ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Rapid Point Cloud Generation Utilizing a Kinect and Vicon Data

Spring Term 2016

Supervised by:
Zachary Taylor
Alexander Millane

Author:
Philipp Egger

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Rapid Point Cloud Generation Utilizing a Kinect and Vicon Data

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Philipp	Egger

Student supervisor(s)

Zachary	Taylor
Alexander	Millane

Supervising lecturer

Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

In this semester thesis a fast and flexible approach for acquisition of ground truth 3D maps is presented. It relies on a low-cost rgb-d sensor and a motion capture system. Through a marker-based calibration approach the connection between the sensor and the tracking system is established. Instead of working with point clouds our implementation relies on probabilistic map representations such as Octomap and the Truncated Signed Distance Function (TSDF). Resulting scans are finally evaluated against Lidar scans, which to date serve as ground truth.

Symbols

Symbols

${}_AT_{AB}$	Transformation between frames A and B as seen from A
${}_At_{AB}$	Translation between frames A and B as seen from A
${}_AR_{AB}$	Rotation between frames A and B as seen from A

Indices

x	x axis
y	y axis

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
ASL	Autonomous Systems Lab
UAV	Unmanned Aerial Vehicle
TSDF	Truncated Signed Distance Function
ICP	Iterative Closest Point
SLAM	Simultaneous Localization and Mapping

Chapter 1

Introduction

For evaluation of algorithms producing 3D models such as those utilized by quadrotors for navigation, an accurate model of the same area needs to be available, as a comparison to the real environment is not possible. These precise scans, so called ground truth, are then assumed to reflect the real environment in all it's details. The 3D representation in question is then compared to this true model by calcuating the distance of each point to the true surface / closest point. This allows for having a measure of quality for 3D models. Obviously this is only a reliable measure if the model is very close to the real world. Furthermore it is very important that the ground truth is available everywhere, where there is data for comparison. If, for instance, a ground truth scan does not cover the area beneath a table, but the model in question does, the result is biased.
Currently ground truth is provided by scanning the area with a Lidar scanner. While being highly accurate, each scan takes around half an hour. In order to minimize uncovered areas, scans from many different viewpoints have to be combined. Therefore making a scan of a small area takes up several hours and mapping a complete office room with chairs and desks, is nearly impossible. This project aims at making this process more efficient. By using a hand-held depth sensor like the Microsoft Kinect and a motion capture system to track the sensor we want to provide a fast tool to generate ground truth maps of the environment. In fig. 1.1 a scan made with our system is presented.

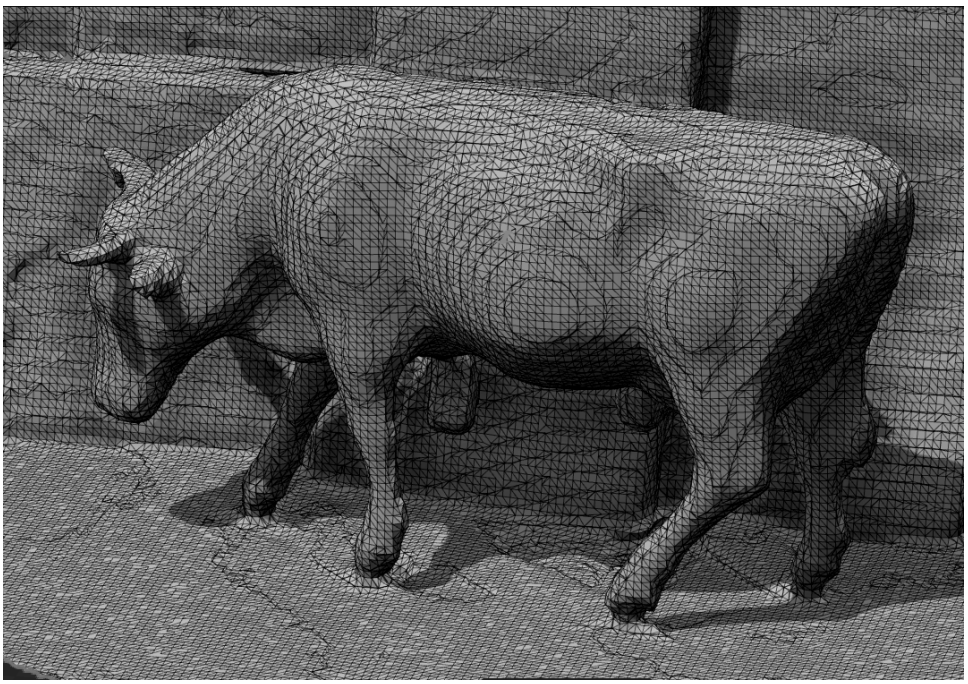

Figure 1.1: Example of a 3D model.

Chapter 2

Background

2.1 Related Work

For a long time equipment for acquiring 3D data for scanning purposes was rather expensive and not available to a wide audience. This changed when Microsoft launched the Kinect in 2010 for just 149 Euros. It wasn't long until first attempts to use it as a hand-held 3D scanner started. KinectFusion [1] was the first published work to present a high-quality scanning system using the Kinect. It featured two main parts: *A map building process* that continuously adds the information from every new scan into a global map representation. For this purpose they made use of the Truncated Signed Distance Function (TSDF)(see section 2.5.1) that allows averaging over multiple scans. However, in order to fuse the incoming data into the global map, the position of the scanner must be known at all times. Therefore a second process, the *sensor pose estimation* must be run simultaneously. This process tries to fit every new scan onto the continously built global map by using a multi-scale iterative closest point (ICP) algorithm. In this manner, it is estimated where, relative to the global map, the camera currently is.

KinectFusion was able to produce some very good scans (e.g. fig. 2.1), but it only worked in a predefined, limited area. Whelan et al. [2], further improved the approach by dynamically allocating space for mapping. Furthermore areas that were far away from the current sensor-position, like a previously scanned room, were no longer stored as TSDF but exported to dense point clouds. This allowed for scanning larger scenes, as the representation with a TSDF is very memory-intensive, with the limitation that no new information could be added to these exported areas. However, like any system relying on simultaneous localization and mapping (SLAM) the localization is never perfect and suffers from problems like drift. As the map building process relies on the position estimate to be exact, it adds data to the wrong position, if this estimate is incorrect. As result more or less small distortions in the maps occur. Areas that have little features, like flat walls, are very likely to produce errors in the pose estimation.

This is where our approach has some clear advantages. We still rely on similar methods for building maps, but, by having the luxury of a motion capture (MoCap) system that provides us with very precise position data, we can fully focus on the mapping process. Common SLAM problems like drift are taken care of. Furthermore all scans are in the coordinate frame of the MoCap System and can therefore be extended and combined any time.

Sturm et al. [3] already tracked a Kinect with a MoCap system. However, it was only used to provide a common measure for evaluating performance of different SLAM algorithms and not for mapping purposes. Nevertheless, the setup could be

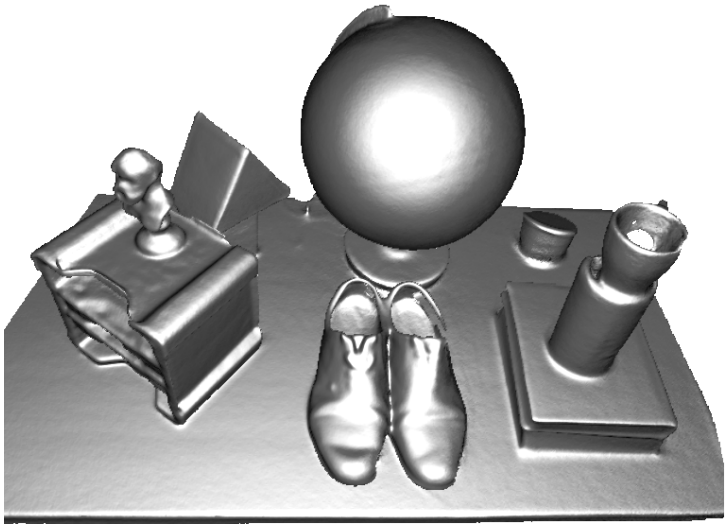

Figure 2.1: Scan from KinectFusion. [1]

expanded to serve as a mapping tool.
Previous to this work, a semester project [4] was already undertaken with the same goal of produsing high quality scans with a MoCap System and a Kinect. However, the resulting scans were very noisy and overall of much worse accuracy than a Lidar scan. We decided to take different approaches in two main points. Firstly we rely on a *probabilistic*, instead of a deterministic representation of the global map. This should help eliminating sensor noise from the fairly cheap Kinect. Furthermore we calibrate the Kinect against the MoCap System in a *marker-based* fashion whereas Hottiger [4] used a SLAM-based approach. (comparison: section 2.2)

2.2 Marker-based vs SLAM-based calibration

The main accomplishment of this project is building a connection between Vicon and Kinect to work together. In order to do so, it is essential to know where the Sensor is in relation to the Vicon system. There are to main approaches to close this gap.

- The *SLAM-based* approach calculates a trajectory of the IR-Camera's Frame by using the position estimates gained through ICP search. This trajectory then is compared to the trajectory of the vicon tracking and, by making use of the fact that the relation between the two frames is time-invariant, the transformation can be calculated. The main advantage of this procedure is, that no additional hardware is required. However, common SLAM-related issues can affect the result.

- The *marker-based* approach on the other hand requires an intermediate that can be detected by both hardware parts. Like this a connection between the prior separated systems is made.

We decided to go with the marker-based method for two reasons:

1. The marker-based calibration is very intuitive and therefore allows tracking down errors more easily.

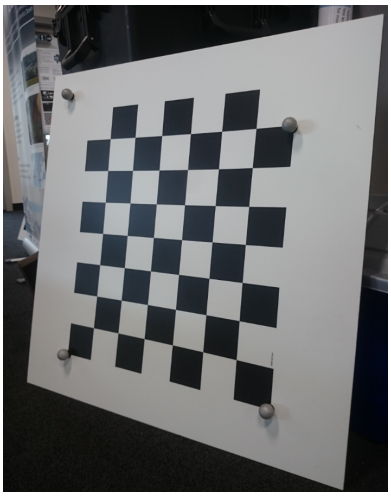

Figure 2.2: Building the connection between Kinect and Vicon: Chessboard with Vicon-Markers.

2. Hottiger [4] used a SLAM-based approach in his thesis and pointed out that misalignment errors in the final point clouds probably are due to errors in the extrinsic calibration.

As connecting medium we used the a chessboard with Vicon markers. (fig. 2.2) The same kind of connecting device already proofed to give accurate results for the SLAM-benchmark by Sturm et al. [3].

2.3 Sensor

For this project we rely on the original Kinect but want to give a quick overview on other affordable sensors that could be used for the same purpose:

- *The Asus Xtion Pro Live* has very similar specifications as the Kinect v1. The only obvious advantages of this sensor is that it is lighter and does not need an additional power supply like the Kinect v1 and therefore makes moving around the room and scanning easier.
- *Intel's RealSense* is even more portable. However, it was monstly developed to perfom well at short distances like scanning people in front of the computer. For our project good quality scans at distances beyond 1.5m are desireable.
- *The Kinect One* is twice as heavy and also bigger than the Kinect v1. However, Fankhauser et al. [5] showed that it has overall improved performance. Especially larger distances of up to 3m still give little errors. The 15W power consumption demands for an additional power supply

2.4 Motion Capture System

At the Autonomous Systems Lab, there are two rooms equipped with a Vicon system available. Vicon is a vision based tracking system. It uses Infrared Light sources and cameras to track small reflective balls in space. At least three of these markers, fixed onto a rigid body, allow for a complete determination of the position and orientation in space. Any additional markers improve the estimation through

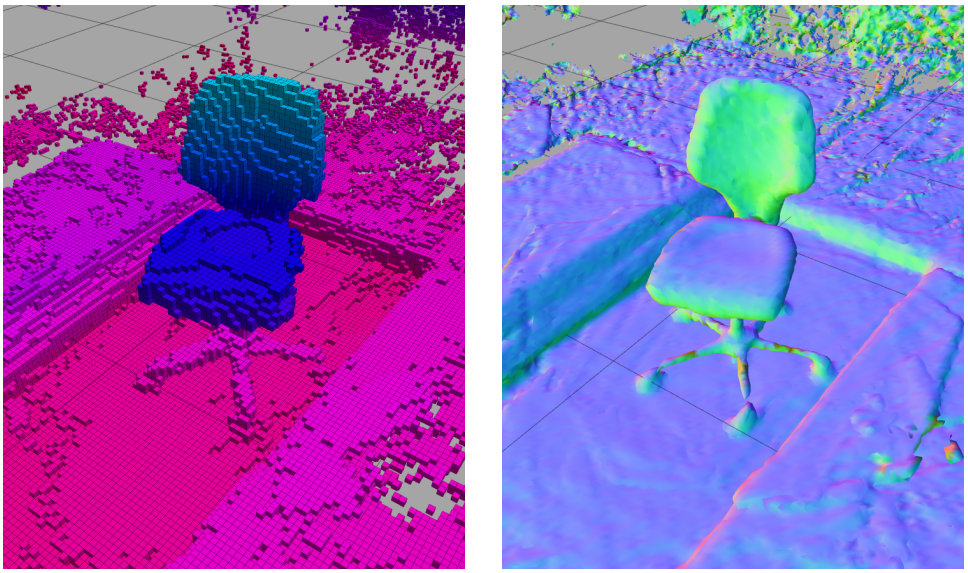

Figure 2.3: A chair represented with the two methods. In the left Picture the occupied voxels of the Octomap are shown. On the right is an image of the mesh created with the TSDF.

least-square minimization and allow for tracking when one or more markers are covered.

2.5 Map Representation

The project relies on two different map representations: *TSDF* and *Octomap*. Figure 2.3 shows an Octomap and TSDF of the same dataset. They are quite different in nature and have different strengths. Nonetheless, they have two very important properties in common:

1. They are dynamic, meaning, that the map is expandable in all directions as information on new areas is added.

2. They represent the world in a probabilistic manner.

We believe that the second point is absolutely essential when using an affordable sensor like the Kinect, as noise is filtered out by capturing sufficient data. Whereas, with a Lidar scanner on the other Hand, it makes sense to use the discrete point cloud directly as is of very high quality.

2.5.1 Truncated Signed Distance Function (TSDF)

When a new surface is scanned, the TSDF splits the space in front and behind a surface scan into volumetric cubes, so called voxels. It then calculates for each voxel the distance to the surface and assigns this value to the voxel. Voxels behind the surface are assigned the negative value of their distance. The truncation distance defines the maximum distance for which this values are calculated. Figure 2.4 illustrates this very clear. Any consecutive scan of the same surface adds new values to the voxels and helps eliminating noise. The actual surface is described implicitly and can be restored by calculating for the zero-crossing between the voxels.

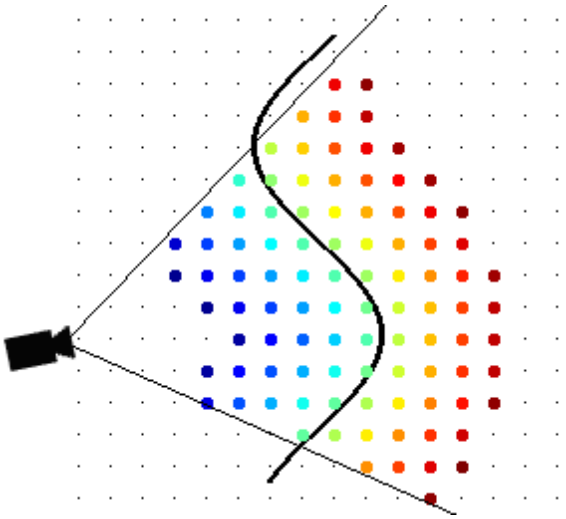

Figure 2.4: Working Principle of the TSDF. [6]

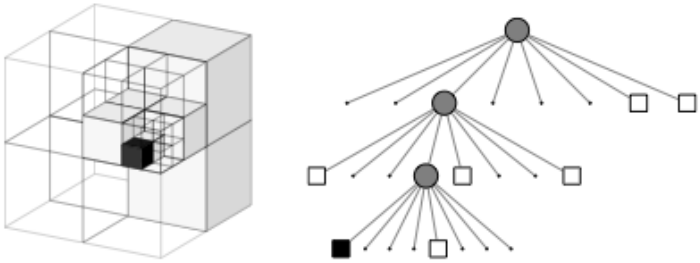

Figure 2.5: A small Octomap example with the corresponding tree-structure. [7]

2.5.2 Octomap

Octomap also discretizes the 3D space into voxels - not only close to surfaces but also further away. Each voxel is described by a likelihood of being occupied. A threshold then splits the voxels into either free space or occupied cells. When a surface is scanned all the voxels between the surface and the scanner decrease in likelihood and the voxels on the detected surface become more likely to be occupied. This probabilistic aspect, again, helps filtering any sensor noise. Furthermore the voxels are internally saved as octrees: Eight voxels form together a higher-level voxel and this is continued recursively. Figure 2.5 illustrates this aspect well. This allows for querying different level of details as needed for the application. An UAV for instance needs a precise map for localization but is not interested in free space with less then 0.5m in diameter for pathplanning. Furthermore big homogeneous volumes, like an empty space, take very little memory as the branches, which wouldn't contain any additional information, are not stored. Octomap is much more memory-efficient than the TSDF representation, but it does not calculate for the actual surface as walls are often thicker than one voxel.

Chapter 3

Method

3.1 Overview

In order to produce 3D maps with the Kinect and the Vicon System a couple of steps are needed. Figure 3.1 tries to give a clear overview on how each part fits in with the rest. The Hardware components are depicted in blue and the end-result in black.
The Kinect provides us with a raw depth image. This data is then converted into a point cloud using a sensor-specific *intrinsic calibration*. This cloud reflects the surface as seen from the infrared camera. In other words it is in the infrared camera's coordinate frame. In order to be able to match the points to those gathered from a different sensor position we need to transform each point of every scan into a global coordinate frame. This is were the Vicon comes into play by providing for every scan the position of the Kinect. Unfortunately this is not yet the exact position of the infrared camera, but instead these two frames are connected through the *extrinsic calibration*. Accordingly the Vicon data and the extrinsic calibration together transform the pointcloud from the camera's frame to the global frame. Once in the global frame, the new information of every scan can be added to the *global map*.
The intrinsic calibration is sensor specific and only has to be calculated once for a sensor. The Extrinsic calibration depends on the room and needs to be recalculated when moved to a new testbed.

3.2 Intrinsic Calibration

For the intrinsic calibration we used the ROS-wrapper of the OpenCV monocular camera calibration.[8] In order to avoid any interference with the chessboard detection we blocked the IR-Emitter with black tape. In fig. 3.2 you can see the interface of the tool. By looking at the live image one can directly see where in the camera's image the chessboard currently is. By moving the chessboard to all sides and closer and further away from the camera the bars slowly fill up. When finished, the calibration data is directly stored in the camera's driver. Not only the focal length in x and y direction are calculated but also the correct principal point and distortion parameters.
We used a 6x7 Chessboard with 7cm square-size. At the beginning results varied significantly between runs. It was found that this was due to insufficient infrared ambient light. By moving closer to a window finally a fairly good repeatability was achieved.

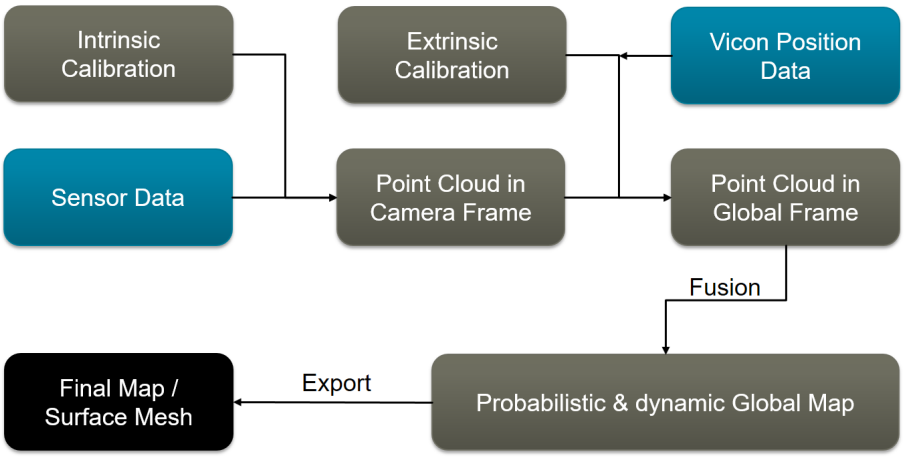

Figure 3.1: Overview on the system. The two hardware parts involved are depicted in blue: Kinect and Vicon.

Figure 3.2: Interface of the monocular camera calibration tool. [8]

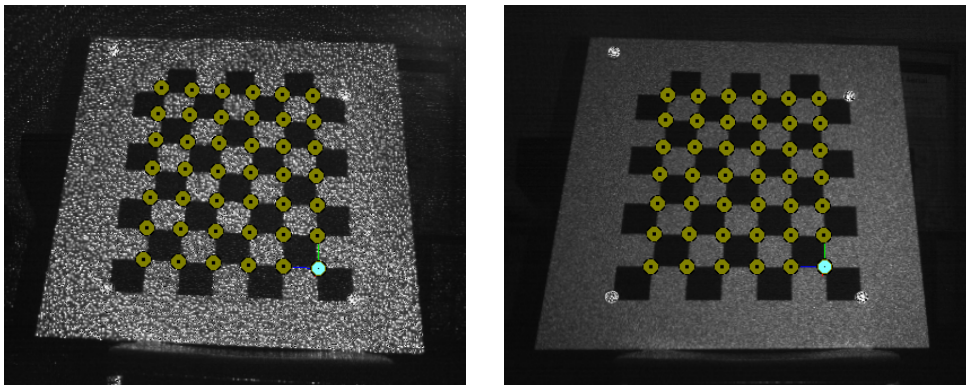

Figure 3.3: No tape vs dim tape on the IR-Emitter.

Instead of fully blocking the IR-Emitter it can also be blurred by using dim, transparent tape. In this manner, we can still take advantage of the light source while minimizing distortions through the structured light pattern. The improvement over the direct IR pattern can be clearly seen in fig. 3.3.

3.3 Extrinsic Calibration

The extrinsic calibration builds the connection between the Kinect and the world, in our case the Vicon system. As can be seen in fig. 3.4, our problem features two coordinate frames that are rigidly connected to the Kinect. Denoted with "I" is the IR-cameras frame from where we take images. The coordinate frame "S" is somewhere on the Kinect and is being tracked by the Vicon system. Therefore we know the position and orientation of "S" relative to the world coordinate frame "W". In order to know where "I" is relatively to the the world frame we need to know the time-invariant connection between S and I. This problem would be non-existent if we could just exactly place S onto I in the Vicon system. However, in the interface we only know where the markers are and have no chance to exactly place S at the correct position. Therefore a calibration process is needed in order to calculate for the transformation between the two coordinate frames.

3.3.1 Coordinate Frames

In this section we would like to give the reader a clear overview on the different coordinate frames involved in the calibration process. Figure 3.5 shows all frames involved and the known connections between them. The Indices are as follows:

W	World/Vicon Frame
I	IR-Camera Frame
S	Sensor Frame: Frame that is tracked by the Vicon system.
C	Chessboard Frame: The Frame that is tracked by the Vicon System.
D	Detected Frame: The point which is returned from the chessboard detection.

The Vicon system provides us with information of the position and rotation of C and S relative to the Vicon Frame W. In other words it gives us the Transformations ${}_WT_{WS}$ and ${}_WT_{WC}$. The chessboard detection calculates the Transform of D relative to I (${}_IT_{ID}$).

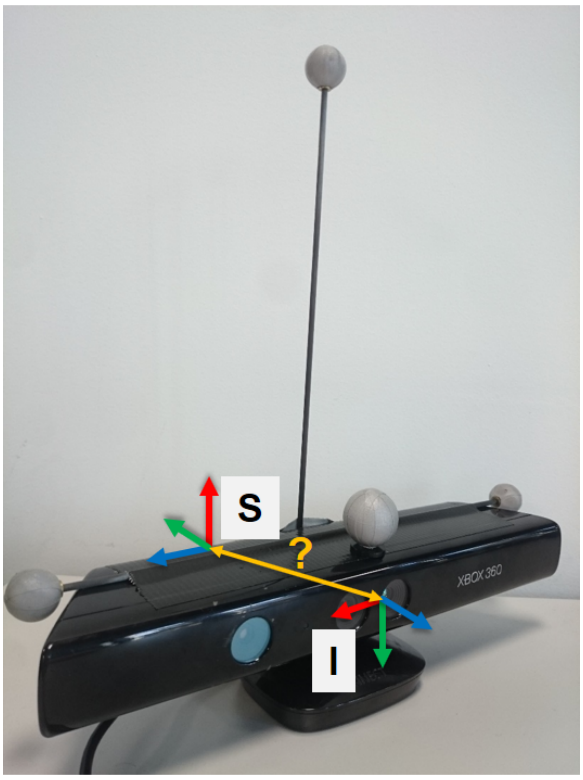

Figure 3.4: The coordinate frames on the Kinect Sensor.

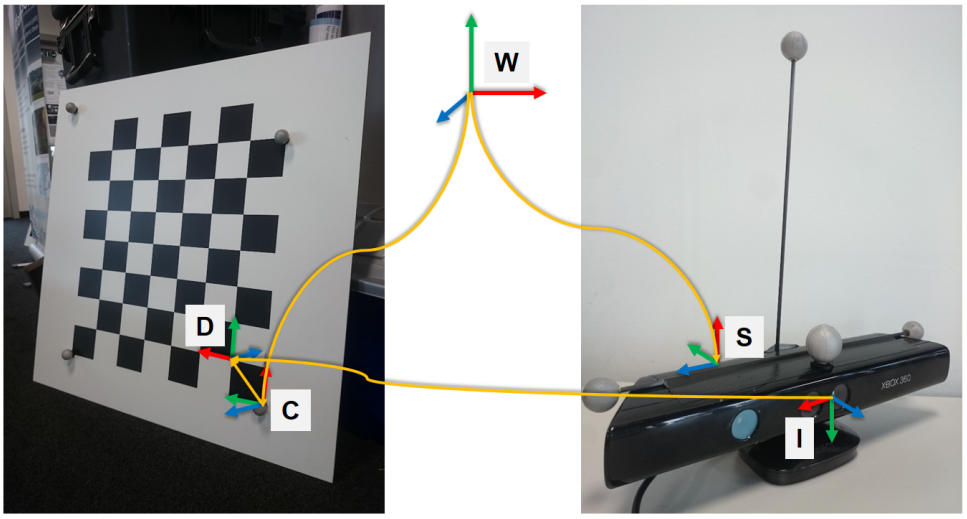

Figure 3.5: An Overview over all relevant coordinate frames involved in the calibration process.

Since we placed the Vicon markers exactly in the edge of the Chessboard we were able to initialize the Chessboard frame at this position as well. Therefore we know the Transform between C and D to be:

$$ {}_{C}T_{CD} = \begin{bmatrix} 0 & 1 & 0 & 0.07 \\ 1 & 0 & 0 & 0.07 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

3.3.2 Calculating the Transform

From section 3.3.1 we already know all the orange Transformations from fig. 3.5.

$$ Given: \quad {}_{W}T_{WS},\ {}_{W}T_{WC},\ {}_{I}T_{ID},\ {}_{C}T_{CD} \tag{3.2} $$

In order to transform the point clouds from the IR-cameras frame to the world frame we need to know where the camera is.

$$ {}_{W}T_{WI} = ? \ = \ {}_{W}T_{WS} *_{S} T_{SI} \tag{3.3} $$

${}_{W}T_{WS}$ is given as the inverse of ${}_{S}T_{SW}$ and we can express ${}_{S}T_{SI}$ with other known transformations:

$$ {}_{S}T_{SI} =_{S} T_{SW} *_{W} T_{WI} = ({}_{W}T_{WS})^{-1} * ({}_{W}T_{WC} *_{C} T_{CD} * ({}_{I}T_{ID})^{-1}) \tag{3.4} $$

This finally provides us the prior unknown connection between the Kinect and the Vicon system.

3.3.3 Averaging Transforms

In order to filter out sensor noise and errors in the chessboard detection we moved the chessboard within the field of view of the camera and averaged ${}_{S}T_{SI}$ over all frames. It is necessary to look at the rotation (denoted as R) and translation (denoted as t) separately. For averaging the translation, calculating the mean value is sufficient:

$$ \bar{t} = \frac{1}{n} * \sum_{i=1}^{n} t_i \tag{3.5} $$

For the rotation, on the other hand, it is not so obvious. Taking the mean over all rotation quaternions does not return a unit vector. However, as it is very close to a unit vector, normalization of the average already yields a good estimate.
A mathematically more correct approach is calculation for the centroid of all quaternions in 4D.

$$ \bar{R} = exp(\bar{v}) * R_0 \tag{3.6} $$

with

$$ \bar{v} = \frac{1}{n} * \sum_{i=1}^{n} t_i \tag{3.7} $$

and

$$ v_i = log(R_0 * Ri^{-1}) \tag{3.8} $$

Equ. (3.6) and (3.8) are also known as box-plus/box-minus functions and *exp* and *log* refer to the exponential/logarithmic map, respectively. Further information can be found in the kindr documentation [9].

3.4 Mapping

After having fully calibrated the system we are able to collect data, fuse it together and build global maps. For building these maps we rely on Octomap and TSDF. (see section 2.5 for an overview on the methods) It is possible to build only one kind of map at a time or both simultaneously. The map can be built live or from previously collected data.

Chapter 4

Implementation

4.1 ROS

The whole implementation is based on ROS, a flexible framework for software development in robotics. The main benefit of using ROS for the project, is the modularity and the visualization tools. The software is split up into different *nodes* that interact through *topics* and *services*. In this Project ROS-Indigo was used. For further information visit `www.ros.org`.

4.2 OpenCV

OpenCV is an Open-Source computer vision library and was used in various parts of the project. For further information visit `www.opencv.org`.

4.3 Point Cloud Library (PCL)

Parts of our code rely on the Point Cloud Library (PCL) for feature extraction and conversion. For further information visit `www.pointclouds.org`.

4.4 Kinect Driver

In order to receive data from the Kinect we used the OpenNI driver.[10] A quick overview on the relevant topics:

- `\camera\ir\image_raw`

  In this topic the raw IR-image is being published. It is used to calibrate the camera intrinsics.

- `\camera\ir\image_rect_ir`

  On this channel a rectified version of the IR-image is published using the intrinsic calibration data. This topic is used for the chessboard detection in the extrinsic calibration process.

- `\depth\points`

  In this topic a point cloud of the depth scan is published based on the intrinsic camera parameters. The point cloud is in the cameras frame. This topic is used for the actual mapping process.

4.5 Camera Calibration

For calibrating the intrinsic parameters of the camera we made use of the readily available OpenCV monocular camera calibration. [8] The chessboard and the image-source is specified in the launch file prior to the calibration. As we were interested in the depth information we used the image stream from the infrared camera. When finished, the calibration node hands the calculated calibration data directly to the OpenNI driver via a service call (setcamerinfo) where it is stored permanently. Therefore intrinsic calibration only needs to be done once per scanning device.

4.6 Chessboard Detection

In order to detect and estimate the position of the chessboard for our extrinsic calibration we used a ROS-Wrapper [11] for the openCV checkerboard detection. This node automatically detects any chessboards within an image-stream and calculates for the position of it by knowing the chessboard size and the camera intrinsic parameters.

4.7 Transformations

For dealing with transformations ROS-TF as well as minkindr, a lightweight version of kinematics and dynamics for robotics (kindr) [9] were used. While all the calculations were done using minkindr, tf was still used for visualization purposes in rviz. Rviz is a visualization tool that comes with the standard ROS distribution.

4.8 Mapping

For building the octomap the volumetric mapping package from asl was used. Voxblox, another asl package, served for calculating the TSDF. Voxblox is still at an early stage in development, as the time of writing but it runs stable.

4.9 Nodes

In this section we want to give the reader an overview on which nodes are being used in the main scenarios, which are: intrinsic calibration, extrinsic calibration and mapping.
For the *intrinsic calibration* only the OpenNI interface and the monocular calibration nodes are needed.
Figure 4.1 shows the interplay of the nodes for the *extrinsic calibration*.The image conversion is needed as the chessboard detector expects a 8bit image and the OpenNI publishes a 16-bit image. In the transform_calc node ${}_{S}T_{SI}$ is calculated.
This transform is then used in the ir_camera_tf node to hand the camera's position over to the map nodes in the *mapping* process. (fig. 4.2)

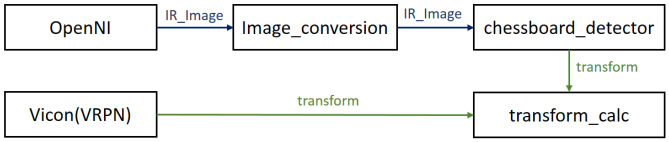

Figure 4.1: Active Nodes for extrinsic calibration.

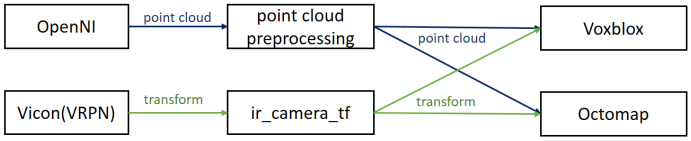

Figure 4.2: Active Nodes in the mapping process.

Chapter 5

Results

5.1 Intrinsic Calibration

The Following intrinsic parameters resulted from the intrinsic calibration process:

$$\text{camera matrix:} \quad \begin{bmatrix} 585.1 & 0 & 323.8 \\ 0 & 583.8 & 244.5 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

$$\text{distortion coefficients:} \quad \begin{bmatrix} -0.0658 & 0.1356 & 0.001829 & 0.0006211 & 0 \end{bmatrix} \tag{5.2}$$

$$\text{rectifiction matrix:} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$\text{projection matrix:} \quad \begin{bmatrix} 584.2 & 0 & 323.5 & 0 \\ 0 & 582.1 & 244.7 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.4}$$

We believe that the intrinsic calibration is necessary for good results as the values vary significantly from openNI's standard Kinect parameters:

$$\text{standard projection matrix:} \quad \begin{bmatrix} 575.8 & 0 & 319.5 & 0 \\ 0 & 575.8 & 239.5 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.5}$$

5.2 Extrinsic Calibration

We calculated the transform ${}_S T_{SI}$ while moving the chessboard relative to the camera. Equation (5.6) shows the rotation as euler angles around z,x',z". It can be seen that the uncertainty is very small with a standard deviation of less than 1 degree. Accordingly the rotation estimate is expected to be very accurate. In eq. (5.7) on the other hand the standard deviation is considerably higher with up to 10mm compared to actual values around 30mm's. However, over the scanning distances of 1-3m an uncertainty in the translation of 1cm does not influence the results noticeably. Angular errors over that distance have much larger impact.

$$\text{Euler Rotation } (^\circ): \quad \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 68.8 \\ 152.9 \\ -176.5 \end{bmatrix} \qquad \text{stdDev:} \quad \begin{bmatrix} 0.79 \\ 0.43 \\ 0.73 \end{bmatrix} \tag{5.6}$$

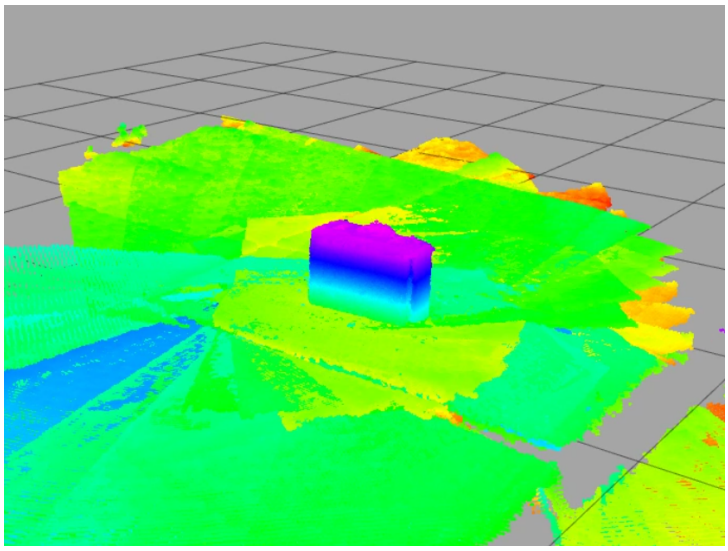

Figure 5.1: Bare pointclouds plotted while scanning a briefcase.

$$\text{Translation (mm):} \quad \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} -30.7 \\ 24.0 \\ -56.2 \end{bmatrix} \qquad \text{stdDev:} \quad \begin{bmatrix} 9.9 \\ 9.7 \\ 5.8 \end{bmatrix} \tag{5.7}$$

5.3 Calibration Verification

Besides being able to produce very accurate scans with the calculated calibration values (see section 5.4), we looked for other measures to verify the results. For visual evaluation we plotted the pointclouds on top of each other while scanning a briefcase. Alignment errors would clearly be visible on the rectangular shape. Figure 5.1 shows the resulting overlay. The quality was not as good as with Octomap or TSDF, as noise was not filtered, but the pointclouds aligned very nicely with no major outliers.
As a second quality measure we scanned the flat floor in the Vicon lab and calculated the angular deviation between the scan and the Vicon coordinate frame. Figure 5.2 shows the results for fast rotational movements of the Kinect. For slower movements the amplitudes relative to the mean values are smaller. The mean values reflect an orientational difference of the physical floor and the world frame. There is various possible sources for the errors. One thing that we noted, is that they increase with rotational speed. Around frames 60 and 170 there were very fast rotational speeds and therefore higher deviations. Possible error sources include:

- Time offset between Kinect and Vicon data.
- Errors in the extrinsic calibration.
- Movement of the Vicon markers due to insufficient fixation.
- Kinect internal errors due to motion blur.

A time-shift comparison was done in a small range of 0.05s with no clear result. Compared were the mean values of the deviation from a given dataset different time-shifts. However, from fig. 5.2 follows, that, in order to account for a offset of the floor, standard deviations should be considered instead. More work could be

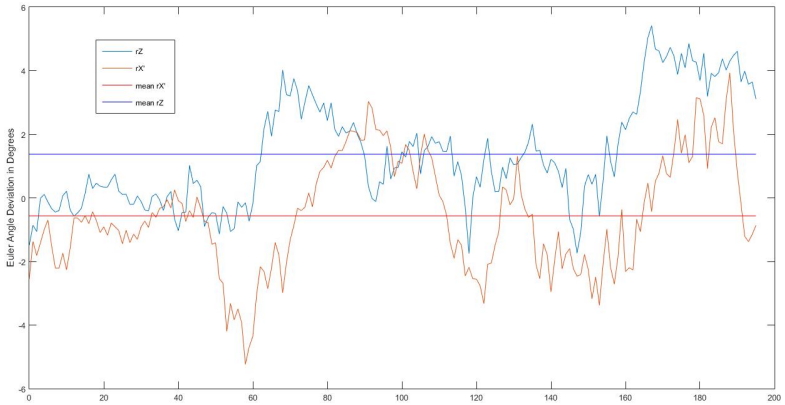

Figure 5.2: Euler deviation from Vicon coordinate frame around the two relevant axes. The mean values reflect the orientation of the floor.

done in this area, though it is important to point out, that in a natural scanning behaviour fast rotations like in fig. 5.2 don't occur and therefore the errors stay small.
Of much higher importance is a rigid fixation of the Vicon markers onto the scanner. There is a notable deviation of up to 5 degrees (without moving the scanner) just by pressing gently against the vertical rod with markers. (see fig. 5.3) The rod was intended to provide better accuracy in pitch. However, insufficient fixation had the opposite effect. We consider a better fixation to be the most important improvement on the current setup.

5.4 3D Scans

To test the capabilities of our system various scans were made. In fig. 2.3 we already presented scanning results of a chair with the two representations in comparison. Figure 5.4 shows an Octomap of the whole Vicon testbed from outside. The holes reflect uncovered areas that could be added subsequently. For our final evaluation served a model of a cow. (see also fig. 5.3) Figure 1.1 in the introduction shows a surface mesh of the cow generated with TSDF. In fig. 5.5 this map is shown as it is built with the TSDF. One thing we noted when capturing the data, is that the Kinect becomes too noisy for our purposes at distances beyond 2.5-3m. Figure 5.6 shows the top left corner of the cow-scan we only collected little data from far away. The map is of very bad quality and useless as ground truth. A sensor with bigger range like the Kinect v2 or filtering points out beyond a certain distance would solve this problem.

5.4.1 Comparison Lidar Scan

Since this project aims at providing a fast tool for ground truth generation, we wanted to quantify the quality against what is used as ground truth to date. We therefore collected point cloud data with a Lidar scanner from three different positions. While, collecting the data of the Lidar scan took 1.5 hours, it only took us less than a minute with our system. For comparing the scans we used CloudCompare, an open source project for cloud and mesh processing. Figure 5.7 shows the two point clouds aligned with ICP. We had to cut down the Kinect scan considerably

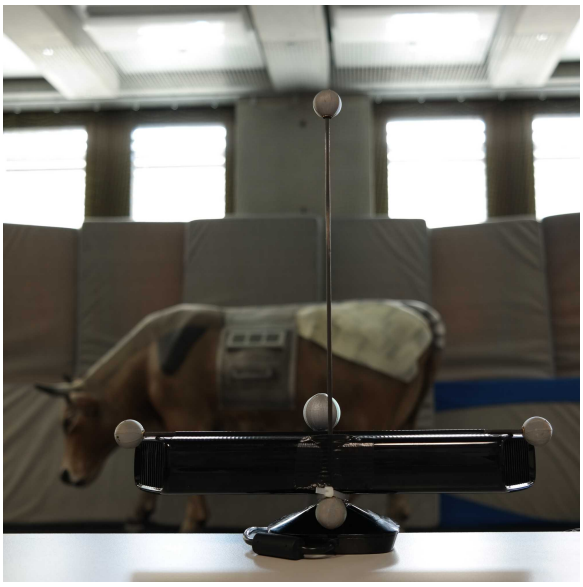

Figure 5.3: Image of the back of the Kinect. The fixation of the vertical markers with tape and cable tie is not rigid enough.

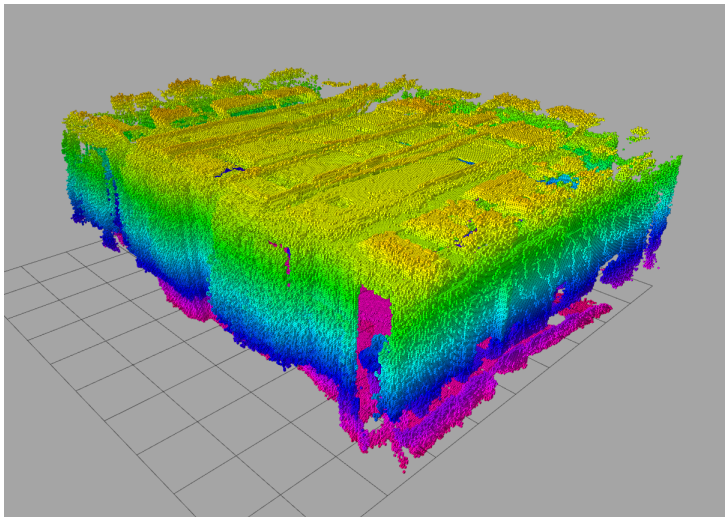

Figure 5.4: An Octomap of the whole testbed seen from outside.

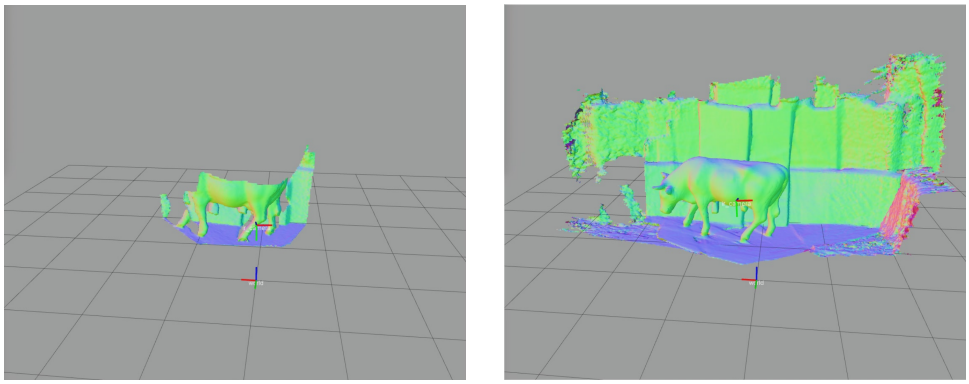

Figure 5.5: Building a surface mesh with TSDF.

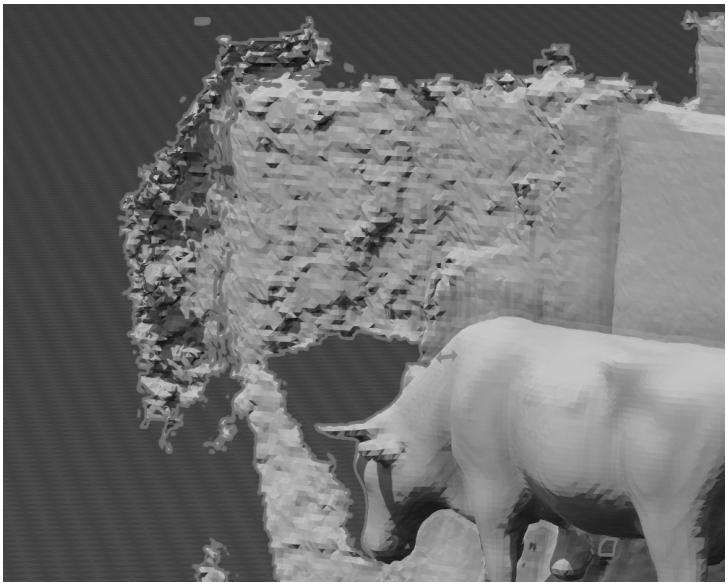

Figure 5.6: Noisy part of the map in the top left corner.

Figure 5.7: Point cloud from the Kinect (white) aligned with the Point cloud from the lidar scans. RMS=13mm

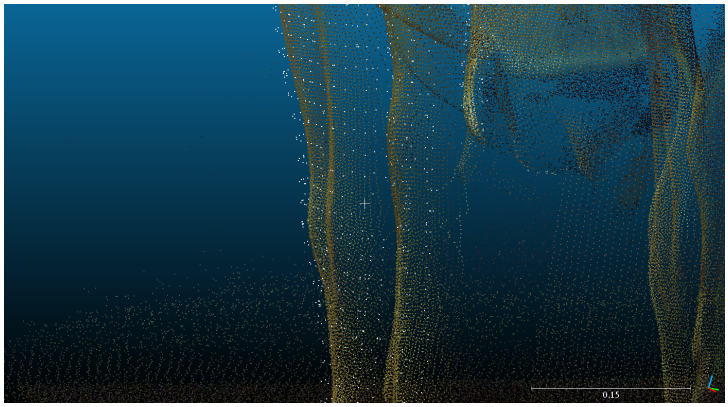

Figure 5.8: Difference of clouds on the leg.

in size, as, within one minute, more area was covered with the Kinect than with the Lidar scans in 1.5 hours. Behind the cow for instance the Lidar point cloud had a much larger hole in the map. When fully aligned, a mean error of 13mm was calculated from our cloud to the one of the Lidar scan. When visually inspecting the alignment, it looked to be much closer. Only on one of the legs a noticable error was found (fig. 5.8). Accordingly the result of 13mm is possibly still dominated by outliers.

5.5 Updating maps

One very useful consequence of using the Vicon frame for the mapping process is the possibilty to fuse new information into a map at any given time. In fig. 5.9 the same dataset as in fig. 5.5 was used to build an octomap of the cow. A day later the cow was removed, a chair added to the scene and new data was fused into the map. While the cow disappears and the chair shows up, the background not only stays the same, but the whole in the back of the cow is also filled. This feature allows for scanning the whole testbed thoroughly once and then update small changes whenever needed.

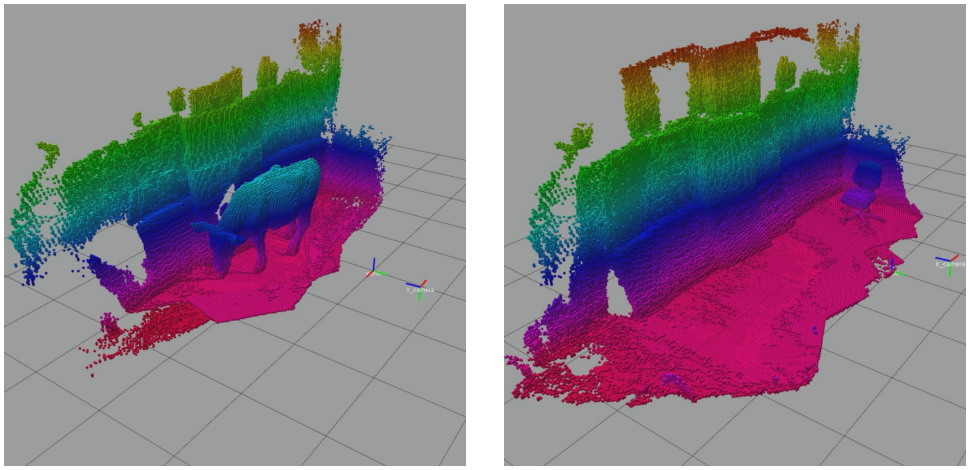

Figure 5.9: Octomap of the cow on the left. New data was added later. The updated map is shown on the right.

Chapter 6

Conclusion

In this report, we presented an approach for 3D ground truth acquisition with a RGB-D sensor and a motion capture system. We showed that our system is able to outperform common methods by both speed and flexibility, while not compromising quality significantly.
Two steps are involved to be able to scan with the setup. First the Intrinsic Parameters of the sensor need to be calibrated. Secondly an extrinsic calibration of the sensor is required in order to build the connection to the motion capture system. We used a marker-based approach to detect a medium both from the sensors camera as well as from the motion capture system. We evaluated the accuracy of the extrinsic calibration by calculating angular deviations of scans relative to a flat floor. Results showed that for natural scanning behavior the calibration is accurate enough, whereas for very fast rotational movements errors become significant.
For actually building maps we rely on Octomap and the Truncated Signed Distance Function. Both are of probabilistic nature and therefore help filtering sensor noise. We believe this is essential to our approach as our sensor is less accurate then normally used Lidar scanners. We showed scans of different scenes in both representations. Furthermore we also demonstrated that parts of stored maps can be updated any time while keeping the previously collected parts.
In order to have a measure of quality, we mapped a reference target both with our system and a Lidar scanner. A mean error of 13mm was calculated between our scan and the reference scan. It is possible that this number is dominated by outliers as visual inspection promised an even better result. While capturing the data with the Lidar scanner took around 1.5h, we only needed one minute with the new setup.
We identified two quality-compromising aspects of our current setup, both of which can be easily fixed in the future. Firstly, markers from the motion capture system need to be attached to the sensor in a more rigid fashion. Secondly, filtering out points outside a certain range before adding them to the global map can further improve scan results.

Bibliography

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

[2] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[3] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[4] G. Hottiger, "High quality terrain scanning," 2014.

[5] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *Advanced Robotics (ICAR), 2015 International Conference on*, July 2015, pp. 388–394.

[6] J. Morgan. (2012) Real-time 3d reconstruction using signed distance functions.

[7] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *In Proc. of the ICRA 2010 workshop*, 2010.

[8] J. Bowman and P. Mihelich. Monocular calibration.

[9] C. D. Bellicoso, M. Bloesch, R. Diethelm, P. Fankhauser, P. Furgale, C. Gehring, and H. Sommer. Kinematics and dynamics for robotics (kindr).

[10] P. Mihelich. Ros openni launch.

[11] R. Diankov. Ros checkerboard detector.