Delft University of Technology
Master's Thesis in Embedded Systems

Improving RGBD Indoor Mapping with IMU data

Bas des Bouvrie

embedded software

Improving RGBD Indoor Mapping with IMU data

THESIS

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

In

Embedded Systems

By

Bas des Bouvrie
born in Rotterdam, the Netherlands

Embedded Software Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.es.ewi.tudelft.nl

Improving RGBD Indoor Mapping with IMU data

Author: Bas des Bouvrie
Email: xilconic@gmail.com

Abstract

With the release of RGBD-cameras (cameras that provide both RGB as well as depth information) researchers have started evaluating how these devices can be used in fields of localization, mapping and ubiquitous computing. Intel Seattle Research proposed an indoor mapping algorithm making use of such a camera. The algorithm itself is vulnerable to violations of the static environment assumption and image based localization failures that can be caused by, for example, featureless environments. The goal of this master thesis is to augment the indoor mapping algorithm with additional Inertial Measurement Unit (IMU) data to enhance the robustness to these vulnerabilities. To this end the characteristics and limitations of the Microsoft Kinect are investigated and an enhanced mapping algorithm is proposed. IMU orientation estimates are fused with pose estimates based on image data, which give an initial guess to the Iterative Closest Point (ICP) algorithm that is used to align point cloud data to create a final map. In case visual localization fails, the algorithm of Intel uses a constant velocity assumption as fallback mechanism while the IMU data provide more accurate orientation estimations than the constant velocity assumption can provide. The IMU-enhanced algorithm shows similar mapping quality in ideal mapping conditions compared to the plain mapping algorithm. While a series of corner case tests show that the IMU-enhanced algorithm was unable to improve the results compared with the plain mapping algorithm, it potentially generates improvements in mapping quality when dealing with non-static environments.

Thesis Committee:

Chair:	Prof. Dr. K.G. Langendoen, Faculty EEMCS, TU Delft
Committee Member:	Dr. S.O. Dulman, Faculty EEMCS, TU Delft
Committee Member:	Dr. E.A. Hendriks, Faculty EEMCS, TU Delft

Acknowledgements

First off, I'd like to thank my friends and family for giving me support and showing patience while I was working on my master thesis.

Even more thanks goes to my girlfriend who gave me much support, love, affection and encouragement in times I really needed it.

I also like to thank Kavitha Mutukrishnan who was able to give me a lot of encouragement and ideas and especially for being available for an advisory role after she had to quit supervising me halfway during the thesis.

I would like to thank Jose-Luis Blanco-Claraco for helping me understand MRPT and help me with issues I was having with the software. I'm very grateful this wonderful piece of software is publically available.

I also like to thank the people who made Open-Kinect and OpenCV available to the public. You guys and girls did a wonderful job.

Lastly I'd like to thank Koen Langendoen for helping me finalize my thesis and reviewing my writing.

Contents

1. Introduction

RGBD-cameras, which are cameras that capture both regular RGB images as well as depth information, are a recent product. PrimeSence in cooperation with ASUS [1] released such a camera in 2010 with the aim of providing intuitive PC entertainment and introducing a new way of interacting with a PC [2]. But currently the most well known RGBD-camera available is the Kinect from Microsoft. Not only is this of interest to consumers, also researchers in the fields of robotics, Simultaneous Localization and Mapping (SLAM) and ubiquitous computing are using these devices in their research. While the use of range information, from for example laser range finders, is not a new subject, the fact that laser range finders are expensive limits their application. The RGBD-camera from PrimeSense and the Kinect give new alternatives at a cheaper price albeit with a downgrade in maximum depth range, field of view and accuracy compared to laser range finders. Research with the use of modern RGBD-cameras has been performed for various purposes [3][4][5][6]. The research topic of interest to this thesis is the use of RGBD cameras for indoor mapping [7]. Indoor mapping is creating a representation of an indoor environment, which can be used for automatic localization in that environment or the reconstruction of an environment such as creating a 3D digital representation of that environment. RGBD-cameras give a means to create metric maps of an environment with known scale (most camera-based visual mapping algorithms can only create a metric map with unknown scale) at a unit price much cheaper than laser range finders. An example of a map is shown in Figure 3 of a room shown in Figure 1 and Figure 2. The mapping algorithm proposed by Intel [7] does have limitations, as it is vulnerable to violations of the static environment assumption (for example a door opening while recording that door) and failures in visual localization. This can cause misalignments in the map, which could result in warped walls for example.

The goal of this master thesis is to augment the indoor mapping algorithm for the Microsoft Kinect with additional Inertial Measurement Unit (IMU) data to enhance the robustness to scenarios where the proposed algorithm of Intel Seattle Research would fail, such as loss of depth information or featureless environments. IMU sensors provide acceleration and angular velocity measurements, which can be used to calculate an orientation estimate that is insensitive to violations of the static environment assumption and can provide this information even when visual localization fails.

The IMU-enhanced algorithm is developed for using the Kinect holding it by hand and aims to have usability in mind. This means trying to minimize the number of restrictions necessary to use the Kinect for mapping, such as removing the need to move the Kinect very slowly to prevent the mapping algorithm from failing. It is assumed that the ideal way to capture the environment by hand is how people would capture an environment with a regular camera, which corresponds to a maximum rotational velocity of around 30 degrees/s and regular velocities up to 1 m/s. The software is developed for the Windows operating system. Data gathered with the Kinect and IMU are processed offline. The Kinect is characterized in order to investigate the strengths and weaknesses of the Microsoft Kinect. As the source code for the implementation of Intel Seattle's algorithm proposed in [7] is not publically available, an implementation based on the paper has been written. This implementation differs in that it does not feature any form of loop-closure (using all recorded data to reduce the global error of the final result) as it is a post-processing step whose output quality is determined by the quality of the input. The general performance of the mapping algorithm is then tested using a dataset in good mapping conditions (good lighting, conforming to the maximum in angular velocity and regular velocity as described earlier, fluid motion of the Kinect and no dynamics

in the environment) with a ground truth model based on physical measurements as shown in Figure 3, where the room can be seen in Figure 1 and Figure 2. After confirming the IMU-enhanced algorithm generates similar results as the original mapping algorithm both algorithms are used to evaluate their performance with corner cases. These corner cases cover scenarios that violate the static environment assumption and cause failure in visual localization such as featureless environments.

The structure of this thesis document is as follows. Chapter 2 provides a review of relevant research in the area of localization and mapping algorithms. Chapter 3 discusses the Microsoft Kinect in detail, covering the calibration process and the characterization of the Kinect. In Chapter 4 the algorithms used in this research are discussed, as well as proposing the augmented algorithm based on the work of Intel Seattle Research. Chapter 5 compares the performance of the proposed algorithm and Chapter 6 presents the conclusions and recommendations regarding the proposed algorithm.

Figure 1 - Photograph of the room that has been mapped (left wall)

Figure 2 - Photograph of the room that has been mapped (right wall)

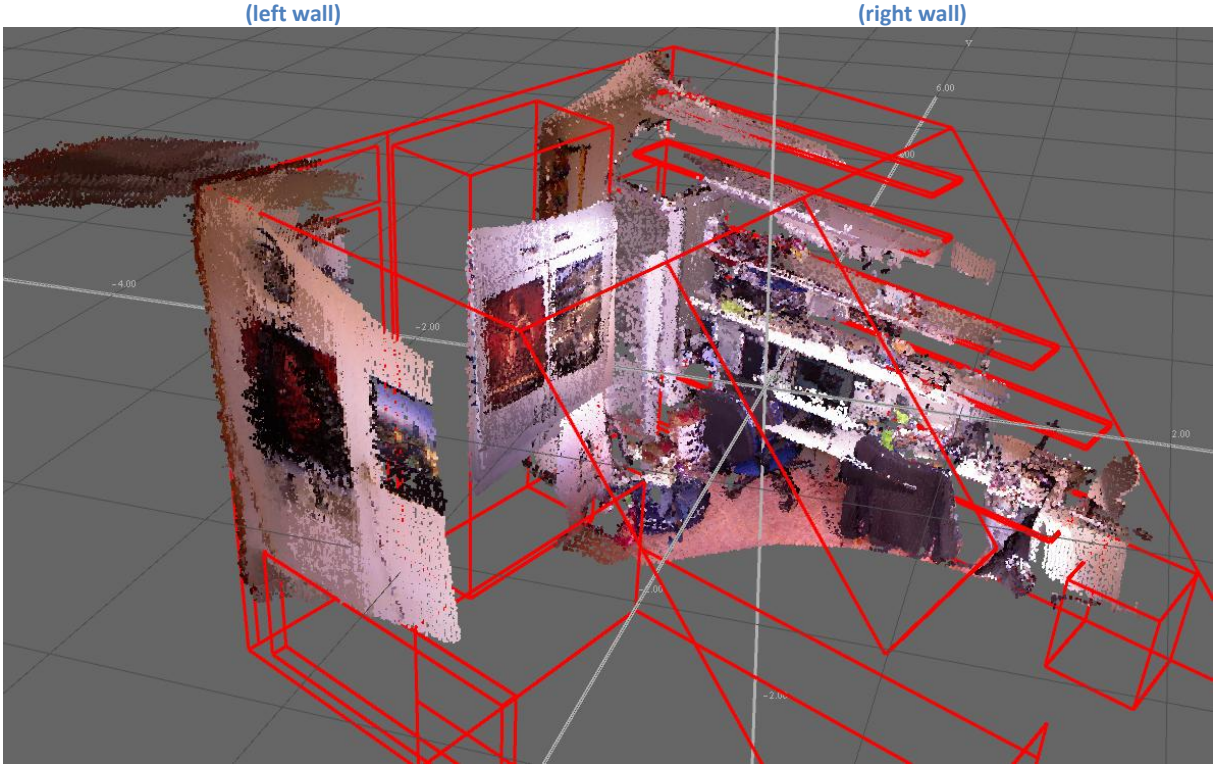


Figure 3 - Example of a point cloud based map created with the IMU-enhanced algorithm (red wireframe shows ground truth based on physical measurements)

2. Related work

This Chapter reviews relevant research into localization and mapping algorithms. The first section gives an overview of localization and mapping algorithms and provides some benchmarks of a few of those algorithms. The second section gives an overview of applications where the former algorithms could be applied in.

2.1. Localization and/or mapping algorithms

Localization algorithms often depend on two core ingredients: a map and a way to localize using this map. The map presents some kind of reference frame which is needed for a localization algorithm to find a location based on the inputs. If an algorithm is able to create or update a map and calculate a position on that map at the same time it is named a Simultaneous Localization and Mapping (SLAM) algorithm. There are also algorithms that calculate a location based on a static map created beforehand instead of creating one on the fly, or use a temporary map that only stores a reference frame for the local environment at that time. Most algorithms rely on the environment to remain static, but there are algorithms that are either robust to slightly dynamic environments or will change the map to capture the dynamics.

Visual localization has been done using only visual sensors like RGB cameras [8][9][10][11][12], RGB cameras with visual markers (fiducials) [13][14][15][16], Time-of-Flight cameras [17][18], laser rangefinders [19] or a combination of visual sensors [20]. But with the coming of RGBD cameras, which give both RGB as well as depth information, there has been research into exploiting these type of cameras for localization and mapping purposes [6][7][21].

Other algorithms combine vision sensors with non-visual sensors using GPS [22], inertial sensors like accelerometers and gyros (also named IMU) [22][23][24], or use Wi-Fi received signal strength measurements [23]. The additional data provide a way to find the initial position, where GPS and Wi-Fi for example can be used to find the general area of where the to-be-localized object is, and then the visual sensors to refine this. Or it might prove useful as fallback mechanism.

There are also algorithms proposed that do not rely on the usage of visual sensors and instead use audio sensors [25], Wi-Fi signal Strength sensors [26][27][28], IMU [26][27][29] or GPS localization techniques [27][29]. These techniques prove useful in situations where no visual sensors could be employed, for example in people localization using a simple tag or from a pocket worn smart phone.

A selection of the above mentioned algorithms is shown with benchmarks in Table 1. The table shows the reported mean accuracy and refresh rates. As most of these papers did not report the variation in the error it is not provided in this table. Algorithms using visual information often have sub-meter accuracy; while non-visual sensor based systems have accuracies in order of meters.

Algorithm	Type	Accuracy (mean) [m]	Refresh rate [Hz]
[7]	Visual	0.16	1.37
[20]	Visual	0.0224	N/A
[25]	Non-Visual	4	N/A
[26]	Non-Visual	3.59	N/A
[27]	Non-Visual	1.57	1
[22]	Visual + Non-Visual	0.75 – 2.3	N/A
[23]	Visual + Non-Visual	.78	30

[24]	Visual + Non-Visual	0.008	12.5

Table 1 - Localization and Mapping Algorithm benchmarks

2.2.Applications for localization and/or mapping algorithms

Mapping and localization technologies can be employed for various applications. An overview of some applications is presented in this section, particularly those that could potentially benefit from the inclusion of depth information.

Localization and pose estimation of robotic devices is one of the applications for which localization and SLAM algorithms are used, such as in [10][21][19][30][31] and [32]. Real-time performance is often a requirement for these types of applications, where automatic and/or autonomous navigation or pose correction is necessary. Visual based systems that use visual features often create a sparse point cloud where the visual features are landmarks used for navigation and orientation.

Another application for localization technologies is object tracking and object recognition. This could mean tracking the recording device itself [11][33] or tracking objects in field of view [22][34][35][36] [37][38] or in case of marker-based systems both the device as well as the marker [13][16][39][40]. Object tracking would be useful for people tracking and labeling, which would allow the possibility of automatically keeping a person in the field of view of a security camera. Face tracking would allow digital cameras to zoom in to optimally fit a person's face for a portrait. Augment Reality applications track objects to overlap it or supplement it with additional information, like a name label or a game entity like a 3D model of a racecar. With depth information it can be used as an additional model descriptor to recognize and track an object. Games for the Microsoft Kinect show this potential already, but the Kinect can also be used where the added value of depth information is useful for object recognition [3].

Nowadays car navigation systems are not a new sight any more. Personal or pedestrian navigation systems could also prove useful, especially in unfamiliar buildings like a visit to a hospital or a very big mall while on vacation. Research has been performed in this area [29][41][42], and Nokia Research Centre has started working on the release of an indoor navigation system [43] as well as companies like Point Inside [44] and Navteq [45]. It would also location based advertisement, where one would be able to view a store's current sale-deals by walking near it, or allow for location based audio-guide in a museum.

Modeling of objects is also an application of mapping, but from a different perspective. Instead of being 'inside' the object you map, you are 'outside' of it. These models could be used for the movie and game industry, and medical purposes. Research like [3][46][47] shows how depth images can be used for modeling, while [9][48][49][50] show how RGB information can be exploited for modeling purposes.

3. Characterization of the Microsoft Kinect

The objective of this chapter is to understand the performance limits and optimal working conditions of the Kinect platform and to assess its suitability for indoor and outdoor mapping purposes. The outline of this chapter is as follows.

First a technical overview of the Kinect is presented, explaining the mechanics behind the depth estimation process. Then the necessity of basic camera calibration techniques is discussed, followed by a section about different drivers available for the PC. Then an explanation is given on the tests that have been performed with the Kinect and a discussion on the results follows. Lastly the conclusion is presented on how well suited the Kinect is for mapping purposes.

3.1. The Microsoft Kinect

The Microsoft Kinect is a special RGBD camera created for Microsoft's XBOX 360, to be used as a controller substitute and an extra input device for specific games that exploit the use of the Kinect[51]. But because this sensor has a normal USB connector and gives the possibility of depth data for a cheap unit-price, it also found the interest of people to make the Kinect available to PC users by making custom drivers.

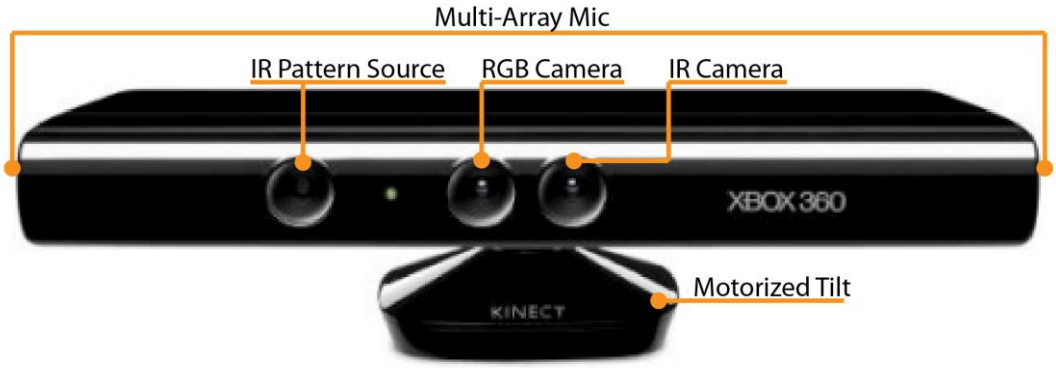


Figure 4 - The Microsoft Kinect

The Kinect is able to grab RGB images of 640x480 pixels in 8 bit depth with a Bayer color filter [52] and IR images of 640x480 pixels with 11 bit depth. It has a frame rate of 30Hz and an angular field of view of 57 degrees horizontally and 43 degrees in the vertical axis. It needs its own power source other than the USB connector [53], which is provided with the stand alone kit of the Kinect. The base of the Kinect houses an electro motor that allows the Kinect to tilt. Furthermore there is a multi-array microphone built in the Kinect, towards the sides of the Kinect and it also has an accelerometer (3 dimensions).

The depth acquisition technology is named Light Coding™ that the company PrimeSense has patented [54][55]. It has an IR Pattern Source, which has a single transparency with a fixed pattern with an IR light source to project a complex pattern of light dots (see Figure 6) onto an object. The IR camera takes images of the object that has been illuminated with this pattern and the image data is then processed to reconstruct a three dimensional model of the object using the knowledge of the IR light pattern.

Figure 5 - A image taken with the Kinect

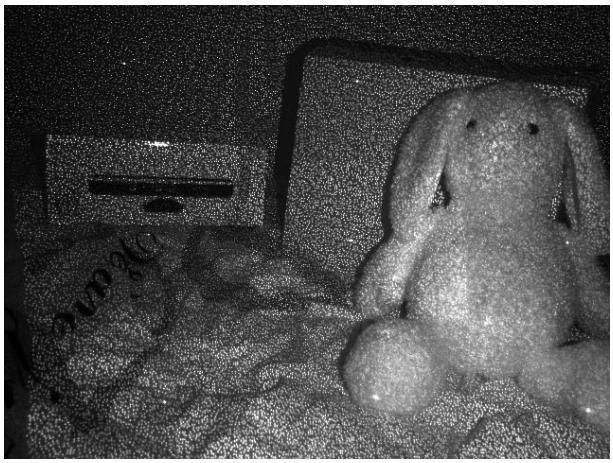


Figure 6 - The corresponding IR image with pattern dots

3.2.Kinect Camera Calibration

In order to correctly create 3D maps based on the 2D depth images from the Kinect, the intrinsics of the depth and RGB camera of the Kinect should be known as well as the pose difference between the two cameras. The camera intrinsics are used to create 3D point clouds and the pose difference between point clouds are used to piece the different clouds together into a greater whole. Point clouds for this application are collections of points with color information in 3D space. How these are created from data captured by the Kinect is explained in Section 4.1. To understand the concepts behind camera calibration an overview is given. For more in depth information and details on the math, the reader is referred to Chapter 11 and 12 of [56].

3.2.1. Pinhole camera model

The simplest model of a camera is the pinhole camera model. In this model, light enters from the scene (i.e. light source, reflection from an object) but only as a single ray from any point in that scene. This point in real space is "projected" onto an imaging surface, also called the *image plane*. Figure 7 illustrates the scenario, but a mathematical abstraction can be used to make the math easier and is shown in Figure 8. Both scenarios are mathematically equivalent, but the latter is easier to work with and easier to illustrate. The size of the image relative to the size of the object is defined as the *focal length* and the relation can be written as follows:

$$-x = \frac{fX}{Z} \qquad 1.$$

In this relation f is the focal length of the camera, Z is the distance from the camera to the object, X is the length of the object and x is the object's image size on the imaging plane. The point at the intersection of the image plane and the optical axis is referred as the *principal point*. For Figure 8 the relation would be $x/f = X/Z$ as there is no inversion and it also gives a more clear indication of a triangular relationship. However the principal point of the camera is not the center of the imager, because of inaccuracies in the manufacturing process. Therefore two additional parameters, c_x and c_y, are used to model the possible displacement from the optical axis. This gives the following equations:

$$x_{image} = f_x \frac{X}{Z} + c_x \qquad 2.$$

$$y_{image} = f_y \frac{Y}{Z} + c_y \qquad 3.$$

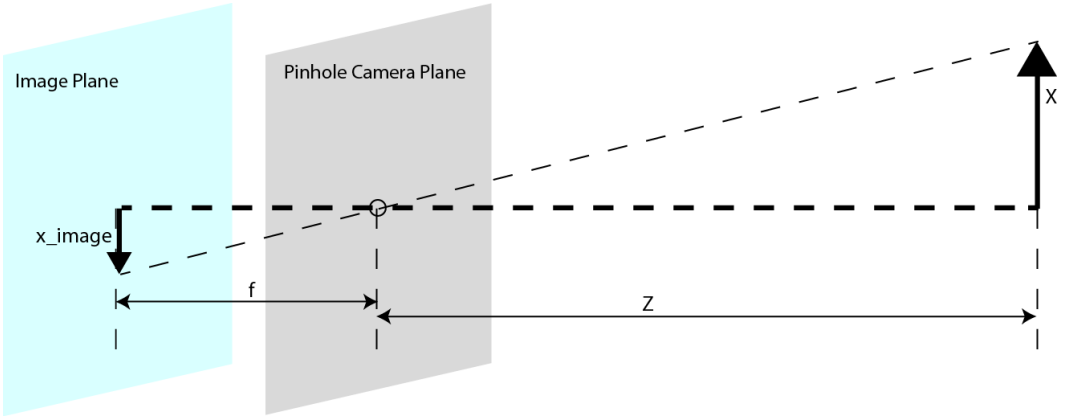


Figure 7 - Projection of 1D object X onto Image plane

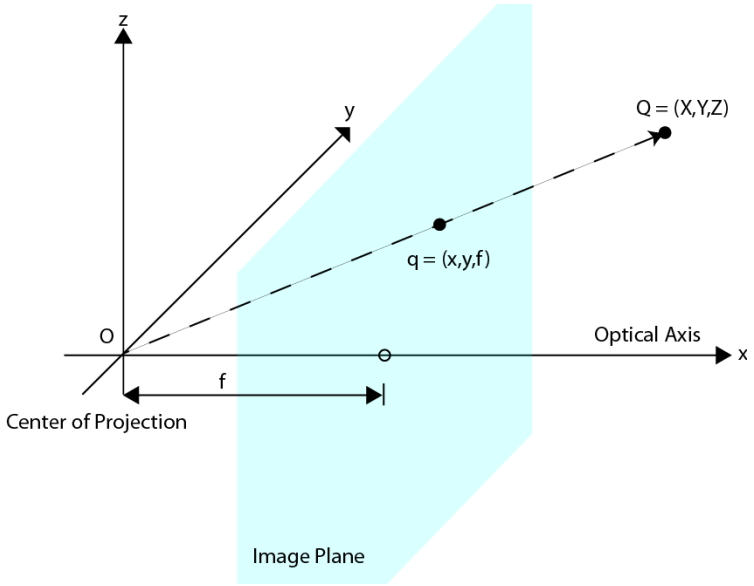


Figure 8 - Projection of 3D point Q onto image plane

Note that the focal lengths in equations 2 and 3 have a different subscript, which is because typical low-cost imagers have rectangular pixels instead of square ones. These focal lengths can also be expressed as $f_x=Fs_x$ and $f_y=Fs_y$ where the s_x and s_y are scaling factors and F the physical focal length. These scaling factors and F cannot be estimated separately, and only the combinations can be derived without actually dismantling the camera and measuring the components.

The transformation from the physical three dimensional world with coordinates (*X,Y,Z*) to the points in the image plane with coordinates (*x,y*) is called a *projective transform*. When working with transforms it is often convenient to use *homogeneous coordinates* that allow the expression of points at infinity (the horizon for imaging) with finite numbers. Using a homogenous coordinate representation for the image plane allows us to define the pin model camera with a single 3x3 matrix. This 3x3 matrix is called the *camera intrinsic matrix*, which gives:

$$q = MQ, where\ q = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad 4.$$

Equation 4 gives an ideal pinhole model for three-dimensional geometry. However this model does not take lenses into account, whose imperfections cause distortion. The distortion can often be modeled with two components: *radial distortion* and *tangential distortion*. The first is caused by the use of "spherical" lenses instead of the mathematically ideal "parabolic" lens, and the measure of distortion is dependent on the shape of the lens. The second is caused by inaccuracy of alignment of the lens and imager, thus is dependent on the assembly process of the camera itself. Other types of distortion models do exist, but these do not significantly improve the results and give rise to numerical instability [57][58].

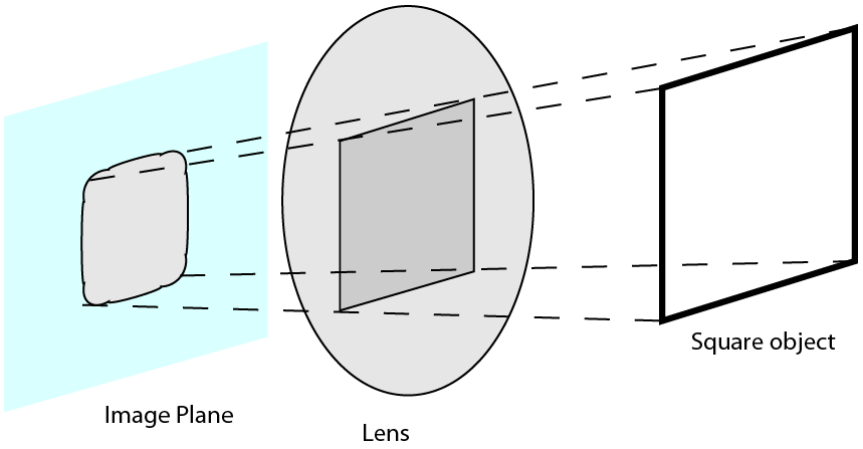


Figure 9 - Radial distortion.

Figure 9 illustrates the effect of radial distortion, where the distortion is zero at the optical center of the imager and becomes more severe towards the edges of the image. This type of distortion can be described by the first few terms of a Taylor series expansion around *r*=0. The distortion can be expressed as follows, where the coordinates (*x,y*) are the distorted locations on the imager, and ($x_{corrected}$,$y_{corrected}$) the corrected coordinates:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad 5.$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad 6.$$

Figure 10 illustrates the effects of tangential distortion for the lens not being exactly parallel to the imaging plane. This can be expressed as follows:

$$x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)] \quad 7.$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 x] \quad 8.$$

All five distortion parameters can be put in a 5x1 matrix [k_1,k_2,p_1,p_2,k_3]. Both the camera intrinsics and distortion parameters can be estimated in a process called *camera calibration*.

3.2.2. Camera Calibration

In order to perform camera calibration, one needs a *calibration object*. In principle, any object that is fully characterized for the camera calibration implementation can be used. For example it can be a

three-dimensional object or a planar object with known dimensions. As this thesis uses the OpenCV library implementation used in the RGBDemo-0.4.0 by Nicolas Burrus[1], the calibration object is a *chessboard*, shown in Figure 11 and Figure 12. It is a planar object with alternating black and white squares with known dimensions. The calibration object is held in various poses in front on the camera, trying to cover as much of camera's field of view with these different poses. Corner points of the calibration object are determined up to sub-pixel accuracy using an automatic corner feature detector, for which the detected corners are superimposed in Figure 11 and Figure 12.

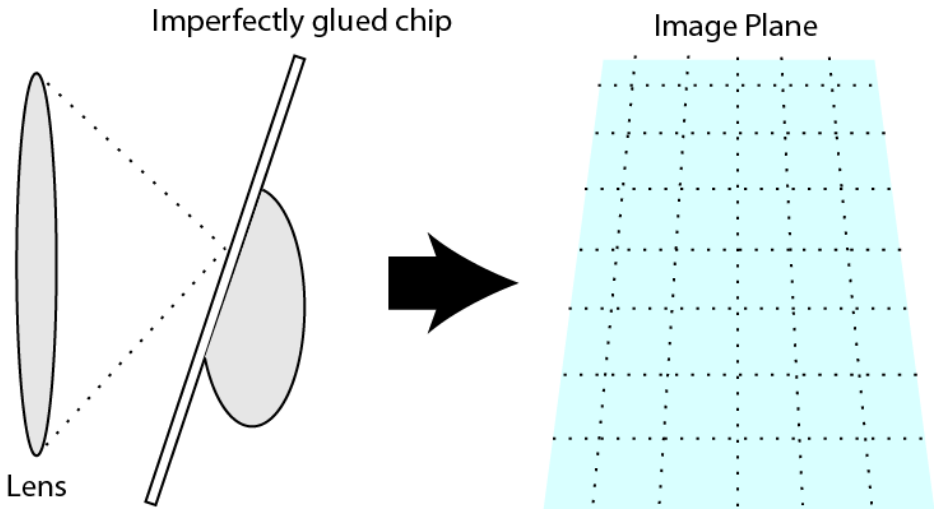


Figure 10 - Tangential distortion

As the calibration object is fully known, meaning it is known how many corners are present horizontally and vertically and the distance between these corner points, all images give constraints to solve the camera intrinsic matrix and the distortion matrix. The poses of the calibration object can also be estimated in this process, which is called *camera extrinsics*, which allows us to calculate the pose of the RGB camera with respect to the IR camera. As only the intrinsic and distortion parameters are of interest in this application, the math for determining the camera extrinsics is not covered here. For those interested into the math please refer to [56].

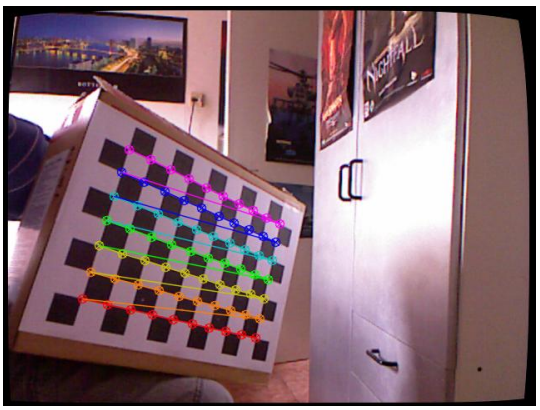


Figure 11 - RGB image calibration frame

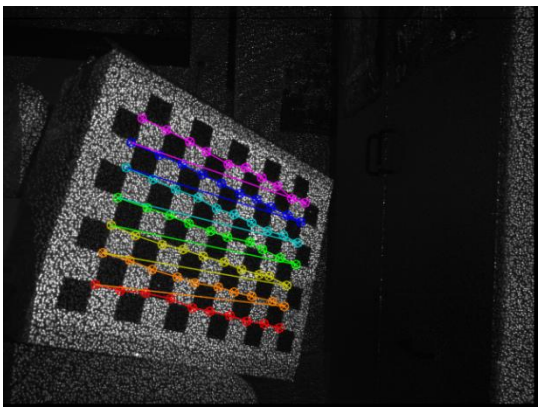


Figure 12 - IR image calibration frame

After calibration the six degrees of freedom, being rotation and translation, between the RGB camera and the IR camera are known and the intrinsic and distortion of both individual cameras are also known. The pose difference between the two cameras is used to determine which pixel in the RGB

[1] http://nicolas.burrus.name/index.php/Research/KinectRgbDemoV4?from=Research.KinectRgbDemo

image corresponds to which pixel in the IR image. The distortion matrix is used to counteract the distortion, shown in Figure 11, which causes the black borders around the image.

3.3.Available software and possible applications

As the Kinect allows RGBD data for a cheap unit cost, various people have started to create drivers for the PC. There are currently four drivers available for the PC. The goal for this thesis is to create a mapping solution using the PC, but there are also other applications for the use of the Kinect. Therefore this section covers the drivers currently available and other possible applications for the use of the Kinect with a PC.

3.3.1. Open Kinect

Website: http://openkinect.org/wiki/Main_Page

Open Kinect, also known as libfreenect, is an open source library for Windows, Linux and Mac. Initially the goal was to create driver-level software but interest has grown to develop additional features. Currently it allows access to the RGB data, the Depth data, the LED, the tilt-motor and the accelerometer. It has wrappers for Python, C, C++, C# and Java among others. It aspires to get additional features like audio access, hand and/or skeleton tracking, 3D reconstruction, point cloud generation and additional wrappers for other platforms like Matlab, OpenCV and Labview.

3.3.2. Code Labs NUI

Website: http://codelaboratories.com/kb/nui

This is another open source library for Windows that allows interfacing with the Kinect. Currently it is able to access RGB and Depth data, the accelerometer and the tilt-motor. It has wrappers for WPF/C# as well as some support (although not fully yet) for C and C++. It has a driver for the audio hardware, but no software yet to access the data though. The project is aiming at finalizing the audio part of the Kinect driver and adding Java support, but at the time of writing the project has not received any updates since its first release on the 8th of December 2010.

3.3.3. OpenNI/NITE

Website: http://www.openni.org/ , http://www.primesense.com/?p=515

OpenNI is an industry-led, non -profit organization formed to certify and promote the compatibility and interoperability of Natural Interaction (NI) devices, applications and middleware. With Natural Interaction devices they mean devices that would allow interaction with electronic devices like we would with humans, for example using speech and gestures. Devices that would fall under this category would be cameras of any kind of microphones.

NITE is middleware developed by Primesense, who have the patent behind the technology implemented in the Kinect. The NITE engine has algorithms for user identification, feature detection and gesture recognition, as well as a framework that manages the tagging of users in the scene and the acquisition and release of control between users.

It offers C++, C# and Flash API's for Linux and Windows that allows access to RGB and Depth derived data, like Full Body Analysis, Hand Point Analysis, Gesture Analysis and Scene Analysis (detection of the floor plane, back ground, foreground, people recognition and labeling). It is not clear if direct access to raw RGB and depth data is possible with this software.

3.3.4. Kinect for Windows SDK

Website: http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/default.aspx

Microsoft released a software development kit to create applications with the Kinect. It provides interfaces for C++, C# and Visual Basic. It allows access to the depth data, RGB data and audio sensor array. Skeletal tracking, acoustic noise suppression, echo cancellation, beam formation and integration with the Windows speech recognition API are provided in this SDK.

3.3.5. Applications

The Kinect has potential in other applications than SLAM. The aim of OpenNI and NITE is to be able to use a device like the Kinect as a new way to interface with electronic devices, which is very interesting for the field of Man-Machine Interaction. It could also be used as a surveillance camera where people recognition, labeling and tracking would be very valuable. The same applies for object tracking. Motion and 3D model capture is also a possibility for the Kinect, adding a new experience to gaming, AR and VR.

3.4. Test Setup

In order to assess the suitability of using the Kinect for 3D mapping purposes, a series of tests are performed to characterize the quality of the depth data in various circumstances. It is investigated how the data quality is affected by distance from an object and by the angle the camera is facing an object. Also the behavior of the depth data is analyzed for different materials like glass, reflective surfaces, wood and fabric. Another aspect that is put to the test is how the Kinect behaves with lighting conditions.

The first setup is to have the Kinect face a plain wall at 90 degrees. The distance from the wall is changed from 60cm to 380cm with increments of 20cm while keeping the angle with the wall fixed. Figure 13 illustrates the scenario, where the red line depicts the distance *d* from a point on the wall. This test is to see how the depth estimation and its accuracy depend on the distance from an object. A sampling grid is used in the depth image to use pixels of the wall. A set of 100 images is used, while the grid itself subsamples an area of 280x210 pixels in a grid with 10 pixel increments in both horizontal and vertical axis as illustrated in Figure 15. It is assumed that the natural variation in depth measurements in both horizontal as well as vertical axis is comparable between different values of *d*.

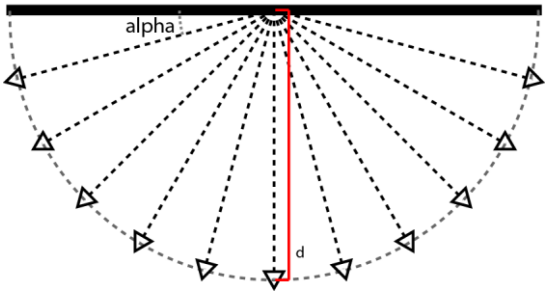


Figure 13 - Setup for Test 1 and Test 2

When looking at Figure 15 one can see that the bottom half of the depth image is black, as well as a blob in the center of the image. The black color for pixels in the depth images presented throughout the thesis means a *faulty measurement* where the Kinect was unable to get a depth measurement for that pixel.

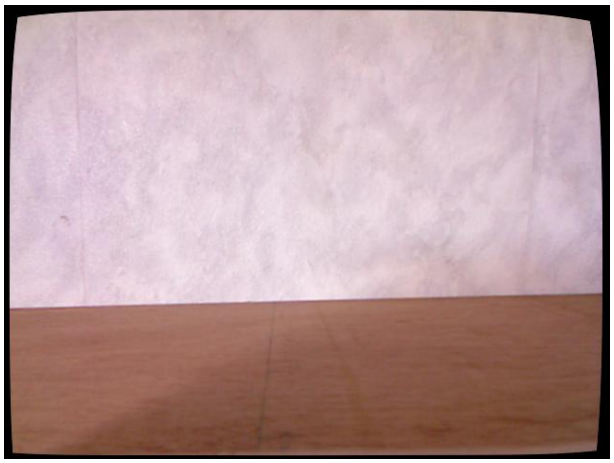


Figure 14 - A RGB image with d=60cm

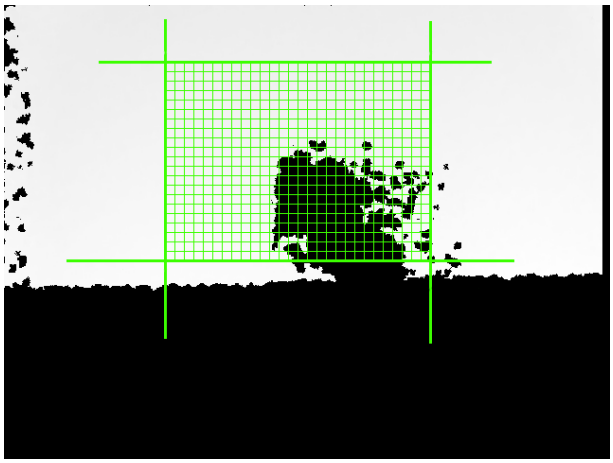


Figure 15 - Corresponding Depth image with sampling grid superimposed

The second test setup is having the Kinect face the wall at different angles while keeping the distance *d* equal to 1,5m. The angle *alpha* is changed from 15 degrees to 165 degrees with increments of 15 degrees. Figure 13 depicts the different poses of the camera in the arc of interest. This test is to show how the depth estimation process is affected by the angle incident to an object. For each angle 100 images are taken and depth pixels are used in the vertical axis of the image (on a single line), instead of a grid as used in the first test. This is done to prevent a significant increase in variance for angles of 15 and 165 degrees, as there is a much stronger depth estimation gradient compared to an angle of 90 degrees. A total of 240 samples are used along the line, which is illustrated in Figure 17. The green area illustrates the possible locations where could be sampled, where the green line shows the specific line for this test case. The red horizontal lines mark the bounds on the sampling in the vertical direction.

Figure 16 - A RGB image at an angle = 30

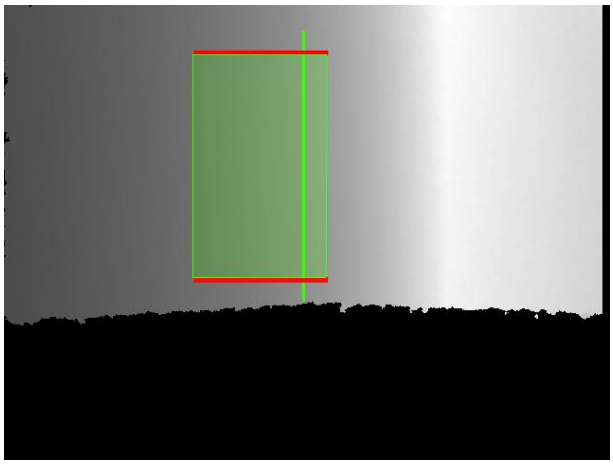


Figure 17 - Corresponding Depth image with sampling line superimposed (general area highlighted for other images)

The third setup is to see how the Kinect depth estimation deals with mirrors. Figure 18 shows the scenario for this test. The Kinect is placed at a 1.5m distance from the mirror surface with an object whose centre will be at 0.5m distance from the mirror surface. The angle of the Kinect with respect to the mirror is changed from 15 degrees to 75 degrees with increments of 15. The object angle is changed to keep the object both in direct field of view and indirect field of view of the Kinect via the mirror surface. A sampling area of 40x45 pixels is used, where the horizontal axis is subsampled with an increment of 2, on the object location in the mirror surface to measure the estimated distance to

the mirror where the object seen in the mirror. Figure 20 illustrates this, where the green area depicts the sampling area used in this test case.

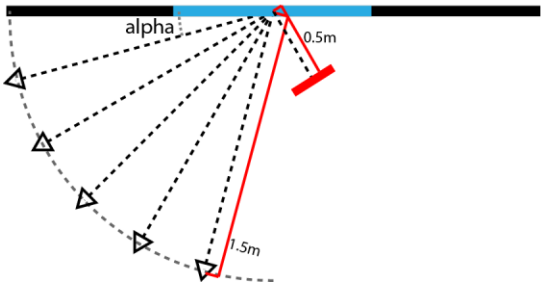


Figure 18 - Setup for Test 3

Figure 19 - A RGB image at alpha = 75

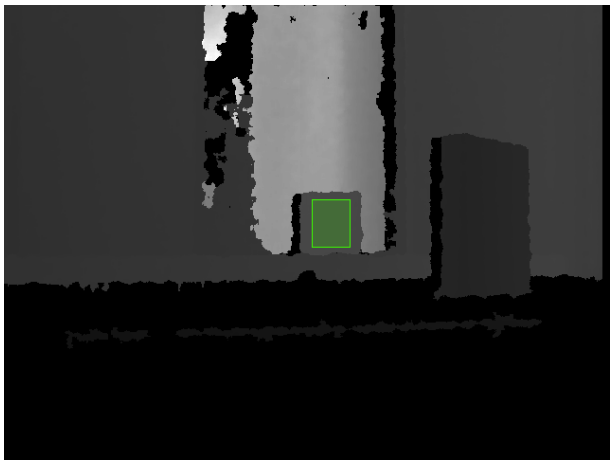


Figure 20 - Corresponding Depth image with the sampling area as overlay

The fourth test case is to put the Kinect behind a pane of glass to see how the depth estimation is affected by a window. The Kinect is put at a fixed distance of 1.5m while the angle is kept variable from 30 to 90 degrees with increments of 15 degrees. Figure 21 illustrates the setup. A sampling grid of 65x40 pixels is put over the object behind the pane of glass as illustrated in Figure 23, which is subsampled with an increment of 2 for both axes.

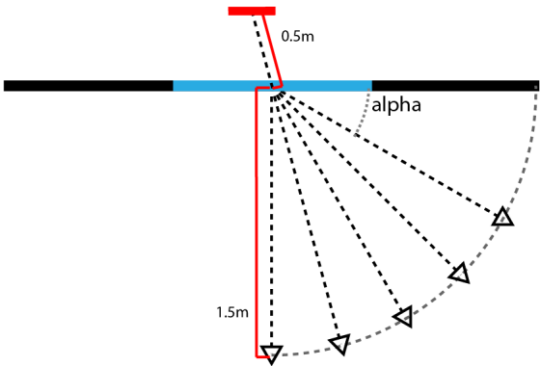


Figure 21 - Setup for Test 4

Some general recordings with the Kinect are analyzed in different scenarios, to see if there are materials that present strange behavior in the depth estimation. These scenarios are:

- Living room environment

Figure 22 - A RGB image for alpha = 90

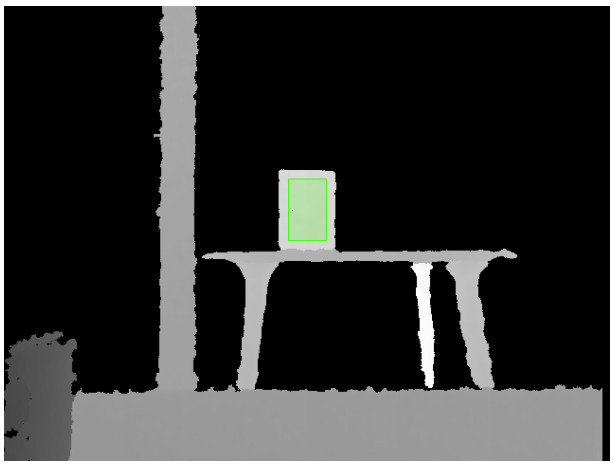


Figure 23 - Corresponding Depth image with sampling area overlay

- Kitchen environment
- Bedroom environment
- Outdoor garden

Furthermore the sensitivity of the Kinect to (sun)light has been investigated. Lastly the possibility of lens flare with the Kinect has been investigated with respect to the IR camera. A quick summary of the number of samples used per test scenario is given in Table 2.

	Goal	# Samples
Test 1	Distance sensitivity	63800
Test 2	Angle sensitivity	24100
Test 3	Angle sensitivity for reflective surface	96600
Test 4	Angle sensitivity for glass	69300

Table 2 - Summary number of samples per test scenario

3.5.Kinect Characterization

As stated in Section 3.1 the Kinect is able to give 11 bit depth estimation per pixel in the depth image. This raw sensor data needs to be converted into a sensible metric to compare it with the known distances of the Kinect based on the four tests. The following 'Depth-to-Range' conversion has been chosen as presented at [59]:

$$Range(m) = \frac{1}{-0.0030711016 * Depth\_measure + 3.3309495161} \quad 9.$$

The Depth-to-Range function has been illustrated in Figure 24. The green and blue areas in the image show the set of depth measurement values that belong to a small range span. It demonstrates that the variance in the measurement values is bigger for a distance around one meter than around 3.6 meters.

3.5.1. Test 1

The goal of the first test is to investigate if the behavior of the depth estimation of the Kinect changes for different distances to an object. The results are shown in Table 3, giving the error values at the 50th and 90th percentiles for the tested distances. It shows that the depth estimation error is within 7.65% of the actual distance with the average being 4.05% based on the 90th percentile. This means that on average one can expect the uncertainty to be 4.05% of the distance, i.e. 8.1 cm error

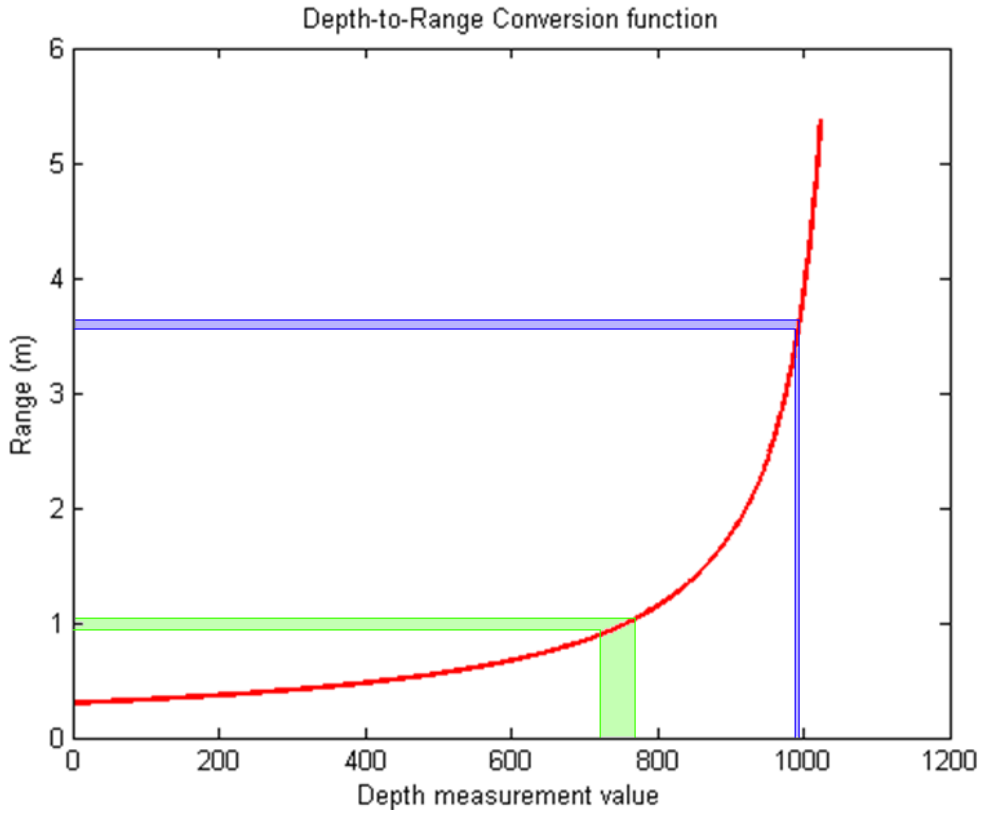


Figure 24 - The implemented Depth-to-range conversion

for a distance of 2 meter. Secondly, the test data show that the Kinect underestimates the range of the Kinect for this scenario.

d (m)=	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	2.2
50%	0.0382	0.0331	0.0496	0.0396	0.0523	0.0613	0.0644	0.0457	0.0380
90%	0.0459	0.0385	0.0606	0.0519	0.0634	0.0685	0.0736	0.0573	0.0523

d (m) =	2.4	2.6	2.8	3.0	3.2	3.4	3.6	3.8
50%	0.0190	0.0126	0.1003	0.1085	0.0874	0.0980	0.0840	0.0834
90%	0.0509	0.0126	0.1225	0.1085	0.1168	0.1636	0.1584	0.1663

Table 3 – Absolute error(m) 1st test setup for 50th and 90th percentile

The uncertainty as relative error with respect to the distance for the first test is plotted in Figure 25. It shows the uncertainty decreases up to 2.6m and increases after that. This suggests that the Kinect depth estimation probably works optimally at a distance around 2.6 meters.

The largest error is found for the d=60cm case, which is caused by 14110 fault samples that were in this test case. The Kinect was not able to get a depth estimate for 22.1% of the sampled pixels on the wall, which can also be seen from Figure 15. The measurement faults are caused by the Kinect being too close to the object. It can be concluded that the Kinect is reliably usable with distances from 0.8 meters up to at least 3.8 meters. Comparing this with [60], which is the technology the Kinect is based on, the stated operation range of 0.8m-3.5m confirms the validity of the conclusion. However, the error in depth as stated in the PrimeSense datasheet of 1cm at a distance of 2 meters does not match with the findings of Test one, which is almost an order of magnitude bigger.

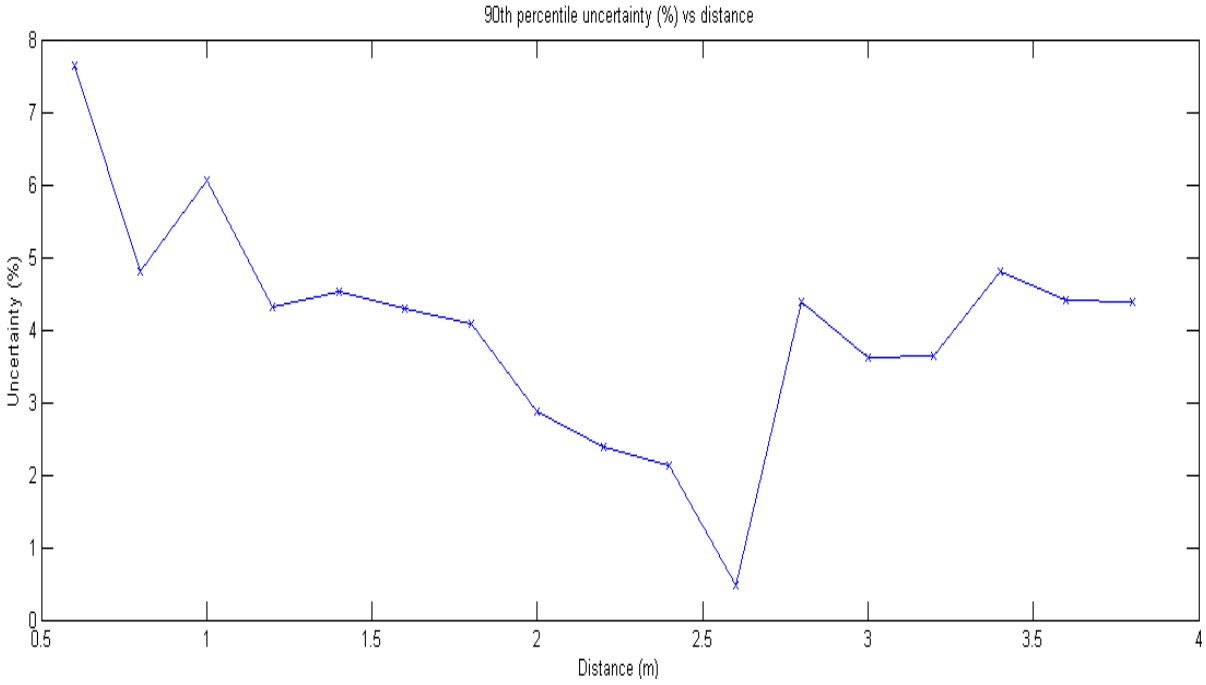


Figure 25 - 90th percentile uncertainty (%) vs distance

3.5.2. Test 2

The second test is to see what the behavior of the depth estimation quality is with respect to at what angle the Kinect faces an object. The results for this test are shown in Table 4, where the 50th and 90th percentiles are shown for the considered angles, with the Kinect being 1.5m away from the wall. The largest relative error measured with respect to the distance is an error of 9.85% for an angle of 165 degrees with the total mean error being 3.73%, both based on the 90th percentile. The largest error for the 165 degree case is caused by a slight slant in the test setup as the lack of available space made it hard to give proper support to keep the Kinect perfectly level. Therefore the sampling line is manually aligned such that it is comparable with the rest of the test data.

α	15	30	45	60	75	90	105	120	135
50%	0.0694	0.0694	0.0757	0.0240	0.0240	0.0173	0.0173	0.0102	0.0102
90%	0.0757	0.0880	0.0819	0.0373	0.0438	0.0373	0.0240	0.0315	0.0172

α	150	165
50%	0.0315	0.1478
90%	0.0534	0.1732

Table 4 – Absolute error(m) 2nd test setup for 50th and 90th percentile

The mean error of Test 2 is smaller compared to Test 1, but this can be explained by the fact that the natural variation in the depth measurement for Test 1 is bigger than that of Test 2. This is because in the first test samples have been chosen along the horizontal axis while for the second test this was not the case. The additional sampling axis allows for more variation of the measured distance and therefore increasing the calculated uncertainty compared to the results from Test 2.

The data in Test 2 show that the Kinect tends to underestimate the distance for angles smaller than 90 degrees and overestimating for angles larger than 90 degrees. Also the data for angles 15, 30 and 45 degrees show a higher error compared to the other angles (with the exception of the 165 error

case because the Kinect was not leveled), which might be caused by of the positioning of the IR Pattern Source. This test shows that there might be a sweet point for operating the Kinect at 90 degree plus/minus 30 degrees with respect to the viewed object.

3.5.3. Test 3

The third test is to see how the Kinect estimates depth when viewing a mirror surface. The test results are shown in Table 5. Interesting is that the measurements result in a distance estimate of around 2 meters (distance to mirror plus distance to object) instead of 1.5 meters (only distance to mirror). This means that for mapping purposes a mirror surface is not detected itself and the distance measured by the Kinect for objects visible in the mirror's surface is equal to the path for a light ray going from the object to the mirror back to the Kinect. The average relative error with respect to 2 meters is 2.01% and the worst relative error measured is 3.30% for an angle of 60 degrees. There was no clear distinction to be found between over- and underestimation in the data.

α	15	30	45	60	75
50	0.0098	0.0148	0.0219	0.0529	0.0274
90	0.0274	0.0274	0.0401	0.0660	0.0401

Table 5 – Absolute error(m) 3rd test setup for 50th and 90th percentile

For the angle of 15 degrees 5.29% of the samples were a fault. For the other angles there were no measurement faults, so it seems that viewing a mirror at angles around 15 degrees allows for measurement faults.

3.5.4. Test 4

The last test setup has the goal to see how the Kinect depth estimation is affected when it views an object behind a pane of glass, performed at different angles. The Kinect correctly measures a distance of 2 meters to the object. The worst relative error measured with respect to 2 meters is 17.2% for the angle of 30 degrees. As a whole the mean relative error is 8.75%, highest among all the test setups. The results of Test 4 are shown in Table 6. The data shows that the depth estimation is negatively affected when viewing through a pane of glass at an angle, with an increase in error for angles further away from 90 degrees.

α	30	45	60	75	90
50%	0.2917	0.1563	0.1458	0.0802	0.0339
90%	0.3438	0.1917	0.1917	0.1026	0.0457

Table 6 - Absolute error(m) 4th test setup for 50th and 90th percentile

At an angle of 30 degrees 3.29% of the samples have fault values, while the other test cases did not have any significant fault count. It seems that viewing through a pane of glass at an angle of around 30 degrees (and possibly lower, but because of the lack of available space in the test environment it was not possible to confirm this) increases the number of faulty depth measurements.

3.5.5. Lens flare

Lens flares are artifacts being caused when light is scattered in lens systems and are often regarded as unwanted effects in the end results. The depth camera can suffer from lens flare effects and can even cause them with its own IR illumination source. To demonstrate this, the Kinect is placed on a chair in front of a mirror at a distance of 1.0m. The back of the chair measures a distance of about 1.4m to the mirror's surface.

Figure 26 - RGB image with lens flare

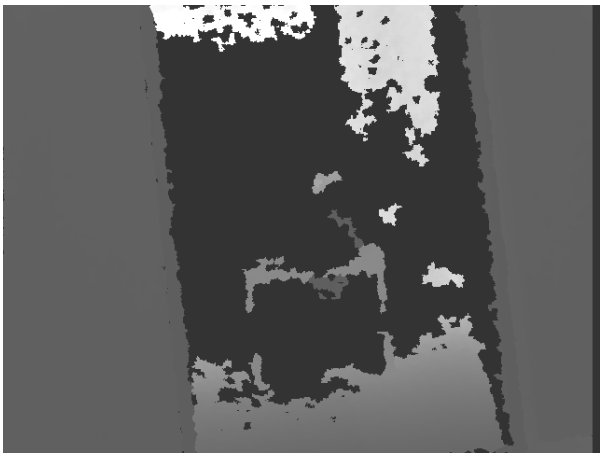


Figure 27 - Depth image with lens flare

Figure 28 - RGB image without lens flare

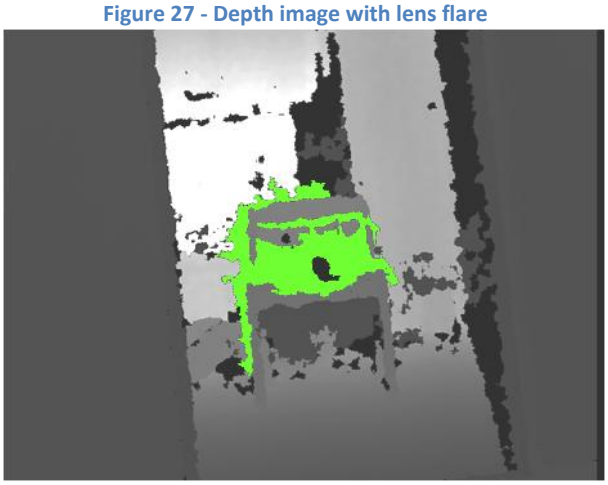


Figure 29 - Depth image without lens flare

Figure 26 shows how the Kinect IR illumination source is being picked up by the RGB sensor and can create lens flare effects, recognizable by the blue circular shapes in the image. The effects are also visible in the corresponding depth image shown in Figure 27, where clearly can be seen how the Kinect cannot get a depth estimate for a large part of the mirror surface because of the lens flare. Figure 28 and Figure 29 show how the images look like without the lens flare effect. There is still an area around the Kinect in Figure 29 (highlighted in green), which gives a range estimate of around 1.0m, which is the distance from the Kinect to the mirror surface. The light grey belonging to the back chair gives a range estimate of about 2.8m, which is two times the distance of the back of the chair to the mirror's surface.

Although causing the lens flare effects with the Kinect illumination source is a rare event, it is an effect that negatively affects the depth image if a reflective surface is able to reflect the light back directly at the sensors. As the illumination source is being picked up by the RGB sensor as a white-bluish light source it can also affect the RGB values when viewing glossy surfaces up close.

3.5.6. Reflective, refractive and IR absorbing materials

As the depth estimate depends on the IR illumination to be reflected into the IR sensor, any reflective and refractive surface can potentially influence the depth estimates. When the IR light is reflected or refracted in such a way that is does not arrive at the IR sensor, it will result in faults in the depth estimation. This means that a glass of water or a fish bowl might not be detected in full with the depth sensor, or the depth measurement gets distorted.

Another effect that can occur with reflective surfaces is that distance estimates are not always up to an object seen in the reflective surface, as stated in Section 3.5.5. Furthermore, reflective surfaces add the possibility of seeing the person carrying the Kinect while moving, which creates issues in most mapping algorithms as these rely on the assumption of a static environment, as explained in Section 2.1.

A last note has to be made about IR light absorbing materials. Although these type of materials did not occur in the indoor environment that have been recorded for this thesis, these might be of a concern. The absorption of the IR pattern makes it impossible to get a depth measurement for these surfaces.

3.5.7. Shadow effect

The previous subsection states that if the IR pattern is visible by the IR sensor then it is not possible to get a depth estimate. Another way depth measurements are affected is by a shadow being cast by the IR illumination. Figure 30, shows how Obstacle 1 casts an IR shadow on Obstacle 2. Because the IR camera is at a different location on the Kinect than the IR Pattern Source, it is able to 'see' this shadow. As a result of this shadow a part of Obstacle 2 cannot be measured because of the lack of any IR pattern in that area, resulting in faulty measurements. An example of this can be seen in Figure 20, where a black edge (the measurement faults) can be seen attached to the left of the object. The appearance of such a black edge attached to an object in the depth image is visible for both the object in direct line of sight of the camera as well as the reflected object in the mirror surface. Because of the positioning of the IR Camera with respect to the IR Pattern Source, a shadow can only be cast and be visible to the left of an obstacle.

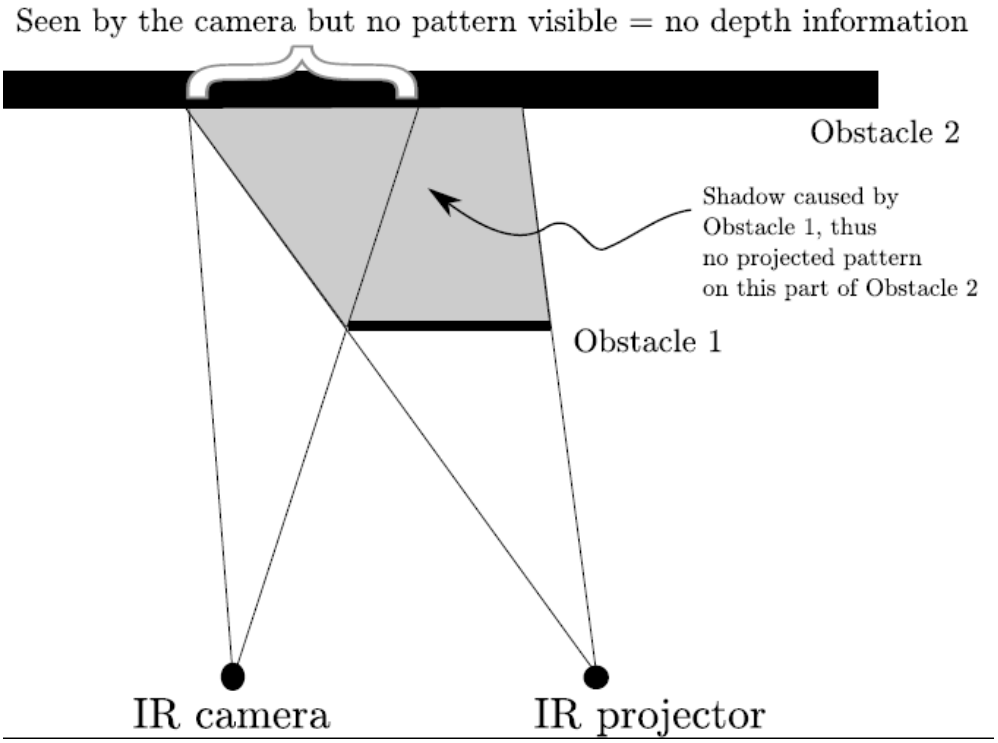


Figure 30 - Kinect Shadow effect (taken from [61])

3.5.8. Gritty surfaces

Gritty surfaces, like some types of floor tiles used for outdoors, are hard surfaces for the Kinect to get a distance estimate for when the camera is held still. When the camera is moved slowly, the estimation process drastically improves, resulting in far less 'black patches' when compared to keeping the camera still. Slightly vibrating the Kinect also reduces the number of faulty measurements, although this might introduce motion blurring into the images. Figure 31 through to Figure 34 show the RGB and depth images for a static Kinect and a slowly moving Kinect.

Figure 31 - RGB image of gritty tiles, keeping Kinect still

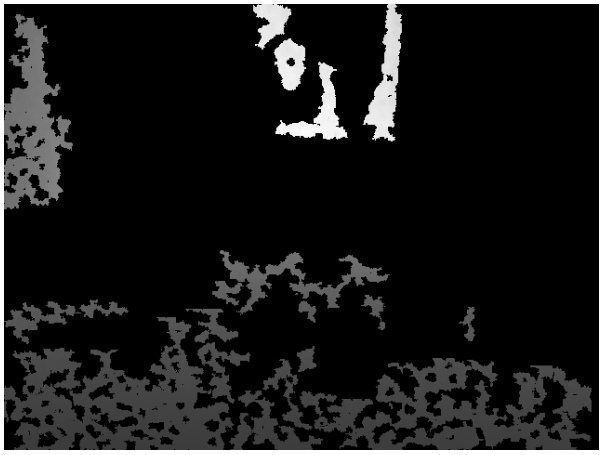


Figure 32 - IR image of gritty tiles, keeping Kinect still

Figure 33 - RGB image of gritty tiles, slowly moving Kinect

Figure 34 - IR image of gritty tiles, slowly moving Kinect

3.5.9. Lighting

The light sensitivity of the Kinect is also analyzed. Light is able to negatively affect the depth image, particularly sunlight. When sunlight falls into the IR sensor (either directly or reflected sunlight, from a mirror for example) almost no depth information can be gathered.

Figure 35 through Figure 40 show the images of a person illuminated by sunlight. From Figure 36 one can see that the upper body is 'hidden' from the depth image because of the sunlight on the upper body while the lower body is in the shade and therefore allowing depth estimation. Figure 38 shows what happens when the sunlight is being mirrored into the sensor, which results in the lower body also being hidden from the depth estimation. Also from Figure 40 it can be seen how sunlight can prevent things from being subject to depth estimation with the Kinect. Because of this, the Kinect is really limited for outdoor purposes.

Figure 35 – RGB image of person in sunlight

Figure 36 - Depth image of person in sunlight

Figure 37 - RGB image of person in sunlight with sunlight reflected into sensor

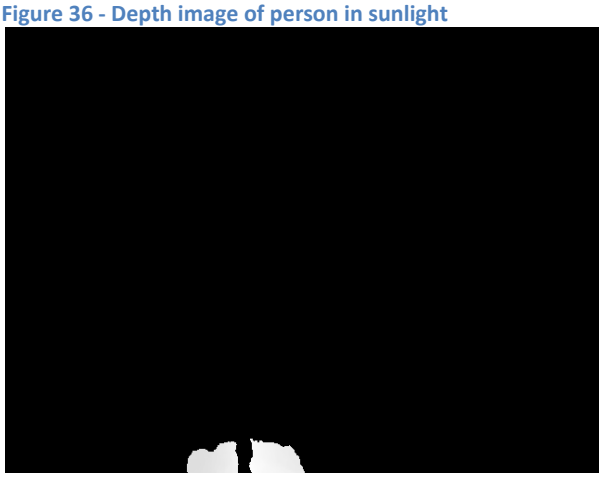


Figure 38 - Depth image of person in sunlight with sunlight reflected into sensor

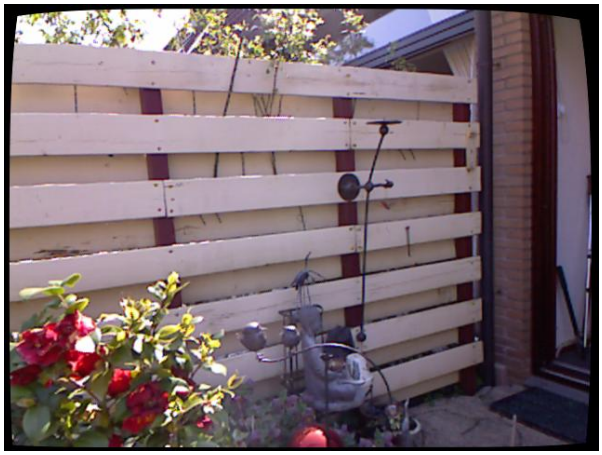


Figure 39 - RGB image garden outside

Figure 40 - Depth image garden outside

3.6.Conclusions

The goal of this chapter is to see if the Kinect is suitable for 3D mapping purposes. The Kinect is able to measure with a relative accuracy of about 4% with respect to the actual distance. The operating range of 0.8 meter up to 3.8 meters is enough to be able to do 3D mapping in indoor environments such as an office or living room, although very spacious areas like an empty storage room of 15x15x5 meter might prove challenging to map. The angle of the Kinect with respect to an object does not really influence the depth measurement quality, although there seems to be a sweet spot around of 30 degrees up to either side when facing the object.

The Kinect does introduce some challenges for mapping. The shadow effect is able to hide parts of the environment, which means that extra work and care might be needed to fully map an environment. Mirrors introduce two causes for concern. One is that the Kinect does not measure up to the mirror itself, but up to objects whose reflection is visible in the mirror, effectively rendering it as a corridor that is not actually there. Second, it might allow the person carrying the Kinect during the mapping to be visible to the Kinect and thus be added to the map itself. This creates errors or outliers in the resultant 3D map. The depth data is negatively influenced by glass when viewing through it at an angle, and any refractive object might introduce distortions or faulty depth measurements into a depth image. Lastly, the Kinect is severely hindered by sunlight, making any attempt of outdoor mapping nearly impossible when the sun shines.

Thus the Kinect proves to be useable for mapping indoor environments where there will be objects to be found within a distance of about 4 meters. The inherent relative accuracy of the Kinect of 4% proves to be far more accurate than non-visual systems presented in Table 1 and similar to the most related research results presented in [7].

4. Implementation

In Section 3.6 it was concluded that the Kinect is a device suitable for indoor mapping purposes. This chapter will explain the techniques used for the indoor mapping performed with the Kinect. The techniques discussed in this chapter are:

- Iterative Closest Point (ICP) algorithm,
- Random Sample Consensus (RANSAC),
- Rigid Body Transformation estimation,
- Image Feature pose estimation,
- IMU pose estimation.

After that the mapping algorithm proposed by Intel Seattle Research and the IMU-enhanced mapping algorithm are explained. But before going into detail, a brief overview of how the above mentioned technique fit together to create a mapping algorithm is given. The ICP algorithm is an algorithm used to put two collections of 3D measurements in the correct position with respect to each other. These measurements, named point clouds, are the measurements of the Kinect and as the Kinect moves these are fitted together to form a bigger 3D representation of what has been recorded with the Kinect, a map. In order to do this an estimate of the rigid body transformation, the translation and rotation, of these measurements needs to be provided to the ICP algorithm. These can be estimated using either image feature information or inertial measurements. The RANSAC algorithm is used to deal with the fact that these initial estimates are not perfectly accurate, increasing the robustness to both imperfect estimates and incorrectly used correspondences between both point clouds.

4.1. Point Clouds

The depth data from the Kinect is used to create point clouds where each pixel in a 2D depth image from the Kinect is transformed into a 3D point with respect to a reference frame, the camera center for example in this work, using the following equations based on Equation 4:

$$P_{3D,x} = \left(x_d - c_{x,d}\right) * \frac{depth(x_d, y_d)}{f_{x,d}} \quad 10.$$

$$P_{3D,y} = \left(y_d - c_{y,d}\right) * \frac{depth(x_d, Y_d)}{f_{y,d}} \quad 11.$$

$$P_{3D,z} = depth(x_d, y_d) \quad 12.$$

Where P_{3D} is the 3D point based on the depth image coordinates (x_d, y_d), and $f_{x,d}$, $f_{y,d}$, $c_{x,d}$ and $c_{y,d}$ are the intrinsics of the depth camera, which are estimated with camera calibration. One depth image of 480x640 provides maximally 307200 3D points (faulty measurements will reduce this count). Putting multiple collections of points in the correct place as the Kinect moves, is called registration of these point clouds. In order to do so, the ICP algorithm is used.

4.2. Iterative Closest Point algorithm

The Iterative Closest Point algorithm (ICP) has been proposed in [62] as a heuristic method to put two sets of 3D curves or maps into registration with each other. In essence, the ICP algorithm requires the input of the following:

- Two point clouds. One is used as reference to which the other (the source) has to be aligned.
- An initial estimate of the transformation between the reference and the other point cloud.
- A neighborhood bound, to limit the search area for which possible correspondences are counted as valid.
- A stopping criterion.

The ICP implementation of the Point Cloud Library (PCL) [63] is used. The algorithm is as follows:

1. Using the provided initial estimation of the rigid body transformation, transform the source. This should put the source more or less aligned with the reference.
2. While not yet converged and not yet reached the maximum allowed iterations do:
   2.1. Find for each point in the source the nearest neighbor in the reference.
   2.2. Save each point correspondence whose nearest neighbor distance is smaller than the provided neighborhood bound.
   2.3. Use the Random Sample Consensus (RANSAC) algorithm [64] to remove outlier correspondences from the correspondence set.
   2.4. Use only the inliers to calculate the rigid body transformation estimate and update the estimate of the transformation between the two point clouds. The rigid body transformation estimation process is explained in Subsection 4.3.2.
   2.5. If not ending the loop: transform the source from its initial position with the current estimate of the rigid body transformation.
3. Return the calculated rigid body transform.

4.2.1. Random Sample Consensus (RANSAC)

The RANSAC algorithm was proposed in [64]. It is an iterative algorithm that works on the assumption that data has inliers and outliers, meaning points that fit some kind of description or not respectively. In this case the RANSAC algorithm is used to generate a pool of correspondences from a superset of correspondences that provide the same rigid body transform and has the least amount of error, which would be caused by the outliers pulling the result away from this solution. The outliers would be points that did not have a true correspondence in the reference because of the lack of depth information or because of a big mismatch caused by a bad initial guess. By removing these false correspondences, the ICP accuracy increases. The algorithm works as follows:

1. From the input correspondences, which have inliers and outliers, a random subset of points is selected. This is the initial Consensus Set for this iteration.
2. Estimate the rigid body transform using the correspondences in the Consensus Set.
3. For each of the other correspondences, add it to the current Consensus Set and calculate the rigid body transform. If the result is close enough to the one calculated in step 2, put the point in the Additional Consensus Set. If it the result was not close enough, discarded it for this iteration.
4. Combine the Consensus Set of step 1 with the Additional Consensus Set of step 3 and use the combined set as the final Consensus Set. Recalculate the rigid body transform using this final Consensus Set.
5. Calculate the error that remains (sum of squared distances).
6. If this final Consensus Set has the least error, keep it as the best found Consensus set.
7. If the maximum number of iterations has not been reached, go to step 1. Else, return the best found Consensus set.

4.2.2. Rigid Body Transform

The rigid body transform estimation performed within PCL has been proposed in [65]. A rigid body transform of a point cloud is defined by the rotation and translation matrix of that point cloud with respect to its starting point. The translation matrix is determined by the center points of the point clouds in the algorithm, while the rotation matrix is determined by the use of Singular Value Decomposition (SVD) where the sum squared error between correspondences is minimized.

4.3.Pose Generation from Image Data

As explained in Section 4.2 the ICP algorithm needs to have an initial guess of the pose difference between the two point clouds. This section describes the implementation of the pose estimation based on RGB and depth data with the Mobile Robot Programming Toolkit (MRPT). This work has used the KLT algorithm [66] for Feature Matching, as it is said to be the most robust tracker in the MRPT library [67], but any other feature matching algorithm should give similar results.

4.3.1. KLT Feature Extraction and Matching

The idea of the KLT feature tracking algorithm is an optical flow algorithm with origins in two papers: [66] and [68]. The algorithm assumes that the displacement in the image between consecutive frames is small. In [66] is stated that the disparity of two images can be expressed as:

$$\boldsymbol{h} \approx \left[\sum_{\boldsymbol{x}} \left(\frac{\partial F}{\partial \boldsymbol{x}}\right)^{T} [G(\boldsymbol{x}) - F(\boldsymbol{x})]\right] \left[\sum_{\boldsymbol{x}} \left(\frac{\partial F}{\partial \boldsymbol{x}}\right)^{T} \left(\frac{\partial F}{\partial \boldsymbol{x}}\right)\right]^{-1} \qquad 13.$$

Where $\boldsymbol{h}$ is the displacement vector of image in image coordinates, F(**x**) is the comparison image, G(**x**) the reference image, $\boldsymbol{x}$ being the image coordinate vector and $\partial/\partial\boldsymbol{x}$ being the gradient operator with respect to **x**. F(**x**) is then displaced with the calculated $\boldsymbol{h}$ and $\boldsymbol{h}$ is recalculated from the new F($\boldsymbol{x}$). Each iteration refines $\boldsymbol{h}$ till convergence in an approach similar to the Newton-Raphson method [69]. The algorithm uses a coarse-to-fine approach during this process, meaning going from the usage of low resolution images to using high resolution images. The smaller the disparity, the better the two images correspond with each other.

The other paper [68] proposes to only keep feature points, which have a window around the point itself, whose eigenvalues of the gradient matrix are larger than some threshold. The lower bound for the value of the threshold is calculated by measuring the eigenvalues for images with an approximately uniform brightness. The upper bound is determined by selecting a set of various types of features such as corners and regions with high texture and examining the maximum eigenvalues in the gradient matrix. The paper states that it is not critical to choose the threshold to be halfway the two bounds.

The KLT algorithm is used to extract features from the first RGB image and compare these features with features extracted from a previous RGB image and the list of previous extracted features that have been retained. In case there are no previously found features a new list of features is composed from the current and previous RGB image. The feature list has a maximum length and features are dropped from the list if a feature was one of the least recently seen features or if it is too near an edge of the RGB image. The removal of RGB features near edges from the list is done as these might not be visible when comparing with the next RGB image and therefore not considered useful to

track. The remaining RGB features are used to generate a pose difference estimate between the current and previous image frame.

4.3.2. Pose generation

For the calculation of the pose difference between the current Kinect data frame and the previous one the corresponding 3D locations of these features are found based on the depth information. The mapping of the RGB image coordinates (x_{RGB},y_{RGB}) to the depth image coordinates is done with the following formulas:

$$x_d = x_{RGB} \quad (14.)$$

$$y_d = y_{RGB} \quad (15.)$$

This is a very simple mapping, but combined with RANSAC in the rigid body transformation estimation process, it proves to be sufficiently accurate. In case the depth image has a fault value at the calculated coordinates, the point is not used for pose generation. All the points that remain are used as an input to calculate the rigid transformation between the current 3D point cloud and the previous 3D point cloud.

4.4.Pose estimation using IMU

Localization can also be performed with IMU sensors. Because IMU data is noisy an Exponential Weighted Moving Average is used. The orientation filter proposed by [70] is used after the filtering. It fuses integration of angular velocity measurements of the gyros with additional information based on the gravitational acceleration vector, reporting accuracy in rotation of around 0.65 degrees. When assuming no external forces working on the system except for gravity, there is a single acceleration vector that will have acceleration components in the x, y and z axis of the accelerometers of the IMU. Normalizing these components, one can calculate two degrees of freedom, being roll and pitch with respect to the earth reference frame.

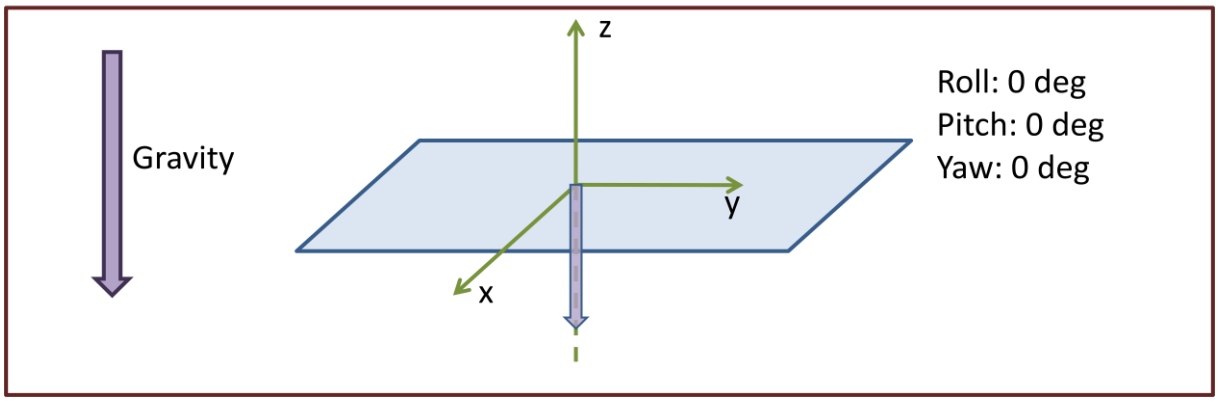


Figure 41 - Acceleration vector decomposition in sensor axis (scenario 1)

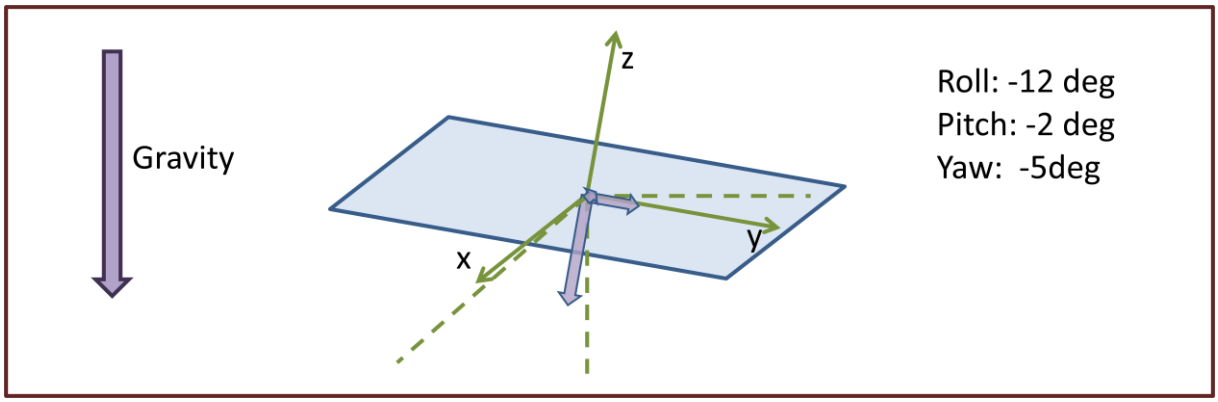


Figure 42 - Acceleration vector decomposition in sensor axis (scenario 2)

One is not able to calculate the yaw of the sensor with respect to the world reference frame. Looking at Figure 41, one can see that any yaw does not change the measurements of the IMU sensor. The measurements for the x and y directions will stay equal to 0 and the measurement in the direction of z would be around -9.8. Performing only a roll or pitch, the rotation does change the measured values. A roll will result in changing the measurements for the y and z axis, and a pitch results in a change in the x and z measurement values. The software used in this research uses the rotation order of "yaw -> pitch ->roll", meaning perform yaw first, pitch second and roll last. Any compound rotation, like the one illustrated in Figure 42, will not introduce changes in the measured values when doing the yaw rotation first. This means that we can use the gravitational force as a feedback for the roll and pitch estimates to combat drift from the gyro measurements, but there is no feedback available to deal with drift that might occur in the yaw component as any yaw does not result in any change in the measurements.

4.5.Implemented mapping algorithms

For this work two mapping algorithms are implemented and compared with each other. The overview of the system as a whole can be seen in Figure 43. The system as a whole is one that revolves around the assumption that the environment does not change while being mapped.

The "Pose Generation: Image Data" block, described in Section 4.3, uses the recorded Kinect data as input and generates a full pose with respect to the previous Kinect data frame if successful. The "Pose Generation: IMU Data" block, as described in Section 4.4, uses accelerometer and gyro measurements as input. Because the IMU samples data at a higher rate than the Kinect, the "Pose Generation: IMU Data" block processes data until the timestamp for the IMU data matches that of the Kinect's current frame. This block only generates the orientation part of a pose. The "Pose Generation: Constant Velocity" block is only used in case the "Pose Generation: Image Data" block fails to calculate a pose. The velocities are calculated based on the previous two poses with respect to the world reference frame, performing an integration to derive position and orientation using the timestamps as time measurements.

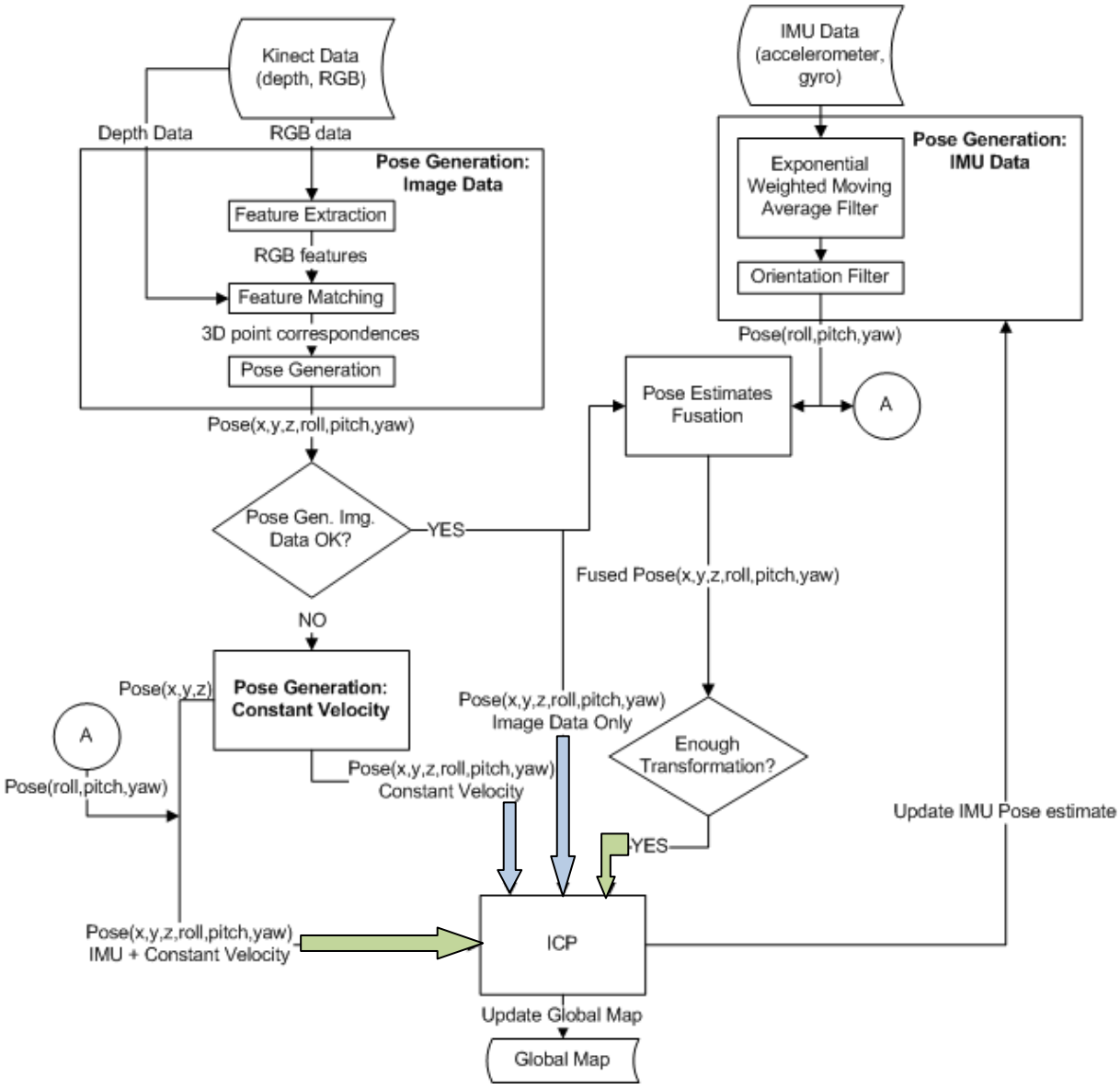


Figure 43 - System configuration

The ICP block, which has been described in Section 4.2, has multiple incoming arcs with different pose estimates. Only one of these is actually used, depending on the specific algorithm and if a pose could be derived from the image data. The blue arcs belong to Intel's algorithm, while the green arcs belong to the IMU-enhanced algorithm. In the specific case of using the algorithm with IMU and image data and the image data failed to localize the Kinect, the "Update Global Map" step is only performed if there was enough movement between the current Kinect data frame and the previous Kinect data frame that was used to update the global map with. Enough movement means that the norm of the displacement is greater than 10cm or one of the orientation parameters is greater than 10 degrees.

4.5.1. Intel's algorithm: ICP and Feature Matching

This algorithm has its roots in the algorithm presented by Intel[7]. It uses only the pose estimate based on image features as input for the ICP algorithm. In case the pose estimation fails to localize the Kinect, it will fall back on the assumption of constant velocity to derive a pose estimate based on previous pose estimates. This algorithm uses every frame that has been recorded. Please take note

that this is not the exact same implementation as the mapping algorithm proposed by Intel Seattle Research, as this implementation does not have loop-closure capabilities (trying to exploit information from all the frames to further align all the frames to reduce global residual error) or the post-processing step that merges nearby points to create a Surfel representation (a representation similar to point clouds, using disks facing the viewer instead of points[71]). For the sake of convenience this implementation will be referred to as “Intel’s algorithm”.

4.5.2. IMU-enhanced algorithm: ICP, Feature Matching and IMU

This algorithm fuses IMU orientation data with pose estimates based on image data. As the IMU data can be affected by drift the Image data can be used as feedback to combat the drift. This is especially the case for yawing the Kinect, as the pose generation based on IMU data does not have any internal feedback to deal with drift in this component. The data is fused as follows:

- If the difference between IMU orientation and visual based orientation in each rotation axis is less than 5 degrees: 50% IMU data, 50% Image data.
- Else if the difference in each rotation axis is within 5 to 10 degrees: 25% IMU data, 75% Image data.
- Otherwise only use Image data.

This fusing scheme has one weakness. When the Kinect is viewing a repetitive pattern like a tiled wall or a barred fence, the aperture problem arises, which is a weakness in optical flow algorithms like KLT. Figure 44 illustrates this problem. Depicted is a scenario where we are viewing a tiled surface with two consecutive image frames, colored blue. Image feature extraction is very likely to track the corners of the tiles. The two frames moved a significant distance, while the corners seen in the first frame are not the same as in the original second frame. However, they are really similar and the Image tracking will consider those corners to be the correspondences. As such, the localization system will wrongly identify the position of the first frame to be as indicated in red, therefore calculating only a small transformation where in reality a far bigger transformation has happened. Also, the pose generated might not even be in the same general direction as the original second frame with respect to the first.

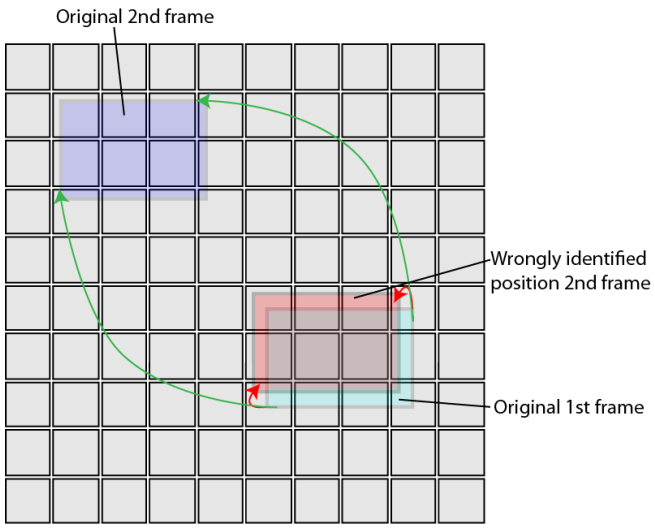


Figure 44 - Illustrating Aperture Problem

The IMU data does not suffer from this problem and the calculated orientation will be more similar to the real orientation. When the disparity between the orientation based on visual information and IMU data becomes too large, the visual information is trusted more while the values from the IMU based calculations represent the real movement better. As a result an inaccurate pose estimate is provided to the ICP algorithm

The algorithm will only use frames to update the global map if there is a big enough transformation from the last used frame. As the algorithm assumes a static environment there will be no new information visible by the Kinect when it is not moving. If the Kinect would remain static Intel's algorithm would continuously add redundant information into the map, wasting both processor and well as memory resources. This can be prevented by only adding data to the map if sufficient movement has been detected. When there is enough transformation or rotation, the fused result is used as input to the ICP algorithm and after refinement the point cloud is added to the map.
In case the localization based on image data fails, the Kinect will fall back on using the orientation estimate based on the IMU data and the position coordinate determined by a constant velocity assumption. ICP is used to refine these estimates, but the frame is only added if the total transformation from the last update frame is big enough.

5. Evaluation

In order to evaluate Intel's and the IMU-enhanced mapping algorithms discussed in Section 4.5, various mini-benchmarks are used to compare them with each other. This will give insight and support to whether or not additional IMU data can be useful when performing indoor mapping with the Kinect.

5.1.Mini-Benchmarks

The mini-benchmarks are various scenarios that pose challenges to the Intel inspired algorithm discussed in Subsection 4.5.1. The tests are performed to show where the addition of IMU data can help overcome these challenges. The tests are run on an Acer TravelMate 5740ZG laptop with a 2Ghz dual core and an ATI Mobility Radeon HD5470 graphics card. To relate closest to real life situations for human operated Kinect recordings, the recordings are made by holding the Kinect in hand and moving it around. The Microsoft Kinect captures at a rate of around 4Hz, which is lower than the actual frame rate of 30Hz and is probably caused by a memory bottleneck. IMU data is provided at a rate measured of 184Hz by a Parrot AR.Drone quadrotor[72] that is attached to the Microsoft Kinect.

5.1.1. Violation of the Static Environment Assumption

The mapping algorithms revolve around the assumption that the environment does not change, as stated in Section 4.5. An event that triggers a violation of this assumption is when a person walks into the view of the camera or the opening of a door while mapping.

Setups

The first thing that is tested is how the Intel's and the IMU-enhanced mapping algorithms behave when the Kinect remains in a static location while a person walks into and out of view. For one recording the person will walk across the field of view of the Kinect and for the second recording a person will walk up to and away from the Kinect. In order to recreate a worst case scenario where localization can fail because a person walks into the field of view of the Kinect, a narrow corridor is used as the environment. The setup is shown in Figure 45.

Figure 45 - Mapping environment static Kinect setup

Figure 46 - Mapping environment moving Kinect setup

The second test is with a *moving* camera where a person walks into the field of view. The Kinect will follow the movement of that person by rotating the Kinect only and therefore the recording takes place in a more spacious environment as shown in Figure 46.

Results

The result of the scenario with the person walking across is shown in Figure 47 and the scenario where the person approaches and walks away from the Kinect is shown in Figure 48. The blue line is the number of features that are found by the KLT feature matching algorithm, which is always higher than the inliers. The tests where the Kinect remained static while a person through of the view of the camera show a drop in inliers (in red), correspondences that are flagged to be reliable and are used for image based localization, at the instance the person walks into the field of view of the camera. This drop can be seen in Figure 47 and Figure 48, where the green highlighted areas indicate the frames where the person is visible by the Kinect.

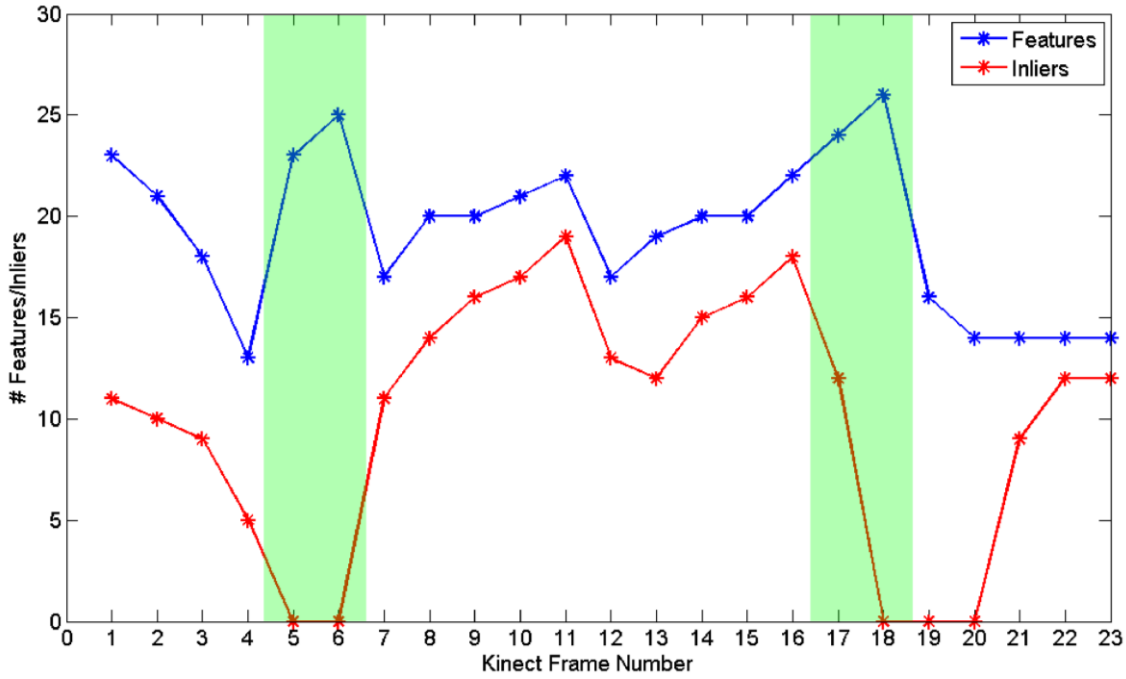


Figure 47 - Features/Inliers found for static Kinect; Person walks across

The features introduced and obstructed by the person walking into view as well as the shadow that is being cast by the person changes the number of features found. The person walking into the field of view changes the depth measurements between correspondences and because of that they are removed by the RANSAC algorithm during the pose estimation step. As the number of inliers drops to zero, localization by RGB information using rigid pose estimation is not possible anymore. During those failures, the fallback mechanism is called to give a pose estimate. In case of Intel's algorithm this estimate comes from the constant velocity assumption and as the Kinect does not move the constant velocity based localization is working perfectly.

IMU orientation estimates also prove to be accurate, making the constant velocity estimates and IMU orientation estimates useable as initial guess for the ICP and provides correct localization.

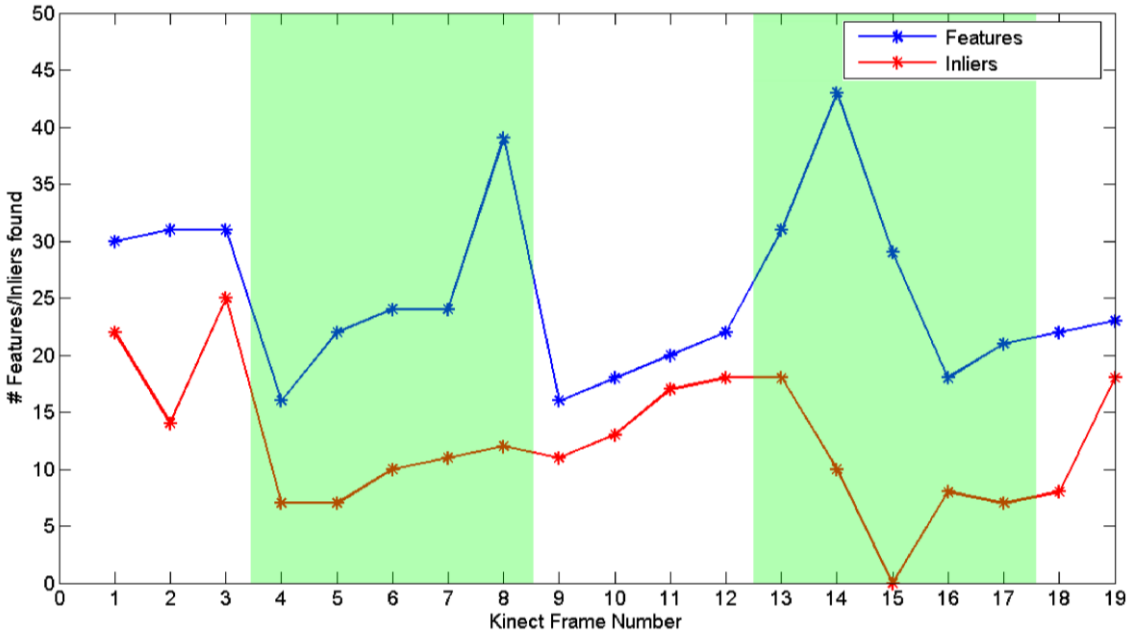


Figure 48 - Features/Inliers found for static Kinect; Person walks up to and from Kinect

The mapping results that correspond to the feature and inlier counts plotted in Figure 47 and Figure 48 can be seen in Figure 49 though Figure 52. The results shown in Figure 49 and Figure 50 are from Intel's algorithm where the first corresponds to the scenario where the person walks across. The latter corresponds to the scenario where the person approaches and walks away from the Kinect. Even though the Kinect localizes correctly, the implementation of Intel's algorithm does nothing to prevent mapping the person walking into the field of view of the Kinect as it uses all frames and adds them to the global map.

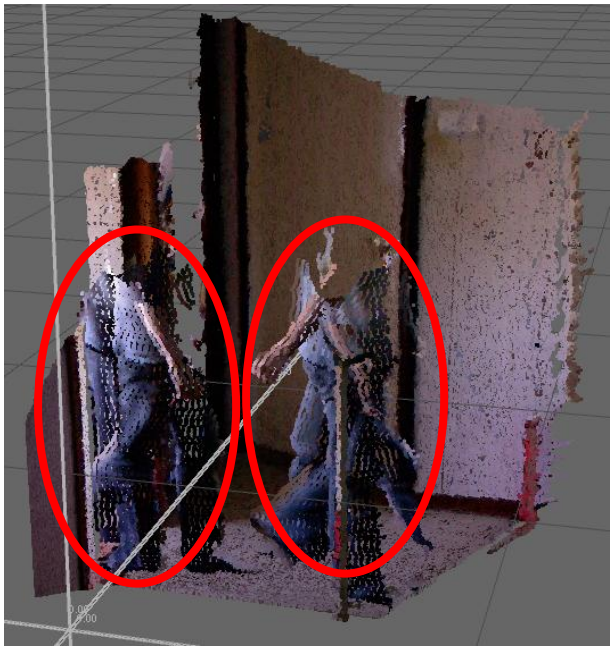


Figure 49 - Mapping Result (Intel's algorithm): Person walking across Kinect

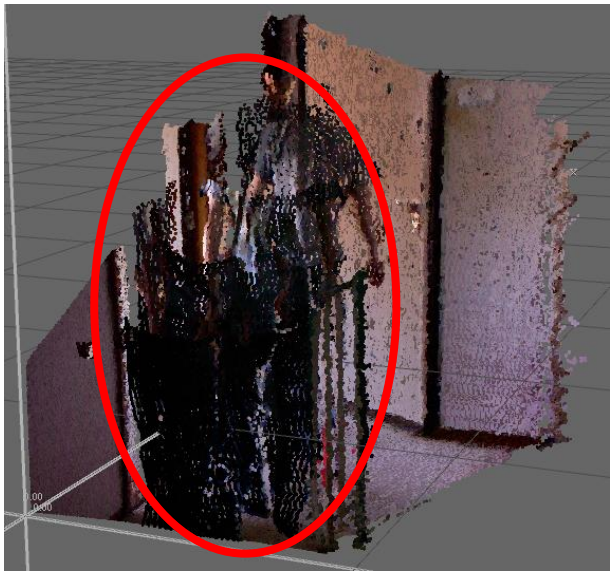


Figure 50 - Mapping Result (Intel's algorithm): Person walking up to and from Kinect

The IMU-enhanced algorithm gives a cleaner result with less unwanted points added into the map as it does not use every Kinect frame. This is shown in Figure 51, which corresponds to the scenario with a person walking across the Kinect, and Figure 52 for the scenario where the person walks to and

from the Kinect. Figure 51 actually only comprises of a single point cloud that is added into the global map, while Figure 52 contains three point clouds. Comparing Figure 49 with Figure 51 is can be seen that the IMU-enhanced algorithm completely prevents the person from being added into the global map, while comparing Figure 50 with Figure 52 shows that the unwanted information, being the person walking in front of the Kinect (highlighted with red in all figures), is less present in the result of the IMU-enhanced algorithm. This is because the IMU-enhanced algorithm only adds frames into the map if sufficient movement of the Kinect is detected.

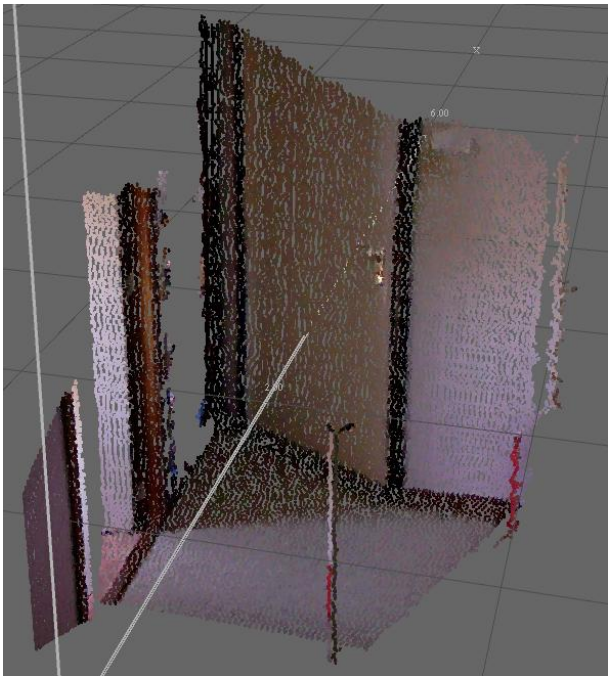


Figure 51 - Mapping Result (IMU-enhanced algorithm): Person walking across Kinect

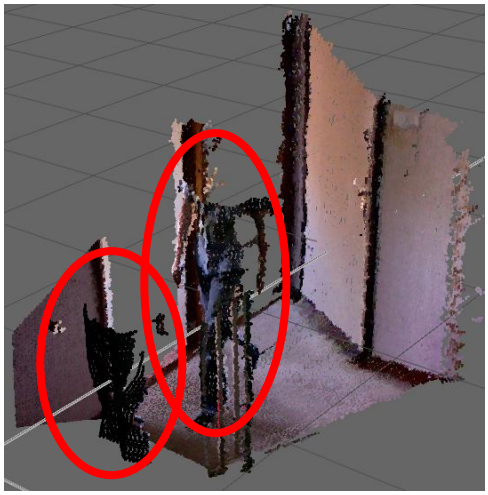


Figure 52 - Mapping Result (IMU-enhanced algorithm): Person walking up to and from Kinect

The second test is similar to the first one, but the main difference is that the Kinect is also moving. The mapping results for both Intel's algorithm and the IMU-enhanced mapping algorithm are shown in Figure 53 and Figure 54 respectively. This data set proved to be more difficult for aligning the various point clouds as image based localization failed a few times and the fall back mechanism was used instead to provide an initial guess for the ICP algorithm. Where Intel's algorithm did a reasonable job, the IMU-enhanced algorithm results have some severe alignment issues mainly because of an imperfect orientation estimate at the beginning. The pitch and roll values of the orientation estimate for the first Kinect data frame are overestimated, with the result that IMU-enhanced algorithm was not correctly aligned with the world reference frame. The algorithm does converge to the correct orientation with respect to the world reference frame, as one can see the left part of the roof that is horizontal, but only does this too late in the recording.

A modified version of Intel's algorithm, where frames are only added to the map after the Kinect has moved enough like implemented in IMU-enhanced algorithm, is also used with this data set to see if this line of thought still improves the results shown in Figure 53. The results for this modified algorithm are shown in Figure 55 and also show imperfect alignment of point clouds which were similar to Intel's algorithm, although slightly worse. While there is less unwanted information, being the person walking into the mapping area, some point clouds have been pulled out of alignment because of the point mass that is from the person in the center. This pulling out of alignment can be

clearly seen from the green railing, which is garbled more in the result of the modified version than in the original. The railing has been highlighted with green in the mapping results shown in Figure 53 though Figure 55.

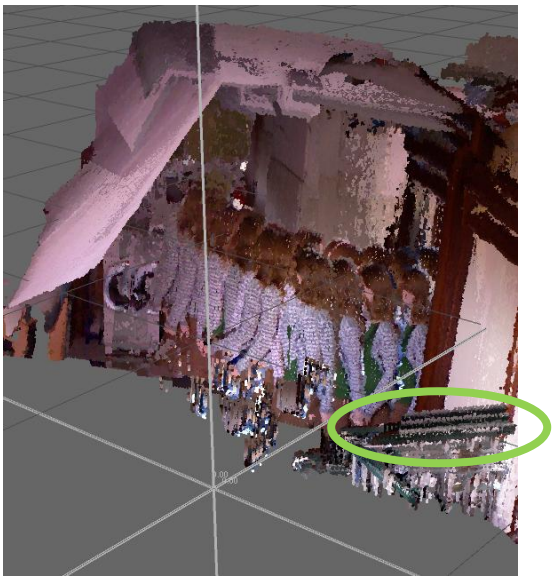


Figure 53 - Mapping Result (Intel's algorithm): Person Walking while Kinect is moving

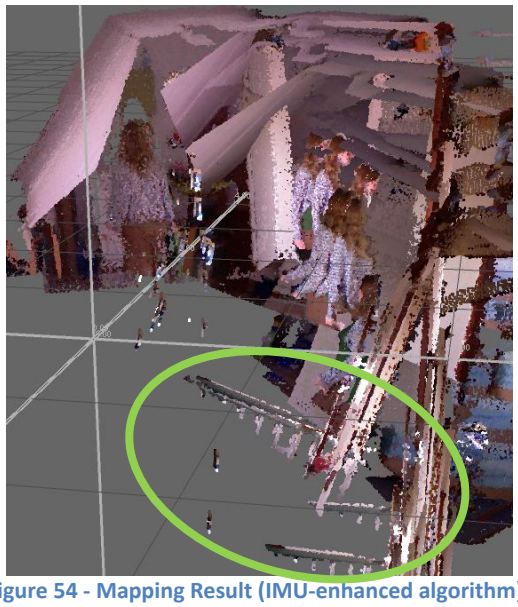


Figure 54 - Mapping Result (IMU-enhanced algorithm): Person Walking while Kinect is moving

Figure 55 - Mapping Result (Modified Intel's algorithm): Person walking while Kinect is moving

5.1.2. Low light and total darkness

Low light and total darkness will make it difficult to perform feature extraction, as these operations rely on contrast differences in an image. Without contrast differences the KLT feature matching algorithm produces no features. This results in the "Pose Generation: Image Data" block to fail calculating a pose.

Setup

To test how Intel's algorithm and the IMU-enhanced algorithm behave in low-light and total darkness conditions, a bed room is mapped in absence of light. The room has its windows covered by drapes, which allows only bit of sunlight in, creating lowlight conditions in areas around these windows. As there are no other light sources available the areas away from the windows, these present total darkness conditions.

Total darkness prevents the generation of pose information based on visual data, where low light conditions allow for only a few usable image features for image based localization. Because of the

lack of visual features that can be extracted, Intel's algorithm will rely mostly on the fallback mechanism of using the constant velocity assumption to provide an initial guess for the ICP algorithm to fit the point clouds. The IMU-enhanced algorithm will also rely on the fallback mechanism and will use the IMU data for orientation information while the constant velocity assumption provides a location estimate. A modified version of Intel's algorithm is also run, for which the ICP algorithm does not get an initial guess of the estimated movement of the camera, in order to investigate if the initial guesses were really a necessity for this test-set.

Results

In order to get a grasp of what the Kinect is recording during this test, Figure 56 provides a sample from the data stream. The right part of the image shows how the drapes in front of the window allow sunlight to pass but not enough to light the room as a whole. As such, the results of the mapping process will be mostly black masses of points that cannot be differentiated by visual inspection. Therefore there will be no images of the resultant maps created by the three mapping algorithms as they do not give useful information to the reader.

Figure 56 - 20th RGB frame from Kinect recording for lowlight and total darkness mapping

When analyzing frames being added one after another in the global map, the results show that when movement is slow enough the Kinect can localize based on the available depth data only, like what is done by using the modified version of Intel's algorithm that does not use any form of initial guess provided to the ICP algorithm.

As was to be expected, there was not enough RGB information to allow feature extraction. Intel's algorithm and the IMU-enhanced algorithm both relied almost only on its fallback mechanism, with the exception when the window became visible. This provided enough features to allow visual localization to be performed for that particular section in the test data.

It has to be noted, that because of the lack of RGB information, the point cloud map is basically a black mass of points which does not allow for easy visual inspection. Although mapping is possible in total darkness scenarios, the 3D map itself is not readable by humans. The gathering of 3D information of the environment is still possible and probably also useful for aiding the path finding of a robot for example.

5.1.3. Movement speed of the Kinect

As the low-light and total darkness test showed: when movement is slow enough, one can actually correctly localize the Kinect without the aid of an initial guess for the ICP algorithm. This leads to the

question: where is the boundary of moving slow enough and is there a threshold where IMU data would allow higher movement speeds?

Setups

A part of a room is mapped by panning the camera at different speeds. The speeds are labeled as: Slowest, Slow, Medium, Fast and Fastest. Medium speed represents panning like a person would use a camcorder to record the environment, which is rotating at a rate of ~30 degrees per second. Slowest is at a rate ~10 degrees/s, Slow at ~20 degrees/s, Fast at a rate of ~40 degrees/s and Fastest at around ~55 degrees/s. Only rotation has been covered as this would be the ideal case of IMU data usage. Both Intel's algorithm and the IMU-enhanced algorithm, and a modified version of Intel's algorithm that does not use an initial guess for ICP, are executed for each of these recordings.

Results

The results of Intel's algorithm are shown in Figure 57 through Figure 61, the modified version of Intel's algorithm has its results shown in Figure 62 through to Figure 66 and lastly the results of the IMU-enhanced algorithm can be found in Figure 67 though Figure 71. The red wireframe is the ground truth model based on the measured dimensions of the main objects in the room and Figure 1 and Figure 2 from the introduction show how the room looks like. Some annotations have been added to the figures to aid the reader in understanding what can be seen in the images.

Figure 62 and Figure 63 show that when using no initial guess for the ICP algorithm in Intel's algorithm for "Slowest" and "Slow" scenarios a sever misalignment takes place with an error of around 50 cm to the side, which affects the placement of all subsequent point clouds for this particular test set. While these two results are of mediocre quality, the results for "Medium" up to "Fastest" (Figure 64 to Figure 66 respectively) scenarios are plain unusable maps demonstrating that the ICP algorithm really needs a correct initial guess when dealing with angular velocities higher than 30 degrees/s.

Intel's algorithm *with* an initial guess provided by the visual data gives reasonable accurate maps for "Slowest" and "Slow" scenarios, which can be seen in Figure 57 and Figure 58, with a few over- and underestimates. There is also a very apparent misalignment at the closet. For the "Medium" scenario (Figure 59) the visual based localization fails because of a lack of inliers for pose estimation and the constant velocity assumption overestimates the rotation of the Kinect creating a curve in an otherwise straight wall. The last two scenarios, "Fast" and "Fastest" (Figure 60 and Figure 61 respectively), the visual localization fails and the constant velocity assumption takes over. In the first case the constant velocity assumption incorrectly provides a pose estimate with an error of around 80cm, which causes all subsequent point clouds to be placed at a wrong place too with respect to the ground truth. The "Fastest" scenario proved to be too fast for visual localization to be possible, thus relying on the constant velocity assumption. As the Kinect was static before it was rotated with an angular velocity of about 55 degrees/s, the constant velocity assumption suggests that there will be no movement. As such, the point clouds are fitted over each other by the ICP algorithm. As such, we can say the mapping has failed for the "Fast" and "Fastest" scenarios.

The IMU-enhanced algorithm shows also reasonable accurate maps for the "Slowest" and "Slow" cases, shown in Figure 67 and Figure 68, just like Intel's algorithm is able to provide for these scenarios. For the "Slowest" scenario the map has an underestimate of the z-coordinates of the point clouds, which results in a mismatch with the ground truth of around 20 cm. The result for the "Slow"

scenario is negatively affected by a single alignment that was incorrect and affected the subsequent point clouds to be placed off. Just like with Intel's Algorithm, the IMU-enhanced algorithm has to rely on its fallback mechanism when the visual localization fails because of a lack of inliers. As the distances between added and ICP corrected point clouds is bigger than in Intel's algorithm, the results are less accurate resulting in this specific situation in estimating an incorrect negative displacement in the z-axis that the ICP algorithm itself was unable to correct for. The same problem also occurs in the "Fast" and "Fastest" scenarios, which are shown in Figure 70 and Figure 71.

Comparing Intel's algorithm, the variant on Intel's algorithm without initial guess and the IMU-enhanced algorithm one can see that for the "Slowest" and "Slow" scenarios Intel's algorithm and the IMU-enhanced algorithm provide a more accurate map than the algorithm not using any form of initial guess, showing the added value of providing an initial guess to the ICP algorithm when aligning point clouds. Interesting to note is that while Intel's algorithm gives a more accurate result than the IMU-enhanced mapping algorithm for the "Medium" scenario, the IMU-enhanced algorithm in turn was more accurate than Intel's algorithm for the "Fast" scenario

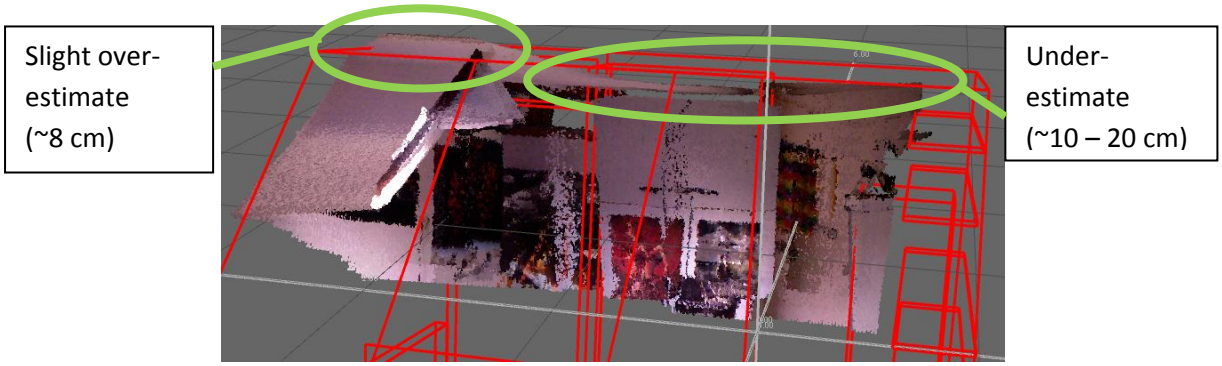


Figure 57 - Mapping Results (Intel's algorithm); Slowest

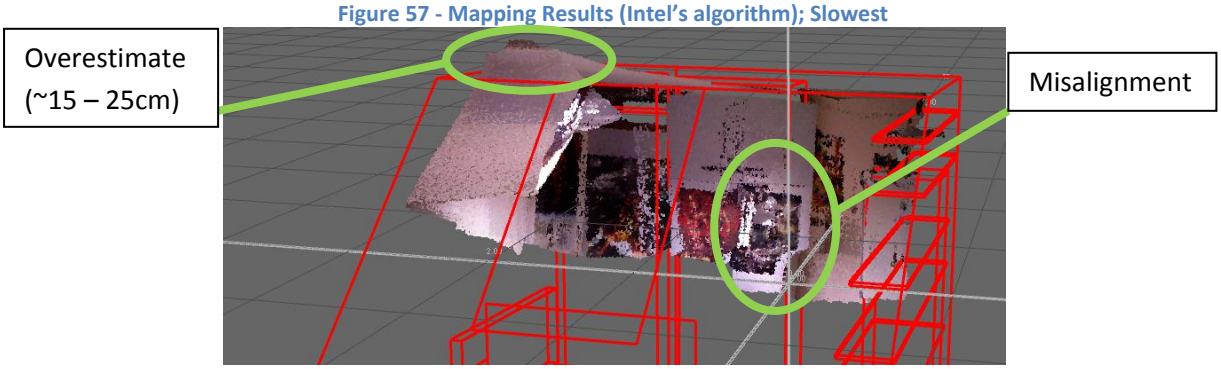


Figure 58 - Mapping Results (Intel's algorithm); Slow

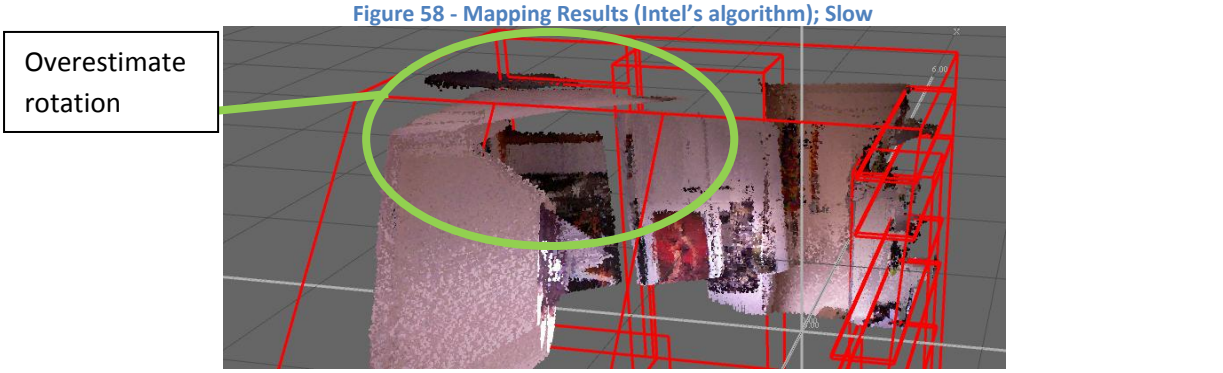


Figure 59 - Mapping Results (Intel's algorithm); Medium

Failing rotation estimation by constant velocity assumption.

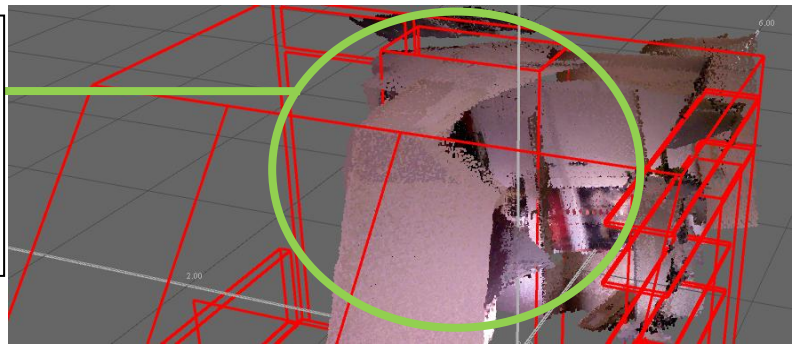


Figure 60 - Mapping Results (Intel's algorithm); Fast

Failing rotation estimation by constant velocity assumption.

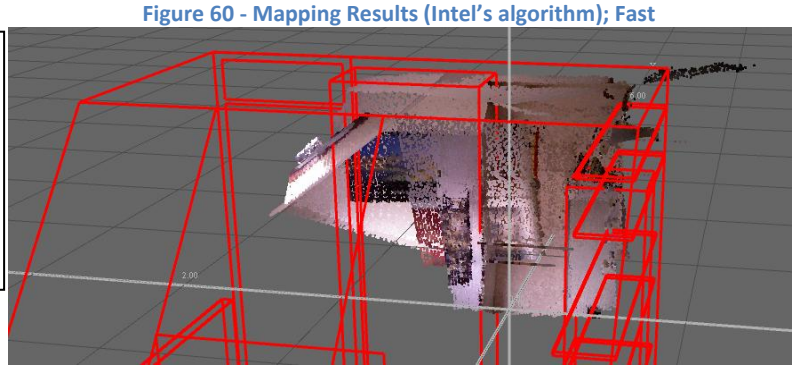
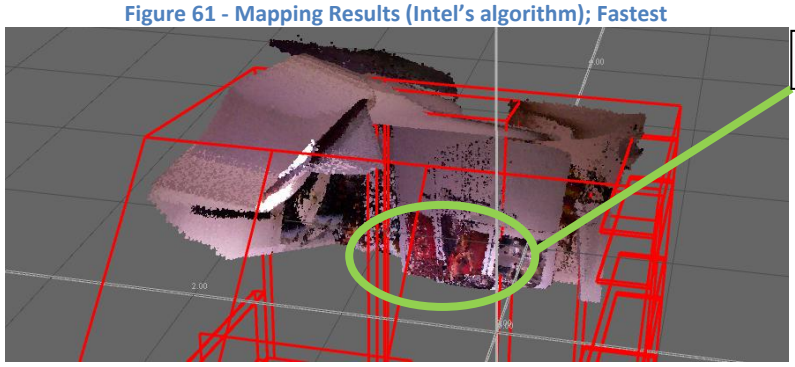
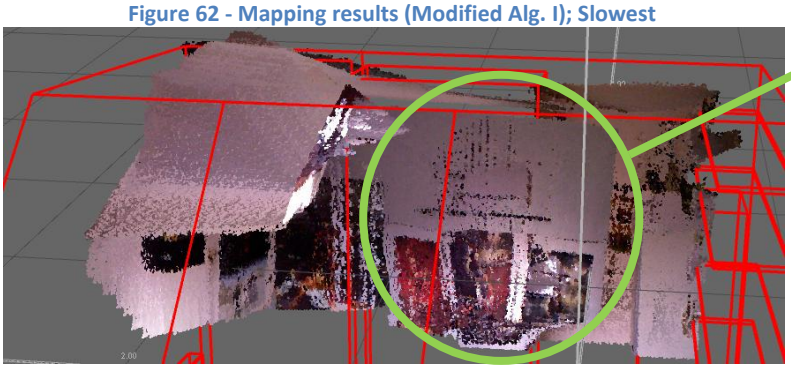


Figure 61 - Mapping Results (Intel's algorithm); Fastest



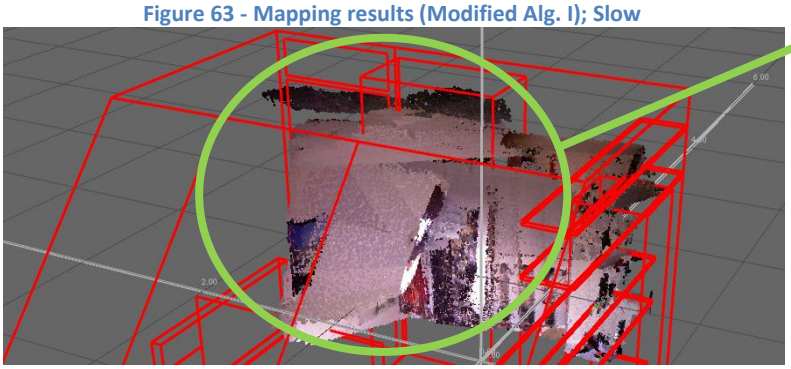
Misalignment

Figure 62 - Mapping results (Modified Alg. I); Slowest



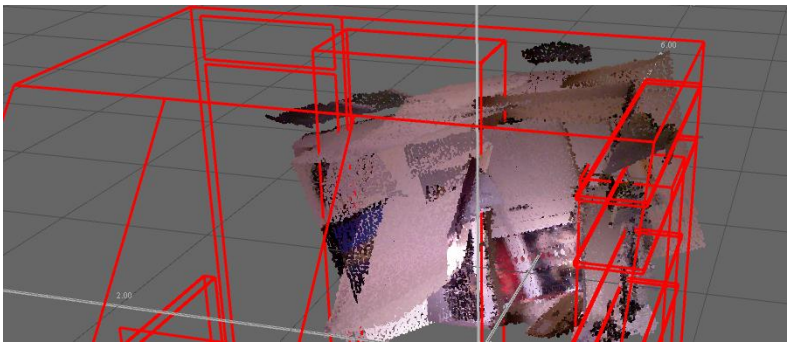
Misalignment

Figure 63 - Mapping results (Modified Alg. I); Slow



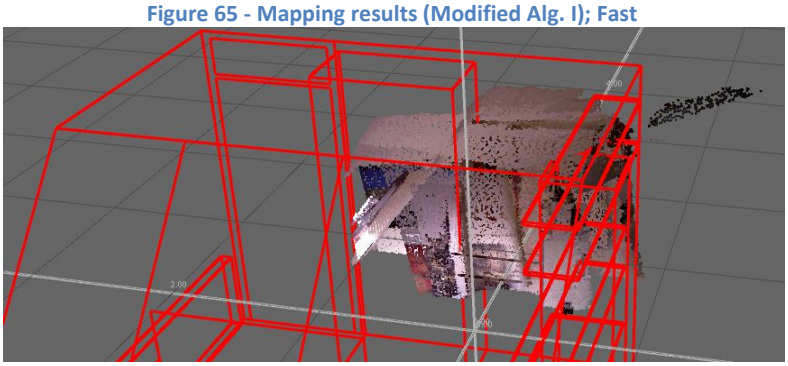
Severe over-estimate rotation

Figure 64 - Mapping results (Modified Alg. I); Medium



Failing ICP fitting

Figure 65 - Mapping results (Modified Alg. I); Fast



Failing ICP fitting

Figure 66 - Mapping results (Modified Alg. I); Fastest

Under-estimate (~20cm)

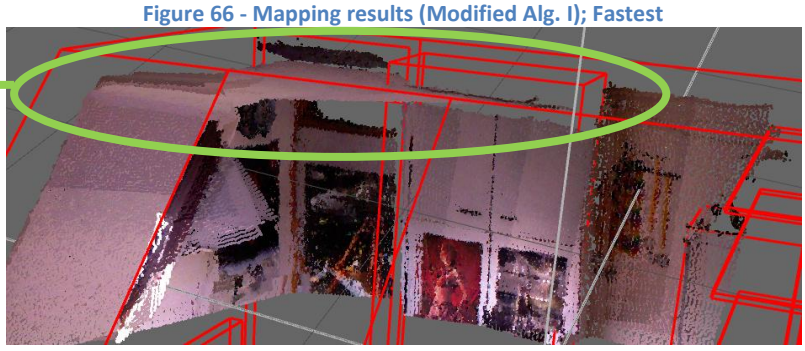
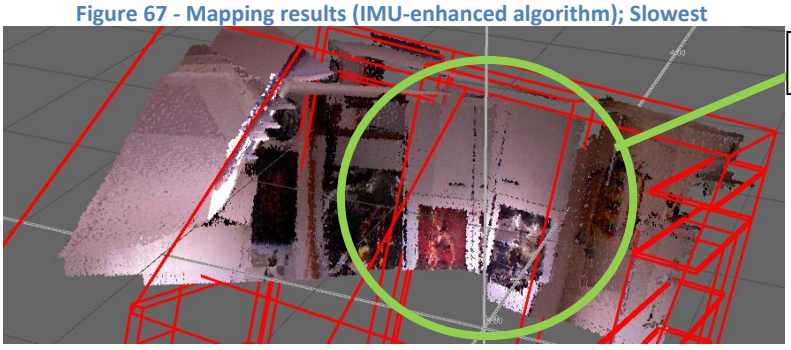


Figure 67 - Mapping results (IMU-enhanced algorithm); Slowest



Misalignment

Figure 68 - Mapping results (IMU-enhanced algorithm); Slow

Constant velicity assumption fails to provide accurate (x,y,z) + overestimate rotation

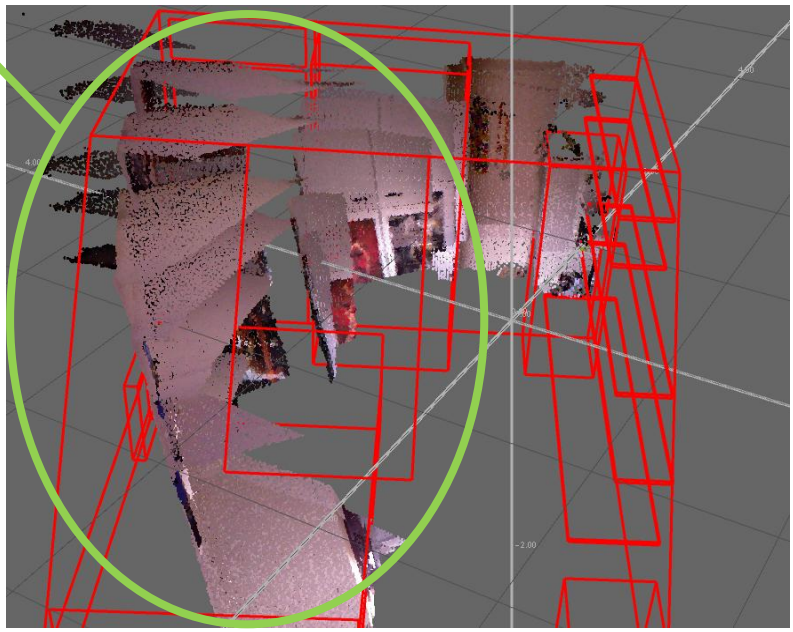


Figure 69 - Mapping results (IMU-enhanced algorithm); Medium

Constant velicity assumption fails to provide accurate (x,y,z)

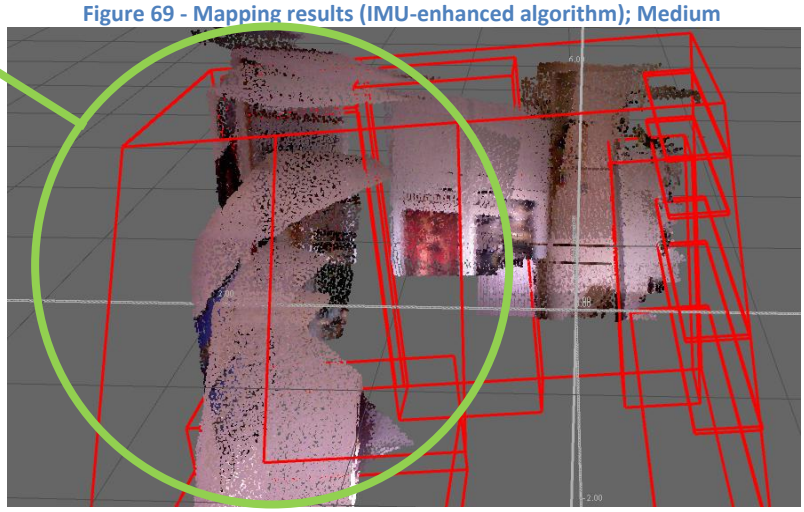


Figure 70 - Mapping results (IMU-enhanced algorithm); Fast

Failure of constant velocity assumption

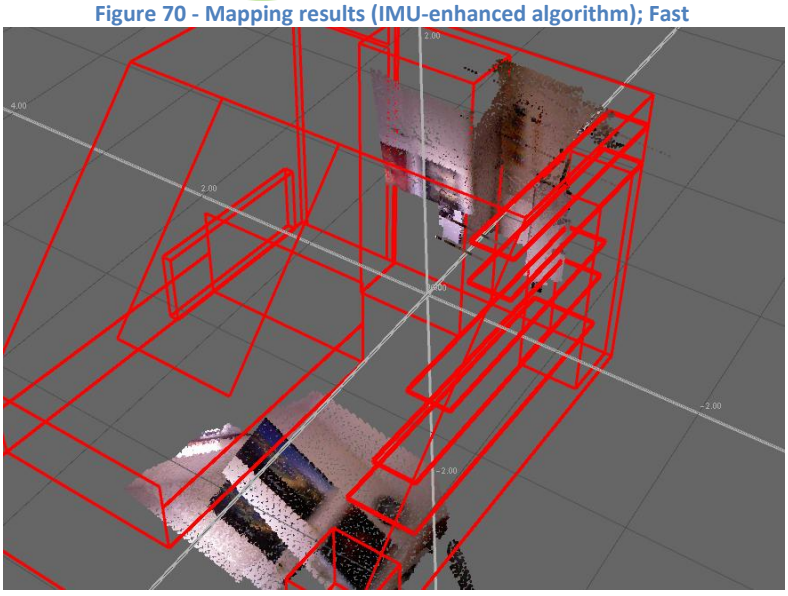


Figure 71 - Mapping results (IMU-enhanced algorithm); Fastest

5.1.4. Convergence time

The ICP algorithm uses an initial translation and rotation estimate of two point clouds, from one point cloud with respect to the other, to align these two point clouds. While the test from Subsection 5.1.2 suggests it is possible to use the ICP algorithm without an initial guess when the pose difference between subsequent point clouds is small (i.e. the camera moves slowly), Subsection 5.1.3 demonstrates that using an initial guess can provide superior mapping quality. However, different types of initial guess methods might have an impact on the time spent doing the refinement.

Setups

For this test four types of input are used:

- Pose estimates based on image data only,
- Pose estimates based on fused Image and IMU data (a variant on the IMU-enhanced algorithm, that uses *every* frame instead of only using frames after the estimated movement of the Kinect is over 10cm or 10 degrees after the frame that has been added the most recent),
- Pose estimates based on the constant velocity assumption,
- The pose of the previous frame.

A data set is created in a bedroom of 3x4m that has posters, two closets and a desk with planks visible in the recording. Pictures of the room are shown in Figure 1 and Figure 2 from the Introduction chapter. The recording starts with the Kinect facing the closet with the two posters, rotates counterclockwise towards the left wall, then rotates clockwise back towards the other wall with the desk and planks. The Kinect is then tilted up and down to record the various planks and then, while still tilted downwards, the Kinect rotates counterclockwise back towards the bottom part of the closet, which it faced when the recording started. The recording was under good conditions without shocks or dynamics in the environment, and the environment was well lit. The recording consists of 210 frames.

The same data set is also used to demonstrate the mapping quality of Intel's algorithm and the IMU-enhanced algorithm in Section 0. As such, the mapping results themselves are discussed in that section instead of in this subsection.

Results

The mapping using only the constant velocity assumption fails quite fast. An overestimate resulted in two point clouds to be not overlapping each other causing the ICP algorithm to be unable to pull it into the right spot. From there the constant velocity estimates grew larger and resulted in the localization to fail. As such, there are no timing measurements to compare with.

The timing results of the other 3 algorithms are plotted as an empirical cumulative density function in Figure 72 in order to visualize the distribution of time measurements. Additionally, the 50^{th} and 90^{th} percentiles are given in Table 7. Looking at the CDF plots one can see that the distributions show similar trends. Intel's algorithm has the best timing results all round, while the IMU-enhanced algorithm and the variation on Intel's algorithm that uses no initial guess have very similar time measurements distributions.

	50%	90%
Intel's algorithm	3.226 s	6.727 s
No initial guess	4.028 s	7.482 s
IMU-enhanced alg. (using every frame)	4.147 s	7.577 s

Table 7 - 50th and 90th percentile of the timing measurements

While Intel's algorithm has a faster convergence time per frame, the normal IMU-enhanced algorithm does normally not use every frame. Instead, it only uses frames after sufficient movement of the Kinect has been estimated. While the IMU-enhanced algorithm might take 1.2 to 0.7 seconds longer per frame, it does not have to spend its convergence time on every single frame as the ICP algorithm is only called when enough movement takes place with the Kinect. This means that in the end the IMU-enhanced algorithm can possible execute faster than the Intel counterpart, depending on the movement of the camera (slow moving Kinect giving the advantage to the IMU-enhanced algorithm).

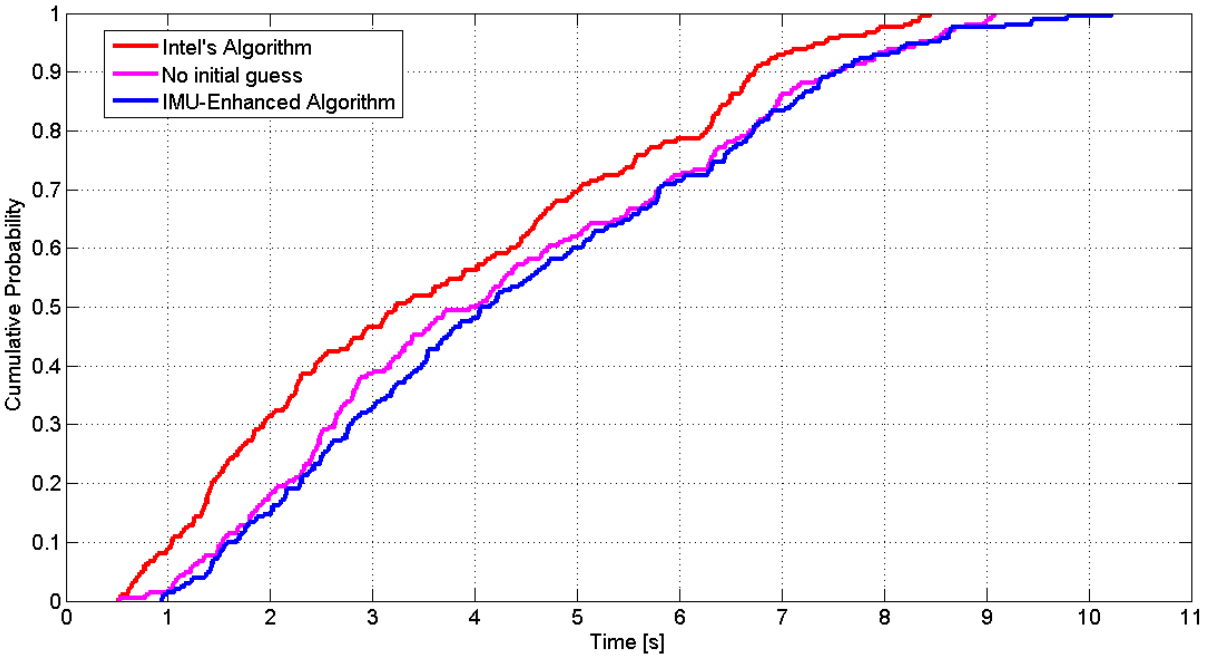


Figure 72 - Empirical Cumulative Density Function of the ICP timing measurements

5.1.5. Reflective surfaces

Reflective surfaces create a challenge when relying on visual information as the surface itself creates a dynamic situation when the Kinect moves while viewing the reflective surface. This can present challenges to RGB-based localization.

Setup

A section of a room is recorded with a dresser that has a mirror on it. The Kinect pans across the dresser in order to see how the two algorithms are able to deal with this challenge as features can be both directly visible on an object and on its mirror-image, which can cause matching-errors, or depth information being perceived in front and behind the mirrors surface as demonstrated in Subsection 3.5.6. The environment is shown in Figure 73.

Results

The mapping result for Intel's algorithm is shown in Figure 74 and the result for IMU-enhanced algorithm in Figure 75, where the mirror edges are highlighted in green. As the IR light reflects in the mirror as discussed in Subsection 3.5.6 the surface itself has no depth information associated with it. The mirror surface is regarded as an opening or corridor where objects are virtually behind the mirror. Both algorithms fail to correctly deal with the reflective surface. Intel's algorithm fails because RGB feature associations are made between the real chair and the reflection in mirror. Therefore there are very few inliers, which causes the RGB localization to fail at times. As a result, the constant velocity assumption takes over and makes an overestimate in distance for which the ICP is unable to correct. This creates the point mass behind the mirror in Figure 74.

Figure 73 - Mapping environment "Reflective surfaces"-test

Misalignments in the results of the IMU-enhanced algorithm are mostly due to over or under estimates of the orientation or location for which the ICP algorithm was not able to correct (perfectly). This is especially visible for the chair towards the front side of the mirror. Here the point cloud belonging to the chair is placed too far below the corresponding chair points in the previous frame. Comparing both results shows that IMU-enhanced algorithm seems to be less affected by the mirror than Intel's algorithm.

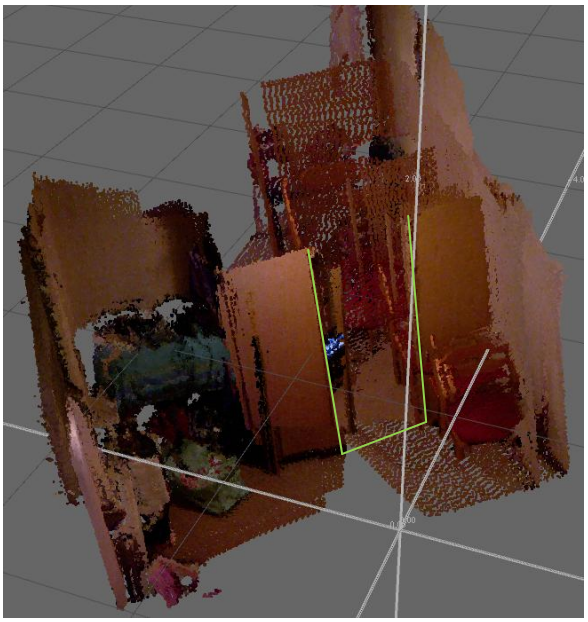


Figure 74 - Mapping Results (Intel's algorithm)

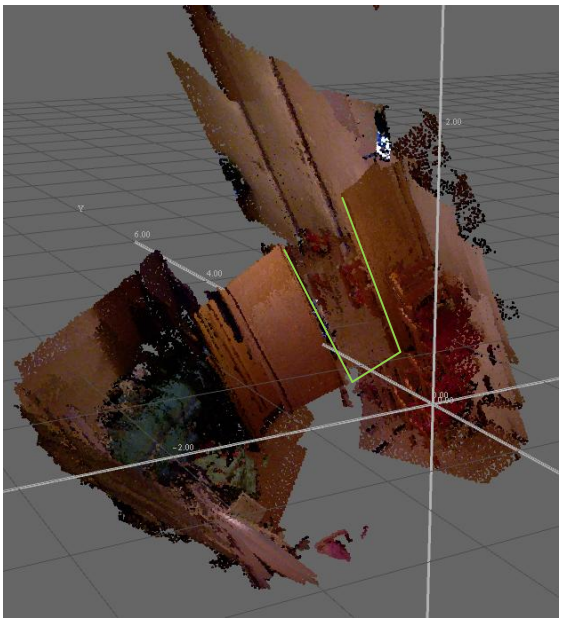


Figure 75 - Mapping Results (IMU-enhanced algorithm)

5.1.6. Featureless environment

A featureless environment such as a plain white wall also causes problems in image based localization because of the lack of contrast. But there is a second way to look at featureless environments by looking at depth data. A single flat surface in itself might introduce challenges for the basic ICP algorithm, as any displacement with the camera will not create a change in what the 3D surface looks like: a plain flat surface. With a given initial guess, the ICP might itself be the cause to misalign the two point clouds even if there is sufficient RGB data for localization. This is a known problem with ICP and a solution to this has been proposed in [7] using the visual features as described. With this in mind, while IMU data might be able to localize in a featureless environment, the basic ICP algorithm might worsen the estimate instead of improving it.

Setup

In order to demonstrate the problems with mapping featureless environment, two recordings have been made of a flat surface. The surface itself is painted with plain white textured paint. While this might allow for very few features to be extracted, the Kinect measurement accuracy will not be able to pick up the texture in its depth measurement. The second recording will have sufficient RGB information in it by means of random pages from a magazine being attached to the surface. The camera will move about two meters in one direction and back again, alongside the flat surface.

Figure 76 - 62th RGB frame from data set for surface with random pages

Results

The results for Intel's algorithm are shown in Figure 77 and Figure 78. The scenario with the plain white surface gives few RGB features for the RGB localization to work with. The frames that did have an initial guess based on RGB information but the estimates do not conform to the movement of the Kinect itself. Most point clouds are fitted over each other as a result of this. RGB localization also fails a few times when the constant velocity assumption takes over and as a result also gives initial guesses with small movement.

The results with additional RGB features (Figure 78) show the localization of the Kinect proves to be conforming to the movement of the Kinect. Occasional misalignments do happen according to the intuition explained in the introduction of this subsection.

The IMU-enhanced algorithm does not perform any better in the scenario with only the plain white surface, shown in Figure 79. Just like with Intel's algorithm, the RGB localization fails and the constant velocity assumption is used. In this case the constant velocity assumption makes an overestimate of the movement of the Kinect, which the ICP algorithm was unable to correct. From that point on, localization breaks down completely.

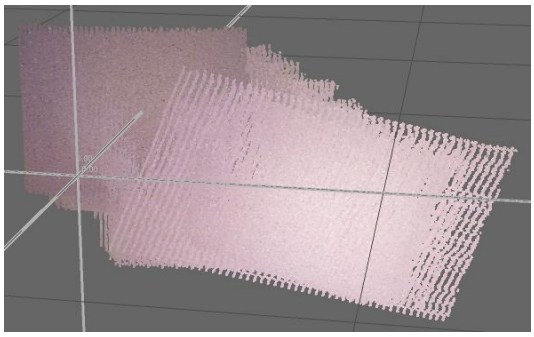


Figure 77 - Mapping Results (Intel's algorithm): Plain white surface

Figure 78 - Mapping Results (Intel's algorithm): White surface with additional RGB information

Misalignments caused by the ICP algorithm are more present in scenario with added RGB information. The results are shown in Figure 80. A total of eight frames are pulled out of alignment because of the ICP algorithm, more than with Intel's algorithm. This might be due to the fact that IMU-enhanced algorithm only uses ICP when enough movement of the Kinect has been estimated. As two point clouds are further away from each other, the ICP algorithm is more inclined to pull them closer to each other.

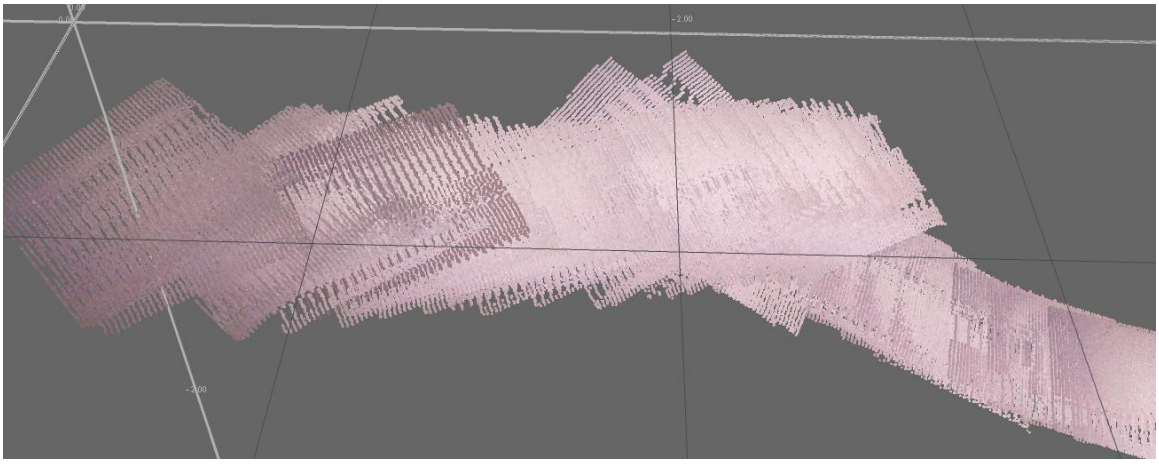


Figure 79 - Mapping Results (IMU-enhanced algorithm): Plain white surface

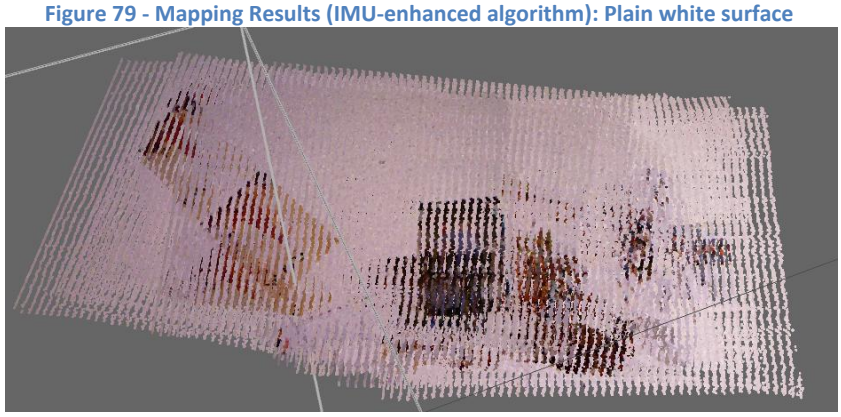


Figure 80 - Mapping Results (IMU-enhanced algorithm): White surface with additional RGB information

In order to have an idea what the ICP algorithm does, the same data sets have been analyzed using Intel's algorithm without ICP in it. These results are shown in Figure 81 and Figure 82, where the former shows the results for only the white surface and the latter the results with additional RGB information. Although not clear from Figure 81 itself, the frames have been added a little bit behind each other. Where Intel's algorithm and the IMU-enhanced algorithm have mapped the recording as a plane, using only RGB for localization you end up with a mass of points consisting of planar point clouds behind each other. An interesting thing to note, demonstrating the intuition of ICP being able

to pull point clouds out of alignment instead of refining the initial guess, is when comparing Figure 82 with either Figure 78 or Figure 80 one can see that the RGB pose estimate without ICP refinement is actually more accurate than Intel's or the IMU-enhanced algorithms that do use ICP, as the ICP algorithm does not collapse the points clouds onto each other.

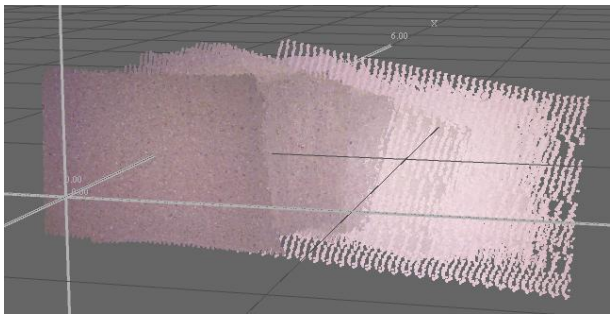


Figure 81 - Mapping Results: Plain white surface; RGB localization only

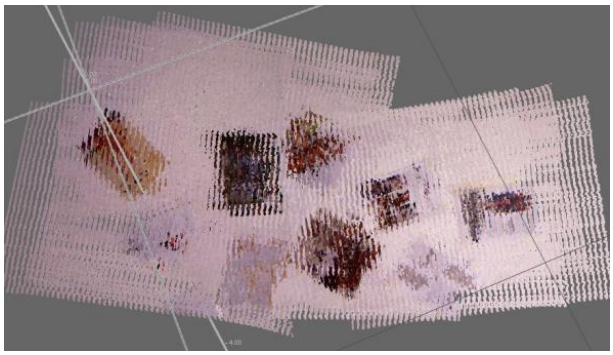


Figure 82 - Mapping Results: white surface with additional RGB information; RGB localization only

5.1.7. Loss of depth information

Localization based on image data as well as using ICP will fail when there is no depth information available. Image data could fail in cases where there is depth information available, but no depth information that would correspond with the image features that have been extracted and tracked. This will result in having the constant velocity and IMU based pose estimation processes to pick up the slack until enough depth information becomes available again. Situations where loss of depth information can happen are when dealing with reflective and refractive surfaces, sunlight and very situational shadowing effects.

Setup

In order to see how both algorithms deal with the sudden loss of depth information, a known data set (the same one used in Subsection 5.1.4 and Section 0) is chosen for which both algorithms show to map correctly. In order to replicate the worst case scenario, which is having total loss of all depth information for a subset of frames, 20 frames (frames 41 to 61) in the data set have been modified to have no depth information any more.

Results

Figure 83 show the mapping progress of Intel's algorithm up to the 40th frame, after which depth data has been removed. For easy comparison, Figure 84 shows the results of mapping with Intel's algorithm without sudden depth information loss while Figure 85 shows the mapping results where depth data is missing. The error that occurs because of the depth data loss can be seen by how point clouds have been misaligned for the scenario with missing depth data. Where in Figure 84 the closet is mapped correctly there is a clear misalignment for the same closet in Figure 85, highlighted in green. The same misalignment is also visible, highlighted in red, at the left wall in the scenario missing depth information, which is not correctly aligned with each other (before and after the depth data loss). Please note that the error is both in translation as well as in orientation.

Figure 83 - Mapping Result (Intel's algorithm): Map from frames up to 40

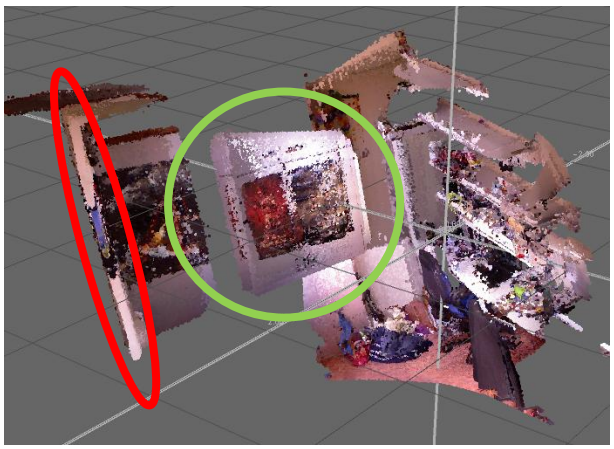


Figure 84 - Mapping Result (Intel's algorithm): Not missing depth information

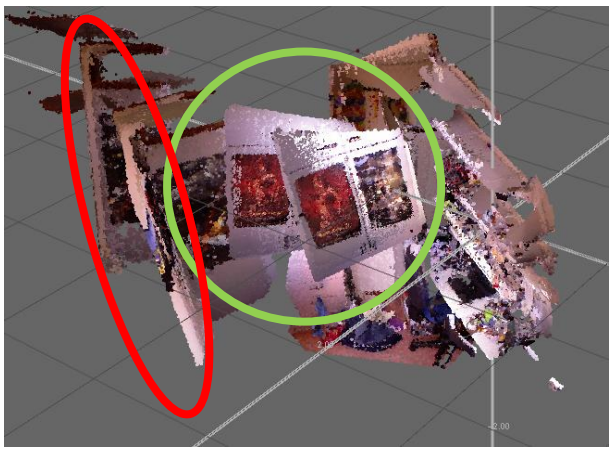


Figure 85 - Mapping Result (Intel's algorithm): Missing depth data for 20 frames

The results for the IMU-enhanced algorithm prior to the loss of depth information is shown in Figure 86. The final results for mapping the data set without depth data loss can be seen in Figure 87, while the results for the scenario *without* depth information for frames 41 to 61 are shown in Figure 88. When comparing the two final results, one can see a very big displacement has occurred in the 20 frames with missing depth information as highlighted in green in Figure 87 and Figure 88. Do note that there is only a significant error in translation. In order to give a better view of the orientation estimate based on IMU data, a part of the map shown in Figure 88 has been re-rendered and shown in Figure 89. The right closet entity in the map is what has been mapped before the missing depth data. The left closet entity in the map has been mapped after depth information loss. It shows that only the (x,y) position is severely off, as can be noticed by the duplication of the posters (highlighted green).

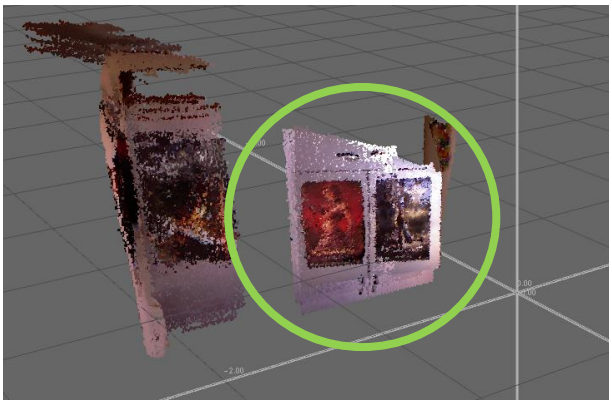


Figure 86 - Mapping Result (IMU-enhanced algorithm): Map from frames up to 40

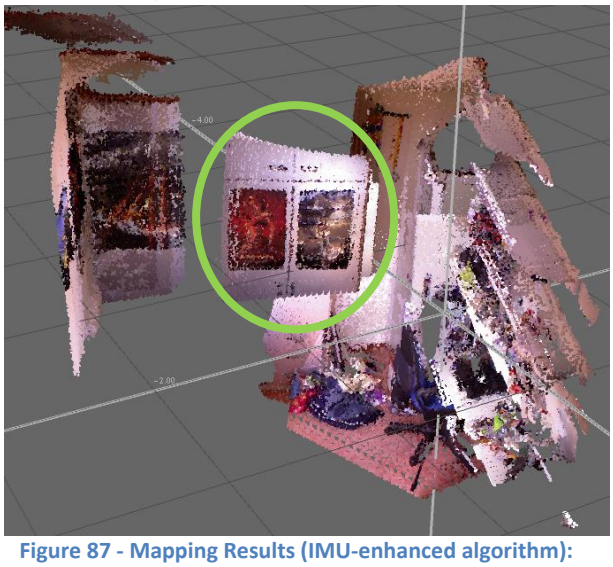


Figure 87 - Mapping Results (IMU-enhanced algorithm): Not missing depth information

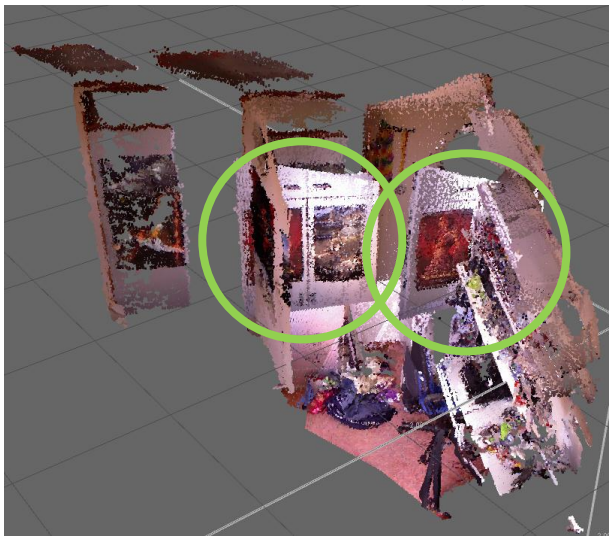


Figure 88 - Mapping Results (IMU-enhanced algorithm): Missing depth data for 20 frames

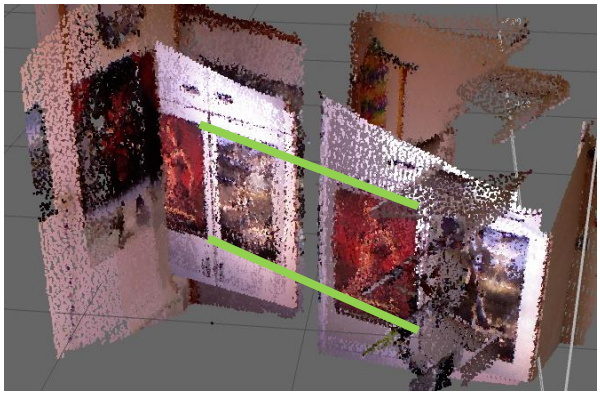


Figure 89 - Mapping Results (IMU-enhanced algorithm): Part of the map with missing depth data for 20 frames

5.2.Regular Mapping

In order to demonstrate the mapping quality of the IMU-enhanced algorithm compared to Intel's algorithm in good mapping conditions, test data is recorded in a room with ample visual features and no challenging sections such as discussed in Section 5.1. The room is a bedroom with floor dimensions of 3x4 meters, containing walls with posters, closets and a desk with planks. The Kinect is rotated and moved slowly starting from facing a closet with posters, rotating counterclockwise to the

left wall and then rotating clockwise back to the right wall with the desk and planks. Next, the planks are recorded by going from top to bottom, going along the floor back to the bottom part of the cupboard where the recording started.

The results for Intel's algorithm are shown in Figure 92 and Figure 93, and the results for IMU-enhanced algorithm are shown in Figure 90 and Figure 91. The results demonstrate than both algorithms give very similar results with only a few differences. The most evident difference is that the point density of the map in Intel's algorithm is higher than in the IMU-enhanced algorithm, which is a logical consequence of the latter not putting every frame into the map. Second is a single misalignment in the map of the IMU-enhanced algorithm, which is visible in two instances:

- The blue and black chair behind the desk is not completely aligned correctly, looking split apart in Figure 90 while this is not the case in Figure 92 (highlighted in green).
- Because of the above misalignment, the right wall section is slightly off. This is visible when comparing Figure 93 with Figure 91 in the lower right corner. The white square, corresponding to a small closet, in the latter should be more inward in the room than it actually is in the final result (highlighted in green).

Figure 90 - Mapping Results (IMU-enhanced algorithm); camera angle 1

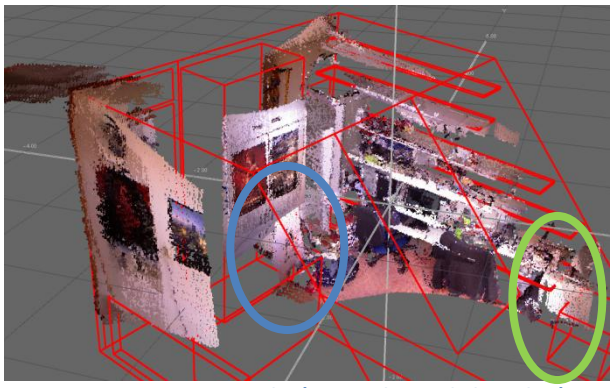


Figure 91 - Mapping Results (IMU-enhanced algorithm); camera angle 2

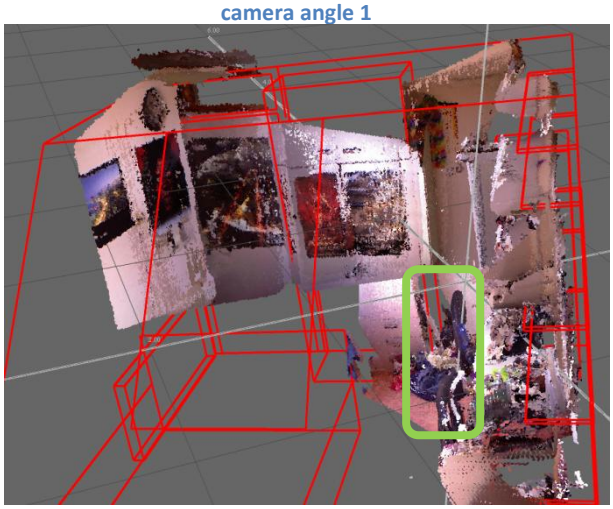


Figure 92 - Mapping Results (Intel's algorithm); camera angle 1

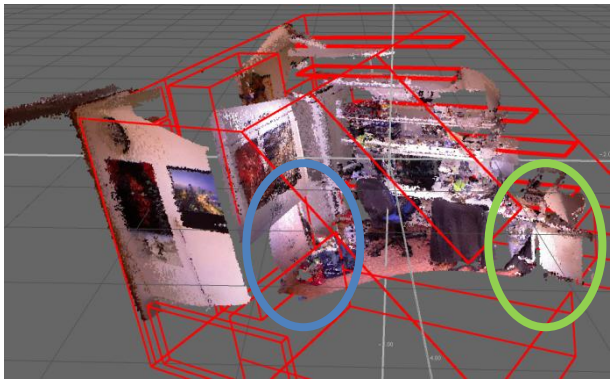


Figure 93 - Mapping Results (Intel's algorithm); camera angle 2

A thing to note the results of both algorithms are the small errors that occur and accumulate while mapping. Highlighted in blue on can see that in Figure 93 and Figure 91 that the bottom part of the closet does not align well with the top part of the closet. This is a known problem when viewing a piece of an environment that has been mapped earlier, where as a result of the accumulation of small errors a misalignment can occur such as demonstrated here. Dealing with these problems is called loop closure, which is considered outside of the scope of this thesis as loop closure is a post-processing step that minimizes global error. The results from using Intel's algorithm or the IMU-enhanced algorithm are assumed to benefit in similar degree from this processing step.

6. Conclusions

With the release of the Microsoft Kinect in 2010, researchers in the fields of robotics, Simultaneous Localization and Mapping (SLAM) and ubiquitous computing have started using this device in their research. The research topic of interest to this thesis is using RGBD-cameras, cameras that capture both color as well as depth information, for indoor mapping [7] to create digital 3D representations of the captured environment. While Intel Seattle Research demonstrated the possibilities of mapping with a RGBD-camera, the mapping algorithm has some weaknesses such as the vulnerability to violations of the static environment assumption and situations where visual localization fails (such as in featureless environments for example).

The aim of this thesis is to investigate the suitability of the Microsoft Kinect for indoor mapping purposes and the possibility of augmenting Intel's algorithm with additional Inertial Measurement Units (IMU) data. As the acceleration and angular velocity measurements of the IMU are not affected by anything related to vision, it can be used as a second source of information for localization.

Chapter 3 discusses the characterization of the Kinect, showing that the Kinect has a 4% relative accuracy with the distance to an object, with a confirmed operating range from 0.8 meters up to at least 3.8 meters, with the best accuracies within 30 degrees of facing an object, demonstrating the Microsoft Kinect is suitable for indoor mapping purposes. The characterization also shows weaknesses inherent to the Kinect such as dealing with reflective and refractive surfaces, which could warp the depth results or cause measurement faults. It also shows a weakness with respect to dealing with sunlight, which prevents depth measurements, limiting its usage to indoor only and the challenge of dealing with depth data loss in case sunlight falls into the IR camera while mapping indoor environments.

Two mapping algorithms have been implemented, which are discussed in Chapter 4. The first one is Intel's algorithm that uses image based feature-matching to provide an estimate of the movement of the Kinect between frames, which is then refined with the aid of the Iterative Closest Point (ICP) algorithm. The second algorithm is an IMU-enhanced algorithm that uses both image based pose estimation as well as IMU data based orientation estimates fused together to provide an initial guess to the ICP algorithm to refine the subsequent data from the Kinect to create a point cloud based map. This implementation only adds frames into the map after enough movement of the Kinect has been detected, in order to reduce redundant information being added into the map as well as to save processor time.

The two mapping algorithms have been evaluated, as discussed in Chapter 5, both in ideal mapping conditions as well as in corner case scenarios that violate the static environment assumption and cause failure in visual localization, such as featureless environments. It is demonstrated that Intel's algorithm and the IMU-enhanced algorithm provide similar mapping results in ideal conditions. It is also demonstrated that only adding frames into the global map if sufficient movement of the Kinect has been detected gives cleaner global maps in scenarios where the static environment assumption is being violated, such as when a person walks into the field of view of the camera. This does come at a potential cost of increased vulnerability to small misalignments due to the point mass

corresponding to the dynamic event in the environment that affects the refinement step with the ICP algorithm more than it would when using each and every frame.

In the corner cases that target situations that can cause visual localization to fail the IMU-enhanced algorithm was unable to provide improved results compared to Intel's algorithm. While the IMU orientation estimate provides a more accurate orientation estimate compared with the orientation estimate based on the constant velocity assumption, the fact that a bad (x,y,z) estimate generated by the constant velocity assumption renders the superior IMU orientation estimate unusable. Exploratory testing also showed that using the IMU accelerometer measurements with double integration introduces too much error to be an alternative for providing an (x,y,z) estimate. Without an accurate (x,y,z) estimate in case visual localization fails, the IMU-enhanced algorithm fails just like Intel's algorithm does.

The answer to the research question, is the IMU-enhanced algorithm able to provide a better robustness in scenarios where Intel's algorithm would fail, is that the IMU-enhanced algorithm is able to provide improved mapping results when dealing with a dynamic environment, although not always. As not every frame is put into the map, fewer duplicate points end up in the global map as well as fewer points that might correspond to unwanted information such as a person or animal walking in view during the recording. Do note that this is actually unrelated to IMU-data usage, as one can separately implement to only use frames when enough movement has occurred without any IMU data processing. The IMU-enhanced algorithm is able to provide mapping results of similar quality to Intel's algorithm, but was not able to provide improvements in other scenarios where Intel's algorithm fails in. Therefore it can be recommended to adopt to only add frame into the global map when enough movement has been detected if one expects to have unexpected dynamics in the environment while recording. One retains the same mapping quality and obtains a reduced negative impact of dynamic objects.

For dealing with scenarios without dynamics or risking non-ideal mapping conditions the increased cost of adding IMU data processing and fusing this data does not provide any gain. Throughout the tests that have been performed, the added value of IMU data only showed up in very few situations where visual localization failed. Also without an accurate (x,y,z) estimate when visual localization fails the IMU orientation estimate is useless information. As the benefit of processing IMU data during this thesis proved to have very few situation where it could add value, it is not recommended to implement IMU data processing when performing indoor mapping with RGBD-cameras.

6.1.Future Work

As a few of the mini-benchmarks show, the orientation estimate based on IMU data is able to provide more accurate estimates than using a constant velocity assumption. It also demonstrated what the (x,y,z) estimate based on a constant velocity assumption is often not accurate enough, rendering the more accurate orientation estimate from IMU data unusable. If this issue would be resolved the IMU-enhanced algorithm's robustness to corner cases that results in mapping failures in Intel's algorithm improves. Exploratory tests have already discarded the idea of integrating accelerometer values of the IMU. The use of Kalman Filtering has been considered but because of a lack of understanding and time this idea has not been pursued. Kalman Filtering might give a more accurate position estimate than constant velocity assumption and therefore could potentially improve the current results of the proposed algorithm without the need of additional sensors.

If there are additional sensors or odometry information available, it could be investigated if these results can be fused in the proposed algorithm.

In the current state the IMU-enhanced algorithm is not using any loop closure, the algorithm only uses the previous data frame. The global error can be reduced if loop closure would be implemented in the algorithm, improving the results significantly when viewing recording areas that have been visited before as knowledge of more previous frames is being used.

Another area where the global map can be improved is in merging points that are very near each other. In the current implementation the points are just put in the map, which often results in many similar point ending up almost ‘on top’ op each other. These almost duplicate points take more memory. If points are merged, fewer points are needed for the global map and thus less memory is needed.

Bibliography

1. **OpenNI.org.** Hardware. [Online] 2010. [Citaat van: 6 June 2011.] http://www.openni.org/hardware.

2. **PrimeSence Ltd & Asus.** PrimeSense Teams Up with ASUS to Bring Intuitive PC Entertainment to the Living Room with WAVI Xtion | Business Wire. [Online] 2011 3-January. [Cited: 2011 6-June.] http://www.businesswire.com/news/home/20110103005276/en/PrimeSense-Teams-ASUS-Bring-Intuitive-PC-Entertainment.

3. **Intel Corporation.** Intel Labs Seatle | Research - RGB-D: Techniques and usages for Kinect style depth cameras. [Online] 2011. [Cited: 2011 6-June.] http://ils.intel-research.net/projects/rgbd.

4. **Technical University Munich.** RGB-D Workshop on 3D Perception in Robotics -- Intelligent Autonomous Systems Group. [Online] 2011. [Cited: 2011 6-June.] http://ias.cs.tum.edu/events/rgbd2011.

5. **F. Endres.** openni/Contests/ROS 3D/RGBD-6D-SLAM - ROS Wiki. [Online] 18 April 2011. [Citaat van: 6 June 2011.] http://www.ros.org/wiki/openni/Contests/ROS%203D/RGBD-6D-SLAM.

6. **MIT, Robust Robotics Group.** Robust Robotics Group | CSAIL Main/Visual Odometry For GPS-Denied Flight. [Online] 2011. [Cited: 2011 6-June.] http://groups.csail.mit.edu/rrg/index.php?n=Main.VisualOdometryForGPS-DeniedFlight.

7. **P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox.** *RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments.* Seattle : Intel Labs Seattle, 2010. White Paper. RSS Workshop on Advanced Reasoning with Depth Cameras, 2010.

8. *Parallel Tracking and Mapping for Small AR Workspaces.* **G. Klein, D. Murray.** Nara : IEEE, 2007. Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on . pp. 225 - 234. ISBN 978-1-4244-1749-0.

9. *Photo Tourism: Exploring Photo Collections in 3D.* **N. Snavely, S.M. Seitz, R. Szeliski.** Boston : ACM, 2006. SIGGRAPH '06 ACM SIGGRAPH 2006 Papers. pp. 835-846. ISBN 1-59593-364-6.

10. **S. Panzieri, F. Pascucci, R. Setola, G. Ulivi.** *A low cost vision based localization system for mobile robots.* Intelligent Off-Road Vehicles, Umeå University. Umeå : Umeå University, 2001. pp. 1-6, White Paper.

11. *Accurate Vision Based Position Tracking Between Places in a Topological Map.* **S. Thompson, A. Zelinsky.** Kobe : IEEE, 2003. International Symposium on Computational Intelligence in Robotics and Automation. Vol. 1, pp. 491-496. ISBN 0-7803-7866-0 .

12. *Monocular SLAM for Visual Odometry.* **R. Munguia, A. Grau.** Alcala de Henares : IEEE, 2007. Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on. pp. 1-6. ISBN 978-1-4244-0829-0.

13. *ARToolKitPlus for Pose Tracking on Mobile Devices.* **D. Wagner, D. Schmalstieg.** St. Lambrecht : Graz University of Technology, 2007. Proceedings of 12th Computer Vision Winter Workshop.

14. *A balanced Approach to 3D Tracking from Image Streams.* **R. Subbarao, P. Meer, Y. Gene.** Vienna : IEEE, 2005. Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on. pp. 70-78. ISBN 0-7695-2459-1.

15. *TRIP: A Low-Cost Vision-Based Location System for Ubiquitous Computing.* **D.L. de Ipiña, P.R.S. Mendonça, A. Hopper.** 3, London : Springer-Verlag, May 2002, Personal and Ubiquitous Computing, Vol. 6, pp. 206-219. ISSN 1617-4909.

16. *A 6-DOF ARTag-Based Tracking System.* **C. Celozzi, G. Paravati, A. Sanna, F. Lamberti.** 1, sl : IEEE, February 2010, Consumer Electronics, IEEE Transactions on, Vol. 56, pp. 203-210. ISSN 0098-3063.

17. *SLAM combining ToF and High-Resolution cameras.* **V. Castañeda, D. Mateus, N. Navab.** Kona : IEEE, 2011. Applications of Computer Vision (WACV), 2011 IEEE Workshop on . pp. 672-678. ISBN 978-1-4244-9496-5.

18. *6 DoF SLAM using a ToF camera: The challenge of a continuously growing number of landmarks.* **S. Hochdorfer, C. Schlegel.** Taipei : IEEE, 2010. Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on . pp. 3981-3986. ISBN 978-1-4244-6674-0.

19. *Localization of an Autonomous Mobile Robot from 3D Depth Images using heterogeneous Features.* **P. Fillastreau, M. Devy.** Yokohama : IEEE, 1993. Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on. Vol. 3, pp. 1881-1888. ISBN 0-7803-0823-9.

20. *Image-Based Localization with Depth-Enhanced Image Map.* **D. Cobzas, H. Zhang, M. Jagersand.** Taipei : IEEE, 2003. Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on . Vol. 2, pp. 1570-1575. ISBN 0-7803-7736-2.

21. *Application of 3D-PMD video cameras for tasks in autonomous mobile robotics.* **A. Prusak, H. Roth, R. Schwarte.** Prague : Elsevier, 2005. 16th IFAC World Conference. ISBN 978-3-902661-75-3.

22. *Hybrid Localization System for Mobile Outdoor Augmented Reality Applications.* **I.M. Zendjebil, F. Ababsa, J.Y. Didier, M. Mallem.** Sousse : IEEE, 2008. Image Processing Theory, Tools and Applications, 2008. IPTA 2008. First Workshops on. pp. 1-6. ISBN 978-1-4244-3321-6.

23. *Sub-Meter Indoor Localization in Unmodified Environmets with Inexpensive Sensors.* **M. Quigley, D. Stavens, A. Coates, S. Thrun.** Taipei : IEEE, 2010. Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. pp. 2039-2046. ISBN 978-1-4244-6674-0.

24. *Using the Marginalised Particle Filter for Real-Time Visual-Inertial Sensor Fusion.* **G. Bleser, D. Stricker.** Cambridge : IEEE, 2008. Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on. pp. 3-12. ISBN 978-1-4244-2840-3.

25. *Did You See Bob?: Human Localization using Mobile Phones.* **I. Constandache, X. Bao, M. Azizyan, R.R. Choudhury.** Chicago : ACM, 2010. Proceedings of the sixteenth annual international conference on Mobile computing and networking. pp. 149-160. ISBN 978-1-4503-0181-7.

26. *Using Intelligent Mobile Devices for Indoor Wireless Location Tracking, Navigation, and Mobile Augmented Reality.* **C.-C. A. Lo, T-C. Lin, Y.-C. Wang, Y.-C. Tseng, L.C. Ko, L.C. Kuo.** Kaohsiung : IEEE VTS, 2009. IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS).

27. *Indoor/Outdoor Seamless Positioning Technologies Integrated on Smart Phone.* **L. Pei, R. Chen, Y. Chen, H. Leppäkoski, A. Perttula.** Colmar : IEEE, 2009. Advances in Satellite and Space Communications, 2009. SPACOMM 2009. First International Conference on. pp. 141-145. ISBN 978-0-7695-3694-1.

28. *Indoor Localization Without the Pain.* **K. Chintalapudi, A.P. Iyer, V.N. Padmanabhan.** Chicago : ACM, 2010. MobiCom '10 Proceedings of the sixteenth annual international conference on Mobile computing and networking. pp. 173-184. ISBN 978-1-4503-0181-7.

29. **A. Parnandi, K. Le, P. Vaghela, A. Kolli, K. Dantu, S. Poduri, G.S. Sukhatme.** *Coarse In-building Localization with Smartphones.* University of Southern California. San Diego : University of Southern California, 2009. pp. 1-12, White Paper (Workshop). Accepted by the Workshop on Innovative Mobile User Interactivity.

30. *Monocular Vision SLAM for Indoor Aerial Vehicles.* **K. Çelik, S.-J. Chung, M. Clausman, A.K. Somani.** St. Louis : IEEE, 2009. Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. pp. 1566-1573. ISBN 978-1-4244-3803-7.

31. *Color-Based Monocular Visuoinertial 3-D Pose Estimation of a Volant Robot.* **N. Kyriakoulis, A. Gasteratos.** 10, sl : IEEE, October 2010, Instrumentation and Measurement, IEEE Transactions on, Vol. 59, pp. 2706-2715. ISSN 0018-9456.

32. *Eye-to-Hand Approach on Eye-in-Hand configuration Within Real-Time Visual Servoing.* **A. Muis, K. Ohnishi.** 4, sl : IEEE, August 2005, Mechatronics, IEEE/ASME Transactions on , Vol. 10, pp. 404-410. ISSN 1083-4435.

33. *A Multi-State Contraint Kalman Filter for Vision-aided Inertial Navigation.* **A.I. Mourikis, S.I. Roumeliotis.** Roma : Dept. of Computer Science & Engineering, University of Minnesota, 2007. Robotics and Automation, 2007 IEEE International Conference on. pp. 3565-3572. ISBN 1-4244-0601-3.

34. *A Robust Real-time Moving Object Tracking Algorithm.* **Y. Wenjie, L. Yun.** Christchurch : IEEE, 2009. Control and Automation, 2009. ICCA 2009. IEEE International Conference on. pp. 1471-1475. ISBN 978-1-4244-4706-0.

35. *Accurate Object Localization in 3D Laser Range Scans.* **A. Nüchter, K. Lingemann, J. Hertzberd, H. Surmann.** Seattle, WA : IEEE, 2005. Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on. pp. 665 - 672. ISBN 0-7803-9178-0.

36. *FPGA-Based Real-Time Visual Tracking System Using Adaptive Color Histograms.* **J.U. Cho, S.H. Jin, X.D. Pham, D. Kim, J.W. Jeon.** Sanya : IEEE, 2007. Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on. pp. 172-177. ISBN 978-1-4244-1761-2.

37. *reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction.* **M. Kaltenbrunner, R. Bencina.** Baton Rouge : ACM, 2006. TEI '07 Proceedings of the 1st international conference on Tangible and embedded interaction. pp. 69-74. ISBN 978-1-59593-619-6.

38. **J. Shotton, A. Fitsgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake.** *Real-Time Human Pose Recognition in Parts from a Single Depth Image.* Cambridge : Microsoft Research Cambridge & Xbox Incubation, 2011. pp. 1-8, White Paper.

39. *Real-Time Visual Tracking of 3-D Objects and Dynamic Handling of Occlusion.* **P. Wunch, G. Hirzinger.** Albuquerque : IEEE, 1997. Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on. Vol. 4, pp. 2868-2873. ISBN 0-7803-3612-7.

40. *Robust and Unobtrusive Marker Tracking on Mobile Phones.* **D. Wagner, T. Langlotz, D. Schmalstieg.** Cambridge : Graz University of Technology, 2008. Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on . pp. 121-124. ISBN 978-1-4244-2840-3.

41. *Low cost vision-aided IMU for pedestrian navigation.* **C. Hide, T. Botterill, M. Andreotti.** Kirkkonummi : IEEE, 2010. Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010 . pp. 1 - 7. 978-1-4244-7880-4.

42. *Personal Position Measurement Using Dead Reckoning.* **C. Randell, C. Djiallis, H. Muller.** White Plains : IEEE, 2003. Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on. pp. 166-173. ISBN 0-7695-2034-0.

43. **Nokia Research Center.** NRC presents: Nokia Indoor Navigation | Nokia Research Center. [Online] 18 April 2011. [Citaat van: 28 June 2011.] http://research.nokia.com/news/11809.

44. **Point Inside.** Point Inside Inc. [Online] 2011. [Citaat van: 28 June 2011.] http://pointinside.com/#.

45. **NAVTEQ.** NAVTEQ Extends the Journey Beyond the 'Front Door' - Mar 21, 2011. [Online] 21 March 2011. [Citaat van: 28 June 2011.] http://press.navteq.com/index.php?s=4260&item=30551.

46. *Object Modeling by Registration of Multiple Range Images.* **Y. Chen, G. Medioni.** Sacremento : IEEE, 1991. Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. Vol. 3, pp. 2724-2729. ISBN 0-8186-2163-X.

47. *3D Pose Estimation, Tracking and Model Learning of Articulated Objects from Dense Depth Video using projected Texture Stereo.* **J. Sturm, K. Konolige, C. Stachmiss, W. Burgard.** Zaragoza : Willow Garage Inc., 2010. Proc. of the Workshop RGB-D: Advanced Reasoning with Depth Cameras at Robotics: Science and Systems (RSS).

48. *Insight3D: a high performance toolkit for advanced visualization of space and terrestrial environments.* **Beasley, G.** Ottawa : Curran Associates, Inc, 2010. Summer Computer Simulation Conference 2010. pp. 328-333. ISBN 9781617387029.

49. *Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets.* **M. Brown, D.G. Lowe.** Ottawa : IEEE, 2005. 3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on. pp. 56-63. ISBN 0-7695-2327-7.

50. *Modeling and Rendering Architecture from Photographs: a Hybrid geometry-and image-based approach.* **P.E. Debevec, C.J. Taylor, T. Malik.** New Orleans : ACM, 1996. SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 11-20. ISBN 0-89791-746-4.

51. **Microsoft.** Kinect for XBox 360 - Xbox.com. [Online] 2011. [Citaat van: 21 June 2011.] http://www.xbox.com/en-GB/kinect.

52. **OpenKinect.org.** Protocol Documentation - OpenKinect. [Online] 2011 24-May. [Cited: 2011 2-June.] http://openkinect.org/wiki/Protocol_Documentation#Cameras.

53. —. Getting Started - OpenKinect. [Online] 2011 25-April. [Cited: 2011 2-June.] http://openkinect.org/wiki/Getting_Started#Please_read_this_before_you_start.

54. **PrimseSense.** (c) 2010 PrimeSense Ltd. | FAQ. [Online] 2010. [Citaat van: 2 June 2011.] http://www.primesense.com/?p=535.

55. **B. Freedman, A. Shpunt, M. Machline, Y. Arieli.** *Depth Mapping Using Projected Patterns. US 2010/0118123 A1* United States, 13 May 2010.

56. **G. Bradski, A. Kaehler.** *Learning OpenCV.* Sebastopol : O'Reilly books, 2008. ISBN 978-0-596-51613-0.

57. *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses.* **Tsai, R.Y.** 4, August 1987, Robotics and Automation, IEEE Journal of, Vol. 3, pp. 323-344. ISSN 0882-4967.

58. *Flexible Camera Calibration By Viewing a Plane From Unkown Orientation.* **Zhang, Z.** Kerkyra : IEEE, 1999. Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Vol. 1, pp. 666-673. ISBN 0-7695-0164-8.

59. **Open Kinect.org.** Imaging Information - OpenKinect. [Online] 2011 24-January. [Cited: 2011 5-June.] http://openkinect.org/wiki/Imaging_Information.

60. **PrimeSense Ltd.** The PrimeSensor(tm) Reference Device 1.08. [Online] 2010. [Cited: 2011 6-June.] http://www.primesense.com/files/FMF_2.PDF.

61. **unknown.** kinect_shadow.pdf. [Online] (Release date unknown). [Citaat van: 6 June 2011.] http://media.zero997.com/kinect_shadow.pdf.

62. *Iterative Point Matching for Registration of Free-Form Curves and Surfaces.* **Zhang, Z.** 2, Hingham : Kluwer Academic Publishers, 1994, Vol. 13, pp. 119-152. ISSN 0920-5691.

63. **R. B. Rusu, S. Cousins.** *3D is here: Point Cloud Library (PCL).* PCL. sl : Willow Garage, 2011. pp. 1-4, White Paper.

64. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* **M. A. Fischler, R.C. Bolles.** 6, New York : ACM, June 1981, Communications of the ACM, Vol. 24, pp. 381-395. ISSN 0001-0782.

65. *Least-Squares fitting of Two 3-D Point Sets.* **K.S. Arun, T.S. Huang, S.D. Blostein.** 5, Urbana : IEEE, September 1987, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 9, pp. 698-700. ISSN 0162-8828.

66. *An Iterative Image Registration Technique with an Application to Stereo Vision.* **B.D. Lucas, T. Kanade.** Vancouver : Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ©1981, 1981. IJCAI'81 Proceedings of the 7th international joint conference on Artificial intelligence. Vol. 2, pp. 674-679.

67. **MRPT C++ reference.** The MRPT project: mrpt::vision:CGenericFeatureTracker Struct Reference. [Online] 6 July 2011. [Citaat van: 6 June 2011.] http://reference.mrpt.org/svn/structmrpt_1_1vision_1_1_c_generic_feature_tracker.html#_details.

68. **C. Tomasi, T. Kanade.** *Detection and Tracking of Point Features.* sl : Carnegie Mellon University, 1991. Technical Report. CMU-CS-91-132.

69. **Stewart, J.** Newton's Method. [book auth.] James Stewart. *Calculus, early transcendentals; fifth edition.* Belmont : Thomson Learning, Inc., 2003, pp. 347-349.

70. **Madgwick, S.O.H.** *An efficient orientation filter for inertial and inertial/magnetic sensor arrays.* sl : x-io technologies, 2010. pp. 1-32, White Paper.

71. *Surfels: Surface Elements as Rendering Primitives.* **H. Pfister, M. Zwicker, J. van Baar, M. Gross.** New Orleans : ACM Press/Addison-Wesley Publishing Co., 2000. Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 335-342. ISBN1-58113-208-5.

72. **Parrot SA.** AR.Drone Parrot - The flying video game - The first quadricopter that can be controlled by an iPhone/iPod Touch/iPad and Android. *Home.* [Online] Parrot SA, 2011. [Citaat van: 24 11 2011.] http://ardrone.parrot.com/parrot-ar-drone/en/.