICP** and **RecFusion** is very sensitive to the exact geometry.

For the abrupt trajectory sequences most methods failed on most objects, apart from the *teddy*, which seemed to be sufficiently large and geometrically distinctive to facilitate registration. Therefore, this type of motion represents a failure case for all methods, and gives insights on their limits. These can be viewed in Table 7.3 (where the biggest errors are highlighted in red) and Figure 7.5. It can be observed that our method outperformed the other techniques on all metrics for the *teddy* sequence. The motion that we estimated for the other models was too erroneous to allow for pose optimisation to converge to the true model. In these cases we usually exhibited the largest maximal angular error, while **RecFusion** had the largest average angular error. However, our average angular error did not exceed 8.3° even in the most extreme cases. **GICP** usually had the lowest angular error overall. There was no clear trend concerning the drift, but we often had the worst maximum drift per pose increment, while the average was object-dependent. We clearly outperformed the other approaches on the *juice box* and the *tea box*, where our average drift is less than 6 mm, while for the others it was almost 3 cm. Our root-mean-squared drift was usually much higher than the average one, which means that our trajectories contained significantly more outliers than the ones estimated by the other approaches. Thus, it can be concluded that our method is rather sensitive to cases with extreme initial angular differences, while it performs better than other methods in moderate cases, which are more common in real tracking scenarios.

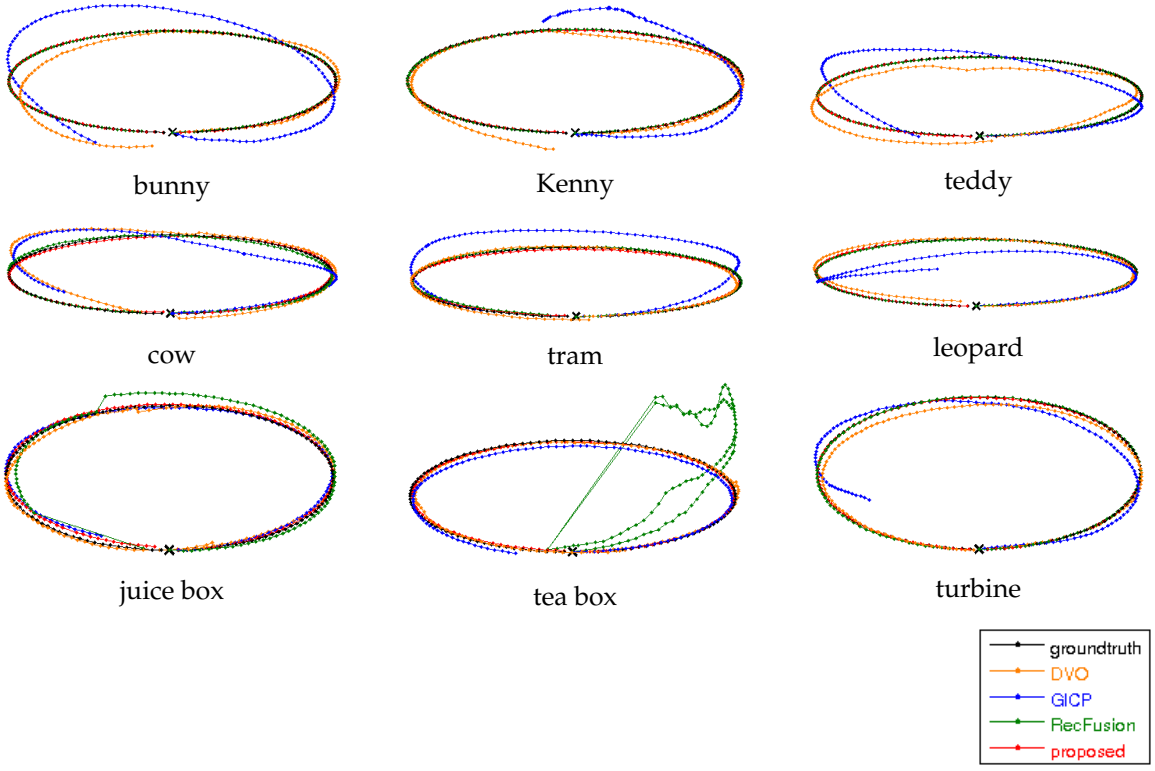


Figure 7.1.: Tracking results on the circular synthetic trajectory

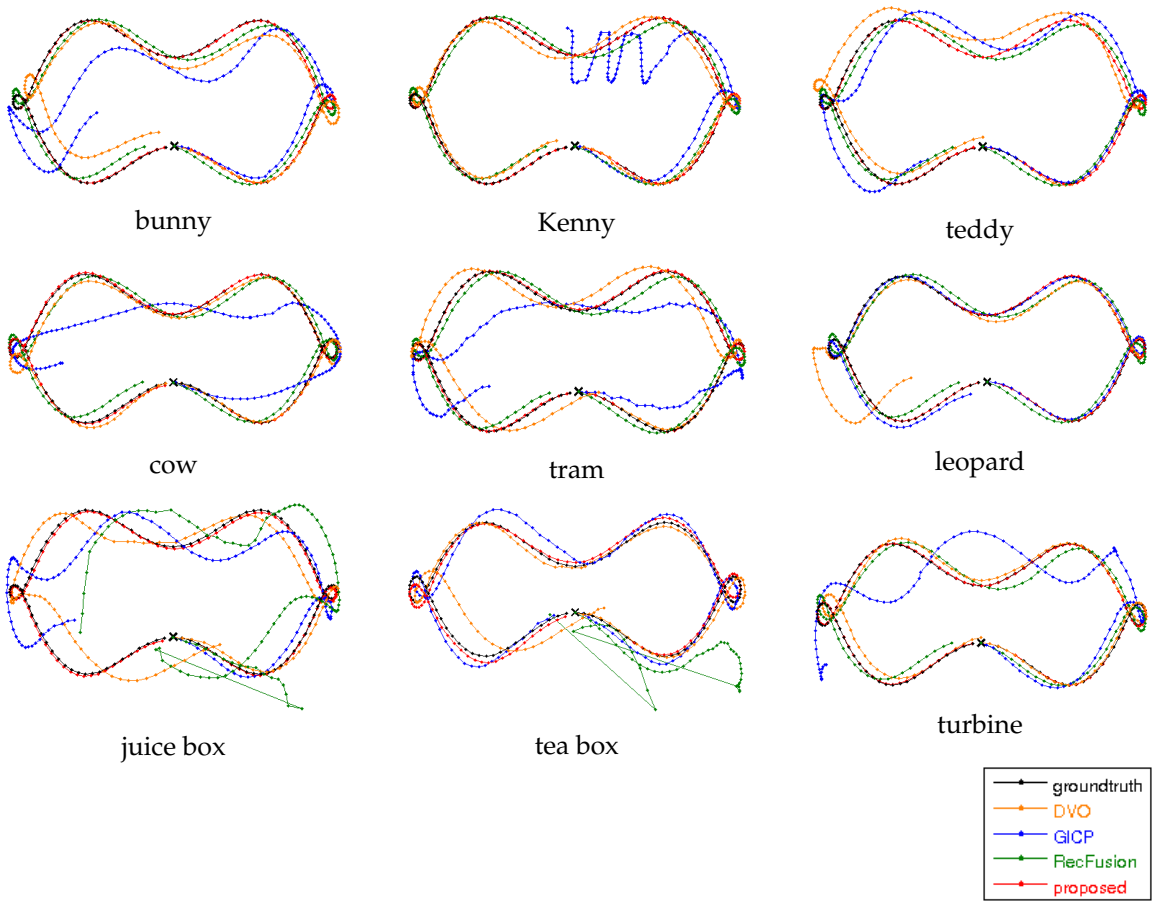


Figure 7.2.: Tracking results on the wavy synthetic trajectory

Reconstruction Evaluation

After tracking, 12 keyframes were selected at regular 10-frame intervals, thus having approximately 30° between poses. Usually if a user is interacting with the system, they can select the keyframes in such a way that all parts of the object are visible. We wanted to keep the fully automatic setting, and therefore have no guarantee that the whole surface was observed. So errors in the final models, which are due to interpolation on unseen regions, can be easily corrected by selecting a higher number of frames. The subsequent pose optimisation converged immediately for the circular and wavy trajectories and the smooth TV-L^1 models were generated. They are displayed in Figure 7.6. The final meshes were rendered using marching cubes on a grid with voxel size 1 mm.

Table 7.4 summarises the mean and standard deviation errors of our models and the ones generated by **RecFusion** in comparison to the real CAD models. **RecFusion** was run with voxel size 0.8 - 1 mm, depending on the size of the object. The error for our models is in the range -1 to 1 mm, which means that the model precision is only limited by the grid resolution. **RecFusion** had lower standard deviation for the *bunny*, *Kenny*, *teddy*, *tram* and *turbine* sequences, indicating that it is slightly more consistent than our approach on geometrically simple models. Nevertheless, we achieve remarkable accuracy, comparable to that of the *KinectFusion* implementation.

object	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
bunny	DVO	0.0024599	0.0022734	0.0009311	0.0051567	0.1758453	0.0000000	0.4111684
	GICP	0.0033875	0.0023616	0.0003695	0.0169163	0.2733477	0.0000000	1.7922591
	RecFusion	0.0032108	0.0004724	**0.0000000**	0.0311766	0.0380466	0.0000000	2.9784925
	proposed	**0.0001066**	**0.0000855**	0.0000052	**0.0003113**	**0.0000000**	0.0000000	**0.0000000**
Kenny	DVO	0.0030508	0.0023196	0.0004022	0.0195652	0.2143841	0.0000000	1.8846462
	GICP	0.0199170	0.0152666	0.0004483	0.0318719	1.4985936	0.0000000	2.9934351
	RecFusion	0.0011975	0.0003051	**0.0000000**	0.0121286	0.0181635	0.0000000	1.1589193
	proposed	**0.0001786**	**0.0001516**	0.0000091	**0.0004230**	**0.0016624**	0.0000000	**0.0395647**
teddy	DVO	0.0027738	0.0023836	0.0004680	0.0082596	0.2500683	0.0000000	0.7813414
	GICP	0.0020756	0.0018890	0.0001434	0.0050433	0.2633821	0.0000000	0.6305581
	RecFusion	0.0013200	0.0002648	**0.0000000**	0.0141296	0.0165594	0.0000000	1.3977118
	proposed	**0.0000992**	**0.0000836**	0.0000041	**0.0002345**	**0.0003325**	0.0000000	**0.0395647**
cow	DVO	0.0048922	0.0022126	0.0001868	0.0337501	0.2495112	0.0000000	3.5262825
	GICP	0.0069215	0.0039269	0.0011914	0.0292515	0.4158586	0.1186941	3.1043471
	RecFusion	0.0048459	0.0014797	**0.0000000**	0.0291084	0.1467638	0.0000000	2.7517320
	proposed	**0.0003814**	**0.0003348**	0.0000189	**0.0009994**	**0.0250437**	0.0000000	**0.1897456**
tram	DVO	0.0010117	0.0008809	0.0000692	0.0026083	0.0726811	0.0000000	0.3956470
	GICP	0.0065058	0.0029694	0.0003481	0.0245058	0.4597992	0.1312212	3.1763843
	RecFusion	0.0041042	0.0012011	**0.0000000**	0.0295501	0.1391812	0.0000000	2.7701605
	proposed	**0.0003479**	**0.0002599**	0.0000391	**0.0016571**	**0.0257776**	0.0000000	**0.2438931**
leopard	DVO	0.0026309	0.0005457	0.0000834	0.0284089	0.0366617	0.0000000	3.0616918
	GICP	0.0017400	0.0014988	0.0000206	0.0065338	0.2116754	0.0000000	1.4182801
	RecFusion	0.0072112	0.0018860	**0.0000000**	0.0313137	0.1726169	0.0000000	2.9934351
	proposed	**0.0001016**	**0.0000863**	0.0000197	**0.0003224**	**0.0006650**	0.0000000	**0.0395647**
juice box	DVO	0.0071818	0.0039408	0.0004820	0.0340482	0.3852918	0.0000000	3.4829499
	GICP	0.0050431	0.0029530	0.0003332	0.0314609	0.2981636	0.0000000	3.0010088
	RecFusion	0.0145186	0.0047376	**0.0000000**	0.1064493	0.3614313	0.0000000	4.4754678
	proposed	**0.0008695**	**0.0007283**	0.0000177	**0.0017794**	**0.0884378**	0.0000000	**0.4597004**
juice box	DVO	0.0073024	0.0046988	0.0005882	0.0271085	0.4691189	0.0000000	2.8233379
	GICP	0.0044593	0.0027046	0.0000963	0.0265907	0.1772689	0.0000000	0.5337572
	RecFusion	0.0856450	0.0289509	**0.0000000**	0.6347480	1.9091880	0.0000000	35.5206507
	proposed	**0.0009695**	**0.0007727**	0.0001261	**0.0031464**	**0.0325118**	0.0000000	**0.3916706**
turbine	DVO	0.0022563	0.0020591	0.0005755	0.0051092	0.1859965	0.0000000	0.5496507
	GICP	0.0062821	0.0041594	0.0000774	0.0307737	0.4490209	0.0395647	3.0150610
	RecFusion	0.0042558	0.0011707	**0.0000000**	0.0304711	0.1044768	0.0000000	2.9350756
	proposed	**0.0002797**	**0.0002198**	0.0000343	**0.0010640**	**0.0085396**	0.0000000	**0.1046783**

Table 7.1.: Evaluation metrics for the circular synthetic trajectory

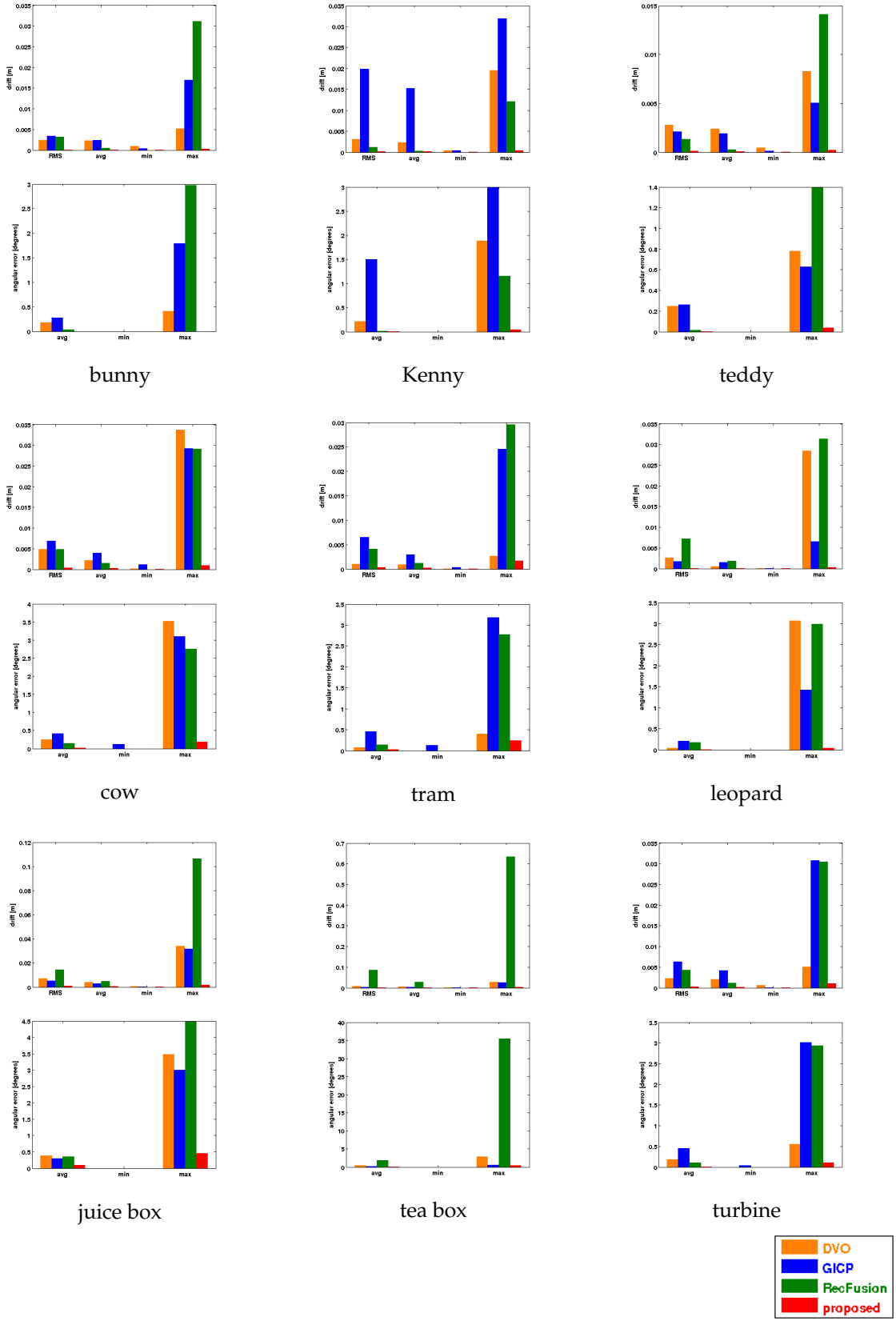


Figure 7.3.: Comparison of evaluation metrics for the circular synthetic trajectory (top: drift, bottom: angular error)

object	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
bunny	DVO	0.0027786	0.0025414	0.0004819	0.0060613	0.2018332	0.0000000	0.5143413
	GICP	0.0030274	0.0024286	0.0001915	0.0152239	0.3007346	0.0000000	1.4793273
	RecFusion	0.0205014	0.0180567	**0.0000000**	0.0407044	1.7930916	0.0000000	3.8878065
	proposed	**0.0000944**	**0.0000754**	0.0000018	**0.0003089**	**0.0016624**	0.0000000	**0.0395647**
Kenny	DVO	0.0034684	0.0024597	0.0004124	0.0231250	0.2337419	0.0000000	2.2510544
	GICP	0.0195245	0.0152978	0.0006450	0.0312916	1.5062747	0.0791294	3.0098640
	RecFusion	0.0205017	0.0180507	**0.0000000**	0.0407962	1.7936643	0.0000000	3.8954506
	proposed	**0.0001577**	**0.0001352**	0.0000112	**0.0004006**	**0.0018001**	0.0000000	**0.0559529**
teddy	DVO	0.0028212	0.0024599	0.0004799	0.0078391	0.2404824	0.0000000	0.7412454
	GICP	0.0021936	0.0018202	0.0000350	0.0055590	0.2479986	0.0000000	0.6954847
	RecFusion	0.0205320	0.0180926	**0.0000000**	0.0407036	1.7965527	0.0000000	3.8884106
	proposed	**0.0001022**	**0.0000860**	0.0000091	**0.0002624**	**0.0018001**	0.0000000	**0.0559529**
cow	DVO	0.0021087	0.0017138	0.0002706	0.0062589	0.1769586	0.0000000	0.9827694
	GICP	0.0163378	0.0144588	0.0016068	0.0328026	1.4332457	0.1582587	3.4404370
	RecFusion	0.0206163	0.0181677	**0.0000000**	0.0404787	1.8099672	0.0000000	3.8645811
	proposed	**0.0003709**	**0.0003183**	0.0000605	**0.0010396**	**0.0293940**	0.0000000	**0.1855748**
tram	DVO	0.0021447	0.0015071	0.0002885	0.0141419	0.1468194	0.0000000	1.7974921
	GICP	0.0131172	0.0094739	0.0003316	0.0351151	0.9951021	0.0000000	4.0303714
	RecFusion	0.0207237	0.0181754	**0.0000000**	0.0411277	1.8073819	0.0000000	3.9280709
	proposed	**0.0003486**	**0.0002922**	0.0000198	**0.0010175**	**0.0311168**	0.0000000	**0.1855748**
leopard	DVO	0.0088432	0.0023056	0.0000458	0.0465716	0.2143957	0.0000000	4.7655477
	GICP	0.0020426	0.0014458	0.0000745	0.0139550	0.2102045	0.0000000	1.2960146
	RecFusion	0.0211109	0.0185269	**0.0000000**	0.0407307	1.8376097	0.0000000	3.8908258
	proposed	**0.0001021**	**0.0000884**	0.0000090	**0.0002236**	**0.0033248**	0.0000000	**0.0395647**
juice box	DVO	0.0056215	0.0035515	0.0004304	0.0307606	0.3446555	0.0000000	3.1906458
	GICP	0.0066370	0.0043935	0.0002030	0.0344170	0.4057555	0.0395647	3.0184342
	RecFusion	0.0379836	0.0264728	**0.0000000**	0.2198723	2.9445899	0.0000000	31.8128039
	proposed	**0.0005720**	**0.0004904**	0.0000376	**0.0013670**	**0.0623454**	0.0000000	**0.2373881**
tea box	DVO	0.0058761	0.0041332	0.0006931	0.0199516	0.4090474	0.0000000	2.0313561
	GICP	0.0069665	0.0057291	0.0004136	0.0218112	0.2648897	0.0000000	0.7454570
	RecFusion	0.0726619	0.0354840	**0.0000000**	0.3768974	2.2342680	0.0000000	7.1747191
	proposed	**0.0007137**	**0.0006348**	0.0000579	**0.0023268**	**0.0513209**	0.0000000	**0.3856290**
turbine	DVO	0.0026870	0.0023325	0.0004093	0.0082212	0.2104359	0.0000000	0.8504126
	GICP	0.0083549	0.0055917	0.0002794	0.0288158	0.5948641	0.0395647	2.8996585
	RecFusion	0.0198975	0.0176736	**0.0000000**	0.0407220	1.7568382	0.0000000	3.8880078
	proposed	**0.0002046**	**0.0001627**	0.0000107	**0.0007892**	**0.0060120**	0.0000000	**0.0884693**

Table 7.2.: Evaluation metrics for the wavy synthetic trajectory

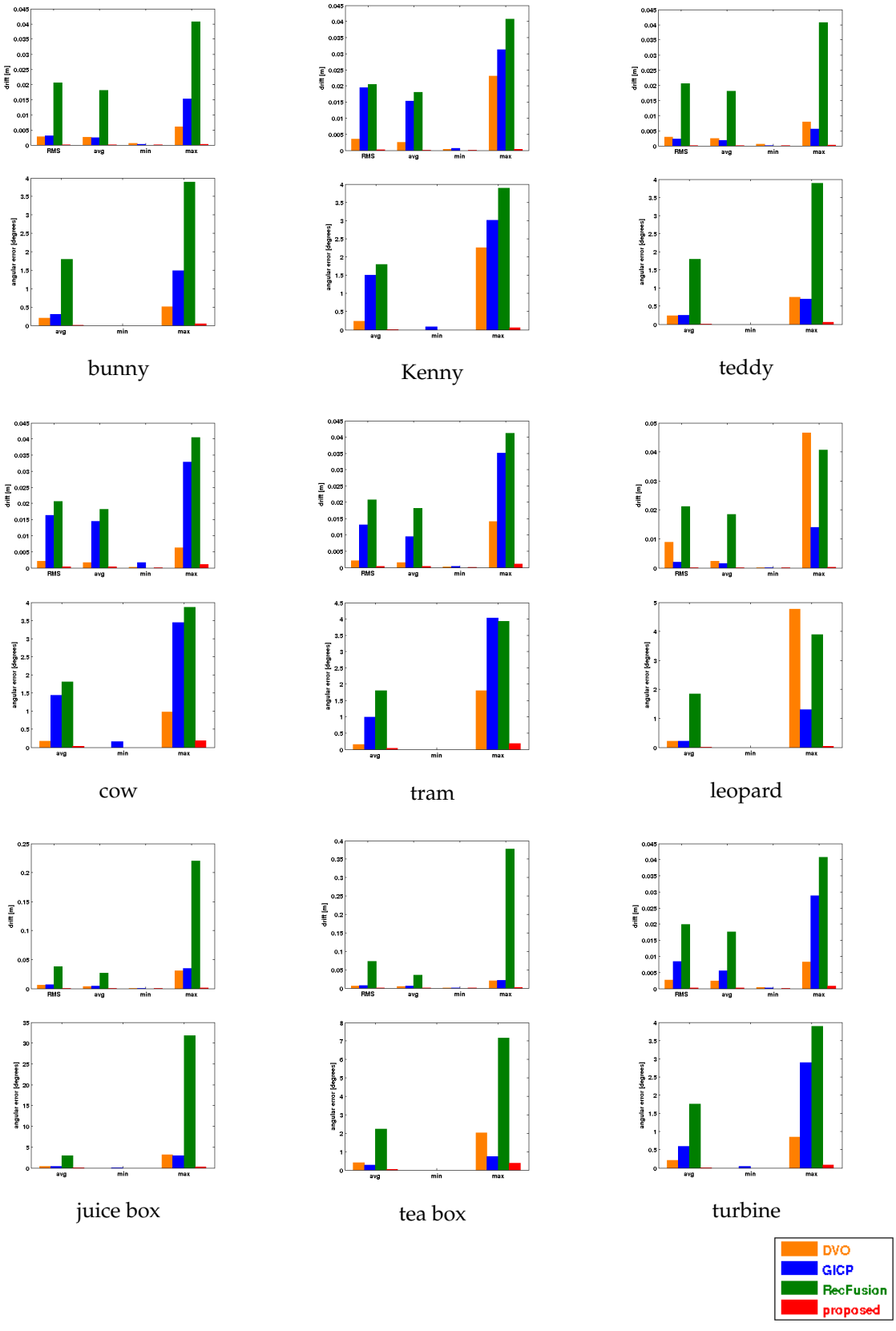


Figure 7.4.: Comparison of evaluation metrics for the wavy synthetic trajectory (top: drift, bottom: angular error)

object	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
bunny	DVO	0.0629367	0.0346562	0.0004623	0.2909223	3.1454326	0.1119058	25.6259566
	GICP	**0.0249703**	**0.0062876**	0.0001910	0.2409567	**0.7662285**	0.0000000	25.0225832
	RecFusion	0.0299282	0.0211625	**0.0000000**	**0.0423249**	3.1718172	0.0000000	**6.3436344**
	proposed	0.0944349	0.0198091	0.0000131	0.6127612	2.5229149	0.0000000	78.9499733
Kenny	DVO	0.0422950	0.0240184	0.0017220	0.1796803	2.2817898	0.0685280	17.4729741
	GICP	**0.0173378**	**0.0131346**	0.0004858	**0.0495157**	**1.4712713**	0.0395647	**5.9275769**
	RecFusion	0.0664603	0.0559500	**0.0000000**	0.0987265	9.1633271	0.0000000	17.2140341
	proposed	0.1522074	0.0651810	0.0000161	0.5585514	6.7392681	0.0000000	59.9807275
teddy	DVO	0.0240464	0.0133096	0.0004610	0.1006452	1.1531106	0.0395647	9.8901242
	GICP	0.0024072	0.0020131	0.0000542	0.0077594	0.2430311	0.0000000	0.8185222
	RecFusion	0.0452263	0.0322792	**0.0000000**	0.0753214	8.9812889	0.0000000	13.9772372
	proposed	**0.0001003**	**0.0000884**	0.0000084	**0.0002084**	**0.0016624**	0.0000000	**0.0395647**
cow	DVO	0.0950952	0.0471491	0.0003188	0.3122357	4.0279981	0.0000000	27.4870854
	GICP	0.1255147	0.1052380	0.0041449	**0.2152359**	10.4554739	0.5719806	**20.1530335**
	RecFusion	-	-	-	-	-	-	-
	proposed	**0.0909144**	**0.0167325**	**0.0000306**	0.6709773	**1.9587471**	0.0000000	88.3638949
tram	DVO	0.1355310	0.0842114	0.0002709	0.3899743	7.3665126	0.0000000	43.8357534
	GICP	0.1134809	0.0744294	0.0001861	0.2028601	7.5406005	0.0000000	20.6716835
	RecFusion	**0.0064877**	**0.0045875**	**0.0000000**	**0.0091750**	**0.4622482**	0.0000000	**0.9244963**
	proposed	0.1638896	0.0519278	0.0000338	0.6488427	5.3320391	0.0000000	83.3658332
leopard	DVO	0.1109295	0.0700776	0.0001664	0.2831688	6.3740521	0.0000000	26.6236281
	GICP	**0.0094400**	**0.0029454**	0.0002241	0.0983127	**0.3911474**	0.0000000	**13.1447652**
	RecFusion	0.0689420	0.0556539	**0.0000000**	**0.0961474**	14.3079033	0.0000000	25.8650873
	proposed	0.1555930	0.0430349	0.0000096	0.7092819	5.1412965	0.0000000	115.8640894
juice box	DVO	0.0459790	0.0206799	0.0006662	0.2231459	1.8786980	0.0000000	23.9210151
	GICP	**0.0075875**	0.0054231	0.0008082	**0.0426052**	0.6508314	0.0884693	**3.1716985**
	RecFusion	0.0390164	0.0275888	**0.0000000**	0.0551776	14.5990746	0.0000000	29.1981492
	proposed	0.0181954	**0.0036401**	0.0000530	0.1459842	**0.4049580**	0.0000000	21.5583841
tea box	DVO	0.0550181	0.0275738	0.0002404	0.1969629	2.6076830	0.0000000	18.2733424
	GICP	**0.0099355**	0.0072210	0.0004189	**0.0236971**	**0.5204841**	0.0000000	**3.2727586**
	RecFusion	0.0397395	0.0281001	**0.0000000**	0.0562002	4.7797527	0.0000000	9.5595054
	proposed	0.0400886	**0.0057858**	0.0000442	0.3066665	0.5512004	0.0000000	30.5782287
turbine	DVO	0.0365144	0.0177515	0.0000957	0.2052455	1.4676542	0.0395647	15.9675410
	GICP	**0.0079515**	**0.0058476**	0.0008510	**0.0268474**	**0.6939415**	0.0884693	**3.1781089**
	RecFusion	0.0795899	0.0626633	**0.0000000**	0.1198169	14.3241052	0.0000000	22.4573001
	proposed	0.1847865	0.0726840	0.0000392	0.7319708	8.2346880	0.0000000	87.2810507

Table 7.3.: Evaluation metrics for the abrupt synthetic trajectory

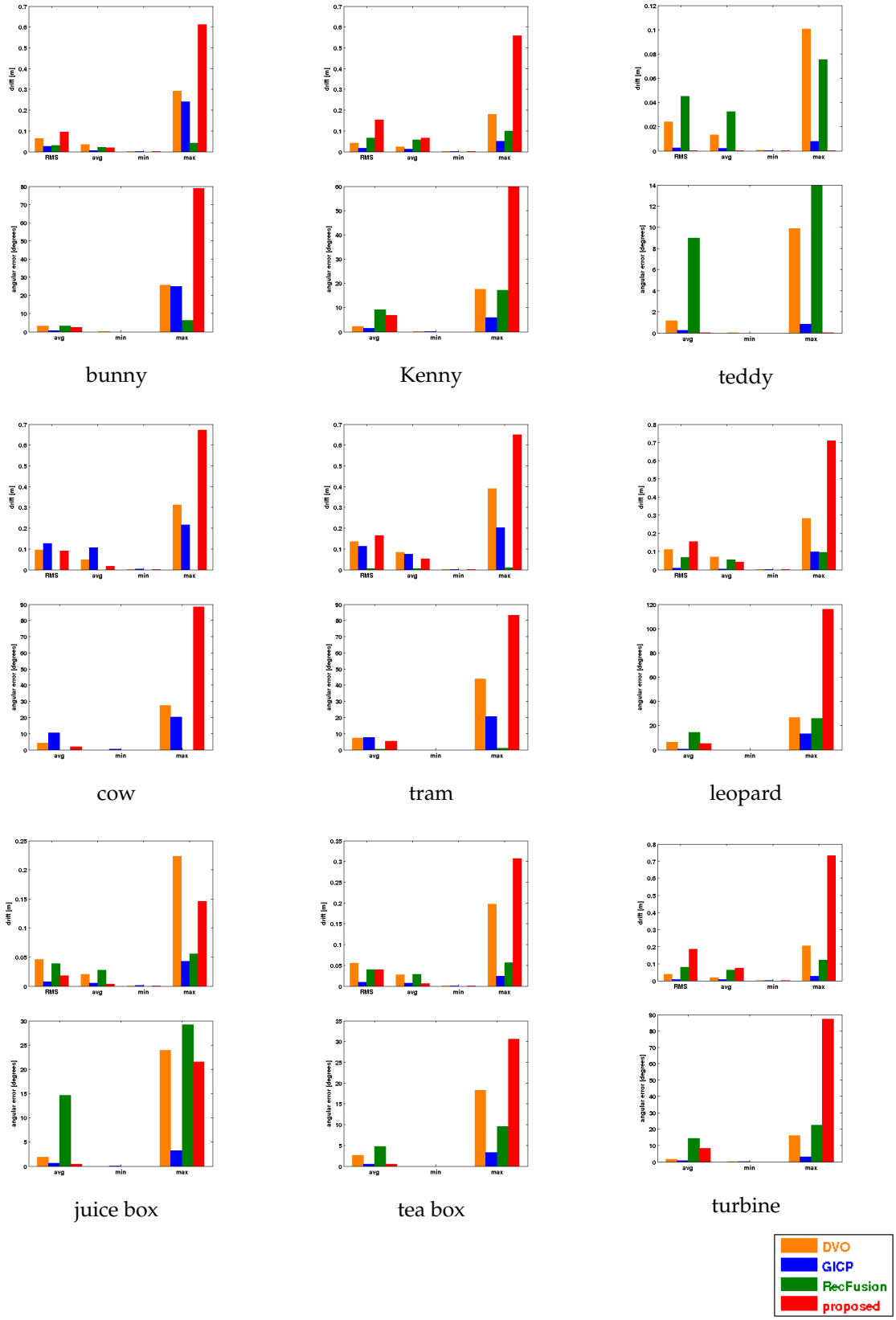


Figure 7.5.: Comparison of evaluation metrics for the abrupt synthetic trajectory (top: drift, bottom: angular error)

Figure 7.6.: Reconstructed models from synthetic data

object	method	mean [m]	std. dev. [m]
bunny	RecFusion	0.000088	**0.001020**
	proposed	0.000336	0.002074
Kenny	RecFusion	0.000173	**0.000796**
	proposed	-0.000069	0.000950
teddy	RecFusion	0.000196	**0.000921**
	proposed	0.000063	0.001409
cow	RecFusion	0.000936	0.001622
	proposed	0.000166	**0.000866**
tram	RecFusion	0.000579	**0.001587**
	proposed	0.000356	0.001627
leopard	RecFusion	-0.001329	0.003638
	proposed	0.000086	**0.001018**
juice box	RecFusion	0.023742	0.015744
	proposed	0.001218	**0.000974**
tea box	RecFusion	0.012059	0.015781
	proposed	0.001785	**0.002405**
turbine	RecFusion	0.000761	**0.000905**
	proposed	0.000093	0.001397

Table 7.4.: *CloudCompare* evaluation metrics against the original CAD models for synthetic data

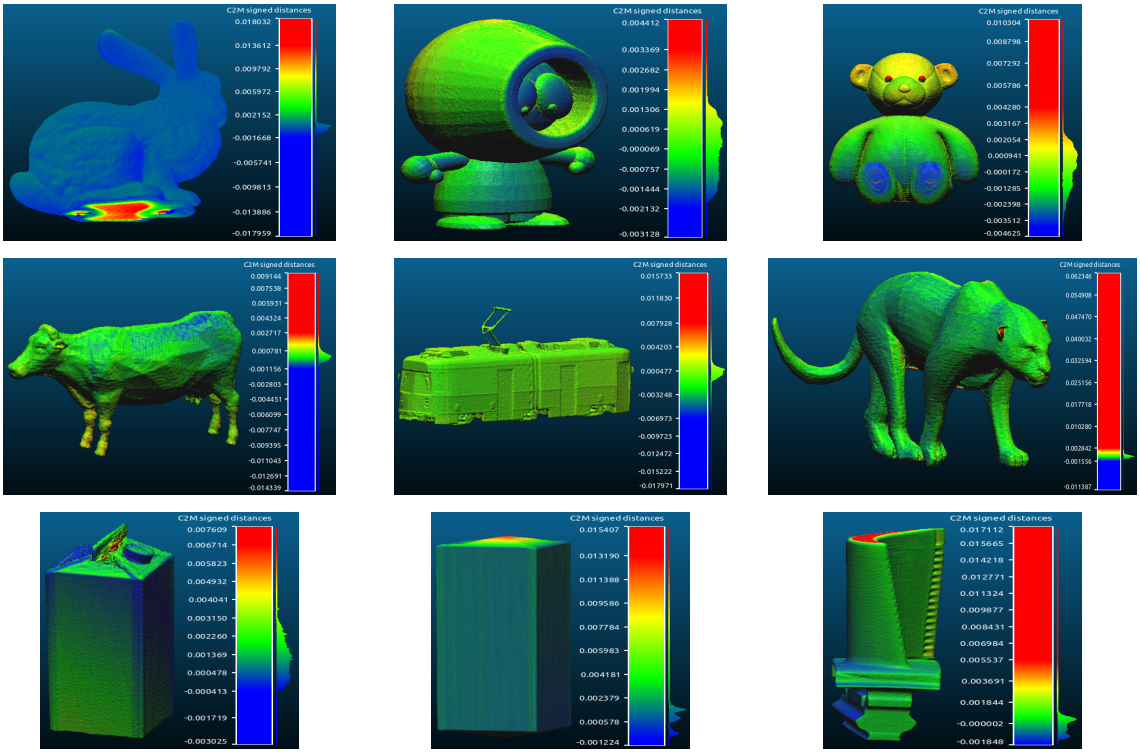


Figure 7.7.: Comparison of our output against the original CAD models in *CloudCompare*

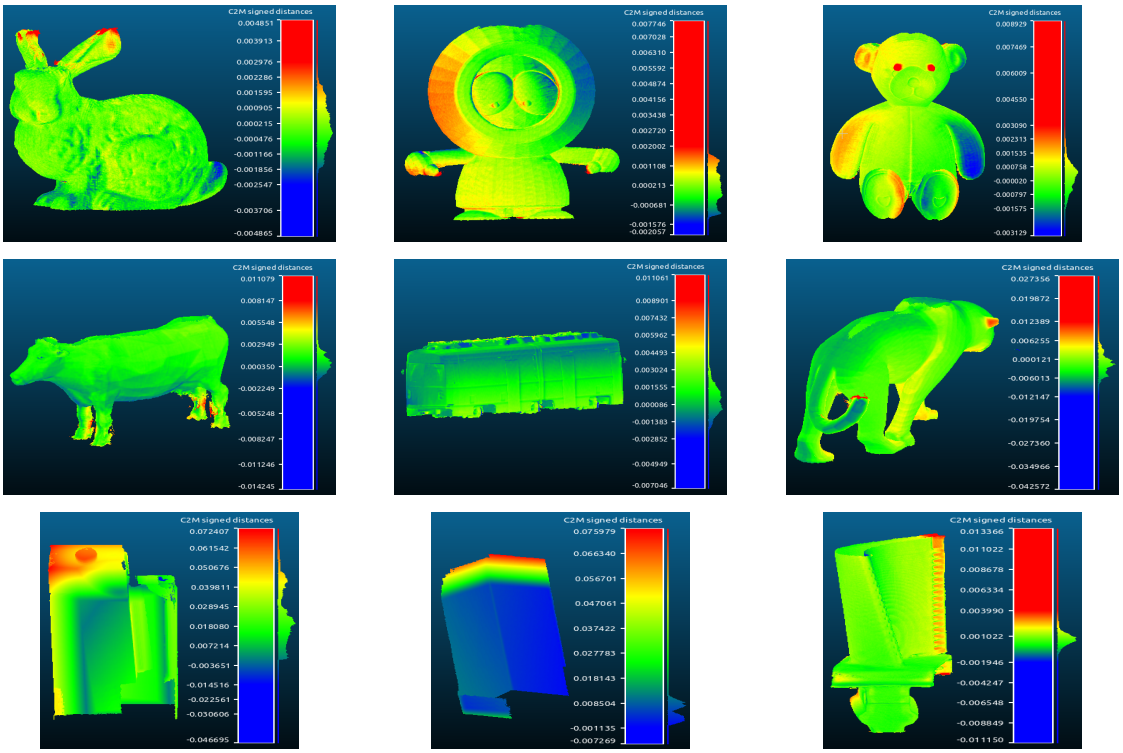


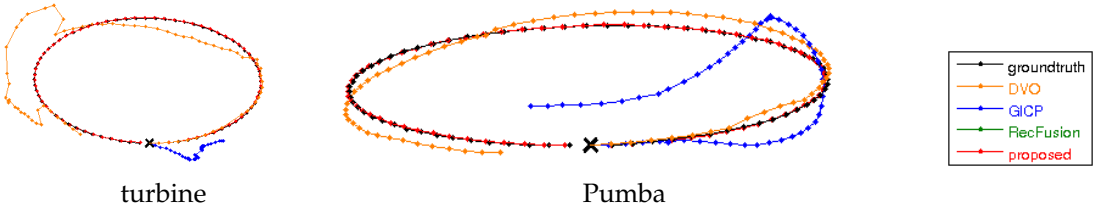
Figure 7.8.: Comparison of the **RecFusion** output against the original CAD models in *CloudCompare*

7.2. Industrial Quality Depth Data

A similar procedure was carried out on the acquired Siemens Global Inspection System sequences, whereby the trajectories were estimated using **DVO**, **GICP**, **RecFusion** and the proposed approach with energy solely based on its geometric component. They were compared to the groundtruth, which was provided by the controller of the scanner motors. Subsequently, our pipeline went into its pose optimisation stage, where it immediately converged and the smooth TV-L^1 model was generated. The same grid resolutions were used as in all experiments: 2 mm voxel size for tracking, coarse-to-fine pyramid with 4 and 2 mm for pose optimisation, and 1 mm voxel size for CTSDF fusion into the final model.

Tracking Evaluation

The estimated trajectories are visualised in Figure 7.9. **RecFusion** lost the tracking as early as the second frame of both sequences and is therefore not visible in the plots. Our implicit-to-implicit registration evidently gave the best results, very close to the groundtruth. **DVO** performed similarly, but was sometimes influenced by the reflective nature of the *turbine* blade. **GICP** performed rather poorly. It can thus be concluded that the 5° pose difference is too large to be handled by ICP-based approaches in an unconstrained scenario. This is, however, not an obstacle for our approach.



turbine Pumba

Figure 7.9.: Estimated trajectories for the industrial depth sensor sequences

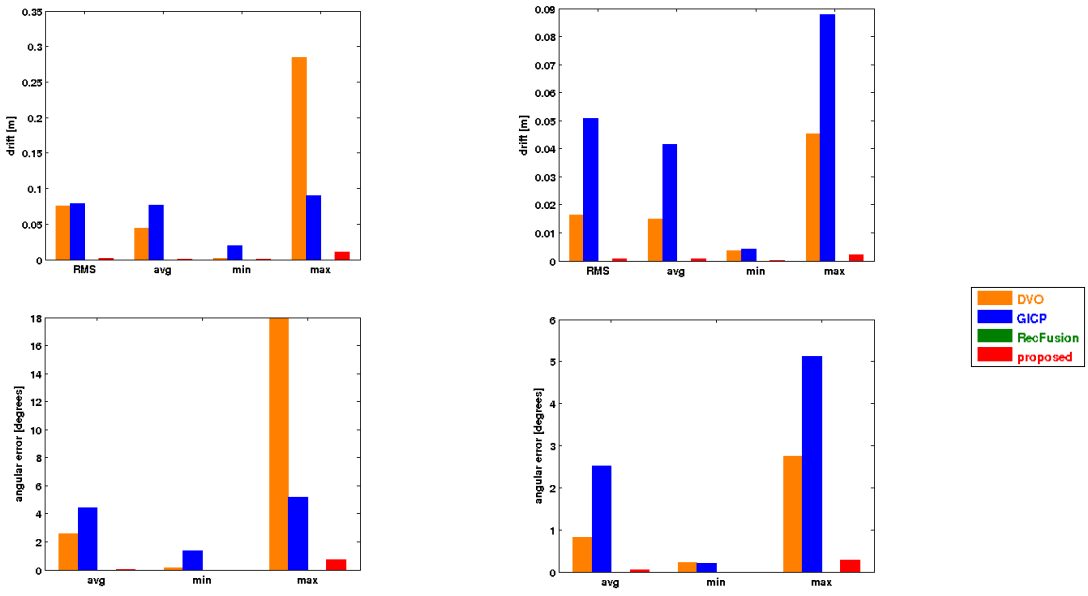
Table 7.5 summarises the evaluation of the drift and angular error for the four approaches and Figure 7.10 juxtaposes these values. Our method has more than a magnitude better results. The average and root-mean-squared drift per pose increment is around 1 mm for both objects, which once again proves that we are only limited by the voxel size. Moreover, the average angular error per transformation is only 0.045° and it never surpasses 0.72°. All metrics are slightly higher for the *turbine* because it is a larger object which is more difficult to handle when viewed from the side, since it is very thin. These tracking errors are small enough to be handled by subsequent pose optimisation or to be compensated during distance field averaging, making our technique suitable for inclusion in an industrial evaluation process.

Reconstruction Results

Figure 7.11 shows the meshes obtained at the end of the pipeline. In addition to the smooth geometry resulting from the highly accurate tracking and the fine depth resolution of the range images, the models exhibit extremely realistic colouring thanks to the high resolution RGB camera. Thus our approach can be successfully applied for non-destructive evaluation of industrial components.

object	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
turbine	DVO	0.0754960	0.0439403	0.0013761	0.2847936	2.5815507	0.1312212	17.8973068
	GICP	0.0788902	0.0766785	0.0192732	0.0895087	4.4412655	1.3853374	5.1599499
	RecFusion	-	-	-	-	-	-	-
	proposed	**0.0017537**	**0.0010118**	**0.0001398**	**0.0105972**	**0.0455488**	**0.0000000**	**0.7187293**
Pumba	DVO	0.0163201	0.0148293	0.0035756	0.0451828	0.8105102	0.2272818	2.7428998
	GICP	0.0508450	0.0414197	0.0041008	0.0877981	2.5231359	0.1938266	5.1182050
	RecFusion	-	-	-	-	-	-	-
	proposed	**0.0008533**	**0.0007369**	**0.0000006**	**0.0022045**	**0.0445724**	**0.0000000**	**0.2797646**

Table 7.5.: Evaluation metrics for the industrial depth sensor sequences



turbine Pumba

Figure 7.10.: Comparison of evaluation metrics for the industrial depth sensor sequences (top: drift, bottom: angular error)

Figure 7.11.: Reconstructed models from the industrial depth sensor sequences

7.3. Kinect-like Depth Data

The tests on the sequences that we acquired with a Kinect-like sensor proceeded in a similar way: we tracked using SDF grids of resolution 2 mm, after which we started a pose optimisation scheme with 4 and 2 mm voxel size, which was concluded by a TV-L^1 fusion into a 1 mm grid, subsequently meshed via marching cubes. Because the data is noisier, we evaluated the effects of incorporating colour and surface orientation constraints, as well as the influence of depth map refinement.

As displayed in Figure 6.4, we put a markerboard on the turntable used for scanning in order to estimate the groundtruth poses. It was, however, never used in registration. We ensured this by applying a binary mask which keeps only the object and discards all other data (the table and the background). The mask is generated by identifying the table as the dominant plane in the image, and selecting the pixels which belong to an object inside the prism whose base is the table. In our approach the mask is additionally used to determine the constrained volume for frame-to-frame tracking, while it was used for removing the non-static background for **DVO** and **GICP**. This procedure is expected to have a negative effect on **DVO**, whose application is SLAM where the entire environment has to be taken into account. The masks were not used for **RecFusion** since there the bounding volume is set by the user. For it we used a voxel size of 0.8 mm for the *bunny*, *cow*, *phone* and *tape*, 0.9 mm for the *blade*, *milk box*, *book*, *muesli box* and *bench vise*, and 1.0 mm for the *Pumba* model. This resolution was chosen according to the largest voxel size below or equal to 1 mm allowed for the selected bounding box.

Tracking Evaluation

Initially we run the pipeline taking every frame in the sequences, which ranged between 466 and 797 shots. However, we noticed that if we allow for a bigger difference between consecutive poses, the estimation is much more robust since the accumulation of noise in overlapping regions is escaped. In the case of every-frame tracking the estimates have the correct orientation, but the translational component is smaller than the true one. Therefore we will provide evaluations of our approach using the whole sequences and taking poses spaced by about 10°.

Figure 7.12 exhibits the trajectories determined by the four methods using the full sequences. **RecFusion** is best tailored to Kinect data and achieves the best results for all models apart from the two objects with multiple symmetry axes: *tape* and *muesli box*. Our approach consistently provides poses which have correct angular displacement, but insufficient translation, as explained above. **DVO** has similar performance, while **GICP** usually fails. Table 7.6 and Figure 7.14 summarise these experiments. It has to be taken into consideration that **RecFusion** outputs poses only when it successfully tracked, thus it seems that the trajectories were better than ours on the *tape* and *muesli box*, but as the plots showed this is not the case. Thus for the Kinect sequences the tables have to be considered in conjunction with the trajectory plots.

The experimental evaluations testify that the angular error of our approach is rather small. Its average is usually around 0.2-0.3°, while it never exceeds 3°. The translational drift is more considerable. Here it is not only influenced by the tracking grid resolution of 2 mm, but also by the severe noise in the Kinect range images. For the objects with simpler

geometry, such as *Pumba*, *bunny*, *cow* and *book*, the average drift is slightly more than 2 mm. For the other objects it is higher, displaying worst performance for the symmetric *tape* and *muesli box* objects, where it is approximately 8 mm. On all tests **DVO** reported comparable values, while **RecFusion** was superior.

We are confident that the insufficient translational estimate is due to noise in the data and not a shortcoming of the approach, because we managed to track accurately when we allowed for bigger difference between frames. To be completely certain, we generated synthetic sequences of 900 - 1000 poses around a circle, thus mimicking a turntable. The tracking was as good as for any other synthetic dataset. Therefore the inability to determine the full translation is attributed entirely to the inferior quality of the data. The solution is to allow for bigger displacement in Kinect sequences, which is what usually seems to be the case in the RGB-D benchmark datasets.

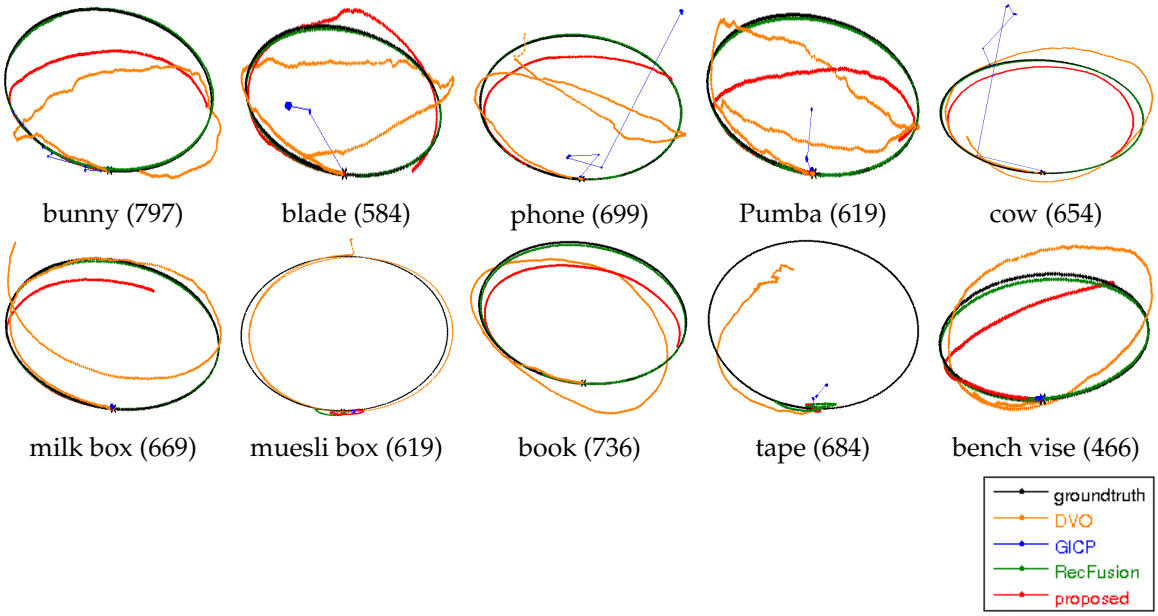


Figure 7.12.: Tracking results on the acquired Kinect sequences (number of frames are shown in brackets)

Figure 7.13 shows how closely the estimated trajectory follows the groundtruth when a bigger step between frames is taken (apart from the cylindrical objects which still cannot be properly handled).

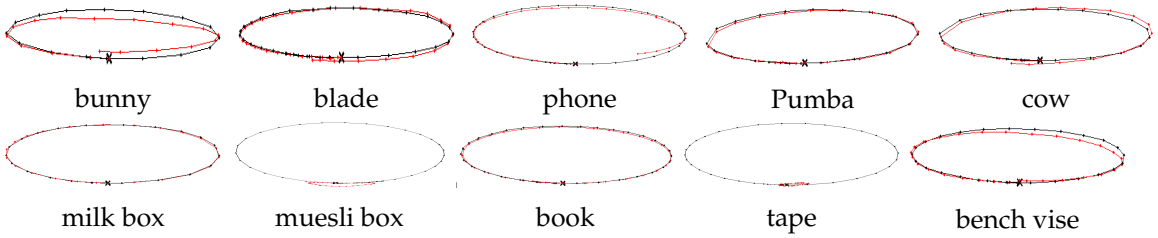


Figure 7.13.: Tracking results on the acquired Kinect sequences with 10-15° pose difference (black: groundtruth, red: proposed approach)

object	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
bunny	DVO	0.0029961	0.0024705	0.0001136	0.0107218	0.1765875	0.0000000	0.8659174
	GICP	0.0162289	0.0065378	0.0000147	0.2915472	0.5508787	0.0000000	21.5782420
	RecFusion	**0.0022874**	**0.0016840**	**0.0000000**	0.0083341	**0.1393537**	0.0000000	0.6954847
	proposed	0.0024197	0.0020535	0.0000517	**0.0070930**	0.1607540	0.0000000	**0.6142098**
blade	DVO	0.0039611	0.0026278	0.0001055	0.0367694	0.2016690	0.0000000	3.1484136
	GICP	0.0161919	0.0086635	0.0000323	0.3002459	0.7426300	0.0000000	23.1740535
	RecFusion	**0.0031469**	**0.0021598**	**0.0000000**	**0.0131423**	**0.1745525**	0.0000000	**1.1021488**
	proposed	0.0036235	0.0024410	0.0001132	0.0343871	0.2030177	0.0000000	2.9430657
phone	DVO	0.0051776	0.0034111	0.0000726	0.0428681	0.2573756	0.0000000	3.3565951
	GICP	0.0531952	0.0097156	0.0000234	1.1668327	0.8938153	0.0000000	108.7421486
	RecFusion	**0.0025258**	**0.0018954**	**0.0000000**	**0.0079023**	**0.1597651**	0.0000000	**0.6620447**
	proposed	0.0034091	0.0027636	0.0001525	0.0132372	0.2324417	0.0000000	1.0623726
Pumba	DVO	0.0037625	0.0032530	0.0000931	0.0116990	0.2545548	0.0000000	0.9004792
	GICP	0.0110224	0.0076240	0.0000230	0.1892488	0.6509678	0.0000000	13.4707395
	RecFusion	0.0027852	**0.0018027**	**0.0000000**	0.0093226	**0.1426995**	0.0000000	0.7813414
	proposed	**0.0025871**	0.0022247	0.0001064	**0.0082168**	0.1741084	0.0000000	**0.7423006**
cow	DVO	0.0027241	0.0023019	0.0000794	**0.0097418**	0.1708640	0.0000000	**0.7121654**
	GICP	0.0556098	0.0106046	0.0001277	1.2443502	0.9210127	0.0000000	122.4836392
	RecFusion	0.0028140	**0.0018914**	**0.0000000**	0.0103314	**0.1508907**	0.0000000	0.9022159
	proposed	**0.0024368**	0.0020346	0.0000709	0.0126151	0.1571830	0.0000000	1.0645805
milk box	DVO	0.0035607	0.0030549	0.0000976	0.0176808	0.2484147	0.0000000	1.5159116
	GICP	0.0068915	0.0065836	0.0000159	0.0131152	0.5655362	0.0000000	1.1232513
	RecFusion	**0.0025251**	**0.0018035**	**0.0000000**	**0.0096997**	**0.1450562**	0.0000000	**0.8411586**
	proposed	0.0038264	0.0032078	0.0000756	0.0097974	0.2655620	0.0000000	0.8586559
muesli box	DVO	0.0040067	0.0034533	0.0002776	0.0206381	0.2956080	0.0000000	1.5354066
	GICP	0.0087010	0.0076520	0.0001121	0.0815659	0.6503062	0.0000000	6.7986422
	RecFusion	0.0018893	0.0009345	0.0000000	0.0114435	0.0760823	0.0000000	0.9159910
	proposed	0.0083629	0.0077459	0.0002388	0.0226854	0.6620541	0.0000000	2.0255682
book	DVO	0.0014849	**0.0012956**	0.0001301	0.0080249	**0.0903972**	0.0000000	0.6293156
	GICP	0.0065536	0.0062131	0.0001803	0.0405928	0.5324232	0.0000000	2.8807000
	RecFusion	0.0037670	0.0016540	**0.0000000**	0.0604979	0.1194941	0.0000000	3.1319614
	proposed	**0.0021326**	0.0019541	0.0001188	**0.0052486**	0.1368279	0.0000000	**0.4073435**
tape	DVO	0.0061668	0.0046230	0.0001064	0.0293352	0.3524186	0.0000000	2.5026526
	GICP	0.0107916	0.0069613	0.0000531	0.1588735	0.5953275	0.0000000	12.2947279
	RecFusion	0.0024238	0.0016331	0.0000000	0.0083547	0.1342302	0.0000000	0.7391305
	proposed	0.0071999	0.0066380	0.0001409	0.0256341	0.5731722	0.0000000	2.2353524
bench vise	DVO	**0.0030094**	0.0024244	0.0001180	**0.0113822**	0.1851294	0.0000000	**0.8613861**
	GICP	0.0102144	0.0095838	0.0000598	0.0224431	0.8200107	0.0000000	1.9077618
	RecFusion	0.0033495	**0.0019318**	**0.0000000**	0.0198757	**0.1526394**	0.0000000	1.6767362
	proposed	0.0056463	0.0044023	0.0001505	0.0201946	0.3660438	0.0000000	1.9007744

Table 7.6.: Evaluation metrics for the acquired Kinect sequences

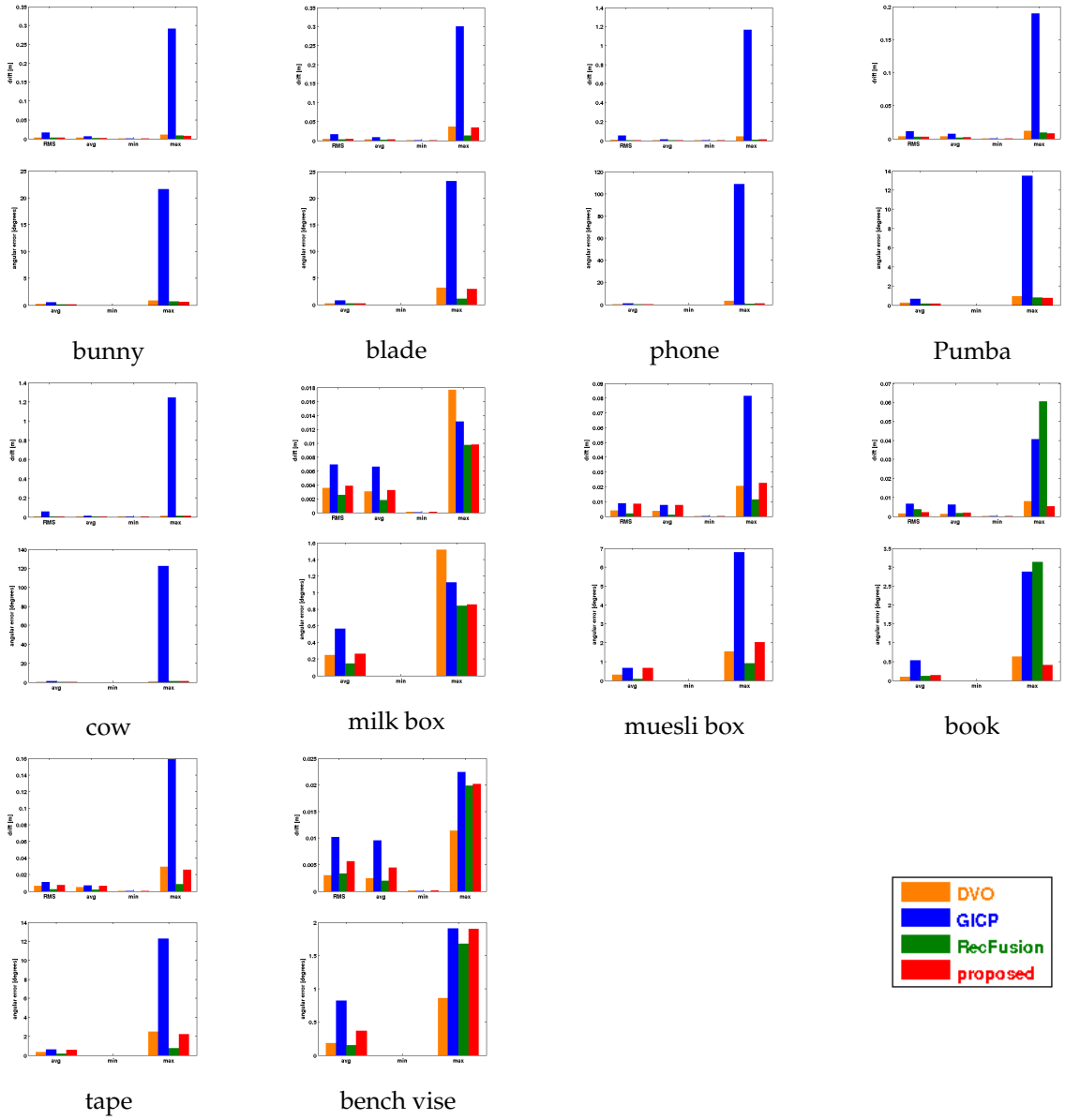


Figure 7.14.: Comparison of evaluation metrics for the acquired Kinect sequences (top: drift, bottom: angular error)

Evaluation of Depth Map Refinement, Surface Colour and Orientation Constraints

For the implicit-to-implicit registration we assessed the effect of adding the remaining components to the objective function. We run tests with 8 types of settings:

- geometry only;
- geometry and surface colour;
- geometry and surface normals;
- geometry, surface colour and normals;
- depth map refinement and geometry;
- depth map refinement, geometry and surface colour;
- depth map refinement, geometry and surface normals;
- depth map refinement, geometry, surface colour and normals.

We applied them to all 10 objects, both with the full and sparse trajectories. In many of the cases the influence of these components was only fractions of millimeters, especially for the complete sequences. When taking frames with difference 10-15°, sometimes adding colour constraints slightly increased the error. This can be attributed to the fact that initially the poses are very different and the colour does not give information which could steer the convergence precisely. Thus it would be safer to start including the colour component only after a certain number of iterations, when the overlap is bigger. Here we will present in Table 7.7 only several of the results, which give good insights into the power of these constraints.

The *phone*, *Pumba* and *book* values show that the separate components usually give a slight boost towards the optimal pose. However, when both the colour and normal constraints are applied on a depth-smoothed image, the outcome is destructive. This is a surprising result, given that even combinations of two of the factors lead to improvements. The weights for the colour and normal energy elements were not changed in the various tests. Therefore, it would most likely be better to decrease them when they are used in combination, so that the reported improvements from separate constraints can be fully taken advantage of in the complete objective function. A similar trend is observed in the *milk box* sequence, however, there depth map refinement generally has a negative effect, and thus the weights have to be tuned in all cases. For the *muesli box* we see that the combination of all three factors actually leads to an overall improvement, since the last row of that section exhibits the lowest errors on all metrics. This is a proof that the complete objective function is well-designed and the only obstacle for it can be caused by the inappropriately chosen weighting factors. For the *tape* sequence using colour constraints is better than using an objective function with geometric component only, both for raw and for smoothed depth maps. Thus, we can conclude that the colour constraints are definitely useful in the case of cylindrical objects, such as the *muesli box* and *tape*. This is a major advantage over methods which are purely based on geometric constraints, such as ICP and

KinectFusion. Although these objects were not tracked perfectly since we were not specifically targeting radial symmetry, this result is a very good indication of the potential of the CTSDF approach. To sum up, various examples have shown that additional constraints are helpful for the registration, but also that the weighs with which they are applied have to be carefully chosen. In order to avoid skipping the optimum, it would be best to pick lower weights and probably wait for more iterations, but converge to a more correct pose.

object	energy components	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
phone	geom.	0.0330152	0.0171178	0.0001015	0.1296510	1.5191253	0.0559529	11.2067923
	geom. & colour	0.0328474	0.0172363	0.0008770	0.1298967	1.5287299	0.1046783	11.2490326
	geom. & normals	0.0332674	0.0172094	0.0007093	0.1311622	1.5264806	0.0685280	11.3167079
	geom. & colour & normals	0.0329447	0.0172852	0.0012678	0.1309204	1.5303922	0.0791294	11.3065895
	smoothing & geom.	0.0257593	0.0136573	0.0009139	0.1314985	1.1882724	0.0685280	11.2179075
	smoothing & geom. & colour	0.0241395	0.0108510	0.0008893	0.1310847	0.9387220	0.0685280	11.2013330
	smoothing & geom. & normals	0.0259263	0.0139940	0.0005483	0.1317983	1.2259654	0.1046783	11.2231483
	smoothing & geom. & colour & normals	0.0581955	0.0198431	0.0011641	0.3122919	1.7314851	0.0685280	27.3844638
Pumba	geom.	0.0068178	0.0062605	0.0026270	0.0135466	0.5346746	0.2272818	1.1521458
	geom. & colour	0.0065507	0.0060126	0.0019612	0.0131974	0.4975070	0.1532334	1.1343455
	geom. & normals	0.0070203	0.0063945	0.0019604	0.0140192	0.5441245	0.1631294	1.1941774
	geom. & colour & normals	0.0069622	0.0064261	0.0017600	0.0134347	0.5307660	0.1312212	1.1480625
	smoothing & geom.	0.0368361	0.0129843	0.0019641	0.1706147	1.1166233	0.2017411	14.6962964
	smoothing & geom. & colour	0.0912488	0.0247310	0.0019871	0.4174388	2.2074584	0.1813082	38.0899068
	smoothing & geom. & normals	0.0068526	0.0061607	0.0019815	0.0138824	0.5096573	0.1813082	1.0594215
	smoothing & geom. & colour & normals	0.1784530	0.0471943	0.0016778	0.7567164	4.0295068	0.1480375	65.0724131
milk box	geom.	0.0141946	0.0122276	0.0009124	0.0242481	1.1100166	0.1769386	2.2103512
	geom. & colour	0.0139616	0.0125923	0.0016950	0.0232424	1.1083245	0.2340679	1.8821526
	geom. & normals	0.0147822	0.0128685	0.0010785	0.0252249	1.1580423	0.1813082	2.2855619
	geom. & colour & normals	0.0146899	0.0132051	0.0010439	0.0252448	1.1570230	0.1897456	2.0298141
	smoothing & geom.	0.0305258	0.0210843	0.0025414	0.1138482	1.7904195	0.2470814	9.4625362
	smoothing & geom. & colour	0.0280429	0.0172924	0.0015985	0.1126510	1.4666432	0.1724585	9.3712569
	smoothing & geom. & normals	0.0311854	0.0218874	0.0013665	0.1145920	1.8723802	0.2502291	9.5268262
	smoothing & geom. & colour & normals	0.0594811	0.0273544	0.0018685	0.2479488	2.3510231	0.1370561	21.4643230
book	geom.	0.0045197	0.0039910	0.0002691	0.0078644	0.2723722	0.0000000	0.5733473
	geom. & colour	0.1841734	0.0374804	0.0004852	0.9915975	3.7095811	0.0685280	100.5907325
	geom. & normals	0.0049316	0.0044107	0.0004044	0.0081536	0.2864446	0.0395647	0.5908274
	geom. & colour & normals	0.1844742	0.0376490	0.0005503	0.9932081	3.7347046	0.0884693	101.1417327
	smoothing & geom.	0.0042161	0.0037154	0.0004122	0.0076038	0.2548893	0.0000000	0.5453620
	smoothing & geom. & colour	0.0033811	0.0029451	0.0002202	0.0066490	0.2155428	0.0000000	0.5066755
	smoothing & geom. & normals	0.0045111	0.0039627	0.0005605	0.0079867	0.2626627	0.0000000	0.5706105
	smoothing & geom. & colour & normals	0.2094471	0.0426291	0.0002953	1.1081301	4.4713694	0.0000000	118.9530980
muesli box	geom.	0.1353218	0.1279968	0.0054243	0.2200940	11.0360982	0.5248852	19.5853143
	geom. & colour	0.1659903	0.1365592	0.0036326	0.5917493	11.8157185	0.3582736	53.1343984
	geom. & normals	0.1348871	0.1282822	0.0070962	0.2156851	11.0548038	0.6921003	19.2021911
	geom. & colour & normals	0.2807137	0.1727565	0.0038308	1.1811228	15.6849775	0.3753437	121.0930540
	smoothing & geom.	0.1400392	0.1325081	0.0059923	0.2546834	11.3801455	0.6039294	21.4891764
	smoothing & geom. & colour	0.1685427	0.1399127	0.0046070	0.5936613	12.0593189	0.4528388	54.0464211
	smoothing & geom. & normals	0.1504152	0.1367141	0.0051215	0.4076284	11.7487826	0.5352216	34.0269120
	smoothing & geom. & colour & normals	0.1288450	0.1223788	0.0045989	0.1805117	10.4546700	0.4545639	16.0608529
tape	geom.	0.1331180	0.1257406	0.0047078	0.2680522	10.8751729	0.4925767	22.3147951
	geom. & colour	0.1298956	0.1199721	0.0024160	0.2349937	10.3584524	0.2564085	21.1676220
	geom. & normals	0.1332232	0.1265783	0.0091866	0.2598073	10.9240637	0.8891079	21.6135696
	geom. & colour & normals	0.1382968	0.1268574	0.0031162	0.2444170	10.9581042	0.2960750	21.8534459
	smoothing & geom.	0.1746014	0.1448523	0.0009956	0.5696706	12.5719655	0.1119058	48.8174809
	smoothing & geom. & colour	0.1619988	0.1433948	0.0009699	0.2866856	12.3952415	0.0969133	24.3845306
	smoothing & geom. & normals	0.1456144	0.1346902	0.0014763	0.3087394	11.6406993	0.1678587	25.7450897
	smoothing & geom. & colour & normals	0.1449206	0.1308702	0.0013265	0.2710937	11.3064499	0.1312212	22.8309145

Table 7.7.: Effect of depth refinement, photometric and surface orientation constraints on the acquired Kinect sequences

Pose Optimisation Evaluation

For all models we evaluated the improvement of the keyframes' poses achieved by pose optimisation. This information is summarised in Table 7.8. Since global pose refinement is sensitive to the initialisation, we assessed its effect on sequences that were tracked rather well. In all cases we see that both the drift and the angular error are reduced thanks to the multiview implicit-to-implicit registration.

object	pose optimisation	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
bunny	before	0.0169128	**0.0138685**	**0.0005075**	0.0284748	**1.1930853**	**0.1046783**	2.5580254
	after	**0.0168520**	0.0140344	0.0015053	**0.0279281**	1.2019509	0.1251145	**2.5466782**
blade	before	0.0223594	0.0172369	**0.0026695**	0.0501029	1.5131647	0.1769386	**4.8436013**
	after	**0.0219157**	**0.0166940**	0.0035916	**0.0499341**	**1.5069114**	**0.1370561**	4.8465098
Pumba	before	0.0106541	0.0100654	0.0060300	**0.0159071**	0.8480610	0.4829489	1.3417113
	after	**0.0105636**	**0.0100503**	**0.0059782**	0.0159401	**0.8462280**	**0.4925767**	**1.3311694**
cow	before	0.0114409	0.0100707	**0.0021650**	0.0201669	0.8720119	**0.1897456**	1.7196012
	after	**0.0112237**	**0.0099255**	0.0022096	**0.0196599**	**0.8701505**	0.1938266	**1.7127599**
bench vise	before	0.0516560	0.0273467	0.0032767	0.1574797	2.2530699	0.3604515	**12.6338549**
	after	**0.0515619**	**0.0271060**	**0.0031035**	**0.1574276**	**2.2459743**	**0.3471788**	12.6606439

Table 7.8.: Effect of pose optimisation on the acquired Kinect sequences

Although the reported improvements here are only slight, we see that there is always a boost. Thus in the case of more severe errors in few of the poses, the effect will be more noticeable and the precise poses will still be successfully recovered. Naturally, there is a limit on the amount of initial error that can be handled, which is investigated later in this chapter.

Reconstruction Evaluation

The final models produced after tracking with sufficient angular difference between frames and subsequent pose optimisation are shown in Figure 7.15. Note that the background removal masks were too inaccurate for the *phone* sequence, so for visual appeal the displayed model is from an earlier test in which we recorded without a markerboard. In addition, the **RecFusion** output models are also shown for comparison.

The models are of similar quality. We achieve the smoothness through the depth map refinement procedure (*cf.* Section 5.2) and the TV-L^1 SDF fusion (*cf.* Section 3.4.4). However, we take care not to apply these techniques too aggressively, so that no false data is introduced into the models. On the other hand, **RecFusion** heavily applies a bilateral filter to every incoming depth map, which is why the models look smoother than ours even when the trajectories are inferior.

Both methods fail to deliver closed meshes for the cylindrical *muesli box* and *tape* objects, because none was successful in tracking. There is a notable difference in the *book* sequence, which our technique handled better. This can be recognised from the letters of the "C++" title, which are blurred in the **RecFusion** model and sharp in ours. Thus we can state that the proposed approach manages objects with poor geometry better than *KinectFusion*.

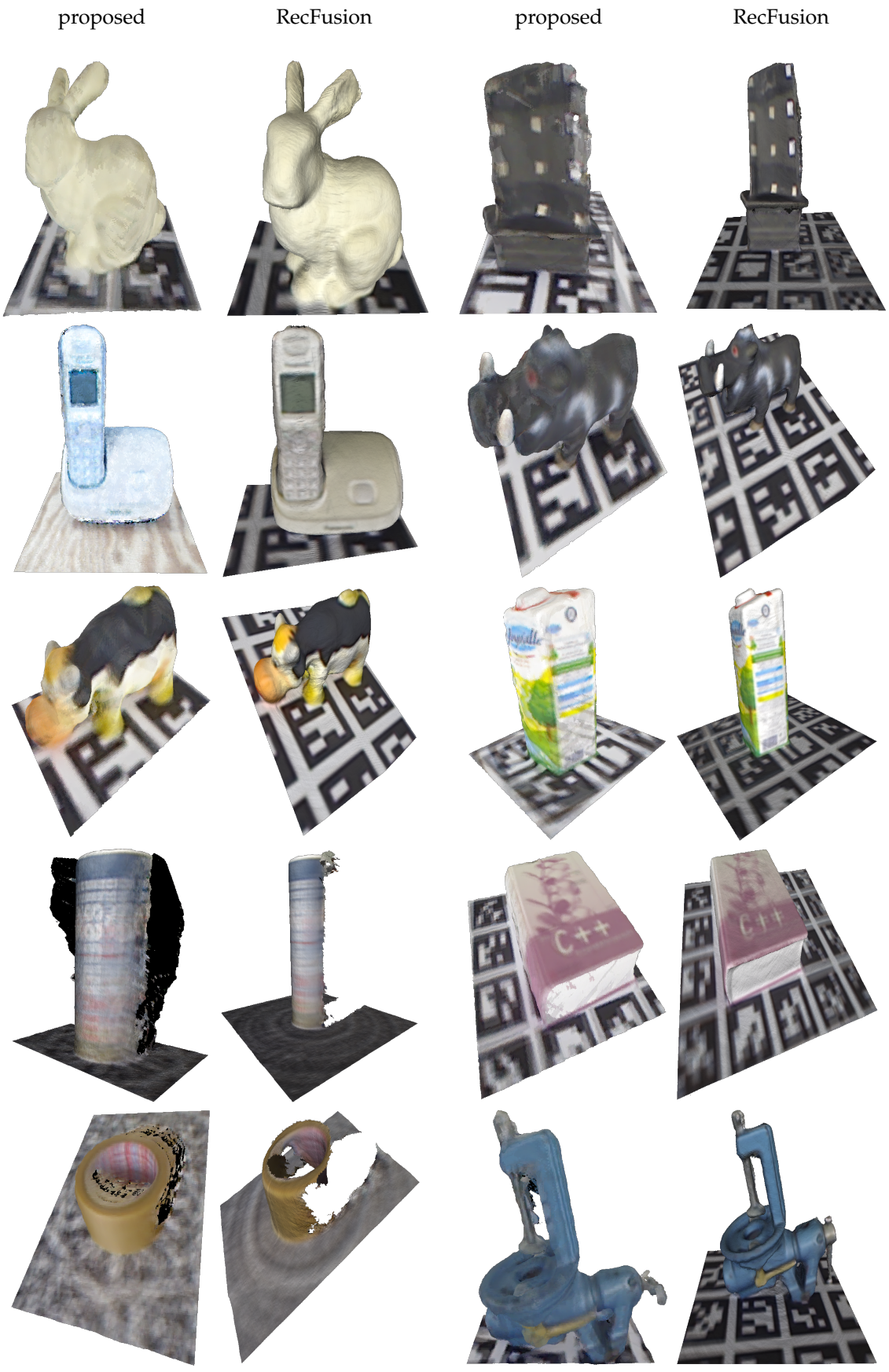


Figure 7.15.: Comparison of reconstructed models by the proposed approach and **RecFusion**

The CAD models for the *bunny* and the *blade* were available, so we further compared our and the **RecFusion** meshes to them, as can be seen in Figure 7.16 and in Table 7.9. The errors of both models are comparable, with the **RecFusion** *bunny* being slightly more precise than ours and our *blade* a bit more accurate than the other one. In combination with the fact that both of our meshes are generated from a grid of voxel size 1 mm, while the **RecFusion** *bunny* was of resolution 0.8 mm and *blade* had 0.9 mm, we conclude that the proposed approach can deliver extremely precise models. The mean error is less than 1.5 mm, which taken into account with the high amount of noise in Kinect data, is rather precise.

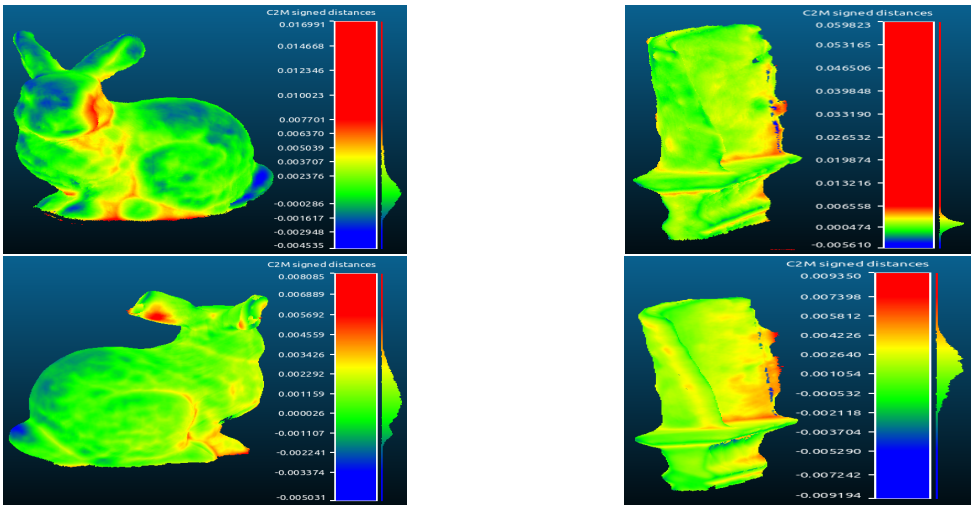


Figure 7.16.: Comparison of output meshes against the original CAD models in *CloudCompare* (top: proposed approach, bottom: **RecFusion**)

object	method	mean [m]	std. dev. [m]
bunny	RecFusion	**0.000829**	**0.001338**
	proposed	0.001107	0.001658
blade	RecFusion	0.001540	0.001695
	proposed	**0.001484**	**0.001645**

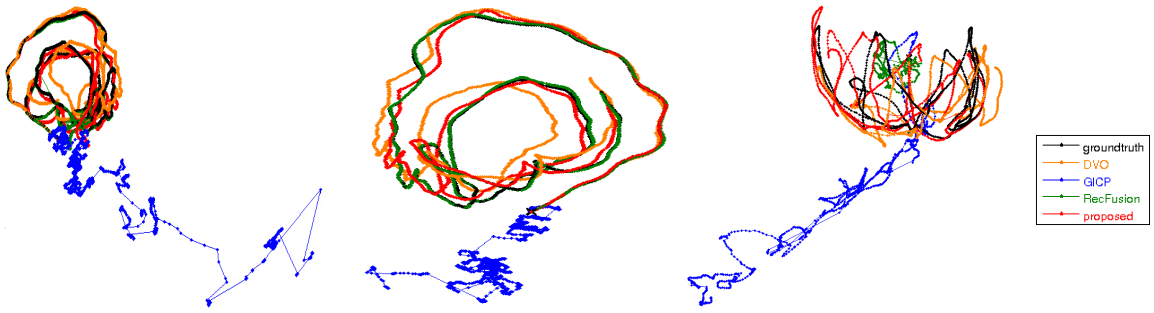
Table 7.9.: *CloudCompare* evaluation metrics against the original CAD models for the acquired Kinect sequences

7.4. RGB-D Benchmark

Since the RGB-D benchmark is composed from Kinect-like images, we followed the same procedure as for our own Kinect sequences, i.e. we included an evaluation of range image smoothing and surface colour and orientation constraints. Contrary to the approach in the previous section, here we did not have masks for the background at our disposal. Therefore, we used a threshold on the distance as a way to only consider regions dominated by the object. For the *fr3/teddy* sequence this was 1.5 m, while for the *fr1/plant* it was 1.2 m.

Tracking Evaluation

Figure 7.17 shows the estimated trajectories on all the 2323 images of the *fr3/teddy* sequence and on a segment with 1500 poses for better visualisation, as well as on all the 1121 of the *fr1/plant* sequence. The motion displayed for our approach is based solely on the geometric component of the energy function, without depth map smoothing. **RecFusion** outputted 2056 poses out of the *fr3/teddy* trajectory and only 366 out of the *fr1/plant* one, indicating the amount of frames which are extremely difficult to handle. It was run with grid resolutions of 2.0 and 1.9 mm respectively. The middle image displays quite clearly how well **RecFusion** can track the *fr3/teddy* sequence, followed by our approach with a slight drift. It can be observed that the three methods, apart from **GICP**, track rather well at the beginning and start gradually drifting around frame 300 and 350 respectively, where the object is at a high distance and thus the noise in the range maps is significant. Based on the failure rate of the ICP implementation in both scenarios, we can state that the overlap between consecutive frames is too small to be handled by such unconstrained approaches that seek explicit correspondences from geometric cues. As the drift is too severe towards the end and the trajectories too long to allow for pose optimisation using a low number of keyframes, we will not proceed with an evaluation of the reconstruction component of the pipeline.



fr3/teddy: whole sequence *fr3/teddy*: first 1500 frames *fr1/plant*: whole sequence

Figure 7.17.: Estimated trajectories on the RGB-D benchmark sequences

Because of the accumulation of drift, we evaluated the sequences in stages of 400 frames. Note that this is not possible to evaluate for the results from **RecFusion**, since the approach outputs only the successfully estimated poses without associating them with the original frame number. More exact tracking comparisons are provided in Tables 7.10 and 7.11. As already observed from the trajectory plots, **GICP** performs worst in all cases.

For the *fr3/teddy* sequence there is a clear trend that our approach provides the best results on the majority of metrics, apart from the range 1601 - 2000. The reason for this is that between frames 1900 and 2000 the camera approaches very near to the object and in some frames only a few pixels on the rims of the teddy's ears are taken into consideration. The one metric on which **DVO** is better than us is the average angular error, but ours is not significantly higher and is within 1° for all frame ranges. Moreover, the errors reported for **RecFusion** are higher than ours on the entire sequence. Thus we can conclude that our approach is performing best overall in tracking *fr3/teddy*.

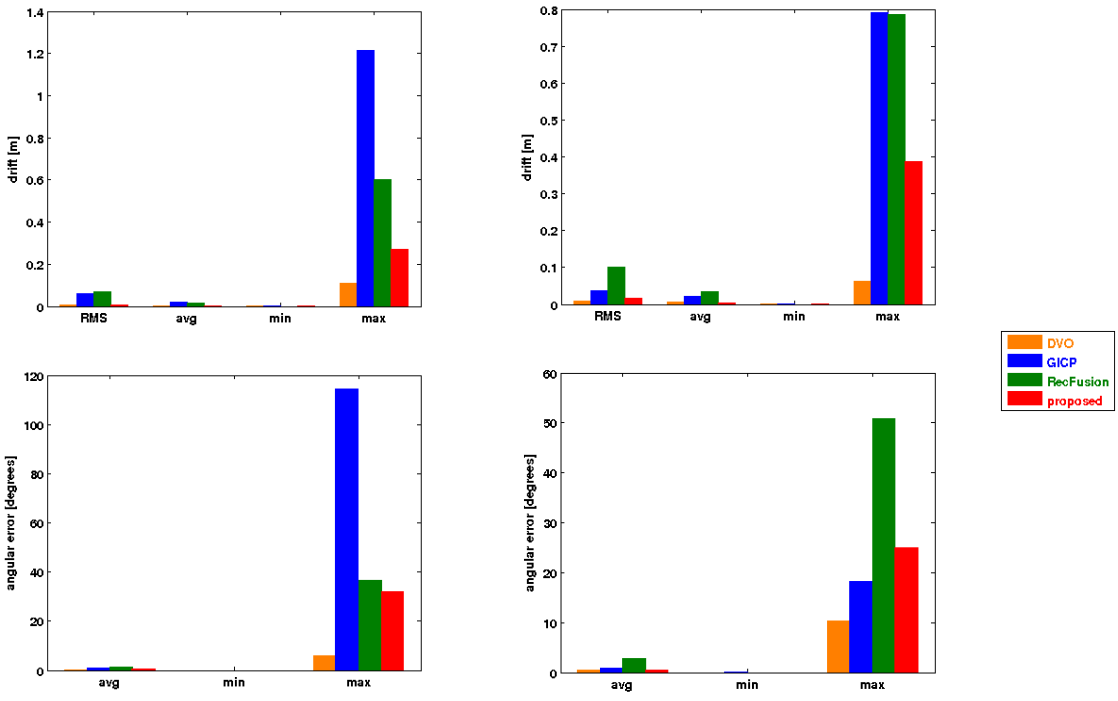
The situation is slightly different for *fr1/plant*, where **DVO** performs best on the first half of the sequence and we prevail in the second. The poorer performance can be attributed to the geometry of the object, since a lot of data can be missing or severely influenced by noise along the borders of the leaves. We, however, consistently show the lowest average drift per frame. Our root-mean-squared drift never exceeds 2 cm per frame and is an order of magnitude lower on average. This is acceptable considering the amount of blurred frames and the fact that we are limited by the 2 mm voxel size for tracking, which is rather big for fine structures like the leaves. The comparative metrics for the complete sequences are visualised in Figure 7.18, where the results discussed above can be verified.

frame range	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
whole	DVO	**0.0062448**	0.0042215	0.0001592	0.1103380	**0.3741795**	0.0000000	**6.0301245**
	GICP	0.0615412	0.0194757	0.0001188	1.2141553	1.1360191	0.0000000	114.3701953
	RecFusion	0.0704261	0.0178351	**0.0000000**	0.6035286	1.3389024	0.0000000	36.7321842
	proposed	0.0084534	**0.0032362**	0.0000313	0.2726533	0.5902023	0.0000000	31.9525648
1 - 400	DVO	0.0041724	0.0033793	0.0003014	0.0269906	**0.2739163**	0.0000000	2.4386559
	GICP	0.0208602	0.0141579	0.0025221	0.3036752	0.7339636	0.1312212	9.2459220
	proposed	**0.0037044**	**0.0030159**	**0.0002271**	**0.0126066**	0.3604114	0.0000000	**1.5825999**
401 - 800	DVO	0.0054855	0.0045785	**0.0001592**	0.0228796	**0.3607833**	0.0000000	**2.2496632**
	GICP	0.0344605	0.0177440	0.0013174	0.4220929	0.9295328	0.1426525	9.0824498
	proposed	**0.0037550**	**0.0031455**	0.0002530	**0.0146447**	0.6164618	0.0000000	2.3768890
801 - 1200	DVO	0.0053592	0.0043560	0.0002273	0.0244846	**0.4194619**	0.0395647	3.8694397
	GICP	0.0168206	0.0116990	0.0007795	0.1346590	1.0982952	0.1769386	4.8587716
	proposed	**0.0026769**	**0.0022252**	**0.0001432**	**0.0124110**	0.5683427	**0.0000000**	**3.1600775**
1201 - 1600	DVO	0.0047603	0.0037578	0.0002788	0.0275269	**0.3868536**	0.0000000	4.3193348
	GICP	0.0425371	0.0190230	0.0020932	0.5302488	1.2164206	0.1724585	12.3893366
	proposed	**0.0038702**	**0.0029836**	**0.0001521**	**0.0160639**	0.6056614	0.0000000	**2.9942195**
1601 - 2000	DVO	**0.0111285**	**0.0059699**	0.0002726	**0.1103380**	**0.5301829**	**0.0000000**	**6.0301245**
	GICP	0.1343815	0.0421182	0.0001529	1.2141553	2.1288933	0.0395647	114.3701953
	proposed	0.0194922	0.0062594	**0.0000313**	0.2726533	1.0399736	0.0395647	31.9525648
2000 - 2323	DVO	0.0042337	0.0033088	0.0002736	0.0163368	**0.2730446**	0.0000000	2.7673334
	GICP	0.0169638	0.0103354	0.0001188	0.1074242	0.6075892	0.0000000	3.6894466
	proposed	**0.0022271**	**0.0016806**	**0.0000998**	**0.0135784**	0.3282948	0.0000000	**2.4951352**

Table 7.10.: Evaluation metrics for different frame ranges of the *fr3/teddy* sequence

frame range	method	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
whole	DVO	**0.0076794**	0.0058513	0.0000918	**0.0636973**	**0.4039900**	0.0000000	**10.3836834**
	GICP	0.0364236	0.0206245	0.0017480	0.7920175	0.9473609	0.0395647	18.2855979
	RecFusion	0.1018840	0.0346787	**0.0000000**	0.7867011	2.8062980	0.0000000	50.8280846
	proposed	0.0153443	**0.0050927**	0.0000718	0.3872918	0.5469910	0.0000000	24.9517047
1 - 400	DVO	**0.0076830**	0.0061924	0.0004915	**0.0344008**	**0.3373028**	0.0000000	**2.3726040**
	GICP	0.0259410	0.0214385	0.0017483	0.1339855	0.9940666	0.0395647	4.4621535
	proposed	0.0199892	**0.0051096**	**0.0001430**	0.3872918	0.5614741	0.0000000	24.9517047
401 - 800	DVO	0.0075951	0.0057697	**0.0000918**	0.0382885	**0.3939977**	0.0000000	4.4141737
	GICP	0.0507847	0.0224434	0.0018565	0.7920175	1.0062904	0.1678587	18.2855979
	proposed	**0.0050090**	**0.0039897**	0.0003906	**0.0326029**	0.4899351	0.0000000	**2.4373715**
801 - 1121	DVO	**0.0131168**	0.0072097	0.0001933	**0.1729702**	0.7709561	0.0000000	62.0822401
	GICP	0.1608326	0.0499305	0.0017480	2.6235206	1.1996066	0.0559529	93.8719731
	proposed	0.0171932	**0.0064801**	**0.0000718**	0.1937379	**0.6013351**	0.0000000	**10.9351049**

Table 7.11.: Evaluation metrics for different frame ranges of the *fr1/plant* sequence



fr3/teddy *fr1/plant*

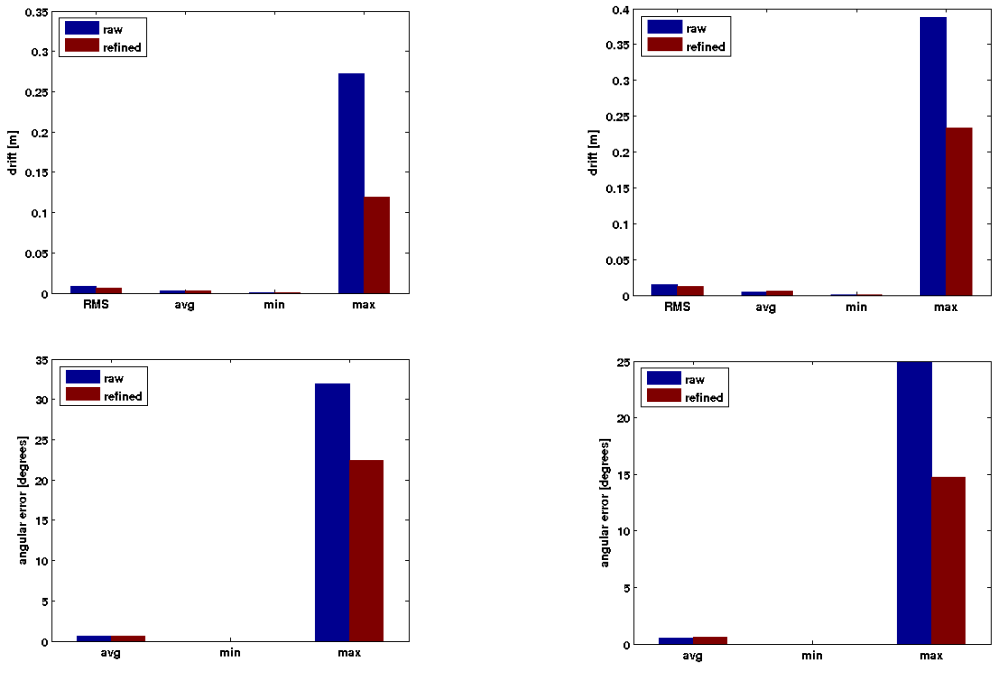
Figure 7.18.: Comparison of evaluation metrics for the RGB-D Benchmark sequences (top: drift, bottom: angular error)

Evaluation of Depth Map Refinement, Surface Colour and Orientation Constraints

Table 7.12 and Figure 7.19 give an overview of the effect of depth smoothing in combination with the geometric component of the objective function on both sequences. Although there is no significant effect on the average angular error, the maximum is decreased by 30-40%. Similarly, the root-mean-squared drift is reduced by 20-30% thanks to the anisotropic diffusion depth refinement. The maximum translational error per transformation is even more remarkably decreased by 40-60%. The minimum drift increases very slightly due to the fact that new values are introduced into the depth maps, but this effect is essentially not noticeable. These results undoubtedly confirm that the chosen depth map refinement scheme is very apt in reducing the pose estimation error stemming from imperfections of the input range measurements.

object	depth map refinement	drift [m/frame]				angular error [°/frame]		
		RMS	avg	min	max	avg	min	max
fr3/teddy	no refinement	0.0084534	0.0032362	**0.0000313**	0.2726533	**0.5902023**	0.0000000	31.9525648
	depth refinement	**0.0060777**	**0.0030907**	0.0000405	**0.1195358**	0.6170335	0.0000000	**22.4242335**
fr1/plant	no refinement	0.0153443	**0.0050927**	**0.0000718**	0.3872918	**0.5469910**	0.0000000	24.9517047
	depth refinement	**0.0122276**	0.0058061	0.0001592	**0.2330571**	0.5955058	0.0000000	**14.7716155**

Table 7.12.: Effect of depth refinement on the RGB-D benchmark sequences



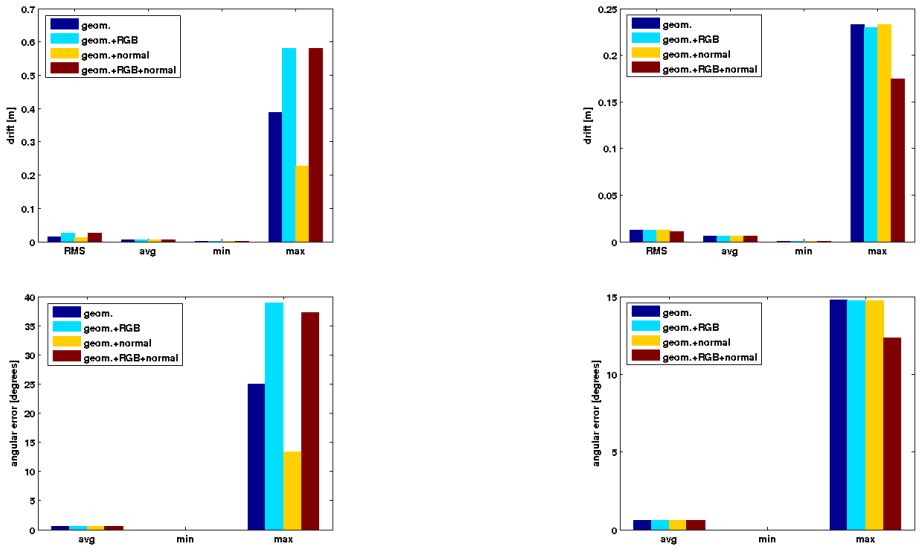
fr3/teddy *fr1/plant*

Figure 7.19.: Comparison of the effect of depth refinement on the RGB-D benchmark sequences

As discussed in the section above, we perform better on the *fr3/teddy* sequence in which the motion is smoother, the object is bigger and fully visible in many frames. Thus we carried out extensive tests on the other sequence, *fr1/plant*, in order to assess the improvement brought by range smoothing, photometric and curvature constraints, and combinations thereof. Table 7.13 and Figure 7.20 display the outcome of these experiments. The conclusion is that including the surface orientation component improves registration in all cases. However, in order for colour to be helpful, the data has to be denoised beforehand, otherwise the error is multiplied. These effects are especially noticeable in the translational component, while the angular error remains only slightly influenced. The root-mean-squared drift is reduced by 30% from using only the signed distance component of the objective function to applying depth map refinement and both surface colour and orientation constraints. To sum up, additional energy components are useful when the data is not noisy.

energy components	drift [m/frame]				angular error [°/frame]		
	RMS	avg	min	max	avg	min	max
geom.	0.0153443	0.0050927	**0.0000718**	0.3872918	0.5469910	0.0000000	24.9517047
geom. & colour	0.0254701	0.0056351	0.0001100	0.5800786	0.5865533	0.0000000	38.9122470
geom. & normals	**0.0121727**	**0.0049778**	0.0002568	**0.2261878**	**0.5379035**	0.0000000	**13.3375963**
geom. & colour & normals	0.0255950	0.0056536	0.0003060	0.5797963	0.5917816	0.0000000	37.2109483
smoothing & geom.	0.0122276	0.0058061	**0.0001592**	0.2330571	0.5955058	0.0000000	14.7716155
smoothing & geom. & colour	0.0122002	0.0057934	0.0002365	0.2292548	0.5917066	0.0000000	14.7502422
smoothing & geom. & normals	0.0122841	0.0058648	0.0002067	0.2327896	0.6010806	0.0000000	14.7501900
smoothing & geom. & colour & normals	**0.0109210**	**0.0057816**	0.0002149	**0.1742158**	**0.5896304**	0.0000000	**12.3558705**

Table 7.13.: Effect of depth refinement, photometric and surface orientation constraints on the *fr1/plant* sequence



raw depth maps refined depth maps

Figure 7.20.: Comparison of the effect of depth refinement, photometric and surface orientation constraints on the *fr1/plant* sequence

7.5. Features

In this section we will summarise the influence of the various registration components that were discussed in detail in the data-specific chapters above.

Effect of Incorporating Colour and Surface Normal Constraints

The sections on Kinect sequences proved that adding the photometric and curvature constraints to the objective function leads to improvements when the depth maps are denoised. The surface normals alone help even in the case of noisy data. From an implementational point of view, calculating the normals does not bring additional burden, since they are already present as the gradient of the signed distance field. Thus, it is advisory to always include them, because they speed up convergence, lead to a more optimal solution, do not occupy additional memory and do not entail a decrease in computational speed.

The photometric component has to be handled differently. It only boosts registration when applied on refined geometry. Moreover, it requires the storage of an additional grid, which might be a severe constraint for very fine voxel resolutions. Therefore, it is advisory to use the colour component either on data from high-quality sensors or in applications where the accuracy is critical, while memory and processing time are of low importance.

Effect of Depth Map Smoothing

The Kinect sequence analysis clearly showed that refining the depth maps is advantageous. Since a non-optimised implementation takes a fraction of a second on a single-core CPU, a parallelisation of the smoothing scheme can lead to instantaneous attainment of improved range images. Thus this preprocessing step is highly recommended for better precision.

Twists versus Augmented Unit Quaternions

All tests reported so far were using twists as parameterisation of rigid body motion. We carried out a substantial amount of experiments using the augmented quaternion representation as well. In the case of tracking it usually converged to a slightly more inaccurate pose after 5% less iterations. Therefore if the goal is speed and more imperfections in estimation are allowed, AUQs can be used for frame-to-frame tracking. For pose optimisation, however, the quaternions lead to a 2-3 times faster convergence in most cases, but with a much bigger error when the initial setup contained more error. Thus AUQs can be used only provided a very good initialisation for multiview registration. Based on these observations, we concluded that it is safer to use twist coordinates and preferred them throughout the experimental stages.

Maximal Initial Misalignment

In the section discussing the sequences that we acquired with a Kinect-like sensor we noted the curious fact that there should be a sufficient difference between poses in order to have successful registration on noisy data. To determine this, we experimented with tracking

on difference from 0.3° to 30° with each sequence. We concluded that it is optimal to have at least 2-3° between poses, while a maximum of 25° displacement can be handled for objects with simple geometry and 15° for more complex models. This means that we require significantly less frames than other common tracking approaches.

As pose optimisation is rather sensitive to the initial setting [74], we executed several tests in order to determine the limits of our implicit-to-implicit frame-to-model multiview pose refinement. To achieve this goal, we used the *bunny* and *teddy* synthetic models and two pose setups: 12 poses in a circle around the mid-height of the object, and 42 equally spaced poses on an icosahedron surrounding the whole model. We then introduced various amounts of noise into the poses and run the pose optimisation, but the depth maps remained unaffected. When all views were influenced by noise, the true poses were successfully recovered if the initial error was within 2-3° and 1.5 cm for an object of dimensions 20 cm^3. Naturally, when only few poses were subject to noise, these boundaries were higher. Thus, when there is a smooth accumulation of drift with not too immense translational error, it can be successfully handled by our optimisation scheme. In case the initial poses are significantly displaced from their correct positions, the refinement will lead to a geometrically optimal model, which is however not the real one. So it is advisory to ensure good tracking, followed by a refinement which will converge in a couple of quick iterations.

Part IV.

Conclusion and Future Work

8. Summary

8.1. The Unified 3D Object Reconstruction Pipeline

We have succeeded in achieving the goals set out at the start of this thesis (*cf.* section 1.2). More specifically, a **complete pipeline** for 3D model acquisition from RGB-D data has been developed, in which the **same objective function** describes both the tracking and pose optimisation stages. In addition, explicit correspondence search has been circumvented by solving the registration problem via a **direct implicit-to-implicit optimisation scheme**. Last but not least, all available data has been used, which not only means **dense registration** with the entire depth maps instead of sparse features, but also incorporation of **colour and surface normal constraints**, which are readily attainable from RGB-D image pairs.

After describing the approach, we presented results from multiple tests on data of various quality and from comparisons with state-of-the-art techniques. We juxtaposed our tracking with a method based on photoconsistency (**DVO**), with an algorithm based on geometrical constraints (**GICP**), and with a *KinectFusion* implementation (**RecFusion**). We evaluated our final models against those delivered by **RecFusion**, establishing the comparison on the true CAD models. These experiments revealed that we achieve state-of-the-art tracking precision, while our 3D model accuracy is comparable to that of *KinectFusion*.

Why This Is Not a Bundle Adjustment Approach

Bundle adjustment is a technique for simultaneously refining the 3D point locations describing the scene geometry and the camera parameters for each viewpoint [81]. This is similar to the problem that we solve through the pose optimisation stage of the pipeline: jointly optimising the camera poses using a global CTSDF, from which the 3D geometry can be extracted. However, our approach differs from bundle adjustment.

Classical BA is based on 2D colour images taken from different viewpoints, in which keypoints are detected. The task is to find the 3D locations of these keypoints which provide for the best consensus between all views. Thus the reprojection error between 3D points and 2D coordinates is minimised [15]. BA modifications exist, in which some of the parameters are eliminated from the optimisation scheme via algebraic manipulation, e.g. the camera orientations [93] or the 3D coordinates [75, 89]. It may seem that our pose optimisation is an instance of BA where the explicit 3D coordinates are removed and eventually derived from the SDF, but in fact the approach is fundamentally different. We optimise only for the camera parameters and generate the global 3D weighted average model using these estimates. Thus we never explicitly refine the 3D geometry. To leverage our method as a bundle adjustment variant, the values in the depth maps can be iteratively modified together with the camera poses. Unless such a step is included, the approach cannot be considered as BA. However, we have proven that it is sufficiently powerful without such complications and have therefore used the term *global optimisation* throughout this work.

8.2. Contributions

The main novelty of the proposed approach is the direct implicit-to-implicit registration, incorporating both photometric and surface orientation constraints into a rather simple objective function, applied both in camera tracking and in global pose optimisation. Through extensive experiments we showed that this technique has the following advantages over existing methods:

- **Tracking component**:
  - more accurate trajectory estimation than **state-of-the-art** approaches, such as visual odometry in RGB-D data, ICP and often *KinectFusion*;
  - robustness via frame-to-frame registration with **failure recovery**, instead of frame-to-model matching which is prone to error accumulation;
  - handling of tracking scenarios with **large displacement** between poses;
  - not limited to a constrained volume, as the bounding box is determined on the fly for every frame;
- **Global optimisation and model reconstruction component**:
  - model **accuracy** comparable to *KinectFusion*;
  - robustness thanks to a **coarse-to-fine scheme** on the voxel grid resolution;
  - much **fewer keyframes** than graph optimisation methods required and, therefore, tendency of the algorithm to be much faster;
- **Improved convergence** in each stage through a combination of **geometric, photometric and curvature constraints** on **refined depth maps**.

A major advantage of our pipeline over *KinectFusion* [36] and existing point-to-implicit tracking and reconstruction methods, such as those by Bylow *et al.* [7] and Canelhas *et al.* [9], is the introduction of the pose optimisation stage, which makes the final mesh even more accurate. Since the proposed global refinement scheme requires much less views than graph optimisation techniques such as g^2o [47], this additional step is a quick and computationally undemanding procedure, which is worth including in any reconstruction routine.

9. Future Work

The follow-up work on the pipeline will address the issues that we identified as unimportant for the current approach in the related work chapter (*cf.* Section 2.1).

A brute-force approach to achieve real-time performance is to process the objective function on the GPU. This is what is done in the original *KinectFusion* implementation and many of the other SDF registration techniques [59, 9, 86]. Therefore we have a straightforward way of significantly decreasing the computational time at our disposal.

In order to decrease the memory requirements, we will represent the TSDF grid as an octree. Since large portions of the volume have a constant value of ± 1, representing empty space or regions which have not been observed, the occupied memory will be significantly reduced. An extension of *KinectFusion* which employs octrees for the SDF representation has already been proposed by Zeng *et al.* [91, 92], where the authors suggest a space-efficient data structure on the GPU, together with specialised SDF update and surface prediction algorithms. A popular representation in the graphics community is the adaptively sampled distance field (ADF) [26, 27], which reduces the memory usage due to its octree nature, and the number of accesses during generation by adaptively sampling the surface according to its local complexity. Another feasible approach is the existing open-source *OctoMap* framework [87, 33], which is a tree-based 3D environment representation using a probabilistic occupancy estimation, similar to the weight field of an SDF. These techniques would allow us to extend the pipeline to full environment reconstruction, and therefore make the approach appropriate for SLAM.

If, however, we still target purely CPU-based approaches, we will need another efficient representation. Due to the fact that most values are ± 1, we could store only the remaining values, as indicated by Canelhas *et al.* [9]. However, this method still needs GPU processing. To avoid this, a combination with an octree partitioning scheme would be needed, or a completely different representation.

Other features of an SDF can also be improved. Apart from the voxel size, which is related to the memory requirements, the SDF has two other parameters: δ which reflects the sensor uncertainty, and η which represents the expected object thickness and is used for the calculation of the weights. As suggested by Moravec [57], the sensor model can be learned, whereby we would not have to estimate these parameters. This will decrease errors due to suboptimal truncation and will require less user input.

To sum up, there are numerous possibilities to improve the SDF representation, to reduce its memory requirements and to decrease the registration speed. The extensions of the pipeline to real-time processing and SLAM seem very feasible, for which we have laid out clear and concrete future steps.

Bibliography

[1] J. Abhijit. *Kinect for Windows SDK Programming Guide*. Packt Publishing, 2012.

[2] M. R. Andersen, T. Jensen, P. Lisouski, A. K. Mortensen, M. K. Hansen, T. Gregersen, and P. Ahrendt. Kinect Depth Sensor Evaluation for Computer Vision Applications. Technical report ECE-TR-6, Department of Engineering - Electrical and Computer Engineering, Aarhus University, 2012.

[3] M. Aubry, K. Kolev, B. Goldluecke, and D. Cremers. Decoupling Photometry and Geometry in Dense Variational Camera Calibration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1411–1418, 2011.

[4] F. Bernardini and H. Rushmeier. The 3D Model Acquisition Pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

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

[6] R. M. Bolle and B.C. Vemuri. On Three-Dimensional Surface Reconstruction Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(1):1–13, 1991.

[7] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions. In *Robotics: Science and Systems Conference (RSS)*, 2013.

[8] D. R. Canelhas. Scene Representation, Registration and Object Detection in a Truncated Signed Distance Function Representation of 3D Space. Master's thesis, Department of Technology, Örebro University, 2012.

[9] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal. SDF Tracker: A Parallel Algorithm for On-line Pose Estimation and Scene Reconstruction from Depth Images. In *2013 IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*, 2013.

[10] Y. Chen and G. Medioni. Object Modeling by Registration of Multiple Range Images. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2724–2729, vol. 3, 1991.

[11] Y. Chiou, J. Tsai, and H. Hang. Depth Map Refinement for View Synthesis using Depth Sensors and Color Image Cameras. In *International Workshop on Advanced Image Technology*, 2013.

[12] P. Claes, D. Vandermeulen, L. Van Gool, and P. Suetens. Automatic, Robust and Accurate 3D Modelling based on Variational Implicit Surfaces. Technical report KUL/ESAT/PSI/0405, Katholieke Universiteit Leuven - Center for Processing Speech and Images, 2004.

[13] P. Claes, D. Vandermeulen, L. Van Gool, and P. Suetens. Robust and Accurate Partial Surface Registration based on Variational Implicit Surfaces for Automatic 3D Model Building. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2005.

[14] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, 1996.

[15] F. Dellaert. Visual SLAM Tutorial: Bundle Adjustment. Technical report, June 2014.

[16] J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, 2006.

[17] M. Dimashova, I. Lysenkov, V. Rabaud, and V. Eruhimov. Tabletop Object Scanning with an RGB-D Sensor. In *Third Workshop on Semantic Perception, Mapping and Exploration (SPME) at the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[18] E. Eade. Lie groups for 2D and 3D Transformations. Technical report, 2013.

[19] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-D Rigid Body Transformations: A Comparison of Four Major Algorithms. *Machine Vision and Application*, 9(5-6):272–290, 1997.

[20] A. Elfes and L. Matthies. Sensor Integration for Robot Navigation: Combining Sonar and Stereo Range Data in a Grid-Based Representataion. In *26th IEEE Conference on Decision and Control*, volume 26, pages 1802–1807, 1987.

[21] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An Evaluation of the RGB-D SLAM System. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[22] S. J. Esses, P. Berman, A. I. Bloom, and J. Sosna. Clinical Applications of Physical 3D Models Derived From MDCT Data and Created by Rapid Prototyping. *American Journal of Roentgenology*, 196(6):683–688, 2011.

[23] H. Euler. Non-Destructive Evaluation Techniques Combined - The Global Inspection System. Technical report, Siemens AG - Corporate Technology, 2010.

[24] J. A. Farrell. Computation of the Quaternion from a Rotation Matrix. Technical report, University of California, Riverside, 2008.

[25] A. W. Fitzgibbon. Robust Registration of 2D and 3D Point Sets. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–10, 2001.

[26] S. F. Frisken and R. N. Perry. Efficient Estimation of 3D Euclidean Distance Fields from 2D Range Images. In *Proceedings of the IEEE / ACM SIGGRAPH Symposium on Volume Visualization and Graphics*, pages 81–88, 2002.

[27] S. F. Frisken and R. N. Perry. Designing with Distance Fields. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 60–66, 2006.

[28] F. B. Gonzalez. Lie Algebras. Lecture notes, Tufts University, 2007.

[29] G. Graber, T. Pock, and H. Bischof. Online 3D Reconstruction Using Convex Optimization. In *1st Workshop on Live Dense Reconstruction From Moving Cameras (ICCV*, 2011.

[30] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, USA, 2nd edition, 2003.

[31] P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch Volumes: Segmentation-based Consistent Mapping with RGB-D Cameras. In *Proceedings of the 2013 International Conference on 3D Vision (3DV)*, pages 398–405, 2013.

[32] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *International Symposium on Experimental Robotics*, 2010.

[33] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: an Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[34] D. F. Huber and M. Hebert. Fully Automatic Registration of Multiple 3D Data Sets. *Image and Vision Computing*, 21(7):637–650, 2003.

[35] V. G. Ivancevic and T. T. Ivancevic. Lecture Notes in Lie Groups. Technical report, ArXiV e-prints, 2011.

[36] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2011.

[37] A. Johnson and S. B. Kang. Registration and Integration of Textured 3D Data. *Image and Vision Computing*, 17.

[38] M. Jones, J. A. Baerentzen, and M. Sramek. 3D Distance Fields: A Survey of Techniques and Applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, 2006.

[39] W. Kehl, N. Navab, and S. Ilic. Coloured Signed Distance Fields for full 3D Object Reconstruction. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[40] C. Kerl, J. Sturm, and D. Cremers. Dense Visual SLAM for RGB-D Cameras. In *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, 2013.

[41] C. Kerl, J. Sturm, and D. Cremers. Robust Odometry Estimation for RGB-D Cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[42] K. Khoshelham and S. O. Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2):1437–1454, 2012.

[43] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[44] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint Bilateral Upsampling. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, 2007.

[45] D. B. Kubacki. Signed Distance Registration for Depth Image Sequence. Master's thesis, University of Illinois at Urbana-Champaign, 2011.

[46] D. B. Kubacki, H. Q. Bui, S. D. Babacan, and M. N. Do. Registration and Integration of Multiple Depth Images using Signed Distance Function. In *SPIE Proceedings*, volume 8296, 2012.

[47] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, May 2011.

[48] A. V. Le, S. Jung, and C. S. Won. Directional Joint Bilateral Filter for Depth Images. *Sensors*, 14:11362–11378, 2014.

[49] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proceedings of ACM SIGGRAPH 2000*, pages 131–144, 2000.

[50] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, 1987.

[51] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

[52] A. Makadia, A. Patterson, and K. Daniilidis. Fully Automatic Registration of 3D Point Clouds. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1297–1304, 2006.

[53] T. Masuda. Generation of Geometric Model by Registration and Integration of Multiple Range Images. In *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, 2001.

[54] T. Masuda. Registration and Integration of Multiple Range Images by Matching Signed Distance Fields for Object Shape Modeling. *Computer Vision and Image Understanding (CVIU)*, 87(1-3):51–65, 2002.

[55] T. Matsuo, N. Fukushima, and Y. Ishibashi. Weighted Joint Bilateral Filter with Slope Depth Compensation Filter for Depth Map Refinement. In *8th International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 300–309, 2013.

[56] S. Milani and G. Calvagno. Joint Denoising and Interpolation of Depth Maps for MS Kinect Sensors. In *IEEE International Conference onAcoustics, Speech and Signal Processing (ICASSP)*, pages 797–800, 2012.

[57] H. P. Moravec. Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids. Technical report, The Robotic Institute, Carnegie Mellon University, 1996.

[58] P. J. Neugebauer. Geometrical Cloning of 3D Objects via Simultaneous Registration of Multiple Range Images. In *Proceedings of the International Conference on Shape Modeling and Applications*, pages 130–139, 1997.

[59] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *10th International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[60] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327, 2011.

[61] D. Nister, O. Naroditsky, and J. Bergen. Visual Odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[62] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153 of *Applied Mathematical Science*. Springer, 2003.

[63] N. Paragios, M. Rousson, and V. Ramesh. Non-Rigid Registration Using Distance Functions. *Computer Vision and Image Understanding*, 89(2-3):142–165, 2003.

[64] V. A. Prisacariu and I. D. Reid. PWP3D: Real-time Segmentation and Tracking of 3D Objects. *International Journal of Computer Vision*, 98(3):335–354, 2012.

[65] K. Pulli. Multiview registration for large data sets. In *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling*, pages 160–168, 1999.

[66] B. Rasolzadeh, M. Björkman, K. Huebner, and D. Kragic. An Active Vision System for Detecting, Fixating and Manipulating Objects in the Real World. *International Journal of Robotics Research*, 29(2-3):133–154, 2009.

[67] C. Y. Ren and I. Reid. A Unified Energy Minimization Framework for Model Fitting in Depth. In *Proceedings of the 12th International Conference on Computer Vision*, volume 2, pages 72–82, 2012.

[68] M. Rouhani and A. D. Sappa. The Richer Representation the Better Registration. *IEEE Transactions on Image Processing*, 22(12):5036–5049, 2013.

[69] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard. Range Sensor Based Model Construction by Sparse Surface Adjustment. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 46–49, 2011.

[70] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *3rd International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[71] C. Schroers, H. Zimmer, L. Valgaerts, A. Bruhn, O. Demetz, and J. Weickert. Anisotropic Range Image Integration. In *Proceedings of Pattern Recognition - Joint 34th DAGM and 36th OAGM Symposium*, pages 73–82, 2012.

[72] C. Schütz, T. Jost, and H. Hugli. Multi-Feature Matching Algorithm for Free-Form 3D Surface Registration. In *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR)*, pages 982–984, vol. 2, 1998.

[73] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proceesings of Robotics: Science and Systems (RSS)*, 2009.

[74] S.-W. Shih, Y.-T. Chuang, and T.-Y. Yu. An Efficient and Accurate Method for the Relaxation of Multiview Registration Error. *IEEE Transactions on Image Processing*, 17(6):968–981, 2008.

[75] R. Steffen, J.-M. Frahm, and W. Förstner. Relative Bundle Adjustment Based on Trifocal Constraints. In *Proceedings of the 11th European Conference on Trends and Topics in Computer Vision - Volume Part II*, ECCV'10, pages 282–295, 2012.

[76] F. Steinbrücker, J. Sturm, and D. Cremers. Real-Time Visual Odometry from Dense RGB-D Images. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011.

[77] T. Stoyanov, A. Louloudi, H. Andreasson, and A. J. Lilienthal. Comparative Evaluation of Range Sensor Accuracy in Indoor Environments. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, 2011.

[78] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

[79] Y. Takase, N. Sho, A. Sone, and K. Shimiya. Automatic Generation of 3D City Models and Related Applications. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 113–120, 2003.

[80] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *Sixth IEEE International Conference on Computer Vision (ICCV)*, pages 839–846, 1998.

[81] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, 2000.

[82] D. Tubic, P. Hebert, and D. Laurendeau. A Volumetric Approach for Interactive 3D Modeling. In *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission*, pages 150–158, 2002.

[83] B. Ummenhofer and T. Brox. Dense 3D Reconstruction with a Hand-held Camera. In *Pattern Recognition (Proceedings of DAGM)*, LNCS, 2012.

[84] J. Van Verth. Understanding Rotations. Technical report, Game Developers Conference, 2012.

[85] K. R. Vijayanagar, M. Loghman, and J. Kim. Real-Time Refinement of Kinect Depth Maps using Multi-Resolution Anisotropic Diffusion. *Mobile Networks and Applications*, 19(3):414–425, 2014.

[86] T. Whelan, J. B. McDonald, M. Kaess, M. F. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[87] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[88] Q.-Z. Ye. The Signed Euclidean Distance Transform and Its Applications. In *9th International Conference on Pattern Recognition*, volume 1, pages 495–499, 1988.

[89] C. Zach. Robust Bundle Adjustment Revisited. In *European Conference on Computer Vision (ECCV)*, 2014.

[90] C. Zach, T. Pock, and H. Bischof. A Globally Optimal Algorithm for Robust TV-L^1 Range Image Integration. In *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[91] M. Zeng, F. Zhao, J. Zheng, and X. Liu. A Memory-efficient Kinectfusion Using Octree. In *Proceedings of the First International Conference on Computational Visual Media*, CVM'12, 2012.

[92] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based Fusion for Realtime 3D Reconstruction. *Graphical Models*, 75(3):126–136, 2013.

[93] J. Zhang, M. Boutin, and D. G. Aliaga. Robust Bundle Adjustment for Structure from Motion. In *IEEE International Conference on Image Processing*, pages 2185–2188, 2006.

[94] Z. Zhang. Iterative Point Matching for Registration of Free-Form Curves and Surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[95] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(11):1330–1334, 2000.

[96] P. Zombo and R. Shannon. Advanced NDE Systems for Flexible Operation and Maintenance of Gas Turbine Components. Technical report, Siemens Power Generation, Inc., November 2006. POWER-GEN International Conference 2006.