Fachbereich 4: Informatik

Erstellung einer 3D-Karte aus RGB-D-Daten unter Benutzung visueller Odometrie

Bachelorarbeit
zur Erlangung des Grades
Bachelor of Science
im Studiengang Computervisualistik

vorgelegt von

Raphael Memmesheimer

Betreuer: Dipl.-Inform. Viktor Seib, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Dipl.-Inform. Viktor Seib, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im Dezember 2014

Kurzfassung

In dieser Arbeit prärentieren wir Methoden zum Schätzen von Kamerabewegungen einer RGB-D-Kamera in 6 Freiheitsgraden und dem erstellen von 3D-Karten. Als erstes werden die RGB- und Tiefendaten registriert und synchronisiert. Nach der Vorverarbeitung extrahieren wir FAST-Merkmale in zwei aufeinander folgenden Bildern. Daraus wird eine Korrespondenzmenge erstellt und Ausreißer werden herausgefiltert. Anschließend projezieren wir die Korrespondenzmenge in 3D um die Bewegung aus 3D-3D-Korrespondezen mittels Least-Squares zu bestimmen. Weiterhin präsentieren wir Methoden um 3D-Karten aus Bewegungsschätzungen und RGB-D-Daten zu erstellen. Dafür benutzen wir das OctoMap-Framework und erstellen wahlweise auch inkrementelle Karten aus Punktewolken. Anschließend evaluieren wir das System mit dem weit verbreiteten RGB-D-Benchmark.

Abstract

In this thesis we present an approach to track a RGB-D camera in 6DOF and construct 3D maps. We first acquire, register and synchronize RGB- and depth images. After preprocessing we extract FAST features and match them between two consecutive frames. By depth projection we regain the z-value for the inlier correspondences. Afterwards we estimate the camera motion by 3D point set alignment between the correspondence set using least-squares. This local motion estimate is incrementally applied to a global transformation. Additionally we present methods to build maps based on point cloud data acquired by a RGB-D camera. For map creation we use the OctoMap framework and optionally create a colored point cloud map. The system is evaluated with the widespread RGB-D benchmark.

Erklärung

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	ja ☐	nein ☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	ja ☐	nein ☐

Koblenz, den 5. Januar 2015

Contents

1 Introduction **9**
- 1.1 Motivation . . . 9
- 1.2 Goals . . . 11
- 1.3 Implementation . . . 12
- 1.4 Notation . . . 13
- 1.5 Outline . . . 13

2 State of the Art **15**
- 2.1 Feature Based Methods . . . 16
- 2.2 Dense Methods . . . 18
- 2.3 Depth Based Approaches . . . 20

3 Visual Odometry **23**
- 3.1 Image Preprocessing . . . 24
- 3.2 Feature Extraction . . . 24
  - 3.2.1 FAST . . . 25
  - 3.2.2 Scale Invariant Features (SIFT) Descriptor . . . 25
- 3.3 Feature Matching . . . 26
- 3.4 Inlier Detection . . . 27
- 3.5 Motion Estimation . . . 28
  - 3.5.1 SVD . . . 28
  - 3.5.2 Least-Squares . . . 29
  - 3.5.3 RANSAC . . . 30
- 3.6 Camera Calibration . . . 31
- 3.7 Depth Registration . . . 33
- 3.8 Time Synchronization . . . 35

4 Mapping **39**
- 4.1 OctoMap . . . 40
  - 4.1.1 Occupancy Gridmaps . . . 40
  - 4.1.2 Octrees . . . 41

4.1.3 Construction . . . 41
4.2 Point Cloud Map . . . 42
4.2.1 Iterative Closest Points . . . 42

5 Evaluation 45
5.1 RGB-D Benchmark . . . 45
5.1.1 Datasets . . . 46
5.1.2 Evaluation Metrics . . . 47
5.2 Results . . . 48
5.2.1 Accuracy . . . 48
5.2.2 Suitability for Mapping . . . 51
5.2.3 Comparison . . . 51
5.2.4 Runtime . . . 51

6 Conclusion 53
6.1 Conclusion . . . 53
6.2 Outlook . . . 54

A Detailed Evaluation Results 61

B Acronyms 75

C List of Symbols 77

D Coordinate System 79

Chapter 1

Introduction

1.1 Motivation

Extracting structure information of an environment is one major goal of the computer vision community.

Lower dimensional visual odometry estimates are already widely spread in consumer devices like optical mouses or mobile scanners. For these use cases the motion estimation is realized in two dimensions, often by supporting light sources.

When we extend this idea to the third dimension, we have to track the camera motion in 6DOF. By relying on a RGB-D camera we are able to regain distance measurements for most of the pixels. With the tracked camera poses and the distance measurements we can reconstruct the environment just by walking through. Additional modifications like markers, special lightning or external camera tracking is not needed.

Using a RGB-D camera, which has multiple sensors encapsulated in one device, simplifies the later usage of the system as well as the sensor data fusion. It also enables us to develop an approach that can be adapted to other RGB-D cameras. With the Microsoft Kinect, that was developed for home entertainment, is a RGB-D camera existing that is already widely spread and achievable for masses.

Currently some companies are investigating a lot in research to bring approaches similar to those presented in this thesis to the consumer market. For instance the Google Tango project[1] aims to bring the PrimeSense Technology [AFMS12] as used in depth cameras like the Microsoft Kinect and Asus Xtion Pro Live into consumer tablet and smartphone devices, which will spread the availability of this technology even more and increases the mobility. The Structure

[1] `tango.google.com`

IO[2] project also aims to bring companion depth cameras to consumer devices that could be attached onto tablets.

The methods presented in this thesis can be useful for several tasks like:

Measuring / Reconstruction Today futuristic sounding scenarios like extending the image sharing process into a third dimension can be reality in a few years. The placing of virtual furniture into your captured flat, will show up if it fits into your room, both optical and in size, just before your bought it. A virtual inspection of a flat you want to buy, a famous building or your next trip are only some use cases that could be imagined in the future.

Augmented- / Virtual Reality Having information about the structure of an environment as well as the position and orientation of a camera enables us to develop Augmented Reality (AR) and Virtual Reality (VR) applications.

The paracosm[3] aims for 3D reconstruction with depth data for bigger scenarios and also captured by different devices. By this it is imaginable that 3D structure information is available more globally and easier accessible. Dynamic augmented informations can be encoded into this data collection for static scenarios. VR will also profit from these techniques, as the 3D map can be used for gaming, tourism or teaching applications in virtual previously scanned environments.

Robotics In unstructured environment, when wheels slip, it is not guaranteed to get reliable movement information from wheel encoders, which also hardens the mapping and localization task. A complete visual approach will be robust against such cases as long as the visual sensor is able to see the environment. Since 2008 the NASA shows how they successfully use a visual odometry approach on two Mars Exploration Rover (MER)s and how it outperforms wheel odometry in unstructured environments (see Figure 1.1).

RGB-D odometry, mapping and Simultaneous Localization and Mapping (SLAM) technologies can also be used on Unmanned Areal Vehicles (UAV) and it was recently shown that it is possible to control quadrocopter based on this input data [KSC13b] and even navigate.

Generating 3D maps will also enable robots to interact better with the environment in many cases such as higher dimensional obstacle avoidance, or a more global grasp planning. A full 3D map of an environment will also enable us to add semantics , i. e. obtained from an object recognition system, to a map and can be used for robot task planning. In general it will enable robots

[2]`http://www.structure.io`

[3]`http://paracosm.io`

to understand the environment better which could improve Human-Robot Interaction (HRI).

In cases where no Global Positioning System (GPS) signals are available, such as tunnels, will visual odometry estimates support the localization.

Inspection Another use case is covered by the Dot Product 3D[4] company which develops mobile devices attached with a depth camera to reconstruct industrial facilities for planning or inspection purposes. In comparison to laser scanner technology they provide low cost mobile scanning technologies giving instant visual feedback about the reconstruction quality. This will lead to a more spread and accessible scanning technology for industrial use cases.

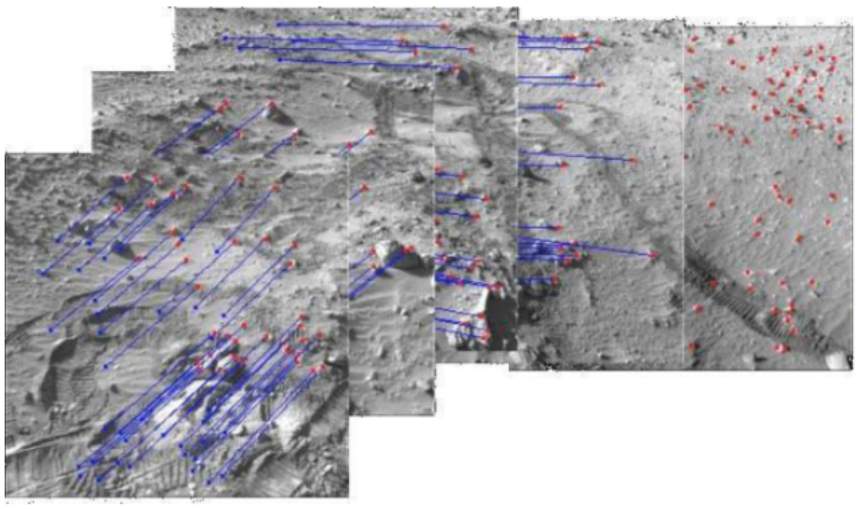


Figure 1.1: This Figure shows the feature tracking component between pairs of images as used by the visual odometry approach by the NASA for motion estimations in unstructured environments on their mars exploration robot Spirit [MCM07]. The dots denote the feature points whereas the lines denote the 2D projection of motion applied between the image pairs.

1.2 Goals

The goal of this thesis is to implement an approach for estimating camera motions in 6 Degrees Of Freedom (DOF) and building 3D maps using only a RGB-D camera as an input device. Thus we have access to color- and depth images. We

[4]`http://www/dotproduct3d.com`

try to achieve this by following an approach that estimates the camera poses using a sparse method but at the same time be able to construct a dense 3D map from registered intensity- and depth images.

1.3 Implementation

The motion estimation and mapping system developed in this thesis was implemented in C++ 11 using the Robot Operation System (ROS) [QCG+09]. The modular nature of ROS enables us to separate the motion estimation from the mapping component and also comes with a tool-chain that supports online interchangeability of the components. Furthermore ROS supports the debugging process by serving tools for visualization as well as data flow inspection.

Image processing was supported by using the Open Computer Vision (OpenCV) [Bra00] library. We made use of all kinds of algorithms from stereo camera calibration, feature extraction and matching over to motion estimation with RANdom SAmple Consensus (RANSAC). The matching process was also supported by the Fast Library for Approximate Nearest Neighbors (FLANN)[ML09]. Additionally we made use of the Fast Odometry From Vison (Fovis) [HBM12] library which implements the approach by Huang et al. [HBH+11] for visual odometry.

For point cloud processing the Point Cloud Library (PCL) [RC11] was used intensively for registration using Iterative Closest Points (ICP), closed form motion estimation using Singular Value Decomposition (SVD) as well as reducing the amount of data using a voxel grid filter to construct colored point cloud maps.

The OctoMap [HWB+13] 3D mapping framework was used for efficient mapping and the comparison against 2D occupancy grid maps generated by a scan matching slam system.

Mathematical operations, particularly matrix and vector operations were realized with the Eigen [GJ+10] Library. For general purpose algorithms, as getting a list of all files in a directory or reading and manipulation configuration files, the Boost Library [SLL01] was used during this thesis.

At some points python scripts were written for testing purposes as well as extraction from encapsulated data in container file formats.

During the creation of this thesis the RGB-D benchmark [SEE+12] was also a great support in the development process with its sequences for testing and debugging as well as for the evaluation.

1.4 Notation

In this thesis, the following conventions are used for denoting mathematical entities:

- Vectors are denoted by lowercase letters like t
- Points are denoted by lowercase letters, annotated with the corresponding space in the superscript and the according reference frame in the subscript like x^c_{rgb} for instance describes a point in camera coordinates with it's origin in the center of the RGB camera.
- Matrices are denoted by bold, uppercase letters like $\mathbf{R}$
- Sets, like point clouds, are denoted by uppercase, italic letters like P
- Images are defined as two dimensional functions, and are denoted by lowercase letters like f. An image element is accessed by its x and y coordinate like $f(y, x)$

An overview of the symbols used in this thesis is given in Appendix A.

1.5 Outline

This thesis is divided into a state of the art chapter 2 which describes related methods and research activities.

Then in the visual odometry chapter 3 we introduce the algorithms to estimate the motion applied to a RGB-D camera. The motion estimate serves as input for the mapping approach (described in Chapter 4), where we introduce algorithms and data structures that are used for building 3D maps. The evaluation methods, using the widely used RGB-D benchmark [SEE$^+$12] as well as the results from our approach, are described in chapter 5.

This thesis closes with a conclusion and suggestions to improve on top off the proposed methods in Chapter 6.

Each chapter starts with a graphical overview like Figure 1.2. This Figure shows the implementation pipeline of each of the both main problems covered in this thesis - the motion estimation and the mapping - in context of the whole system. The steps, data and their relations covered in the chapter are highlighted in front of each chapter. In these graphics rounded boxes denote data while boxes with sharp corners describe actions.

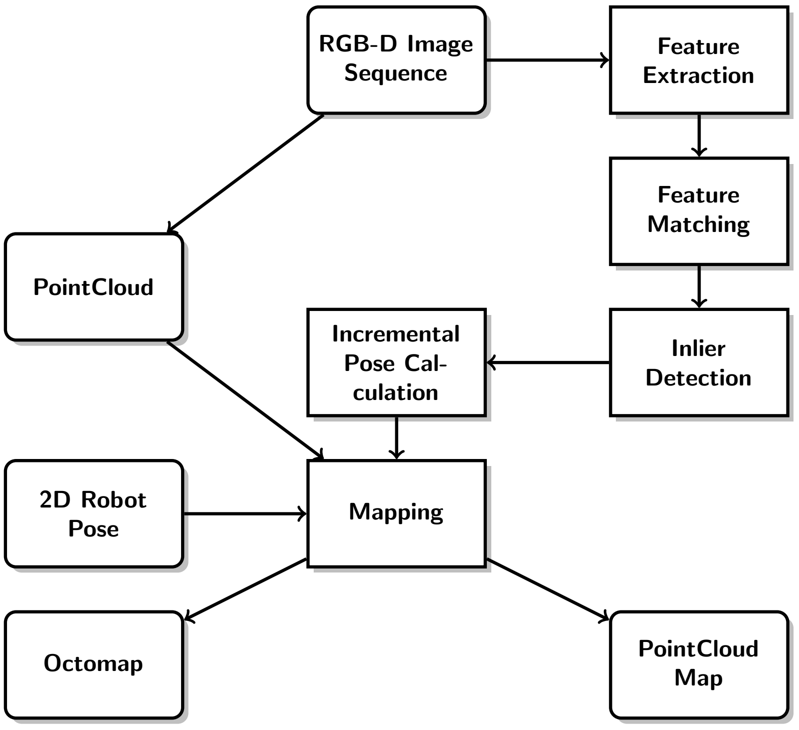


Figure 1.2: Implementation Overview

Chapter 2

State of the Art

There are three basic approaches to estimate the motion of a visual system:

- Feature Based Methods
- Direct Methods
- Depth Based Methods

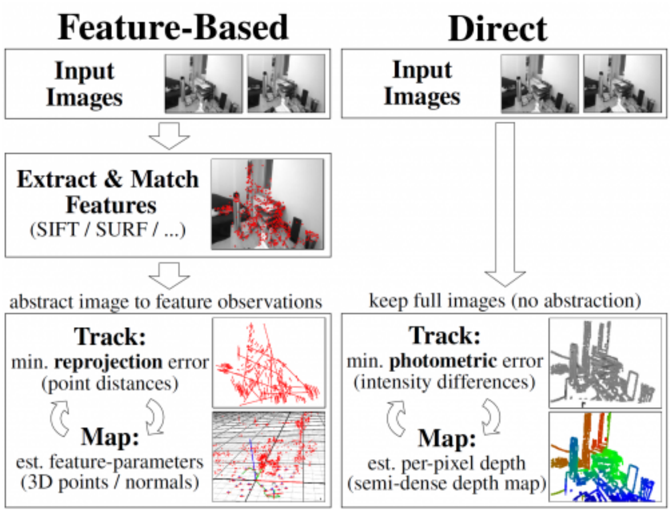
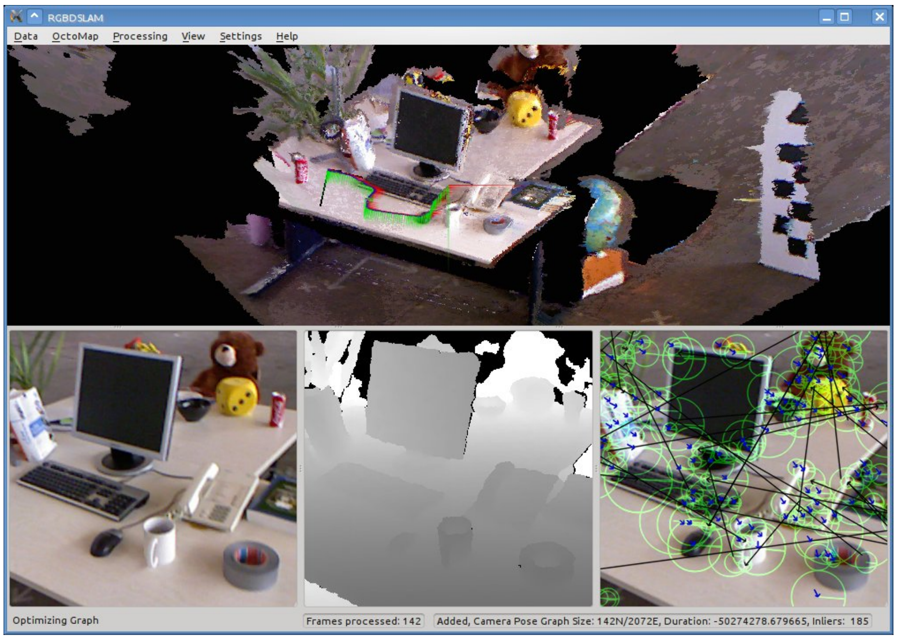


Figure 2.1: Comparison of feature and direct based methods for camera tracking [ESC14]

Feature based (also sparse) methods extract and match interest points between consecutive frames to calculate the camera motion. Direct methods skip the feature extraction and matching by minimization of the photometric error between images. Depth based methods estimate motions given point clouds or directly on depth maps and also can be realized on data obtained by laser scanners. All presented methods have in common that they estimate the camera motion in 6DOF and construct maps in 3D. Figure 2.1 shows the differences between feature and dense methods for motion estimation and mapping.

2.1 Feature Based Methods

We now give an overview of methods used for motion estimation and/or mapping that follow a sparse approach. All these methods rely on a feature extraction and matching step and thus won't be able to give correct motion estimates on for instance blank walls where not enough features could be extracted and matched. Further when relying on features for mapping we only can triangulate 3D points for matched interest points which will lead to very sparse maps that can hardly be interpreted by humans. To overcome this it is possible to interpolate distances in between and map parts of the image to interpolated values. This however will only support the eye candy aspect and won't support accuracy for localization. By additional usage of depth data acquired from RGB-D camera we are able to reconstruct dense maps using sparse motion estimates.



Optimizing Graph

Frames processed: 142

Added, Camera Pose Graph Size: 142N/2072E, Duration: -50274278.679665, Inliers: 185

Figure 2.2: RGB-D SLAM GUI showing the constructed 3D map on top. On the bottom left we can see the input image and in the center the corresponding depth image. On the bottom right we can see the extracted SIFT features and matches [EHS+14].

RGB-D SLAM RGB-D SLAM is a system presented by Endres et al. in 2010 [EEH+11] [EHS+14]. The system for motion estimation developed in this thesis is very similar to the one proposed by Endres et al.. The SLAM-backend extracts sparse image features with SIFT, Speeded Up Robust Features (SURF) or Oriented BRIEF (ORB). Those features are projected into 3D using the depth value at the feature locations. After matching features from consecutive intensity images, the camera pose is calculated incrementally using RANSAC. To speed up the algorithm, they also integrated a Graphical Processing Unit (GPU) version of SIFT. The SLAM front end optimizes the pose graph using *g2o* [KGS+11]. The map is represented either as an octree map using the OctoMap framework [HWB+13] or an incrementally built point cloud. Figure 2.2 shows the RGB-D SLAM in application. In a later publication [EHE+12] the approach was evaluated against the RGB-D benchmark.

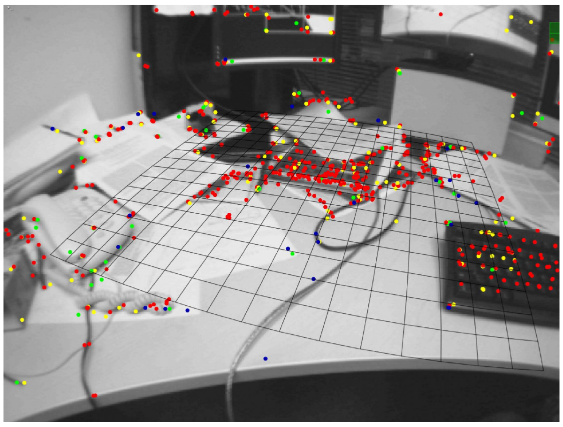


Figure 2.3: Parallel Tracking And Mapping (PTAM) showing an initialized and feature points building the map. The dot colors denote on which coarse to fine level a feature point was extracted.

PTAM PTAM [KM07] was introduced in 2007 for estimating camera pose from a handheld camera in unknown scenes for small AR workspaces. To speed up the tracking and mapping process they decoupled the components each in its own thread. This approach processes relative detailed maps of thousands of features and is able to overlay AR information in a large scale.

Essential Matrix decomposition For monocular cameras a multiple view approach that estimates the Essential matrix which then can be decomposed into a translation and a rotation component [HZ04] is widely used. As with other monocular approaches we won't have any exact distance metrics and thus we will get the transformation only up to scale. There are possibilities to overcome this by fusing the estimation for instance with data of an Inertial Measurement Unit (IMU) [WS11].

2.2 Dense Methods

Dense (also direct) methods estimate the camera motion by direct image alignment that is achieved using a minimization of the photometric error. As Figure 2.1 shows, the motion estimate is based on direct image alignment on the difference minimization, therefore the feature extraction and matching is not needed and one step is skipped. However, as more image information is used, the estimation is more costly and was only recently optimized to run in realtime on a GPU[NLD11], Central Processing Unit (CPU)[KSC13b], or even mobile devices [ESC14]. We now introduce the current state of the art methods for dense motion estimates and mapping.

DTAM In 2010 Dense Tracking And Mapping (DTAM) [NLD11] was published by Newcombe et al. being the first dense tracking and mapping method running online. This was possible due to the usage of optimized algorithms for parallelism that where executed on modern GPU hardware. By minimization of the photometric error between a bundle of frames and a key frame using an exhaustive search algorithm they incrementally constructed an inverse depth map. In comparison to PTAM the DTAM method is more robust against fast camera movements and defocused cameras but will fail on global illumination changes that more likely will converge to a wrong minima. Figure 2.4 shows an incrementally built map and also compares the map details to PTAM.

Real-Time Visual Odometry from Dense RGB-D Images Steinbrucker et al.[SSC11] showed an energy based approach to estimate visual odometry from RGB-D images. Instead of relying on image features this approach minimizes the photometric error. The idea is to compute the rigid body motion which optimally transforms the second image into the first, i.e. the difference image, computed for locations of reliable depth, should be zero [SSC11]. The minimization is executed from a coarse to fine level to be more robust against bigger movements.

DVO Based on the approach proposed by Steinbrucker et al.[SSC11] the Dense Visual Odometry (DVO) [KSC13b] approach extended it to also minimize the depth error and proposed the use of error functions to make it robust against moving objects in the scene[KSC13b]. Kerl et al. make heavy use of Single Instruction Multiple Data (SIMD) to run it on a single CPU core and also on embedded devices which enables it to run on quadrocopters for instance.

Using this visual odometry estimation they later extended the approach to build a SLAM system in [KSC13a] which is accomplished by building a

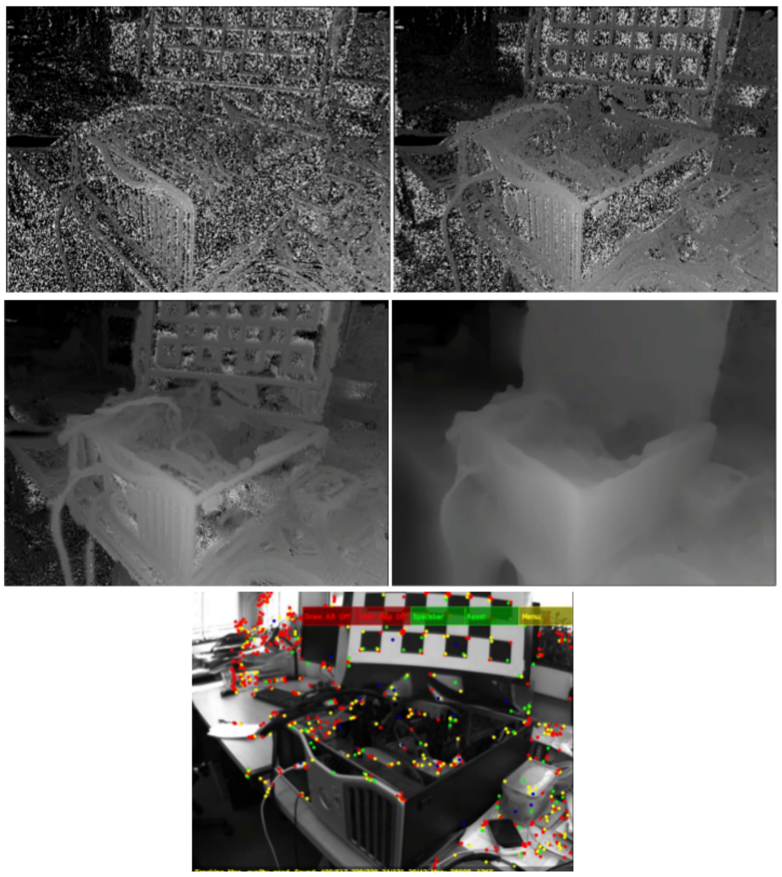


Figure 2.4: Visualization of the incremental constructed inverse depth map over time. [NLD11] On the bottom a map overlay created by PTAM for the same scene containing fewer information.

graph representation and optimization using *g2o* [KGS+11] resulting in global consistent maps.

LSD-SLAM: Large-Scale Direct Monocular SLAM Also based on the approach by Steinbrucker et al. Engel et al. presented a direct monocular SLAM technique able to build large scaled dense maps. The camera tracking is realized using direct image alignment and also generates geometry in form of semi dense depth maps [ESC14] based on the image alignment. For global consistent maps and loop closure detection they build a key frame based pose graph. In comparison to DTAM the technique could be executed on a CPU and even on mobile devices.

2.3 Depth Based Approaches

In this section we introduce methods for motion estimation based on depth maps or point clouds using registration algorithms like ICP [ZHA92].

Kinect Fusion [IKH+11] was introduced in 2011 aiming at rapid, visual impressive reconstruction of scenes. They first convert the depth map of a RGB-D camera into 3D points with normals in camera coordinate space (a). The next step is the camera pose calculation in 6DOF by running ICP on the points between the current and the last frame (b). The global transformation is incrementally calculated based on this transformation. The mapping consists of incrementally fusing 3D meshes instead of the point cloud data using a volumetric surface representation based on Signed Distance Function (SDF) (c). The global transformation is used for the mesh transformation. For rendering they cast rays out of the global pose by a ray casting algorithm. The calculations are done on a GPU in order to achieve higher frame rates (d)[IKH+11]. The complete pipeline as described is shown in Figure 2.5.

Direct Camera Pose Tracking and Mapping With Signed Distance Functions Just like the Kinect Fusion method, this approach estimates the camera motion using SDFs but instead of using ICP on an synthesized depth image created by SDFs Bylow et al.[BSK+13] proposed an approach to run ICP directly on a SDF.

RGB-D Mapping as proposed by Henry et al.[HKH+10] creates colorized 3D maps, only supported by pose estimations from the RGB-D camera. They extract SIFT features and perform alignment using RANSAC. Then they propose to use RGB-D-ICP [HKH+10] which runs the ICP algorithm on the

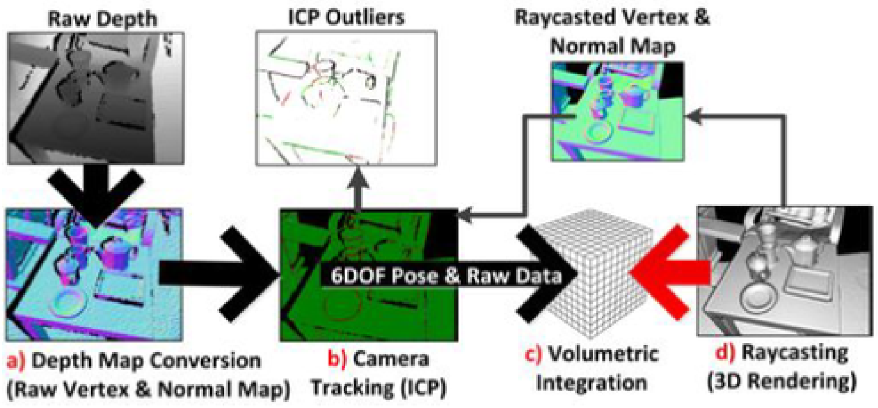
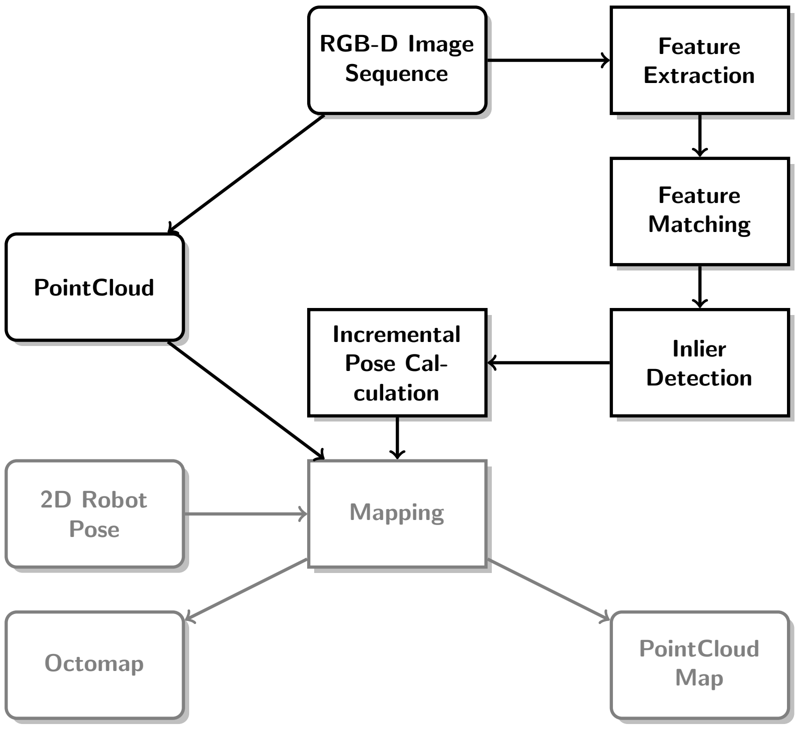

Raw Depth
ICP Outliers
Raycasted Vertex & Normal Map
6DOF Pose & Raw Data
a) Depth Map Conversion (Raw Vertex & Normal Map)
b) Camera Tracking (ICP)
c) Volumetric Integration
d) Raycasting (3D Rendering)

Figure 2.5: Kinect Fusion pipeline. See description in Section 2.3

reduced set of image features. They also optimize the camera pose and detect loop closures using a pose graph. For a more efficient map representation they suggested to use a surfel based structure which combines points at similar locations and colors to reduce the amount of data.

Chapter 3

Visual Odometry



In this chapter we describe the process of estimating the camera transformation. The problem is reduced to a transformation estimation between two consecutive frames based on 3D-3D correspondences. These estimates are then used for an incremental motion estimation.

We start with a brief introduction followed by a description of each step of the motion estimation process, each separated in subsections.

The following steps are executed:

- Image preprocessing

- Feature extraction
- Feature matching
- Inlier detection
- Motion estimation

Odometry describes the position estimate of a robot usually with observations of the wheel spins [Pel11]. Visual odometry however does not rely on wheel odometry and describes the motion of a camera or a robot platform only with visual information obtained from an optical system like a camera. In this thesis the motion estimation is calculated using the color and depth information of a RGB-D camera.

3.1 Image Preprocessing

In the image preprocessing step we obtain the RGB and depth image. The RGB image is converted to gray scale and smoothed with a Gaussian kernel of $\sigma = 0.85$ [HBH+11]. Additionally Gaussian pyramid is constructed for feature extraction on different scales in order to be more robust against motion blur by fast movements. In the preprocessing step we also register and synchronize the intensity information to the depth information. This process is described in Section 3.6.

3.2 Feature Extraction

In this section the methods to extract features from an intensity image are described. The extracted features from image space are associated with the corresponding depth values. If the depth value at the feature location could not be estimated, for instance they are not in the visible depth range or the infrared light was absorbed, they are simply discarded. This step is necessary because we rely on 3D-3D correspondences for the motion estimation step (described in Section 3.5) We project the successfully matched 2D image features into 3D by associating the depth information to the image coordinate of the feature points as described in Section 3.7.

For feature extraction we rely on the Features from accelerated segment test (FAST) feature detector, as described in Section 3.2.1, which is applied on each level of the Gaussian pyramid [HBH+11].

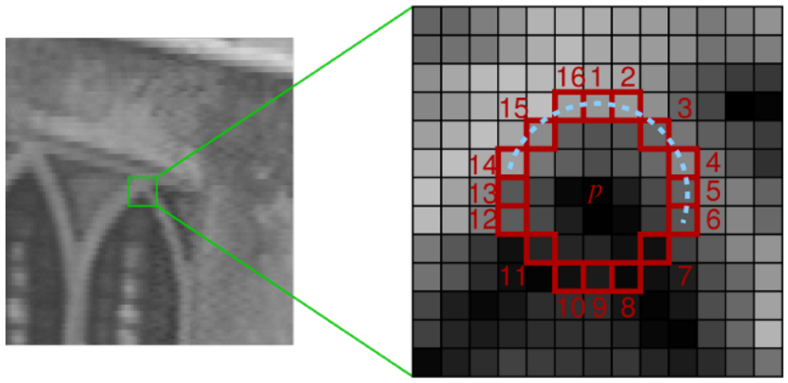


Figure 3.1: FAST corner detector - This Figure shows the Bresenham circle of radius 3 around a center pixel. When a certain amount of contiguous pixels on this circles is brighter and darker than a given a threshold the center pixel is a feature [RD06].

3.2.1 FAST

FAST [RD06] and its successor FASTER [RPD10] are, as the name suggests, optimized for significant faster detection then other methods. For comparison the Harris corner detector is 20 to 30 times slower [Eng11]. They extract features using a simple threshold convention for the brightness of pixels on a circle with a given radius around a center pixel. This simple definition of a feature yields in an efficient implementation with accelerated segment tests. Further improvement is realized with a machine learning approach which generates code, in form of thousands lines of if-statements, for a pre trained decision tree. Beside the efficiency of FAST, due to this simple feature definition, it will also result in a high amount of detected features. For sparse 3D reconstruction using multiple views this would result in a dense map because more features will lead to more correspondences for which a 3D point can be triangulated. On the other hand we have to match more features and dismiss more false positive correspondences resulting in false motion estimates. As we use the depth map of the RGB-D camera we could neglect the reconstruction aspect but have to focus on the correspondence aspect as we show in Section 3.3.

A descriptor is then calculated for each extracted feature, which supports the matching process 3.3

3.2.2 SIFT Descriptor

For the SIFT [Low99] descriptor creation a 16×16 neighborhood around a key point is taken and divided into 16 sub blocks each of size 4×4 as shown in Figure 3.2. For each sub block a 8 bin histogram of orientations, based on the gradient, is calculated. This sums up in a key point descriptor of size 128 [Bra00]. We now have local key point descriptors for each feature supporting the correspondence association between consecutive frames.

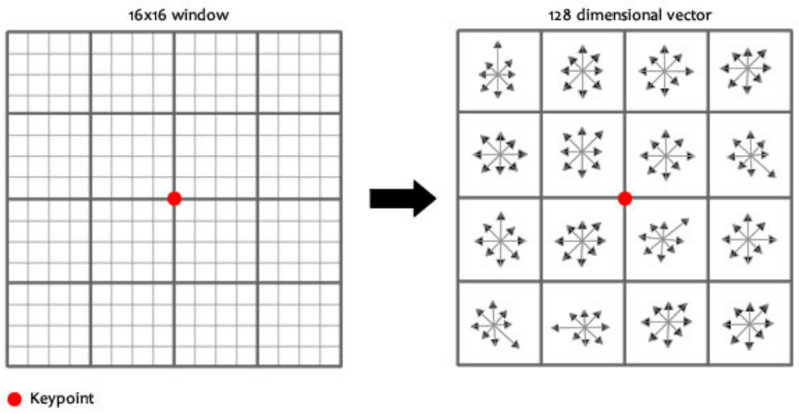


Figure 3.2: SIFT Descriptor - On the left a key point with its corresponding search window of size 16×16 and the sub blocks of size 4×4 are visualized. On the right the visualization of the orientation histograms for each sub block is shown[Yua13].

3.3 Feature Matching

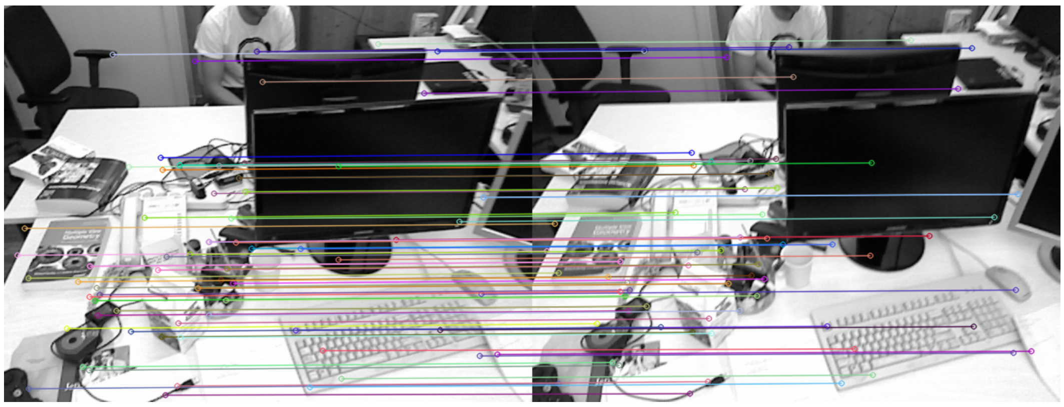


Figure 3.3: Visualization of matched features between two consecutive intensity frames in this case using a SIFT detector and descriptor with an adaptive thresholding on the descriptor distance

Now we are able to extract feature points from images, we want to know which feature feature points are corresponding between two consecutive frames.

Given the descriptors we calculated in the Feature Extraction section 3.2, we can match features by computing the euclidean distance d between their descriptors p and q:

$$d = \left(\sum_{i=1}^{128} (p_i - q_i)^2 \right)^{1/2} \tag{3.1}$$

Features with a low descriptor distance will be marked as matches. Instead of canceling the matching process when the descriptor distance is below a certain threshold we go on and search for features that may have even better descriptor distance. As the matching is a very fundamental step for the motion estimation results we try to find the best feature matches in this step. This however is due to the amount of features and the 128 dimensional descriptor a costly function and could be optimized. We additionally discard feature matches that are above a certain descriptor distance as they most likely will be false matches. Using the FLANN library we found a good trade off between efficiency and accuracy.

The most computationally expensive part of many computer vision algorithms consists of searching for the closest matches to high-dimensional spaces. [ML09] Given a dataset and a degree of accuracy the FLANN library is able to determine the best algorithm and parameter values to find the approximate nearest neighbour. FLANN represents the descriptor space as randomized k-d trees or k-means trees, depending on the input, which have be shown to perform best on approximate neighbour search. In Figure 3.3 the result of the matching process is shown.

This approach will in general lead to very good associations by itself, but it is still possible to have false key point matches between frames that will lead to wrong motion estimations. To further optimize this we have to detect false matches.

3.4 Inlier Detection

As we still get some false associations due to false matches with low descriptor distances we have to add an additional step for inlier detection. A greedy algorithm is executed to approximate the maximal clique on a graph of consistent feature matches [HIG02]. This leads to a runtime of $O(n^2)$ depending on the number of feature matches. We could neglect this due to the efficiency of the consistence checks. A comparison between this method and RANSAC has been shown by Huang et al. in [HBH$^+$11]. The motion estimation process is canceled for correspondence sets with fewer then ten inlier to get more robust motion estimates.

3.5 Motion Estimation

We now introduce methods for motion estimation given two 3D point sets with known correspondences. We implemented methods by:

- SVD
- Least-Squares
- RANSAC

Beside these methods there are other approaches like ICP [ZHA92] or Maximum Likelihood Estimation (MLE) [MSS09].

3.5.1 SVD

Given a set of 3D-3D correspondences we are able to compute the transformation of two consecutive frames in closed form using the SVD. A complete deviation can be found in [N$^+$06] P. 33-34.

First we calculate the center of mass of the source and destination depth values at the extracted features points.

$$\bar{p} = \frac{1}{n}\sum_i p_i \tag{3.2}$$

and

$$\bar{q} = \frac{1}{n}\sum_i q_i \tag{3.3}$$

Then we have to calculate the covariance matrix **H**.

$$\mathbf{H} = \sum_{i=0}^{N} (p_i - \bar{p})(q_i - \bar{q})^T \tag{3.4}$$

[Nüc09]

The SVD says that each $n \times m$ Matrix could be decomposed into 3 matrices:

$$\mathbf{H} = U\Sigma V^T \tag{3.5}$$

If $rank(H) = 3$ holds, the optimal solution is unique and the rotation is given by:

$$\mathbf{R} = VU^T \tag{3.6}$$

To calculate the translation between the two sets we can simply subtract the centroid of the source set and subtract it by the rotated centroid of the destination set:

$$t = \bar{p} - \mathbf{R}\bar{q} \tag{3.7}$$

The complete transformation between source and target is then given by:

$$\mathbf{T} = \mathbf{R}p_i + t \tag{3.8}$$

3.5.2 Least-Squares

A Least-Squares formulation for the transformation of two 3D point sets was given by Horn [AHB87] and Umeyama [Ume91].

Given two point sets p and q and the transformation $\mathbf{T}$ by $\mathbf{R}$ and t the residual error can be written as [Yan14]:

$$e_i = q - \mathbf{R}p - t \tag{3.9}$$

We want to minimize the sum of squared errors:

$$\sum_{i=1}^{n} ||e_i||^2 \tag{3.10}$$

First we have to calculate the centroid (as in equations 3.2, 3.3) and then translate the point sets into their centroid with:

$$p'_i = (p_i - \bar{p}) \tag{3.11}$$

and

$$q'_i = (q_i - \bar{q}) \tag{3.12}$$

We now can write the error term as:

$$e_i = q' - \mathbf{R}p' - t \tag{3.13}$$

where

$$t' = t - \bar{q} + \mathbf{R}\bar{p} \tag{3.14}$$

This yields a Least-Squares formation by:

$$\sum_{i=1}^{n} ||e_i||^2 = \sum_{i=1}^{n} ||q'_i - \mathbf{R}p'_i - t||^2 \tag{3.15}$$

3.5.3 RANSAC

The RANSAC algorithm was first proposed by Fischler and Bolles in 1981 [FB81] to fit a model to noisy sensor data using minimal data to represent a model. After outlier detection a least-squares estimation is executed on the inlier set to further improve the estimate.

In our case we want to estimate the best transformation $\mathbf{T}$ given two sets of 3D correspondences $\mathbf{P}$ and $\mathbf{Q}$. This leads us to the following equation:

$$\mathbf{Q} = \mathbf{TP} \tag{3.16}$$

This can be read as we want to find a transformation $\mathbf{T}$ such that $\mathbf{Q}$ fits best into $\mathbf{TP}$. In this case the model is given by $\mathbf{T}$. The best fit is defined as the lowest distance between all elements of $\mathbf{P}$ and $\mathbf{Q}$ [Fre13].

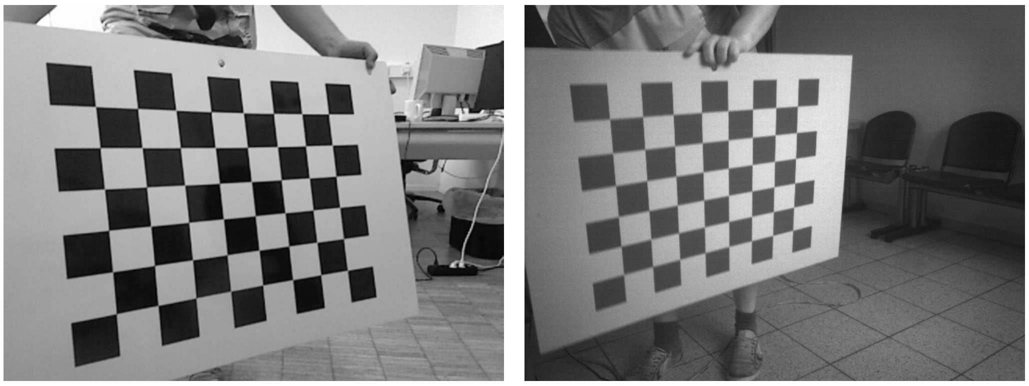

(a) Mono Calibration Image **(b)** Infrared Calibration Image

Figure 3.4: Calibration Images - The images above are images from the calibration process. While the mono calibration image (a) could be recorded without special conditions the infrared image (b) on the right was recorded in a well-lit environment with a blocked Infrared (IR) projector.

3.6 Camera Calibration

The goal of the camera calibration process is to find the intrinsic parameters that describe how the light information maps to a camera sensor and the extrinsic parameters that describe the geometric transformation between a world reference frame and the camera reference frame. Extrinsic camera parameters from different cameras according to the same world reference frame also deliver the relation between cameras.

A good calibration is prerequisite for accurate motion estimation algorithms. While the Microsoft Kinect delivers a standard intrinsic calibration by itself, it is not suitable for accurate computer vision tasks as it does not model lens distortion.

To calibrate a RGB-D camera, the RGB- and the IR camera need to be calibrated. Both cameras were calibrated using a chessboard pattern with a camera calibration method proposed by Zhang [Zha00]. An additional stereo calibration process between the RGB- and IR camera gives us the extrinsic parameters and therefore the geometric relations between them. We show in Figure 5.5 example calibration images acquired by the RGB- and IR camera.

While the IR projector disturbs the IR camera during the calibration process Smisek et al. [SJP13] suggest to light the calibration pattern by a halogen lamp to block the IR projector. Another possibility is to cover the IR projection, while this leads to a darker IR image and corner detection will be harder.

The image taken by a camera projects 3D information in a world reference frame with a transformation $\mathbf{T}$ composed of rotation $\mathbf{R}$ and translation t also $[\mathbf{R}|t]$

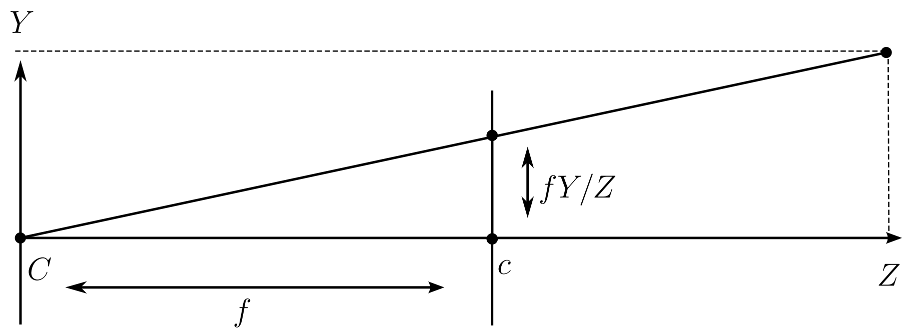


Figure 3.5: Pinhole camera model - f denotes the focal length. Y, Z are object coordinates in the camera frame, c denotes the principal point (center of projection) on the image plane

into pixel coordinates. The image recorded by a camera projects points given in the camera frame onto image coordinates coordinates by the camera calibration matrix $\mathbf{C}$

$$x^c = [\mathbf{R}|t]x^w \tag{3.17}$$

Using the focal length and the principal point of a camera we can build the camera calibration matrix $\mathbf{C}$ (see Equation 3.19), which projects a point given in homogeneous camera coordinates x^c to homogeneous image coordinates x^i:

$$x^i = [\mathbf{C}|0]x^c \tag{3.18}$$

Due to the 3D-2D projection with $\mathbf{C}$ we loose scale information and thus exact distance information which we can regain using a depth projection as described in Section 3.7.

We now explain the intrinsic and extrinsic parameters more detailed.

Intrinsic parameters

- Focal length f
- Principal point c_x, c_y on image plane
- radial distortion coefficients κ_1, κ_2

The camera matrix is composed by:

$$\mathbf{C} = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

With the radial distortion coefficients we are able to calculate a rectified (or undistorted) version of the image using

$$\begin{pmatrix} x^{i'} \\ y^{i'} \end{pmatrix} = (1 + \kappa_1 r^2 + \kappa_2 r^4) \begin{pmatrix} x^i \\ y^i \end{pmatrix} \tag{3.20}$$

.

where r is defined as:

$$r = \sqrt{x^{i^2} + y^{i^2}} \tag{3.21}$$

The success of the rectification can be visually inspected by observing the edges of the resulting image: all edges should be straight. If this is not the case there are most likely poses of the calibration pattern missing in the calibration images.

Note that we will now assume that the image coordinates are already undistorted. As the undistortion process removes the influence of the lens onto the projected image coordinates we are now able to use the pinhole camera model, which simplifies calculations.

Extrinsic parameters

The extrinsic camera parameters

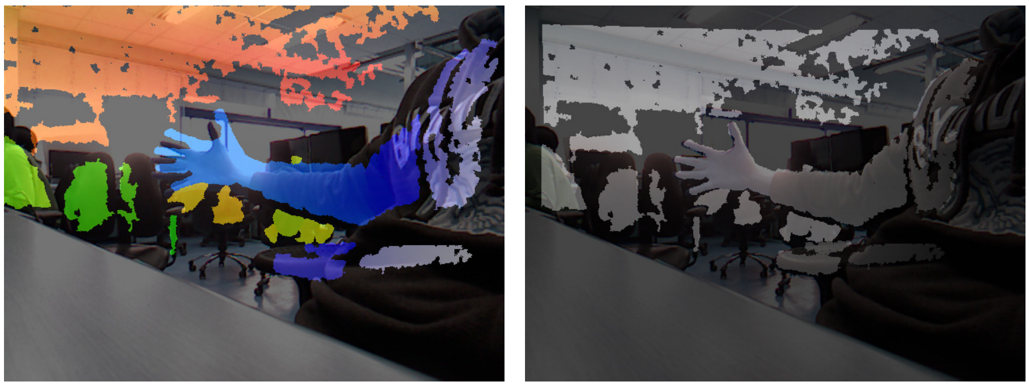
- Rotation $\mathbf{R}$
- Translation t

describe the geometric relation to a reference, in our case the center of the chessboard pattern, in the world frame for each camera. Given the extrinsic parameters we can calculate the relative transformation between both cameras. For simplification we name the camera relations now rotation $\mathbf{R}$ and translation t as we don't need the world transformation any longer.

Those parameters are needed for the transformation of the IR camera coordinates into the RGB camera frame as described in the depth registration section 3.7.

3.7 Depth Registration

Image registration is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints and/or by different sensors. It geometrically aligns two images - the reference and sensed images [Bro92]. Figure 3.6 shows the difference between registered and unregistered depth data. This



(a) Unregistered depth **(b)** Registered depth

Figure 3.6: Depth Registration - The image on the left shows an image overlay with unregistered depth information. Depth values are not corresponding to the image information and thus will lead to false depth associations. The right image shows an registered version where depth information match the corresponding image information. [col11]

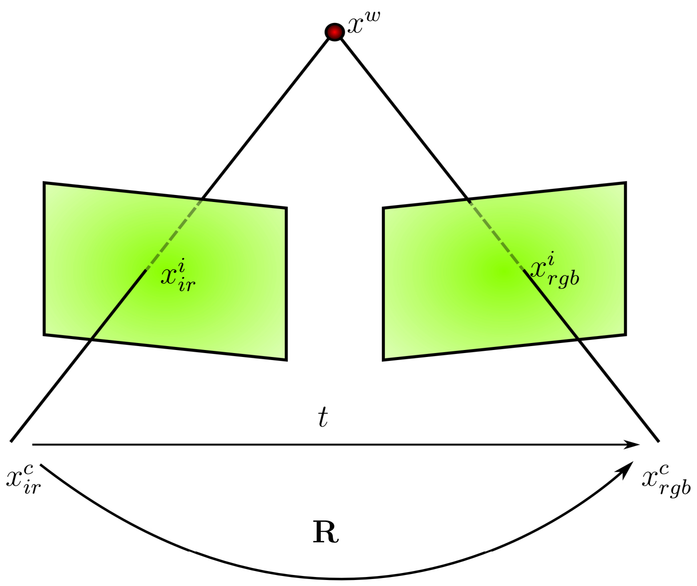


Figure 3.7: Depth Color Registration

section describes the registration process for two different sensors (RGB- and IR camera). Their viewpoints are nearly parallel but horizontally translated.

With the camera matrices $\mathbf{C}_{ir}$, $\mathbf{C}_{rgb}$, the orientation $\mathbf{R}$ and the translation t between the cameras we are able to register the data in a way that the depth values which have their origin in the infrared camera coordinate system can be projected into the RGB camera coordinate system. This enables us to lookup corresponding depth values given a pixel coordinate in the RGB camera frame.

This process is essential for the motion estimation process described in Section 3.5, because otherwise we would not be able to determine correct motion estimates between two consecutive image frames if the depth values are already wrong. Figure 3.7 visualizes the problem of the image registration process.

$$x_{ir}^{c} = {\mathbf{C}_{ir}}^{-1} x_{ir}^{i} \tag{3.22}$$

x_{ir}^{c} can be transformed to the RGB camera frame by the relative rotation $\mathbf{R}$ and t.

$$x_{rgb}^{c} = \mathbf{R} x_{ir}^{c} + t \tag{3.23}$$

Then, we project x_{rgb}^{c} onto the RGB camera image and we obtain a 2D point x_{rgb}^{i}.

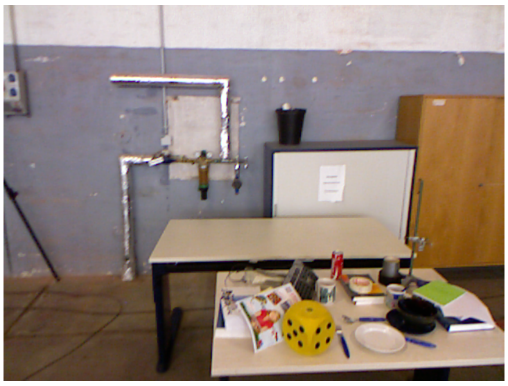
$$x_{rgb}^{i} = \mathbf{C}_{rgb} x_{rgb}^{c} \tag{3.24}$$

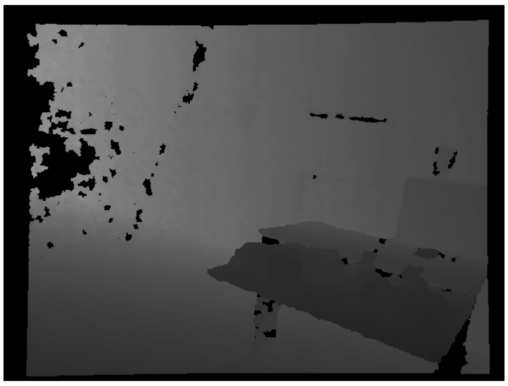
Finally, the depth value corresponding to the location x_{rgb}^{i} in RGB image is x_{rgb}^{c}'s corresponding z value[col11].

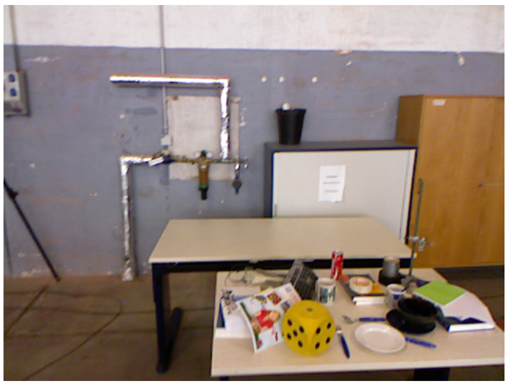
3.8 Time Synchronization

A problem similar to the registration problem could appear if the depth and color information are not synchronized. A RGB-D camera needs more time to produce depth images than it needs to produce the color images. On a Microsoft Kinect the depth information are received by the driver around 20 ms later then the color information [SEE$^+$12].

This will lead to the effect that the color information are transfered earlier then the depth information and could lead to false associations between extracted image features and depth values. This effect could be neglected for static cameras or very slow movements as we get RGB and depth information with 30 Hz, but will affect the motion estimation with fast camera movements dramatically as we get old depth associations for the current frame. The faster the camera is moved, the higher will be the influence of unsynchronized color/depth images. Figure 3.8 illustrate the synchronization problem.


(a) Color image


(b) Unsynchronized depth image


(c) Color image

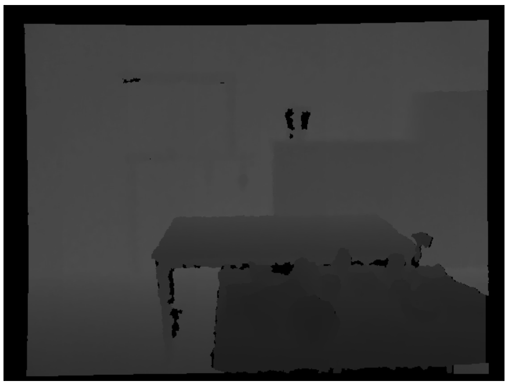

(d) Synchronized depth image

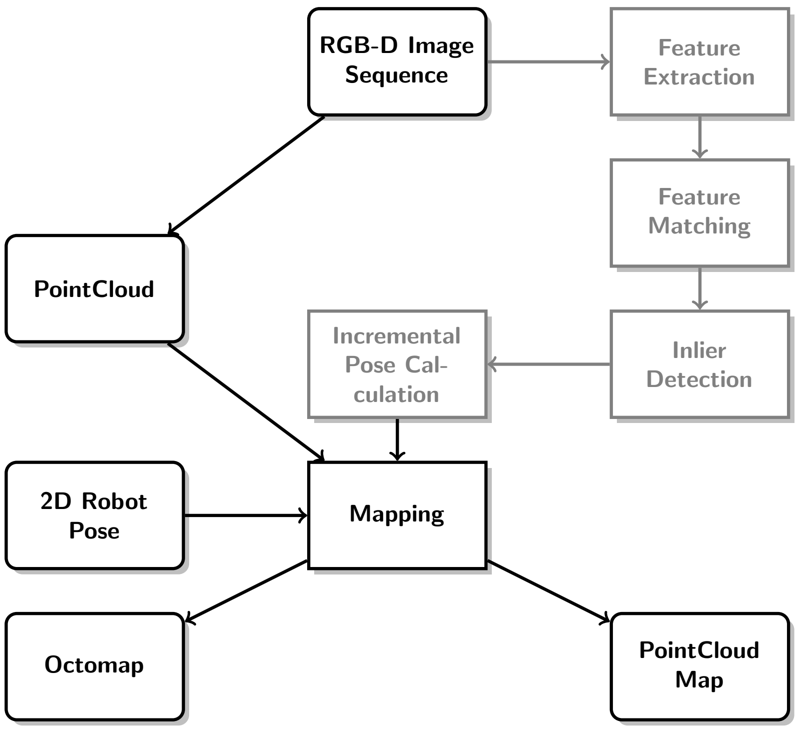
Figure 3.8: Comparison between synchronized and unsynchronized color- and depth images. These figures show for illustration purposes the synchronization problem. While the upper images show a color image (a) and its delivered depth image (b) by the RGB-D camera which belongs to a color image a few frames behind. The color image (c) and the time synchronized depth image (d) match together. Now imagine if we would associate depth values at interest points in the upper unsynchronized case with a consecutive frame, we would get useless motion estimates.

The ASUS Xtion camera provides the possibility to associate the depth and color frames with a timestamp given by a hardware clock. This timestamp is stored when the infrared and RGB shutter is opened. We assume that the frames with the closest timestamps belong together. When using a Microsoft Kinect, which does not store a hardware timestamp, we have to associate the depth values that are incoming approximately 20 ms later to the current color frame.

An other possibility that leads to lower frame rates is to use an exact association (up to nanosecond precision), which means that only frames with an exact matching timestamp are considered. As this is not the case in general the consequence will highly be a reduced frame rate.

Chapter 4

Mapping



In this chapter we describe how to incrementally build maps based on depth data and the global transformation from the previous chapter. The mapping component is decoupled from the motion estimation system. Thus we are able to use other motion estimates as input for the mapping. For instance we can use a pose from a domestic robot or integrate motion estimates from a motion capturing system without relying on the motion estimate from the previous chapter.

Further improvement i.e. by constructing a pose graph for global pose optimization to reduce drift. Sensor fusion using a Kalman filter can be integrated modularly and is suggested.

4.1 OctoMap

For mapping we use the OctoMap [HWB+13] framework. OctoMap focusses on an efficient representation for scalable maps. The underlying data structure, is as the name suggests an octree. The idea of occupancy grid maps using a probabilistic representation for the occupation of a cell is extended to the third dimension. This results in an efficient mapping framework that can cope with sensor noise. Using an 3D OctoMap we can also project a 2D occupancy gridmap.

4.1.1 Occupancy Gridmaps

Maps can be represented using an occupancy grid data structure. An occupancy grid is a grid which holds for each cell the probability that this cell is occupied. Cells that are very likely occupied hold a maximum value whereas the cells that are most likely not occupied get a minimum value. This raster based map representation was first introduced by Moravec and Elves in 1989 [Elf89]. This representation is often used for 2D gridmaps in robotic systems because it gives a good foundation for path planning, obstacle avoidance and map supported localisation. [Pel11]

In practise the grid is discretized. Each cell represents a specific size like for example $5cm^2$. In 2D this amount of data can be represented uncompressed even with bigger maps. When we try to extend this representation to the third dimension the uncompressed size of a map will grow exponential. Thus the data needs to be compressed. Equations 4.1 and 4.2 show this problem by example.

$$\left(\frac{40}{0.05}\right)^2 = 640.000\, cells = 4.88\, mb \tag{4.1}$$

Memory consumption for a $40\, m^2$ occupancy grid map in 2D, where each element is $0.05\, m^2$

$$\left(\frac{40}{0.05}\right)^3 = 512.000.000\, cells = 3.8\, gb \tag{4.2}$$

Memory consumption for a $40\, m^3$ occupancy grid map in 3D, where each element is $0.05\, m^3$ [Stu13]

4.1.2 Octrees

An octree is a hierarchical data structure for spatial subdivision in 3D [Mea82]. Each node is a subdivision in space of it's parent node. The benefits for mapping are that the volumes are only allocated when needed and it scales to multiple resolutions. Thus more detailed parts of the map get more nodes with more structure information. Figure 4.1 shows the data structure. For mapping each node stores the occupancy probability of a voxel.

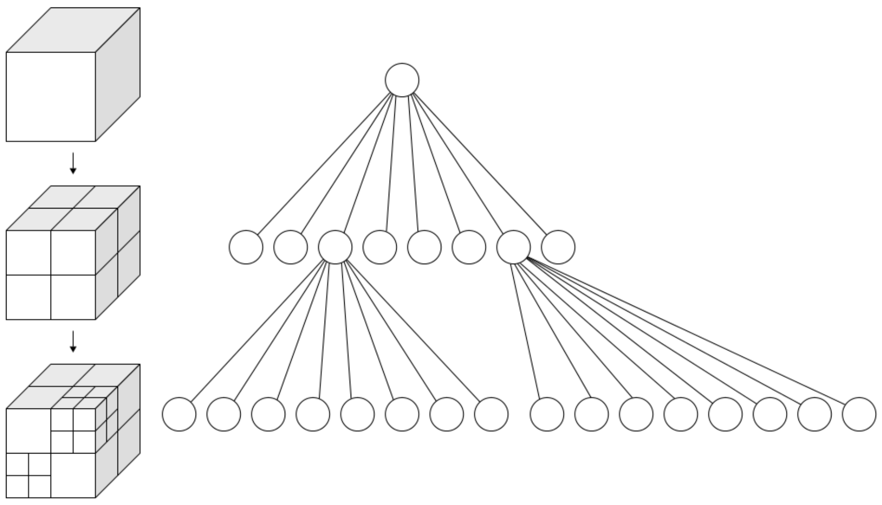


Figure 4.1: Octree data structure - This figure shows a visualization of the octree data structure. At the first level is no subdivision. The second level shows a subdivision for all eight child nodes and on the third level only the third and seventh node are subdivided. Each cube at the left shows an visual interpretation of the corresponding node level on the right. [oct14]

4.1.3 Construction

For node construction we need an inverse sensor model. Based on this model we can calculate the occupation probability by intersection tests along a beam. Initially all nodes are assigned with an occupation probability of 0.5. When a beam does not intersect with a volume it is encoded as free space, else the occupation probability is calculated using the inverse sensor model [HWB+13]. Figure 4.2 shows the input data as well as their occupied and free space octree representations. Figure 4.3 shows an OctoMap of a 360 degree handheld scan in an office environment.

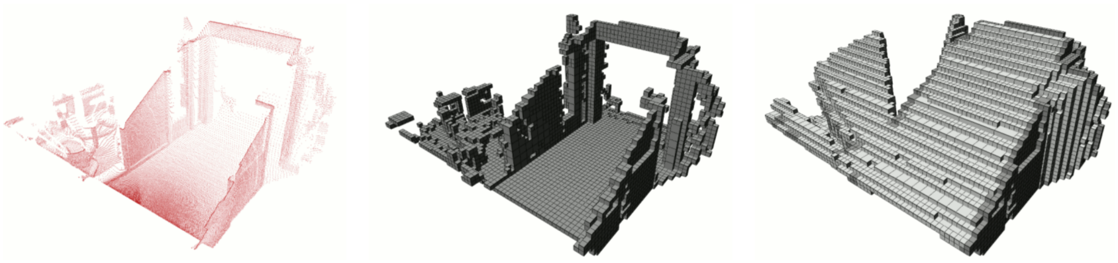


Figure 4.2: This figure shows an example of the incoming point cloud (left) and the constructed octree representations of the occupied cells (center) and the free space (right). [HWB+13]

4.2 Point Cloud Map

A depth camera with a resolution of 640×480 pixels and a frame rate of 30 Hz is able to produce 30 colored point clouds with 307200 points each. This means we potentially get 9216000 points per second. Thus we have to reduce the amount of points. We only process each $n - th$ frame and the second and subsample the point cloud. The subsampling is achieved by removing points with distances below a certain threshold to an other point. We transform the subsampled point cloud using the motion estimate or an other pose source. We refine the transformation then using ICP on the current point cloud and the incrementally global point cloud.

This gives us colored 3D maps by transforming subsampled point cloud. For navigation or path planning it is suggested to convert the resulting point cloud to an OctoMap.

4.2.1 Iterative Closest Points

When we have given two point cloud sets and a transformation estimate between the two sets we can register both sets using ICP. The approach is similar to the dense approach for motion estimates. But instead of minimizing intensity values the ICP algorithm minimizes transformations. The input of the algorithm is an estimated transformation, i.e. from visual odometry, and two point cloud sets. In our case $\mathbf{Q}$ is given by the current point cloud acquired from the depth image and $\mathbf{P}$ will be the global point cloud. Specific points are then given by p_i and q. The ICP algorithm will then estimate the transformation $\mathbf{T}$ by refining the rotation $\mathbf{R}$ and translation t until the overall distance between the points of $\mathbf{P}$ and $\mathbf{Q}$ is below a certain threshold or a maximum number of iterations is exceeded. Correspondence association is handled by the ICP itself i.e. by a K-d tree [GY03] and stored in a correspondece set C. The transformation between the

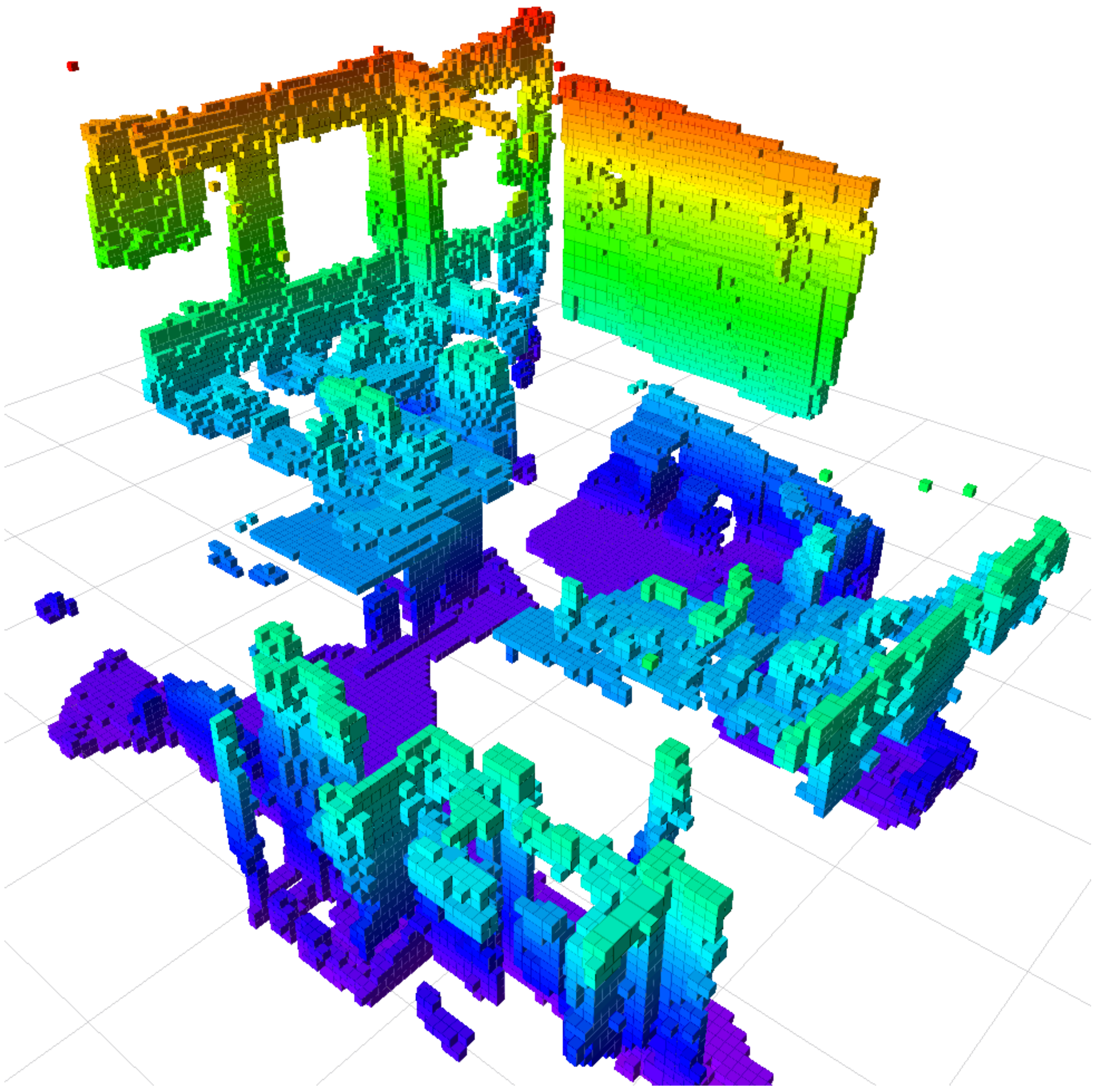


Figure 4.3: Map generated using the fr1/360 sequence from RGB-D benchmark with a resolution of 0.05 m and unfiltered ground. The color encodes the height of each voxel.

correspondences is calculated by the SVD as described in Section 3.5.1. Afterwards the current point cloud is transformed and the errors (as given by Equation 4.3) are calculated between the correspondences of the point clouds [Nüc09]. It is not guaranteed that the ICP converges to the right solution.

$$E(\mathbf{R}, t) = \sum_{i,j \in C} |||q_i - \mathbf{R}p_j + t|||^2 \tag{4.3}$$

Chapter 5

Evaluation

This section presents the RGB-D Benchmark with a brief introduction of the datasets and error metrics used to evaluate the results of this thesis.

5.1 RGB-D Benchmark

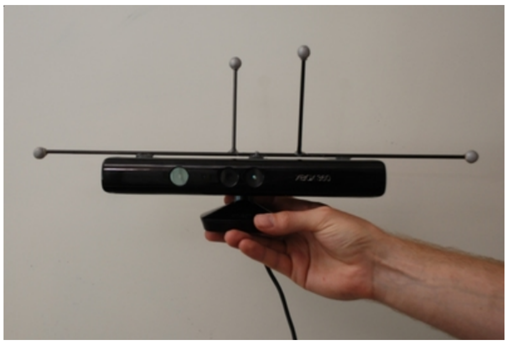


Figure 5.1: This figure shows a handheld Kinect RGB-D camera with attached motion capture markers used during the recording of the dataset. The markers are tracked my a motion capture setup for ground truth [SEE+12].

The RGB-D Benchmark created by Sturm et al. was used to evaluate the implementation. This benchmark contains datasets for benchmarking RGB-D methods for motion estimation and visual SLAM systems.

The dataset contains 39 sequences recorded using three different handheld Microsoft Kinect RGB-D camera and some sequences using manual controlled Pioneer 3 robot platform in different indoor environments.

Beside the image sequences, the dataset also contains time synchronized ground truth data. The ground truth data was recorded using eight 100 Hz motion capture cameras capturing reflective markers that were mounted on the camera as you can see in Figure 5.1

Next to the sequences itself and the ground truth data, the dataset also includes calibration files and image sequences for all three Kinect cameras that were used during the creation of the sequences. The Pioneer dataset also contains odometry data from the robot platform as well as 2D distance measurements from a mounted laser scanner.

5.1.1 Datasets

The RGB-D benchmark contains several datasets for motion estimation and/or reconstruction. In our evaluation we focus on the use of the following datasets:

Testing and Debugging This dataset contains sequences meant for testing purposes and to support the development process. All sequences contain camera movements along the principal axes of the camera in an typical office desk environment. The xyz-Datasets contain sequences translating the camera while the rpy contains sequences with a rotating camera.

Handheld SLAM This dataset contains a lot of different sequences recorded by a person carring a camera. There are scenes with a log trajectory at different speed but all recorded with natural movements. Some scenes are recorded with a down tilted camera to ensure a planar surface. Another dataset is recorded while performing a 360 degree rotation. Most of the scenes are also recorded in an typical office environment.

Robot SLAM This dataset contains sequences recorded on a moving robot platform that was controlled with a joystick through a hall, with bigger distances to objects, and less noisy structure. In addition the dataset contains laser scanner data by an SICK 2D scanner as well as odometry data delivered by the Pioneer robotics platform. Special sequences simulate sensor failures or sensor occlusion by covering the sensor while moving and then uncovering it again which lead to the famous kidnapped robot problem. All sequences are recorded at different speeds but all containing a long trajectory.

In this thesis we focus on evaluation for the Testing and Debugging, Handheld SLAM and the Robot SLAM datasets as they are intended for usage in motion estimation, mapping and SLAM systems. As we did not propose any error function for cases like moving persons we don't evaluate against those datasets.

5.1.2 Evaluation Metrics

We evaluate the local accuracy of the camera trajectory as well as the global consistency. We don't evaluate the map quality as it is hard to construct ground truth maps and not strictly necessary because the map quality heavily depends on the pose estimates.

For the evaluation process the computed poses resulting in a trajectory estimation $\mathbf{P}_1 ... \mathbf{P}_n$ are compared against the ground truth trajectory $\mathbf{Q}_1 ... \mathbf{Q}_n$. For simplification the evaluation metrics presented are time synchronized, equally sampled and have both the same length n. In practice this association step has to be done in advance, as we can't assume the ground truth to be time synchronized and sampled with the same rate [SEE+12].

Relative Pose Error (RPE)

The Relative Pose Error (RPE) measures the local accuracy, in our case in meters, of the trajectory over a fixed time interval Δ. The pose error at timestep i is then defined as [SEE+12]:

$$\mathbf{E}_i := (\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta})^{-1}(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}) \tag{5.1}$$

From a sequence of n camera poses, we obtain in this way $m = n - \delta$ individual relative pose errors along the sequence. From these errors, Sturm et al. propose to compute the root mean square error (RMSE) over all time indicies of the translational component as: [SEE+12]

$$\mathrm{RMSE}(\mathbf{E}_{1:n}, \Delta) := \left(\frac{1}{m}\sum ||trans(\mathbf{E}_i)||^2\right)^{1/2} \tag{5.2}$$

$trans(\mathbf{E}_i)$ refers to the translational components of the relative pose error $\mathbf{E}_i$.

Absolute Trajectory Error (ATE)

Beside the local consistency it is also important to measure the global consistency of the estimated trajectory [SEE+12]. The global consistency can be evaluated by comparing the absolute distances between the estimated and ground truth trajectory [SEE+12]. For comparing the differences we first have to transform them into the same coordinate frame. This can be done in closed form by a least squares as proposed by Horn [Hor87]. The estimated rigid body transformation is stored in $\mathbf{S}$

The Absolute Trajectory Error (ATE) is calculated by:

$$\mathbf{F}_i = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{5.3}$$

Sequence	avg. trans. vel. [m/s]	avg. rot. vel. [deg/s]	RPE	ATE
fr1/xyz	0.24	8.92	0.0218	0.0513
fr1/rpy	0.06	50.15	0.0477	0.0977
fr2/xyz	0.06	1.72	0.0038	0.0130
fr2/rpy	0.01	5.77	0.0037	0.0187

Table 5.1: Evaluation results on Testing and Debugging sequences

As for the RPE Sturm et al. suggest to evaluate the RMSE over all time indices of the translation component:

$$\text{RMSE}(\mathbf{F}_{1:n}, \Delta) := \left(\frac{1}{m} \sum ||trans(\mathbf{F}_i)||^2 \right)^{1/2} \tag{5.4}$$

5.2 Results

We now present the evaluation results on all relevant datasets. More detailed evaluation results are given in Appendix A.

The evaluation was executed with focus on precision of the estimated motion on a workstation notebook holding a Intel Core i7-2720QM CPU @ 2.20GHz and 12GB of memory.

5.2.1 Accuracy

First we show the effects of different velocities (see Table 5.1) by comparing the errors for similar trajectories using the fr1/xyz and fr2/xyz on the Testing and Debugging sequences. The system performs well for low velocities with a RPE of 0.0038 on the fr2/xyz sequence while pose estimates with higher velocities result in a much higher RPE of 0.0218 on the fr1/xyz sequence. This effect could be explained by the motion blur of the images and the rolling shutter of the camera. Also the block matcher, interpolating the depth values, will have a higher influence on bigger movements. By using a RGB-D camera delivering higher frame rates with an equal image quality or distances measurement for each pixel we will be able to further improve the results. Figure 5.4 shows a RPE plot for both datasets. While accuracy of the fr2/xyz is in a centimeter range with an max error of $1.7\,cm$ it will have errors up to $9\,cm$ on the faster fr1/xyz dataset. We also can observe that the accuracy decreases on camera rotation in comparison to translation.

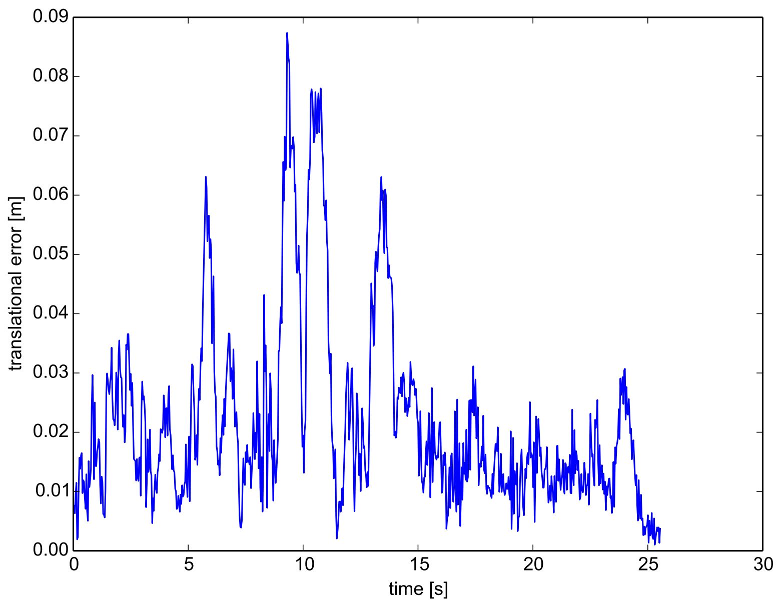


Figure 5.2: RPE on fr1/xyz

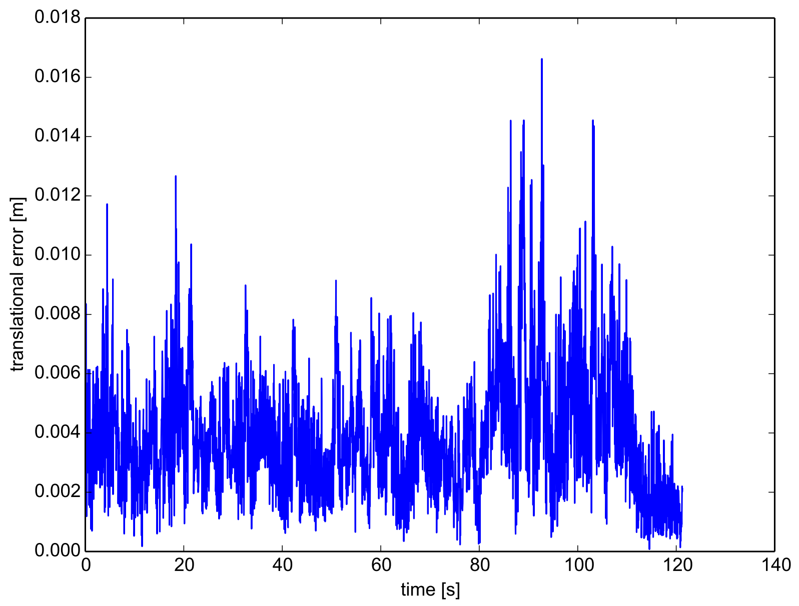
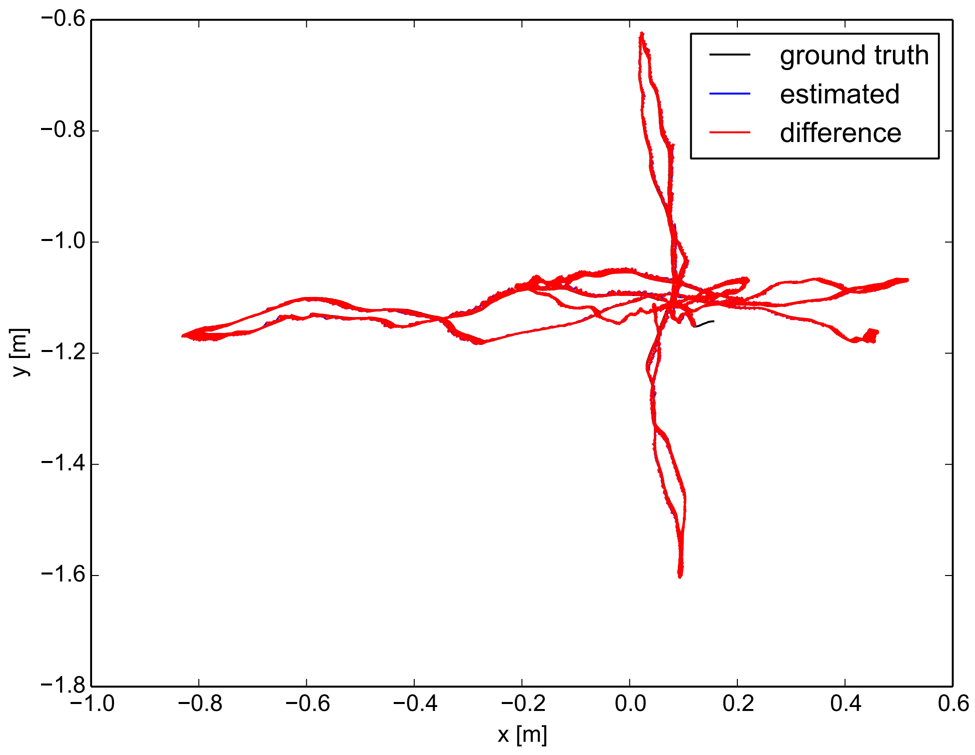
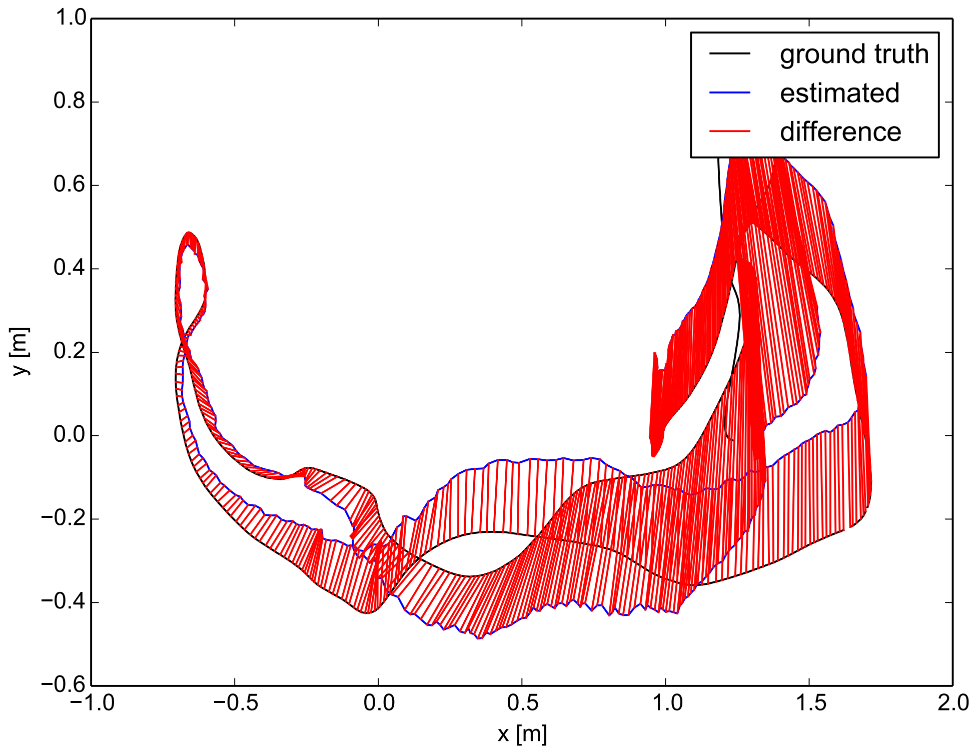


Figure 5.3: RPE on fr2/xyz

Figure 5.4: RPE comparison with different velocities on fr1/xyz and fr2/xyz sequences



(a) ATE on fr2/xyz



(b) ATE on fr1/desk

Figure 5.5: While the ATE on the fr2/xyz dataset (1) is low and thus suitable for mapping, the pose drift on the fr1/desk dataset (2) is to high to create accurate maps.

Method	fr1/desk2 [m/s]	fr1/desk [m/s]	fr2/desk [m/s]
DVO	0.0687	0.0458	0.0188
FOVIS (used by us)	**0.0424**	**0.0327**	0.0118
avg. camera velocity	0.413	0.426	0.193

Table 5.2: Comparison to DVO

5.2.2 Suitability for Mapping

For sequences with a low ATE like fr2/xyz and fr2/rpy we are able to build plausible maps. While for sequences recorded with higher velocities the accumulated pose drift will influence the map quality. Once a motion estimate is inaccurate there is currently no chance to correct for this error. For global consistent maps at higher velocities, we recommend an integration into a graph-based SLAM framework. The pose estimates of the visual odometry fill the pose vector $\mathbf{x}$ while distance measurements of interest points fill the measurement vector $\mathbf{z}$. A keyframe based approach can be used for loop closures as in RGB-D-SLAM [EEH+11].

5.2.3 Comparison

We now will compare our integration of FOVIS with other state-of-art methods. First, in Table 5.2 we compare the visual odometry results with the DVO approach. For global consistency we compare our integration with RGB-D-SLAM [EEH+11], DVO-SLAM [KSC13a] and the open source PCL implementation of Kinect Fusion [IKH+11].

The accuracy outperforms current state-of-art methods for dense visual odometry estimation DVO [KSC13b] in static scenes as shown in Table 5.2 but due to the missing global alignment will perform worse on the absolute trajectory (see Table 5.3). In comparison to Kinect Fusion our ATE is lower for longer trajectories and Kinext Fusion tends to drift more on rotations. Like our approach Kinect Fusion does not handle any global optimization on the estimated poses while RGB-D SLAM and DVO_SLAM integrate into a graph-based SLAM.

5.2.4 Runtime

The runtime, dependent on the parameter set, can be optimized to run up to 2 Hz. For more accuracy the system can take up to 2 s per frame. The rate will be sufficient for handheld or human operated devices, but will not be sufficient for autonomous systems in domestic environments. We therefore suggest to further optimize for parallelism in the motion estimation process. Also the performance

Dataset	Fovis (used by us)	DVO SLAM	RGB-D SLAM	KinFu
fr1/xyz	0.051	**0.011**	0.014	0.026
fr1/rpy	0.097	**0.020**	0.026	0.133
fr1/desk	0.258	**0.021**	0.023	0.057
fr1/desk2	0.124	0.046	**0.043**	0.420
fr1/room	0.184	**0.053**	0.084	0.313
fr1/360	0.140	0.083	**0.079**	0.913
fr2/desk	0.102	**0.017**	-	-

Table 5.3: Comparison to other state-of-art methods. Other results are taken from DVO_SLAM [KSC13a] publication

comparison between the current clique-graph inlier detection and RANSAC approach will be of interest.

Chapter 6

Conclusion

In this chapter a summarization of the presented approaches for the motion estimation and mapping using a RGB-D camera is given. Then in the outlook section (Section 6.2) we describe methods for further improvement and extension.

6.1 Conclusion

In this thesis an approach to create a dense 3D map in real time using only a RGB-D camera was presented.

For the motion estimation a feature based approach was used. After extracting features from the color image, the corresponding depth values where extracted. When the depth values where assigned the extracted features where matched with the previous RGB-D image. This leads to 3D-3D correspondences on which the relative motion between the camera was estimated using a Least-Squares approach or alternatively an direct approach using the SVD. The relative motion estimate is then multiplied incrementally and leads to a global motion estimation.

To represent a 3D map an octree data structure was chosen, to reduce the high amount of data. The incremental calculated motion estimate serves as input to the OctoMap framework which leads to a incremental build 3D map.

The system presented in this thesis then was evaluated using the RGB-D benchmark.

Camera poses and maps for slow camera movements are very accurate while the system needs further improvement on faster moving cameras.

In general the system is very modular. For instance the visual motion estimation can be replaced by an other estimation method i. e. the odometry of a robotic platform or the pose estimation of a robot.

6.2 Outlook

We now suggest further improvements to get more accurate poses and maps.

Graph SLAM While, in general, it would be desirable to be able to extract accurate motion estimates based on this approach, we have to deal with noisy measurements and thus accumulate errors in a way that the motion estimate drifts. To threshold that problem, the visual odometry approach should be integrated in a graph-based SLAM [GKSB10], [N+07] by building a pose graph using the motion estimates. For loop closing, key frames giving strong image features can be saved. If a new incoming frame matches very good with the keyframe we detected a loop and could optimize the pose graph.

IMU integration Li et al. showed in [LM13] and [LM12] that camera based motion estimation could be improved by fusing it with motion data of an IMU using an Extented Kalman Filter (EKF). They demonstrated this on a smart phone camera and an integrated IMU on an one kilometer walk without any pose refinement or loop closing and got an absolute error of only 5 m.

Time of Flight (TOF) evaluation During the creation of this thesis Microsoft released a developer version of their second Kinect version, which uses a TOF approach to produce a depth image. In comparison to the depth cameras with a PrimeSense chip they deliver 640×480 depth map with a real depth value for each pixel in the depth map, in comparison to an estimated depth by an 9×9 block matcher from the previous generation. This should lead to more accurate and detailed maps and also to less errors in the motion estimation process.

Adaptive Depth / Sparse Approach While sparse image feature approaches work quite well for texture rich environments they tend to fail in homogeneous scenes. An idea how to overcome this is when there won't be enough features found we could use a depth based fall-back like extracting and matching 3D geometry features or use direct SDF approach like [BSK+13].

Robustness againt enviromental changes The proposed approach of this thesis will likely fail when the environment changes while estimating the motion. To overcome this Kerl et al. proposed in [KSC13b] to use weight functions on the residuals of an dense approach. An other idea is to track features an calculate an optical flow. When have a higher movement or differ from surrounding features they could be rejected from the motion estimation.

Volumetric Mapping Steinbrücker et al. showed in [SSC14] a visual impressive method to represent highly detailed RGB-D maps. This approach could be additionally integrated into the proposed system. While the approach from Steinbrücker et al. works on ground truth camera poses it also needs to be evaluated how accurate the generated maps would be from visual estimated camera poses.

List of Tables

5.1 Evaluation results on Testing and Debugging sequences 48
5.2 Comparison to DVO 51
5.3 Comparison to other state-of-art methods 52

A.1 Detailed evaluation results 62

List of Figures

1.1 Feature tracking on the NASA Spirit MER 11
1.2 Implementation Overview 14

2.1 Comparison of feature and direct based methods for camera tracking 15
2.2 RGB-D SLAM 16
2.3 PTAM 17
2.4 DTAM 19
2.5 Kinect Fusion pipeline 21

3.1 FAST [RD06] corner detector 25
3.2 SIFT Descriptor 26
3.3 Matched key points between two consecutive intensity frames 26
3.4 Calibration Images 31
3.5 Pinhole camera model 32
3.6 Depth registration 34
3.7 Depth Color Registration 34
3.8 Comparison between synchronized and unsynchronized color- and depth images 36

4.1 Octree data structure 41
4.2 Generated octree map example 42
4.3 OctoMap for fr1/360 sequence 43

5.1 Handheld Kinect 45
5.2 RPE on fr1/xyz 49
5.3 RPE on fr2/xyz 49
5.4 RPE comparison with different velocities on fr1/xyz and fr2/xyz sequences 49
5.5 ATE comparison for mapping 50

A.1 RPE on fr1/xyz 63
A.2 ATE on fr1/xyz 63

A.3 RPE on fr1/rpy 64
A.4 ATE on fr1/rpy 64
A.5 RPE on fr2/xyz 65
A.6 ATE on fr2/xyz 65
A.7 RPE on fr2/rpy 66
A.8 ATE on fr2/rpy 66
A.9 RPE on fr1/360 67
A.10 ATE on fr1/360 67
A.11 RPE on fr1/floor 68
A.12 ATE on fr1/floor 68
A.13 RPE on fr1/desk 69
A.14 ATE on fr1/desk 69
A.15 RPE on fr1/desk2 70
A.16 ATE on fr1/desk2 70
A.17 RPE on fr1/room 71
A.18 ATE on fr1/room 71
A.19 RPE on fr2/360_hemi 72
A.20 ATE on fr2/360_hemi 72
A.21 RPE on fr2/360_kidnap 73
A.22 ATE on fr2/360_kidnap 73
A.23 RPE on fr2/desk 74
A.24 ATE on fr2/desk 74

D.1 World frame convention 79
D.2 Camera frame convention 79

Appendix A

Detailed Evaluation Results

Sequence	Duration [s]	Length [m]	Avg. Trans. Vel. [m/s]	Avg. Rot. Vel. [deg/s]	RPE	ATE	Runtime
Testing / Debugging							
fr1/xyz	30.09s	7.112m	0.24	8.92	0.0218544356933	0.051337	6m11.686s
fr1/rpy	27.67s	1.664m	0.06	50.15	0.0477871834046	0.097700	5m35.695s
fr2/xyz	122.74s	7.029m	0.06	1,72	0.00385282416511	0.013073	27m47.608s
fr2/rpy	109.97s	1.506m	0.01	5.77	0.00374915698089	0.018747	21m42.376s
Handheld SLAM							
fr1/360	28.69s	5.818m	0.21	41.60	0.0720565844545	0.140682	6m37.161s
fr1/floor	49.87s	12.569m	0.26	15.07	0.0449647603323	0.250151	13m42.953s
fr1/desk	23.40s	9.263m	0.41	23.33	0.032735059229	0.258913	17m20.607s
fr1/desk2	24.86s	10.161m	0.43	29.31	0.0424628827188	0.124799	4m50.280s
fr1/room	48.90s	15.989m	0.33	29.88	0.0460558630087	0.184176	11m35.210s
fr2/360 hemisphere	91.48s	14.773m	0.16	20.57	0.104729726504	0.534881	15m32.294s
fr2/360 kidnap	48.04s	14.286m	0.30	13.43	0.124198983592	0.909020	7m44.058s
fr2/desk	99.36s	18.880m	0.19	6.34	0.0118710324953	0.102925	22m11.227s
fr2/large no loop	112.37s	26.086m	0.24	15.09	0.0817289735221	1.059697	81m11.407s
fr2/large with loop	173.19s	39.111m	0.23	17.21	0.119136371982	2.757940	29m12.723s
fr2/long office household	87.09s	21.455m	0.25	10.19	0.0126214725687	0.101096	82m23.346s
Robot SLAM							
fr2/pioneer 360	72.75s	16.118m	0.23	12.05	0.128221063606	0.644579	5m33.814s
fr2/pioneer slam	155.72s	40.380m	0.26	13.38	0.0867684238886	1.000429	66m39.400s
fr2/pioneer slam2	115.63s	21.735m	0.19	12.21	0.0741344135691	1.712665	14m23.118s
fr2/pioneer slam3	111.91s	18.135m	0.16	12.34	0.0709210119758	1.102463	12m58.770s

Table A.1: Detailed evaluation results - During the evaluation we were not able to associate all ground-truth and estimated poses measurements (namely fr2/large with loop, fr2/desk, fr2/large no loop) which results on high RPE and ATE on these datasets even if the estimated trajectory looks plausible.

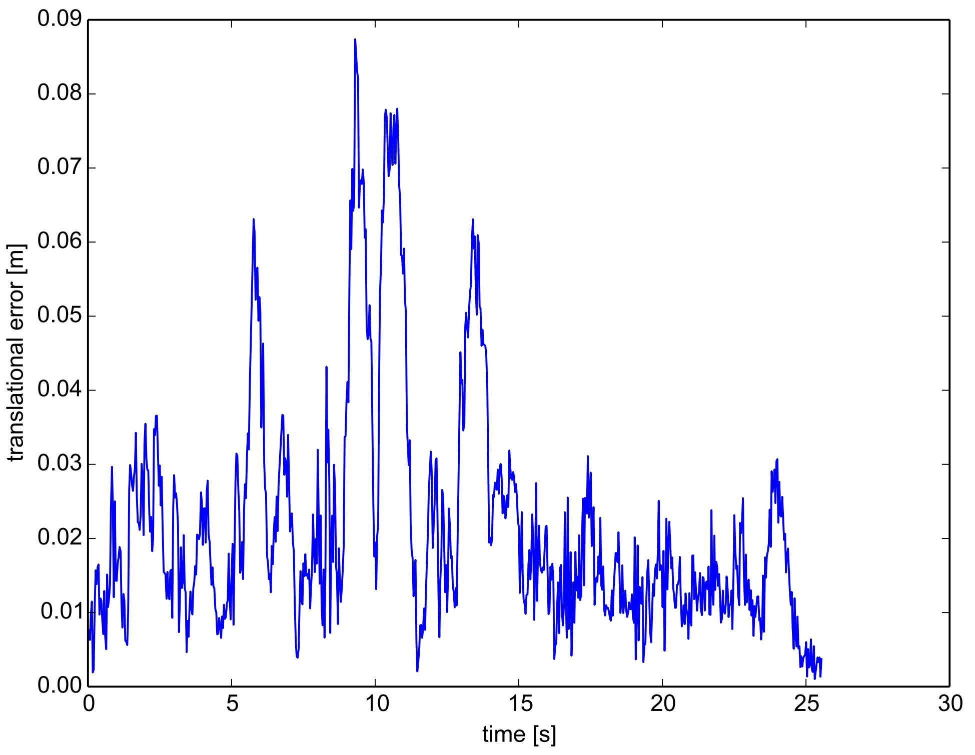


Figure A.1: RPE on fr1/xyz

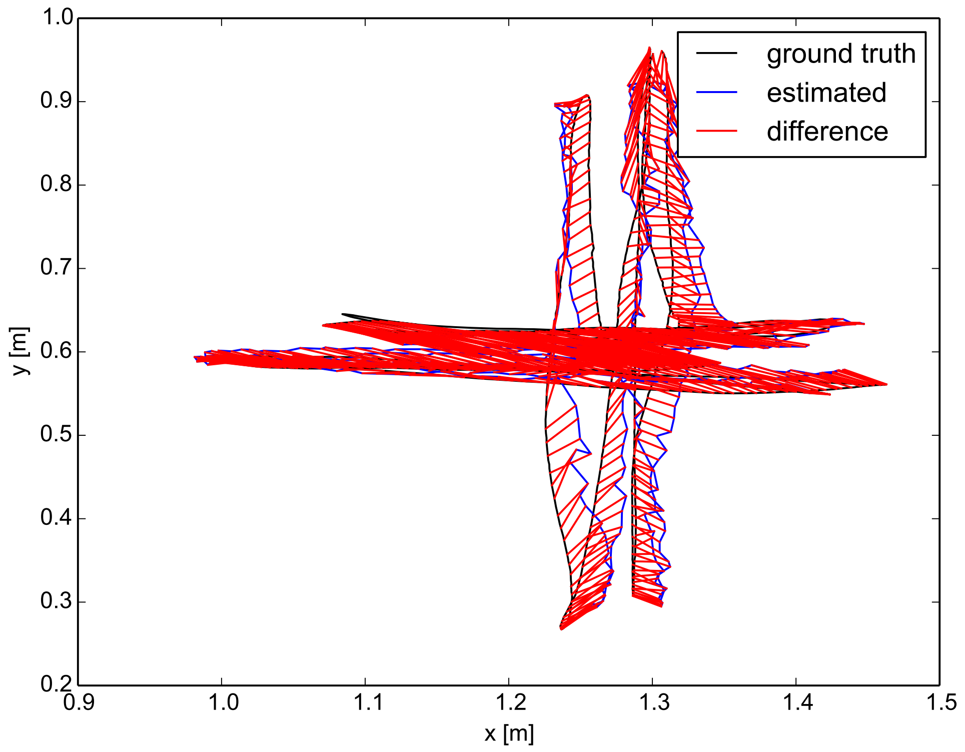


Figure A.2: ATE on fr1/xyz
Evaluation results using the fr1/xyz dataset

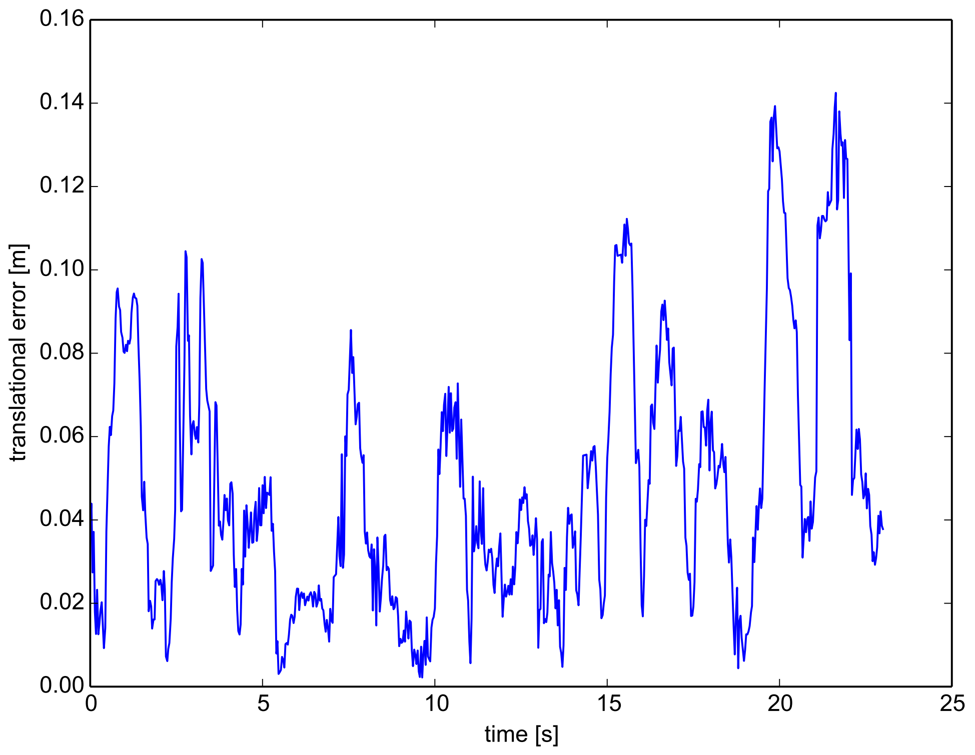


Figure A.3: RPE on fr1/rpy

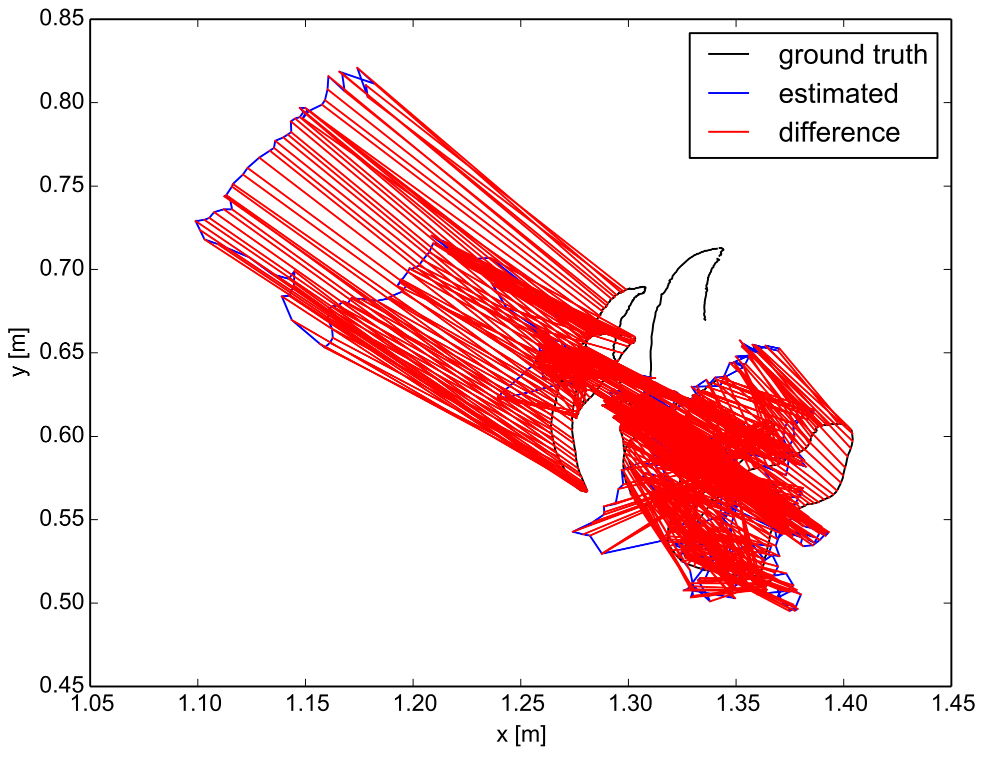


Figure A.4: ATE on fr1/rpy
Evaluation results using the fr1/rpy dataset

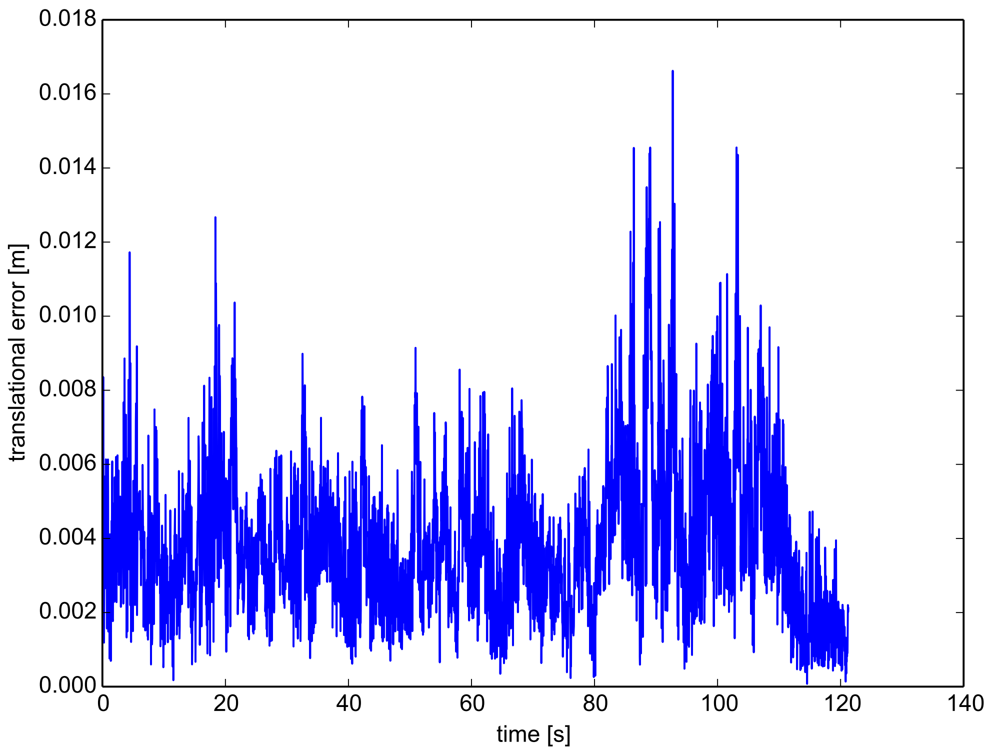


Figure A.5: RPE on fr2/xyz

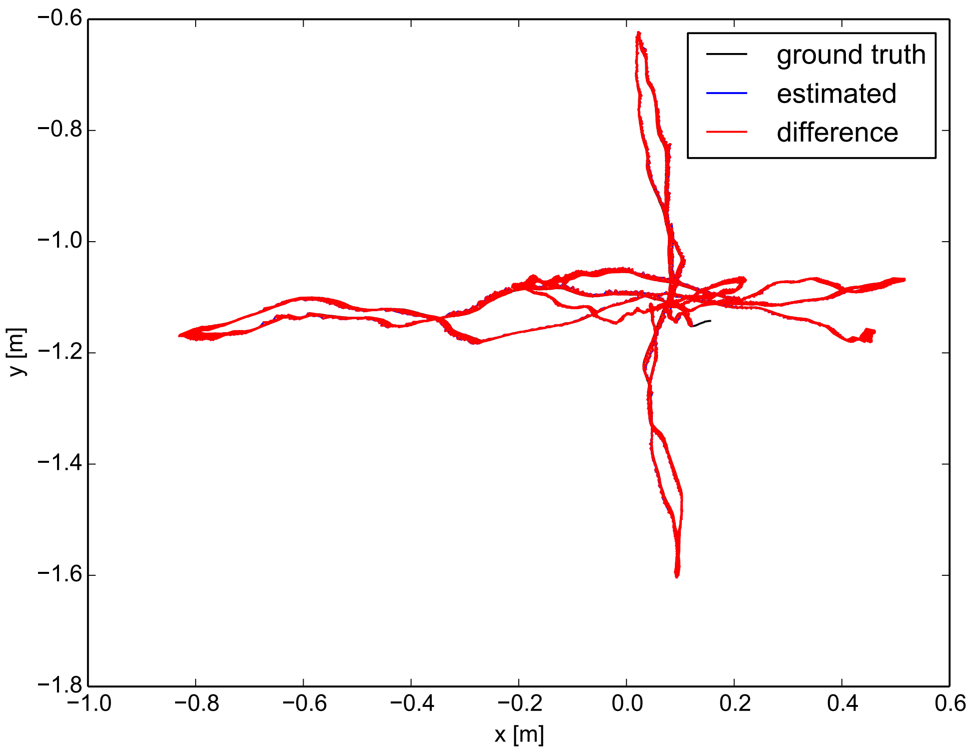


Figure A.6: ATE on fr2/xyz
Evaluation results using the fr2/xyz dataset

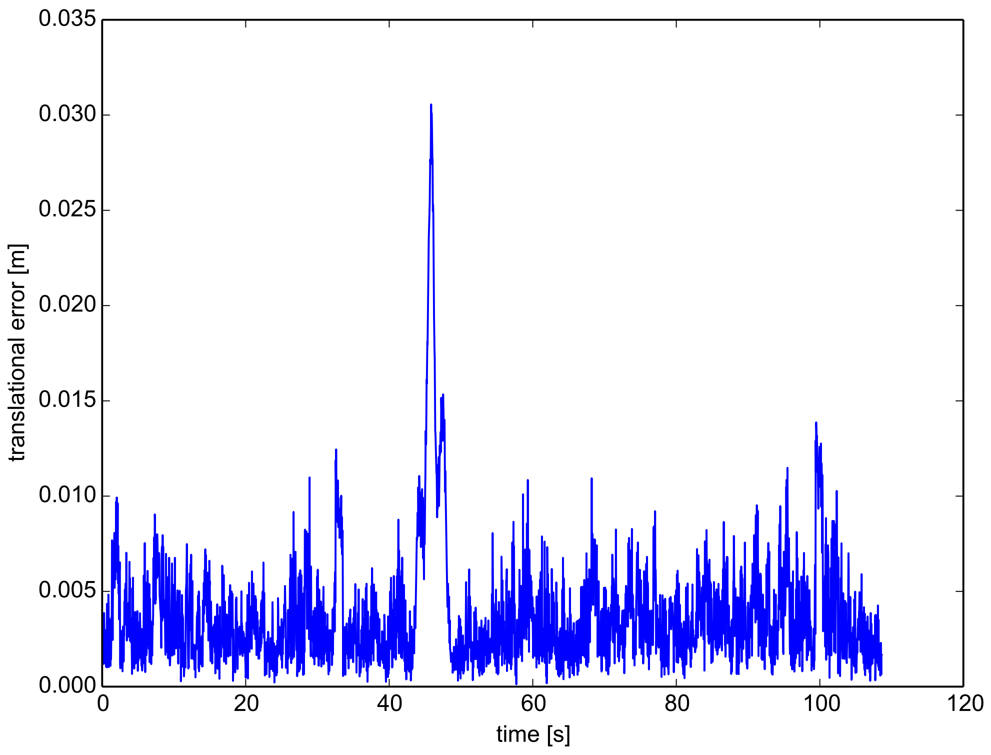


Figure A.7: RPE on fr2/rpy

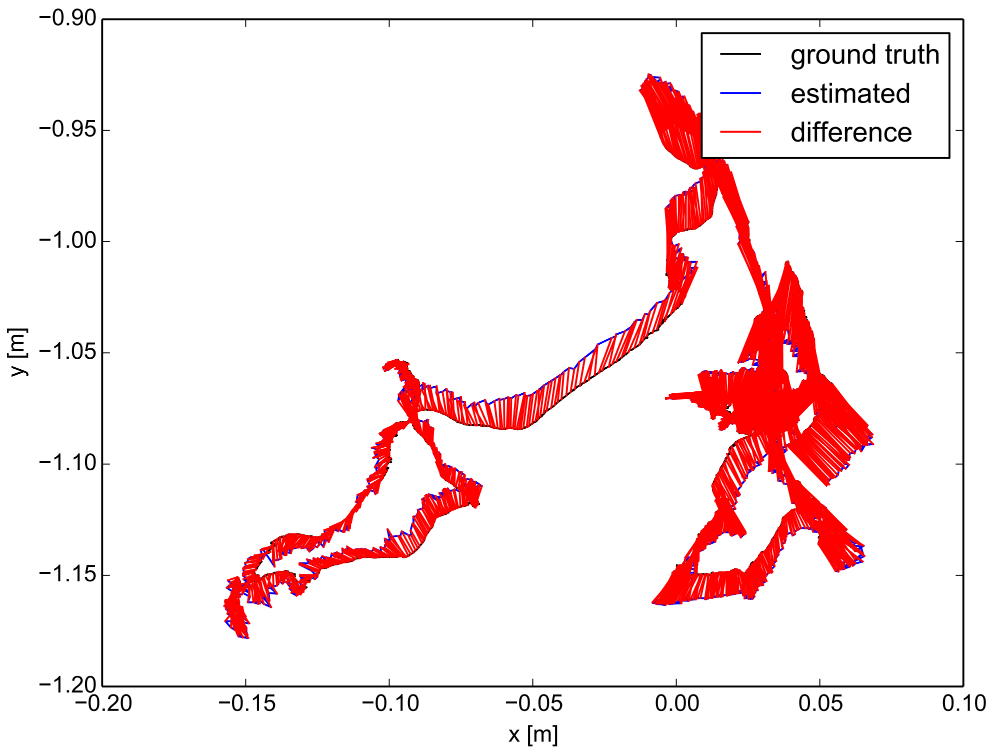


Figure A.8: ATE on fr2/rpy
Evaluation results using the fr2/rpy dataset

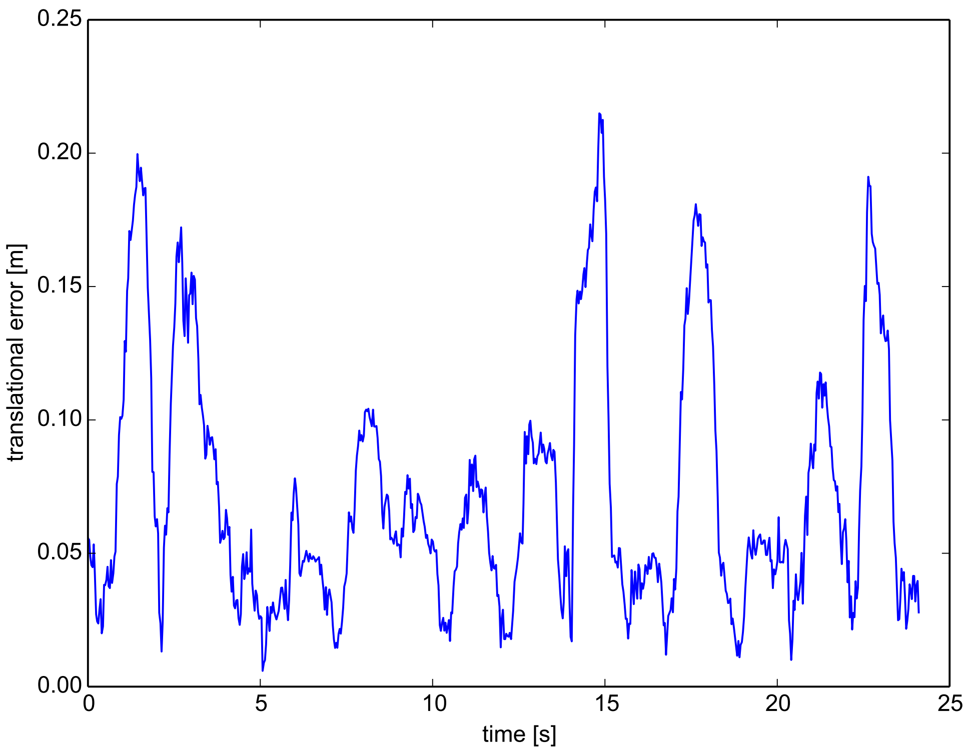


Figure A.9: RPE on fr1/360

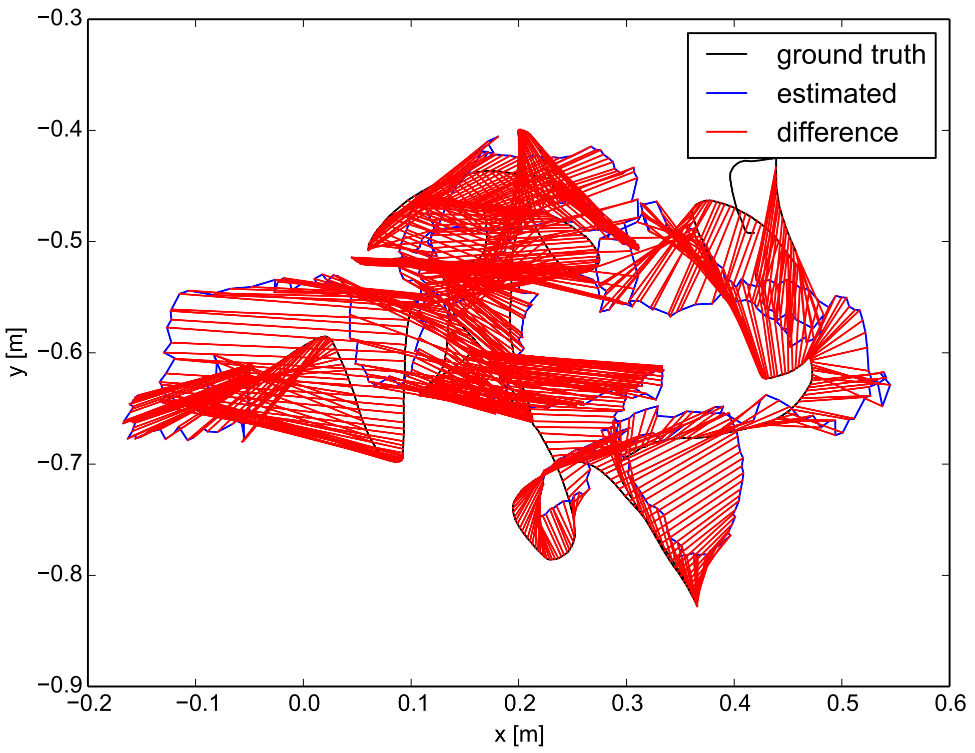


Figure A.10: ATE on fr1/360
Evaluation results using the fr1/360 dataset

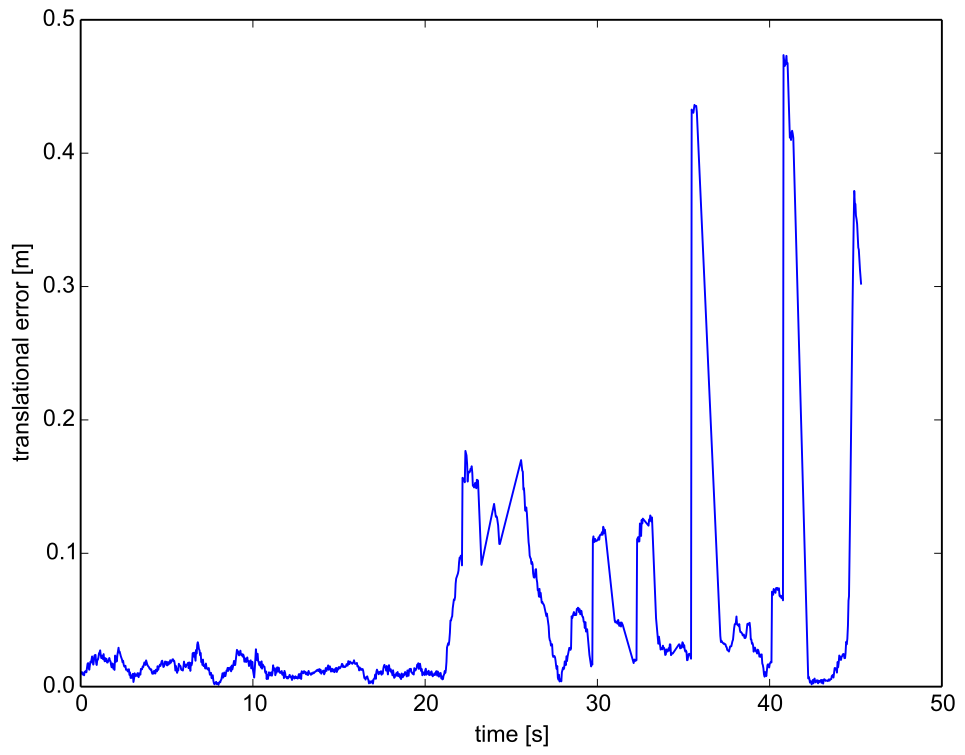


Figure A.11: RPE on fr1/floor

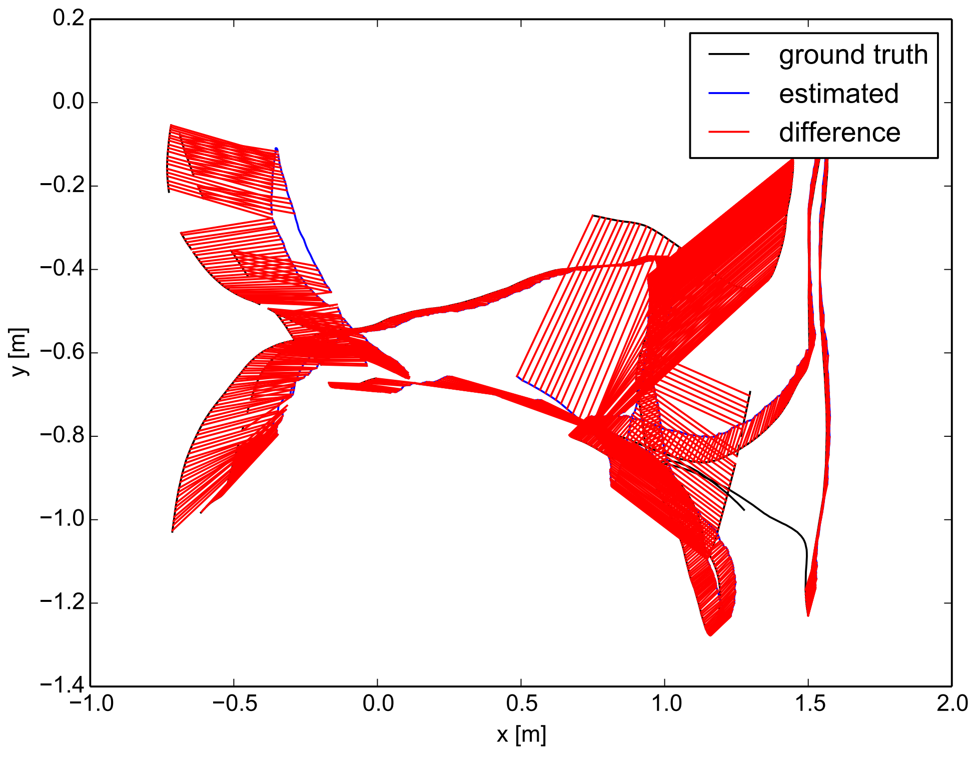


Figure A.12: ATE on fr1/floor
Evaluation results using the fr1/floor dataset

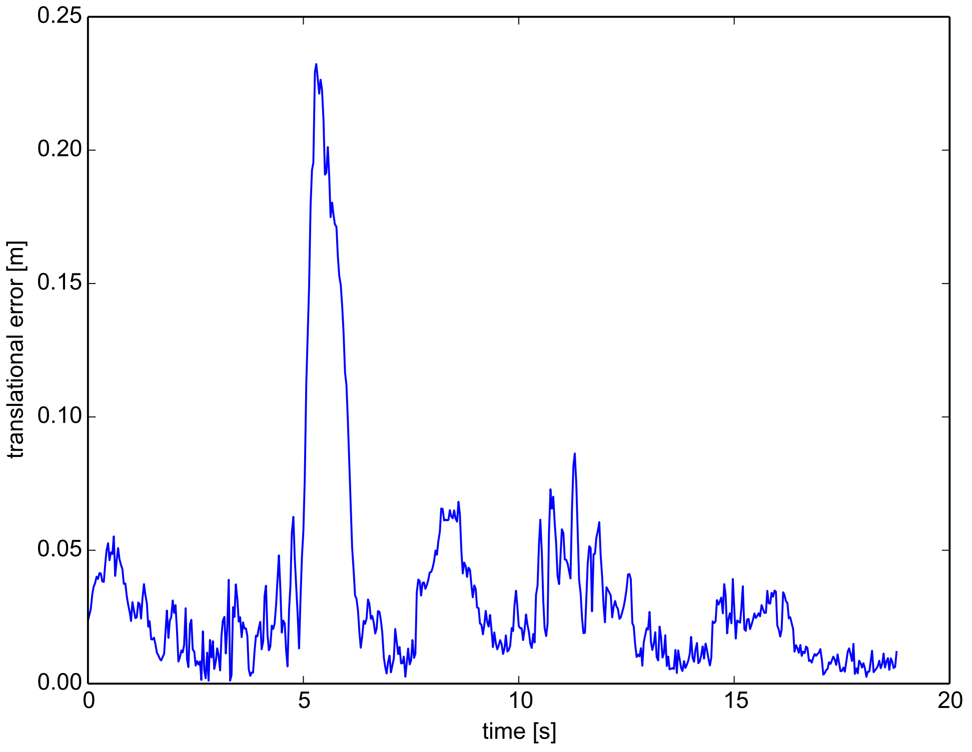


Figure A.13: RPE on fr1/desk

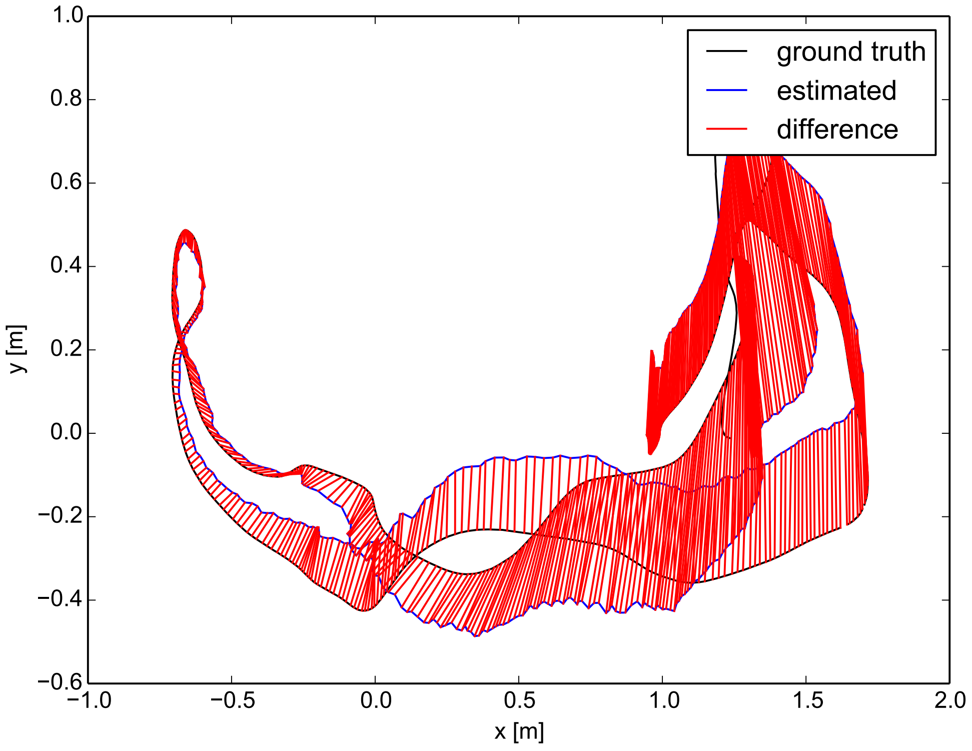


Figure A.14: ATE on fr1/desk
Evaluation results using the fr1/desk dataset

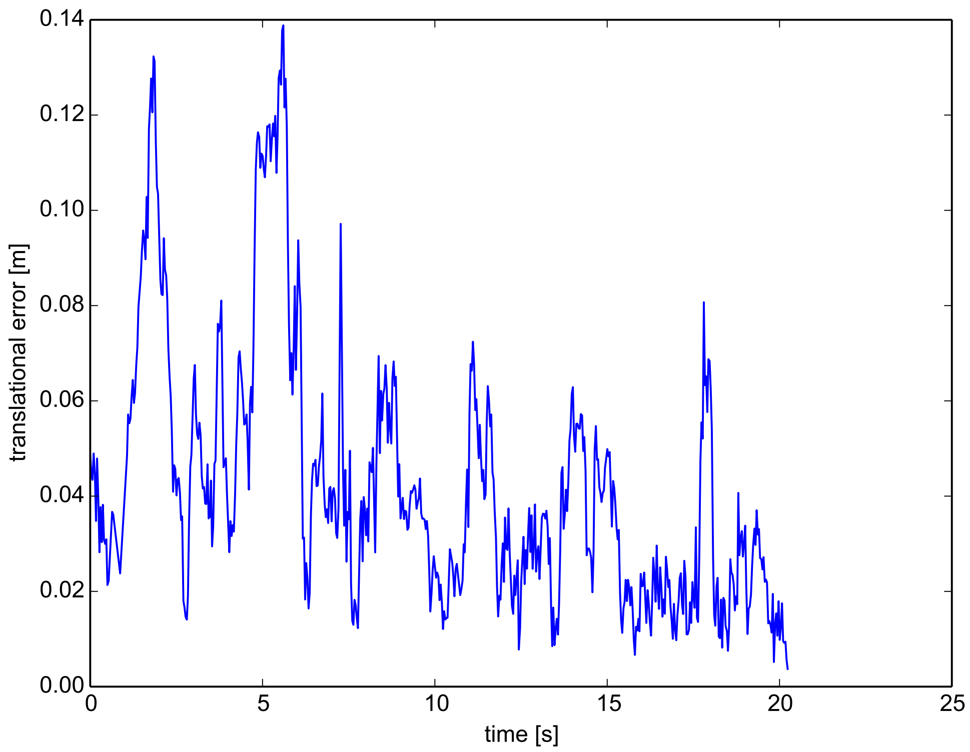


Figure A.15: RPE on fr1/desk2

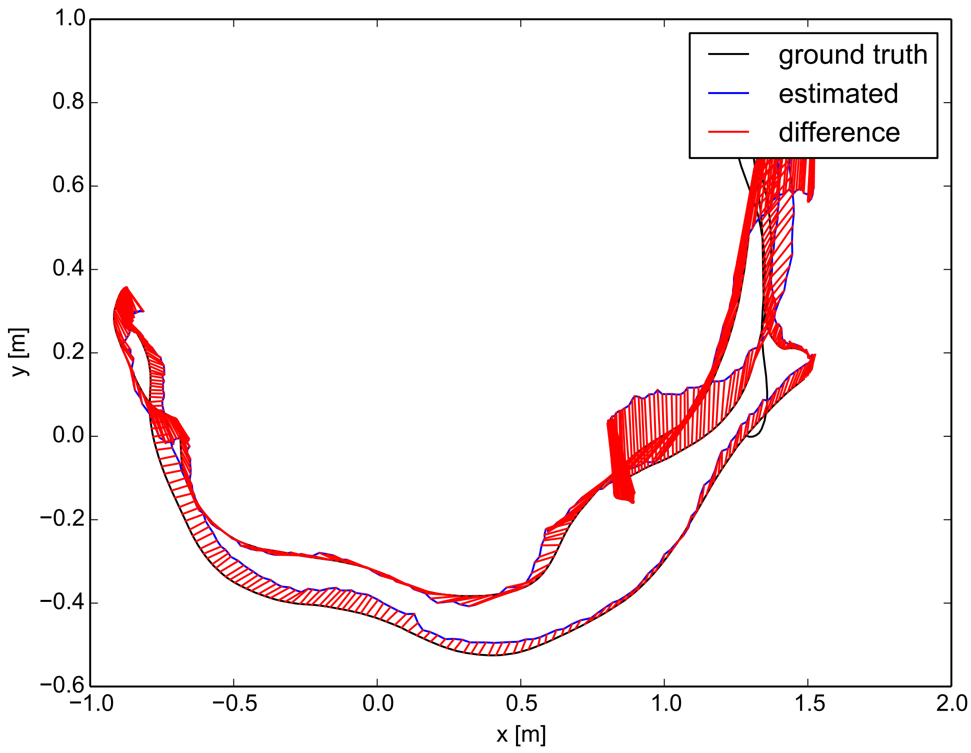


Figure A.16: ATE on fr1/desk2
Evaluation results using the fr1/desk2 dataset

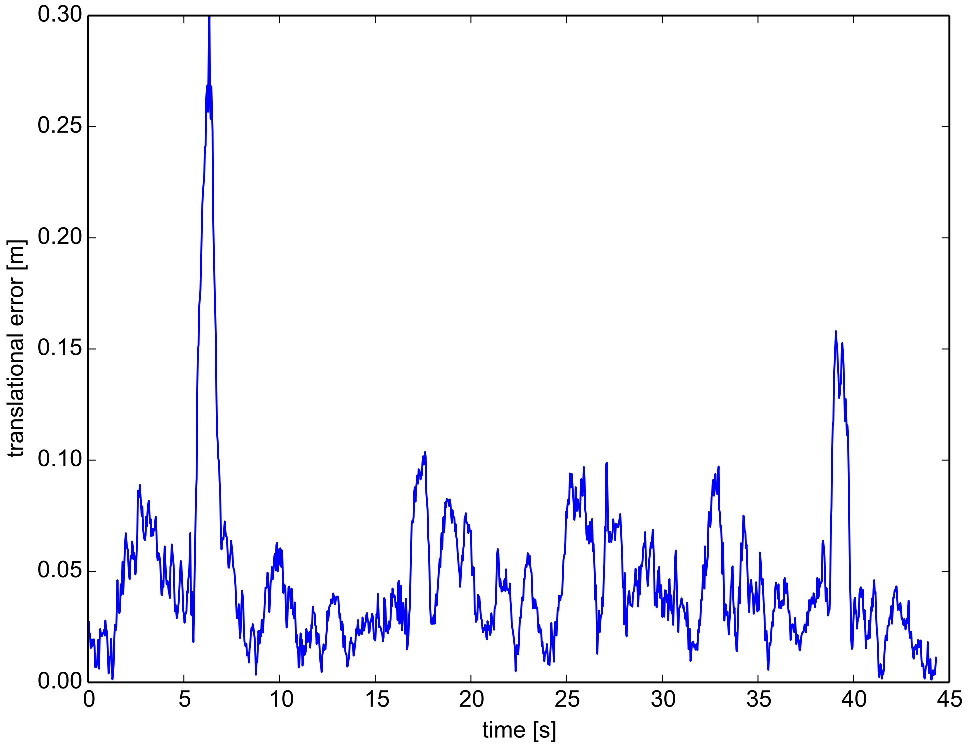


Figure A.17: RPE on fr1/room

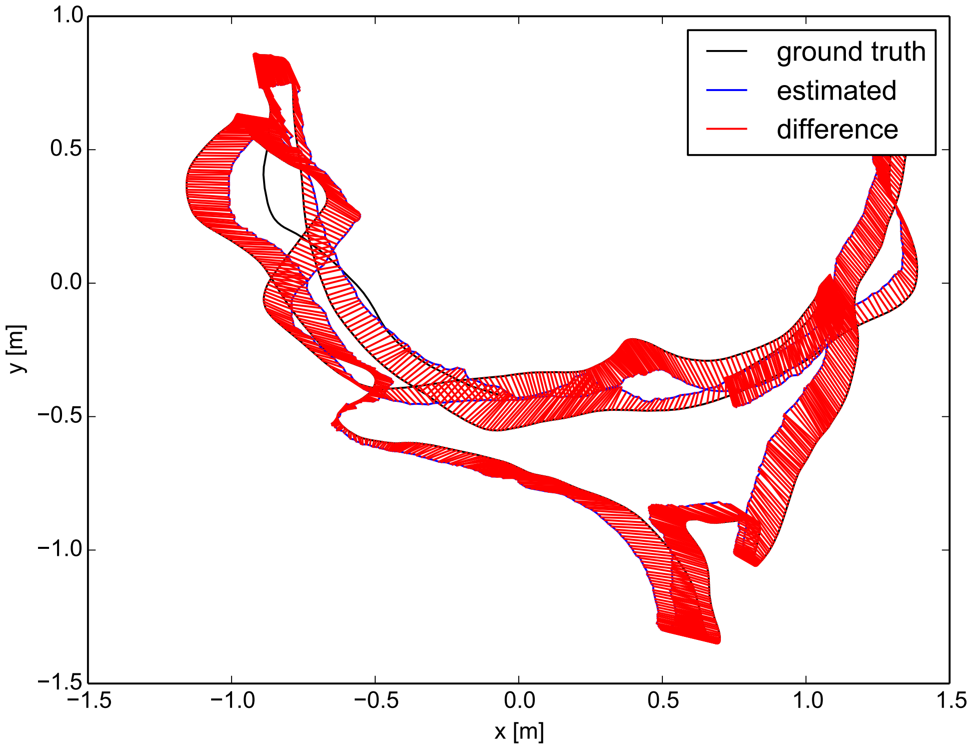


Figure A.18: ATE on fr1/room
Evaluation results using the fr1/room dataset

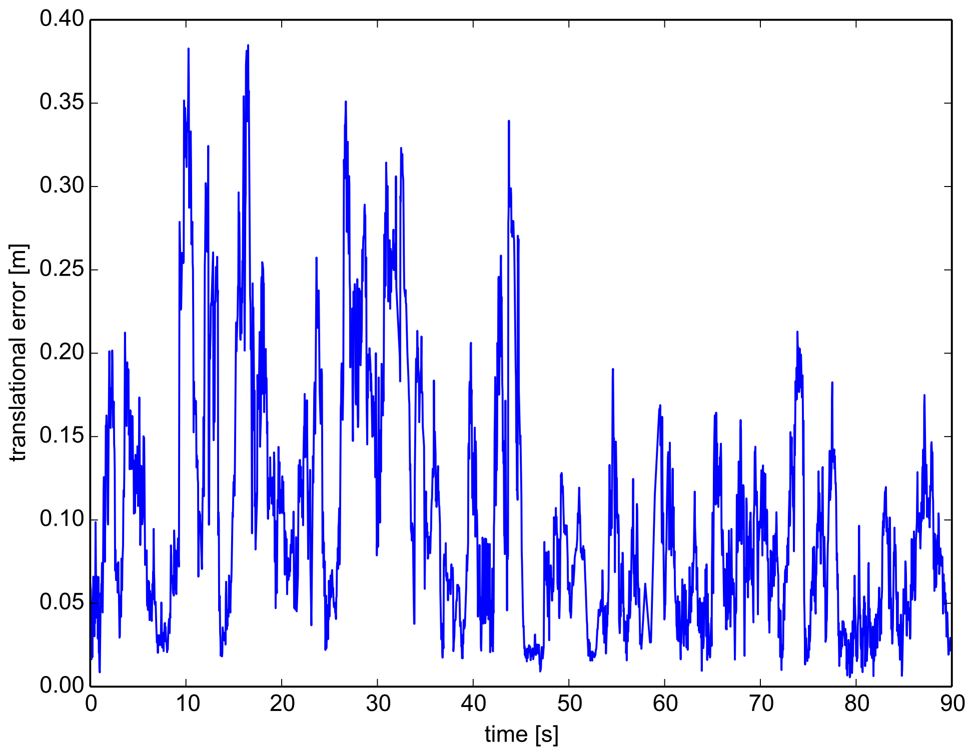


Figure A.19: RPE on fr2/360_hemi

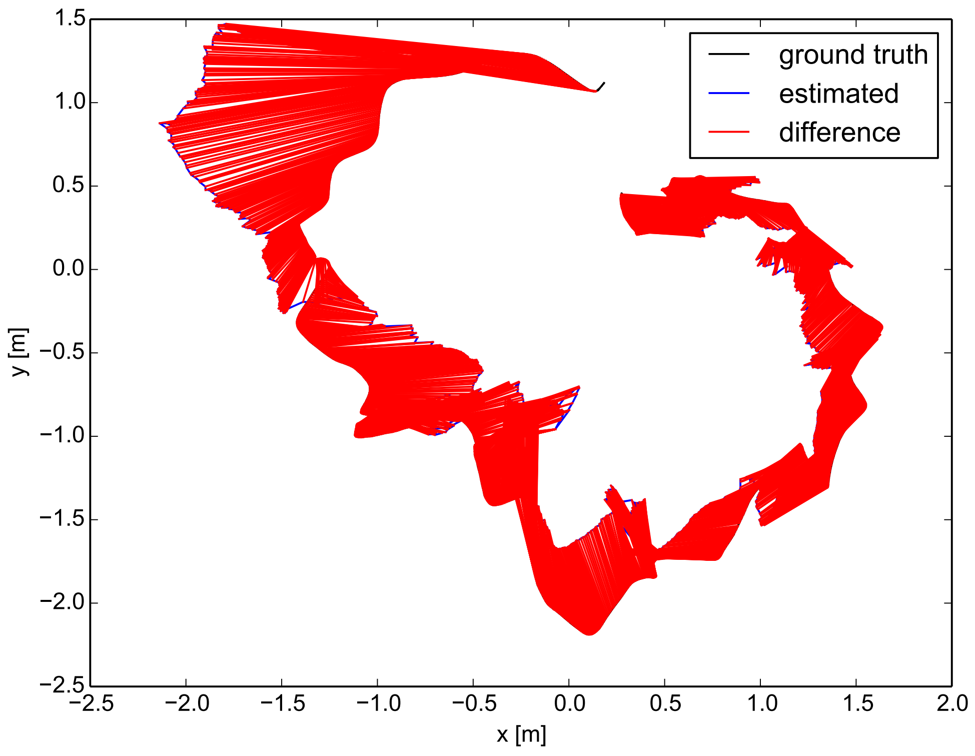


Figure A.20: ATE on fr2/360_hemi
Evaluation results using the fr2/360_hemisphere dataset

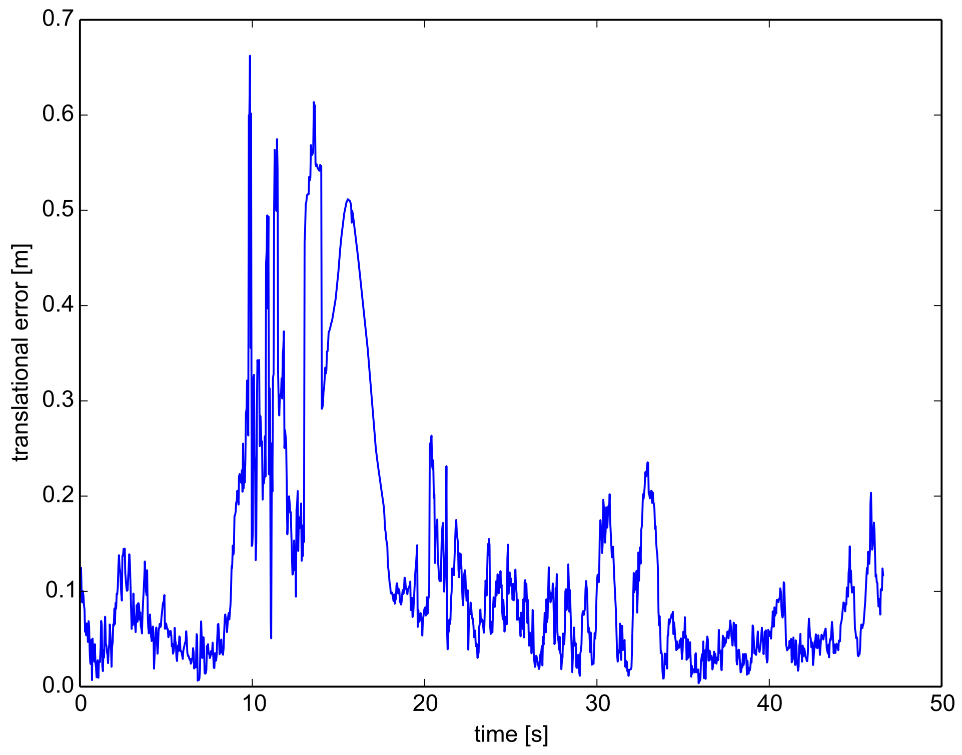


Figure A.21: RPE on fr2/360_kidnap

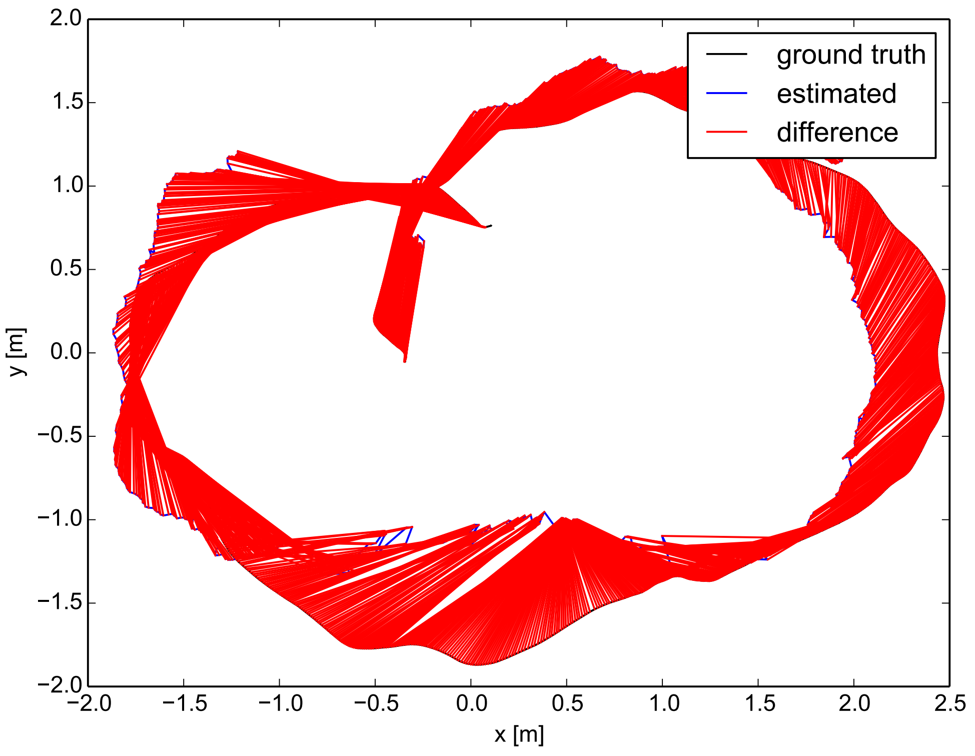


Figure A.22: ATE on fr2/360_kidnap
Evaluation results using the fr2/360_kidnap dataset

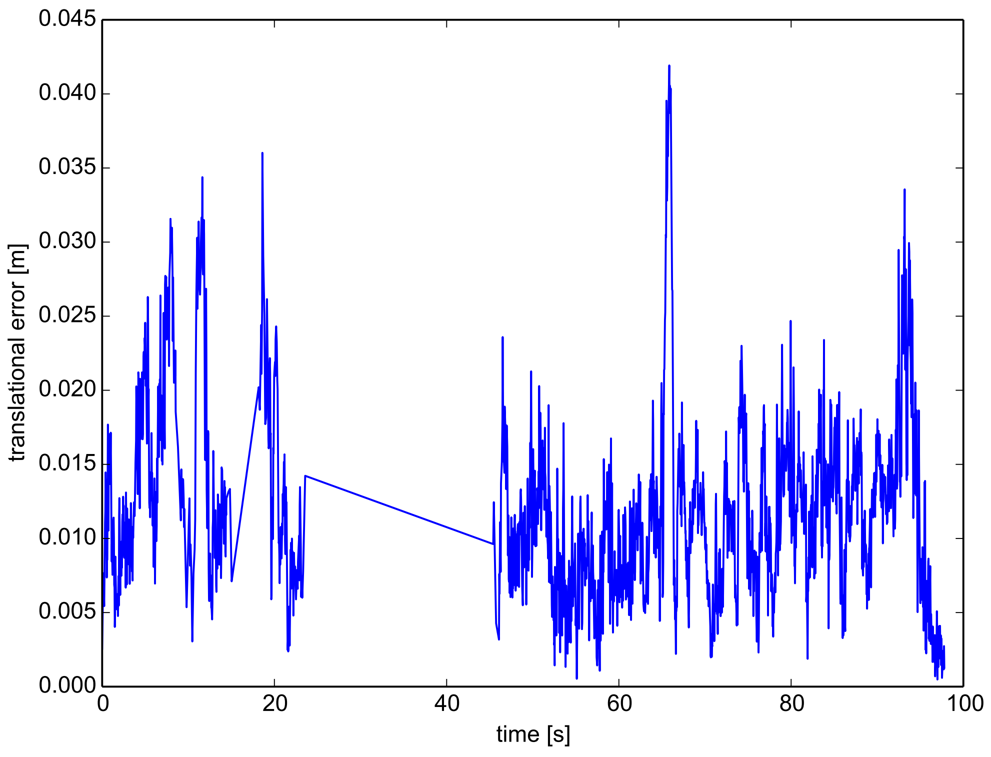


Figure A.23: RPE on fr2/desk

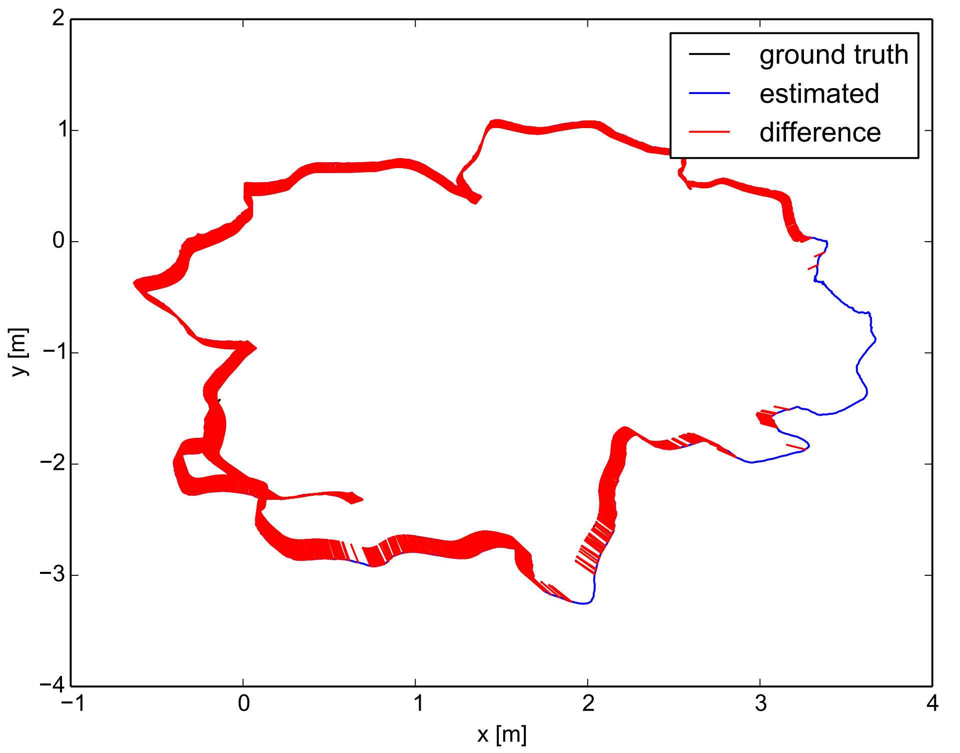


Figure A.24: ATE on fr2/desk
Evaluation results using the fr2/desk dataset

Appendix B

Acronyms

AR Augmented Reality. 10, 17

ATE Absolute Trajectory Error. 47, 48, 50, 51, 62–74

CPU Central Processing Unit. 18, 20

DOF Degrees Of Freedom. 11

DTAM Dense Tracking And Mapping. 18, 20

DVO Dense Visual Odometry. 18, 51, 52

EKF Extented Kalman Filter. 54

FAST Features from accelerated segment test. 24, 25

FLANN Fast Library for Approximate Nearest Neighbors. 12, 27

Fovis Fast Odometry From Vison. 12, 51, 52

GPS Global Positioning System. 11

GPU Graphical Processing Unit. 17, 18, 20

HRI Human-Robot Interaction. 11

ICP Iterative Closest Points. 12, 20, 21, 28, 42, 44

IMU Inertial Measurement Unit. 18, 54

IR Infrared. 31, 35

MER Mars Exploration Rover. 10

MLE Maximum Likelihood Estimation. 28

OpenCV Open Computer Vision. 12

ORB Oriented BRIEF. 17

PCL Point Cloud Library. 12, 51

PTAM Parallel Tracking And Mapping. 17–19

RANSAC RANdom SAmple Consensus. 12, 17, 20, 27, 28, 30, 52

RMSE root mean square error. 47, 48

ROS Robot Operation System. 12

RPE Relative Pose Error. 47–49, 62–74

SDF Signed Distance Function. 20, 54

SIFT Scale Invariant Features. 16, 17, 20, 25, 26

SIMD Single Instruction Multiple Data. 18

SLAM Simultaneous Localization and Mapping. 10, 17, 20, 46, 51, 52, 54, 62

SURF Speeded Up Robust Features. 17

SVD Singular Value Decomposition. 12, 28, 44, 53

TOF Time of Flight. 54

UAV Unmanned Areal Vehicles. 10

VR Virtual Reality. 10

Appendix C

List of Symbols

$\mathbf{H}$ Covariance matrix between two point cloud sets. 28

$\mathbf{C}_{ir}$ Intrinsic matrix for infrared camera. 35

$\mathbf{C}_{rgb}$ Intrinsic matrix for RGB camera. 35

$\mathbf{C}$ Matrix containing the intrinsic camera parameters. 32

$\mathbf{z}$ Measurement vector. 51

$\mathbf{x}$ Pose vector. 51

x^c_{ir} Point in IR camera frame. 35

x^c_{rgb} Point in RGB camera frame. 13, 35

x^i_{ir} Point in IR image frame. 35

x^i_{rgb} Point in RGB image frame. 35

$\mathbf{E}$ Relative pose error. 47

$\mathbf{F}$ Absolute trajectory error. 47, 48

C Correspondence set. 42, 44

e Error value. 29

$\mathbf{S}$ Rigid body transformation for alignment. 47

p Point in 3D. 28, 29, 42, 44

q Point in 3D. 28, 29, 42, 44

$\mathbf{P}$ Set of poses. 30, 42, 47

$\mathbf{Q}$ Set of poses. 30, 42, 47

$trans$ Translational pose error component. 47, 48

$\mathbf{R}$ Rotation Matrix. 13, 28, 29, 31–33, 35, 42, 44

$\mathbf{T}$ Transformation Matrix. 29–31, 42

t Translation vector. 13, 29, 31–33, 35, 42, 44

Appendix D

Coordinate System

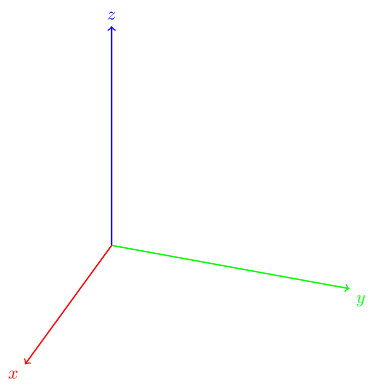


Figure D.1: World frame convention

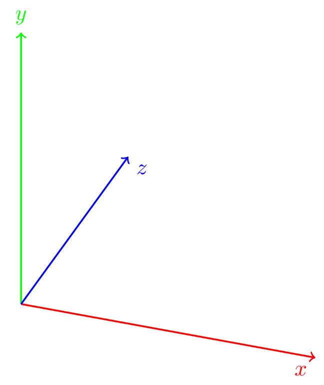


Figure D.2: Camera frame convention

Bibliography

[AFMS12] Arieli, Yoel ; Freedman, Barak ; Machline, Meir ; Shpunt, Alexander: *Depth mapping using projected patterns.* April 3 2012. – US Patent 8,150,142

[AHB87] Arun, K S. ; Huang, Thomas S. ; Blostein, Steven D.: Least-squares fitting of two 3-D point sets. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (1987), Nr. 5, S. 698–700

[Bra00] Bradski, Gary: The opencv library. In: *Doctor Dobbs Journal* 25 (2000), Nr. 11, S. 120–126

[Bro92] Brown, Lisa G.: A survey of image registration techniques. In: *ACM computing surveys (CSUR)* 24 (1992), Nr. 4, S. 325–376

[BSK+13] Bylow, Erik ; Sturm, Jürgen ; Kerl, Christian ; Kahl, Fredrik ; Cremers, Daniel: Direct Camera Pose Tracking and Mapping With Signed Distance Functions. In: *RGB-D Workshop on Advanced Reasoning with Depth Cameras (RGB-D 2013)*, 2013

[col11] *Kinect Color - Depth Camera Calibration.* `http://cv4mar.blogspot.de/2011_03_01_archive.html`. Version: März 2011

[EEH+11] Engelhard, Nikolas ; Endres, Felix ; Hess, Jürgen ; Sturm, Jürgen ; Burgard, Wolfram: Real-time 3D visual SLAM with a handheld RGB-D camera. In: *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden* Bd. 180, 2011

[EHE+12] Endres, Felix ; Hess, Jürgen ; Engelhard, Nikolas ; Sturm, Jürgen ; Cremers, Daniel ; Burgard, Wolfram: An evaluation of the RGB-D SLAM system. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on* IEEE, 2012, S. 1691–1696

[EHS$^+$14] Endres, F. ; Hess, J. ; Sturm, J. ; Cremers, D. ; Burgard, W.: 3D Mapping with an RGB-D Camera. In: *IEEE Transactions on Robotics* 30 (2014), Feb, Nr. 1, S. 177–187

[Elf89] Elfes, Alberto: Using occupancy grids for mobile robot perception and navigation. In: *Computer* 22 (1989), Nr. 6, S. 46–57

[Eng11] Engel, J.: *Autonomous Camera-Based Navigation of a Quadrocopter.* Germany, Technical University Munich, Diplomarbeit, Dec. 2011

[ESC14] Engel, J. ; Schöps, T. ; Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In: *European Conference on Computer Vision (ECCV)*, 2014

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395

[Fre13] Freiling: *RANSAC Point Cloud Alignment.* http://users.csc.calpoly.edu/ zwood/teaching/csc570/final13/freiling/, 2013

[GJ$^+$10] Guennebaud, Gaël ; Jacob, Benoît u. a.: *Eigen v3.* http://eigen.tuxfamily.org, 2010

[GKSB10] Grisetti, Giorgio ; Kummerle, Rainer ; Stachniss, Cyrill ; Burgard, Wolfram: A tutorial on graph-based SLAM. In: *Intelligent Transportation Systems Magazine, IEEE* 2 (2010), Nr. 4, S. 31–43

[GY03] Greenspan, Michael ; Yurick, Mike: Approximate kd tree search for efficient ICP. In: *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on* IEEE, 2003, S. 442–448

[HBH$^+$11] Huang, Albert S. ; Bachrach, Abraham ; Henry, Peter ; Krainin, Michael ; Maturana, Daniel ; Fox, Dieter ; Roy, Nicholas: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: *International Symposium on Robotics Research (ISRR)*, 2011, S. 1–16

[HBM12] Huang, Albert S. ; Bachrach, Abraham ; Maturana, Daniel: *Fast Odometry from VISion.* https://code.google.com/p/fovis/, 2012

[HIG02] Hirschmuller, Heiko ; Innocent, Peter R. ; Garibaldi, Jonathan M.: Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In: *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on* Bd. 2 IEEE, 2002, S. 1099–1104

[HKH$^+$10] Henry, Peter ; Krainin, Michael ; Herbst, Evan ; Ren, Xiaofeng ; Fox, Dieter: RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: *In the 12th International Symposium on Experimental Robotics (ISER* Citeseer, 2010

[Hor87] Horn, Berthold K.: Closed-form solution of absolute orientation using unit quaternions. In: *JOSA A* 4 (1987), Nr. 4, S. 629–642

[HWB$^+$13] Hornung, Armin ; Wurm, Kai M. ; Bennewitz, Maren ; Stachniss, Cyrill ; Burgard, Wolfram: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. In: *Autonomous Robots* (2013). `http://dx.doi.org/10.1007/s10514-012-9321-0`. – DOI 10.1007/s10514–012–9321–0. – Software available at `http://octomap.github.com`

[HZ04] Hartley, R. I. ; Zisserman, A.: *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004

[IKH$^+$11] Izadi, Shahram ; Kim, David ; Hilliges, Otmar ; Molyneaux, David ; Newcombe, Richard ; Kohli, Pushmeet ; Shotton, Jamie ; Hodges, Steve ; Freeman, Dustin ; Davison, Andrew u. a.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology* ACM, 2011, S. 559–568

[KGS$^+$11] Kummerle, Rainer ; Grisetti, Giorgio ; Strasdat, Hauke ; Konolige, Kurt ; Burgard, Wolfram: g 2 o: A general framework for graph optimization. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on* IEEE, 2011, S. 3607–3613

[KM07] Klein, Georg ; Murray, David: Parallel tracking and mapping for small AR workspaces. In: *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on* IEEE, 2007, S. 225–234

[KSC13a] Kerl, C. ; Sturm, J. ; Cremers, D.: Dense Visual SLAM for RGB-D Cameras. In: *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013

[KSC13b] Kerl, C. ; Sturm, J. ; Cremers, D.: Robust Odometry Estimation for RGB-D Cameras. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013

[LM12] Li, Mingyang ; Mourikis, Anastasios I.: Improving the accuracy of EKF-based visual-inertial odometry. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on* IEEE, 2012, S. 828–835

[LM13] Li, Mingyang ; Mourikis, Anastasios I.: 3-D motion estimation and online temporal calibration for camera-IMU systems. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on* IEEE, 2013, S. 5709–5716

[Low99] Lowe, David G.: Object recognition from local scale-invariant features. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* Bd. 2 Ieee, 1999, S. 1150–1157

[MCM07] Maimone, Mark W. ; Cheng, Yang ; Matthies, Larry: Two years of Visual Odometry on the Mars Exploration Rovers. In: *J. Field Robotics* 24 (2007), Nr. 3, S. 169–186

[Mea82] Meagher, Donald: Geometric modeling using octree encoding. In: *Computer graphics and image processing* 19 (1982), Nr. 2, S. 129–147

[ML09] Muja, Marius ; Lowe, David G.: *Flann, fast library for approximate nearest neighbors.* 2009

[MSS09] Myronenko, Andriy ; Song, Xubo ; Sahn, DavidJ.: Maximum Likelihood Motion Estimation in 3D Echocardiography through Non-rigid Registration in Spherical Coordinates. In: Ayache, Nicholas (Hrsg.) ; Delingette, HervÃ© (Hrsg.) ; Sermesant, Maxime (Hrsg.): *Functional Imaging and Modeling of the Heart* Bd. 5528. Springer Berlin Heidelberg, 2009. – ISBN 978–3–642–01931–9, S. 427–436

[N$^+$06] Nüchter, Andreas u. a.: *Semantische dreidimensionale Karten für autonome mobile Roboter.* Aka, 2006

[N$^+$07] NEUHAUS, Frank u. a.: A Full 2D/3D GraphSLAM System for Globally Consistent Mapping based on Manifolds. (2007)

[NLD11] NEWCOMBE, Richard A. ; LOVEGROVE, Steven J. ; DAVISON, Andrew J.: DTAM: Dense tracking and mapping in real-time. In: *Computer Vision (ICCV), 2011 IEEE International Conference on* IEEE, 2011, S. 2320–2327

[Nüc09] NÜCHTER, A.: *3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom.* Springer, 2009 (Springer Tracts in Advanced Robotics 52)

[oct14] *Microsoft Kinect.* `http://en.wikipedia.org/wiki/Octree`. Version: Oktober 2014

[Pel11] PELLENZ, Johannes: *Aktive Sensorik für autonome mobile Systeme.* Der Andere Verlag, 2011

[QCG$^+$09] QUIGLEY, Morgan ; CONLEY, Ken ; GERKEY, Brian ; FAUST, Josh ; FOOTE, Tully ; LEIBS, Jeremy ; WHEELER, Rob ; NG, Andrew Y.: ROS: an open-source Robot Operating System. In: *ICRA workshop on open source software* Bd. 3, 2009, S. 5

[RC11] RUSU, Radu B. ; COUSINS, Steve: 3d is here: Point cloud library (pcl). In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on* IEEE, 2011, S. 1–4

[RD06] ROSTEN, Edward ; DRUMMOND, Tom: Machine learning for high-speed corner detection. In: *European Conference on Computer Vision* Bd. 1, 2006, 430–443

[RPD10] ROSTEN, Edward ; PORTER, Reid ; DRUMMOND, Tom: FASTER and better: A machine learning approach to corner detection. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 32 (2010), 105–119. `http://dx.doi.org/10.1109/TPAMI.2008.275`. – DOI 10.1109/TPAMI.2008.275

[SEE$^+$12] STURM, J. ; ENGELHARD, N. ; ENDRES, F. ; BURGARD, W. ; CREMERS, D.: A Benchmark for the Evaluation of RGB-D SLAM Systems. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012

[SJP13] SMISEK, Jan ; JANCOSEK, Michal ; PAJDLA, Tomas: 3D with Kinect. In: *Consumer Depth Cameras for Computer Vision.* Springer, 2013, S. 3–25

[SLL01] Siek, Jeremy G. ; Lee, Lie-Quan ; Lumsdaine, Andrew: *Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education, 2001

[SSC11] Steinbrucker, F ; Sturm, Jürgen ; Cremers, Daniel: Real-time visual odometry from dense RGB-D images. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* IEEE, 2011, S. 719–722

[SSC14] Steinbruecker, F. ; Sturm, J. ; Cremers, D.: Volumetric 3D Mapping in Real-Time on a CPU. In: *Int. Conf. on Robotics and Automation*. Hongkong, China, 2014

[Stu13] Sturm, J.: *Dense Reconstruction*. `http://vision.in.tum.de/_media/teaching/ss2013/visnav2013/lecture8_dense_reconstruction.pdf`. Version: 2013

[Ume91] Umeyama, Shinji: Least-squares estimation of transformation parameters between two point patterns. In: *IEEE Transactions on pattern analysis and machine intelligence* 13 (1991), Nr. 4, S. 376–380

[WS11] Weiss, Stephan ; Siegwart, Roland: Real-time metric state estimation for modular vision-inertial systems. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on* IEEE, 2011, S. 4531–4537

[Yan14] Yaniv, Ziv: *Point based Rigid Registration*. http://yanivresearch.info/educationalMaterial.html, 2014

[Yua13] Yuan, Eric: *SIFT Image*. http://eric-yuan.me/sift/, 2013

[ZHA92] ZHANG, Zhengyou: *Iterative Point Matching for Registration of Free-form Curves*. 1992

[Zha00] Zhang, Zhengyou: A flexible new technique for camera calibration. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22 (2000), Nr. 11, S. 1330–1334