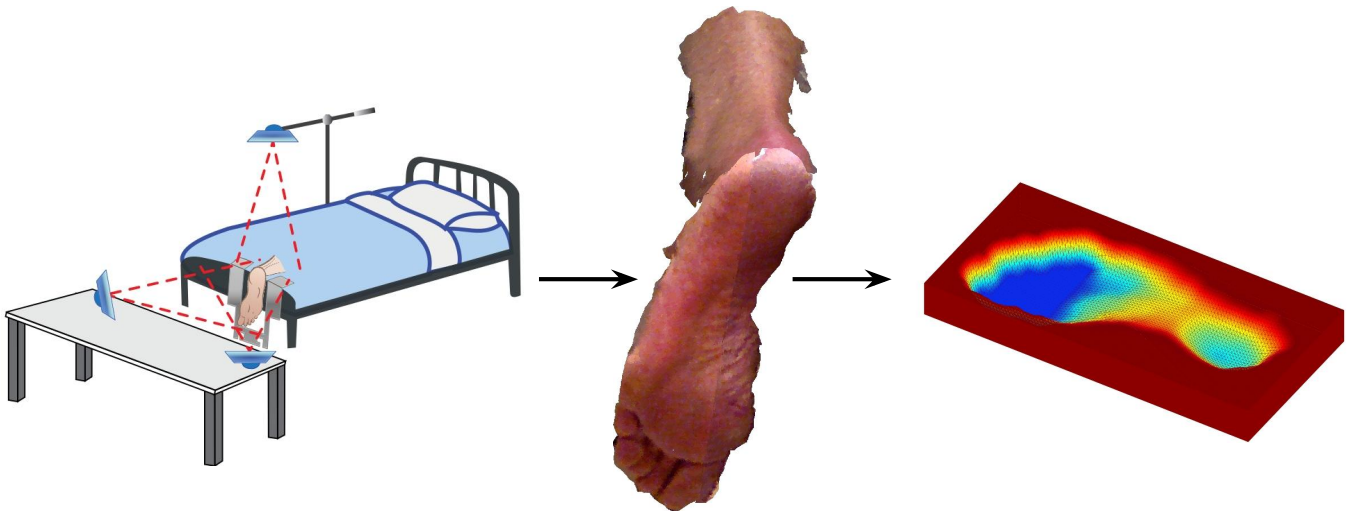
CHALMERS



Insole modeling using Kinect 3D sensors

Daniell Algar and Anton Guldberg

Department of Signals & Systems
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
Gothenburg, Sweden 2013
Master's thesis 2013:EX017

MASTER THESIS IN BIOMEDICAL ENGINEERING

Insole modeling using Kinect 3D sensors

DANIELL ALGAR
ANTON GULDBERG

Department of Signals & Systems
Division of Signal processing and Biomedical engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2013

Insole modeling using Kinect 3D sensors
DANIELL ALGAR
ANTON GULDBERG

Master's thesis 2013:EX017
Department of Signals & Systems
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
SE-412 58 Göteborg

Cover: The proposed system for insole modeling

Chalmers Reproservice
Göteborg, Sweden 2013

Abstract

Lower limb associated pain is a common problem affecting people in all age groups. In addition to the personal discomfort caused by this pain, it is also a load for the health care sector. Custom shoe insoles are a common treatment for lower limb associated pain. Insoles aims to alter the biomechanics structure of the foot, providing load relief in injured areas and additional foot support.

The objective of the thesis was to develop a computerized system for foot insole modeling.

We propose a system for foot insole modeling consisting of three modules: foot 3D scanning, foot 3D reconstruction and insole design. The 3D scanning is performed using Microsoft Kinect 3D sensors from three different views. The Kinect 3D range data is registered to a common reference system using marker data and aligned with RGB images. The registered data is converted to a polygon mesh representation using Delaunay triangulation. The final foot 3D model is obtained by merging the meshes and coloring the mesh faces with color from the RGB images. The foot insole design module consists of functions for adding/modifying pelott elevation and pronation/supination correction.

Results indicated that the accuracy of the Kinect was satisfying for foot and insole modeling using the proposed system. Similarities to existing products was good, although some shape differences exists.

It is concluded that the insole model produced by the system met the requirements regarding accuracy and real world validation, which indicated that the system holds potential in the orthotic field through its simplicity and low investment cost. Possible future work sees improvements to scanning resolution and insole design tools of the proposed system.

Keywords. 3D sensor, 3D scanning, computer-assisted, insole design, insole modeling, Kinect, mesh, orthotics, registration.

Preface

This study was performed as a master thesis project at Chalmers University of Technology, carried out at MedTech West at Sahlgrenska University Hospital. The company FotAnatomi, Gothenburg, came up with the original problem description.

A special thanks to Mikael Persson for steering the project on the right course, Arthur Chodorowski for continuous support and technical supervision, Fernando Manuel da Silva Oliviera for his orthotic expertise, Håkan Torén for helping with the fabrication, Ramin Moshavegh, Yazdan Shirvany and Lisa Snäll.

The authors, Gothenburg June 25, 2013

Contents

List of Figures

List of Tables

Glossary

.stl STereoLithography is a file format commonly used for rapid prototyping and computer-aided manufacturing.

Achilles tendon The tendon connecting the heel to the calf.

CAD Computer aided design.

CAM Computer aided manufacturing.

CNC Computer numerical control.

Coronal plane The plane dividing the body into front and back.

FOV Field of view, describing the angular view from a monocular camera.

FPS Frames per second.

GUI Graphical user interface.

Orthosis An external device applied to modify the functional or structural characteristics of the musculo-skeletal system.

Pelott A locally modified area of an insole aimed to increased support at local areas of the foot.

Plantar aponeurosis The tendons connecting the forefoot and the heel in the bottom of the foot.

Plantar fasciitis An inflammatory process in the Plantar aponeurosis.

Pronation Indication that the Achilles tendon is not straight during standing and gait, which rotates the foot towards the center of the body.

PU-foam Polymer, crisp foam which can be shaped into various forms.

RGB-D RGB data with depth processing, e.g. RGB image with range information for each pixel.

SDK Software development kit.

Supination Indication that the Achilles tendon is not straight during standing and gait, which rotates the foot away from the center of the body.

Transverse plane The plane dividing the body into upper and lower parts.

1 Introduction

Pain or discomfort in the lower limbs (e.g. foot, knee, pelvis, hip joint) is a common complaint with many possible causes. In Europe among adults, every third male and every second female experience pain in the lower limbs each year [1]. Among people in the age group of 65 years of age or more, one third reports foot problems [2, 3]. The pain might originate from injuries, wear of ligaments, joints etc., which results in a degraded physical function. Other than the obvious individual discomfort, injuries of these types leads to a high workload for the health care system as well as for employers [1].

1.1 The orthotic insole

One approach to ease the described pain, or to augment the healing process of an injury, is by using an orthosis [4]. An orthosis is an "external device applied to modify the functional or structural characteristics of the musculoskeletal system" [5]. An example of an orthosis is the foot orthosis or the shoe insole which modifies the foot by adding support by e.g. applying a pelott or local elevation in the insole. There exists general, prefabricated insoles as well as custom molded insoles, designed for individual feet.

Insoles has been proven to improve foot function as well as relieving pain caused by foot problems. A study by Kogler (1995) [6] investigated the effectiveness of the longitudinal arch support mechanism of custom foot insoles and concluded that foot insoles were effective in resisting the "arch flattening moment" occurring in the foot during gait. This was achieved by decreasing the strain in the plantar aponeurosis, increasing the load area. The result was a damping effect in the foot.

Another study by Kato (1995) [7] showed an increased mean contact area of 62.7 % after applying a custom insole. This in turn reduced the mean pressure by 56.3 % during standing, relieving areas normally under high load and spreading the pressure more even.

As mentioned above, strain relief in the plantar aponeurosis is achieved by spreading the force applied on the foot onto a greater area across the sole of the foot [8]. This is important since the natural healing process of plantar fasciitis requires the damaged area to be in rest for optimal healing effect [7].

Another application of custom insoles is counteracting pronation and supination [9]. This is the case when the Achilles tendon rotates during gait and stance, as seen in Figure 1. This will cause the foot to pronate or supinate to the side, imposing an unnatural rotation of the foot that could effect both the feet, knees and the back of the patient. To remedy this, the insole must support the foot in order to straighten the Achilles tendon [10].

Even though custom insoles are effective treatments for foot ache and lower

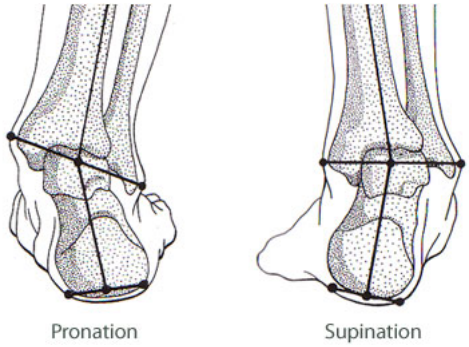


Figure 1: An example of pronation and supination on a right foot. Image courtesy of *Orthopedia Wiki* [11].

limb associated pain, they come at a high price. The price often originates in the production method where in the case of traditional manufacturing using plaster, several hours of work goes into each insole [10] or in the case of computer assisted methods, where the equipment is expensive.

1.1.1 Prefabricated insoles

The simplest form of insoles is the prefabricated insole. This insole is produced to counteract a specific problem and can supposedly be used by anyone, assuming that an insole must not be shaped to an individual foot for sufficient fit. Being general and simple to mass produce, the prefabricated insoles are cheaper and more accessible for the general public.

1.1.2 Custom insoles - traditional plaster cast method

The traditional process of manufacturing insoles involves casting a plaster shell around the foot, placed in a non weight-bearing position, which in turn is used to create a mold [12]. The aim is to provide a plaster replica of the foot and therefore the mold is polished and scrubbed until resemblance is satisfactory. The orthopedist can impose changes directly to the mold, which will affect the insole. An example of a molded replica with pelott imprint can be seen in Figure 2.

After molding the replica, a desired fabric for the insole is chosen. The insole material is heated and vacuum sucked onto the mold for attaining the shape of the mold. After forming, the insole is finalized for customer fit. See *Traditional manual method* in Figure 3 for a schematic overview of the method. Drawbacks of the method are labour intense and time consuming tasks, material inefficiency and the requirement of extensive storage facilities [13].

Figure 2: Molded replica of the bottom of the foot with the forefoot to the left and the heel to the right. A pelott imprint (red) has been placed by an orthopedic. Molded by FotAnatomi, Gothenburg [10].

1.1.3 Custom insoles - computer assisted method

Computer assisted insole production methods are available as semi-computer assisted methods or as fully computer assisted methods.

The semi-assisted methods are achieved by replacing some steps of the manual process with computer resources and the fully computer assisted methods replace all steps with e.g. laser scanning, computer assisted design (CAD) programs, and computer numerical control (CNC) cutting.

By being able to omit the large and cumbersome plaster mold from the process, some of the problems mentioned in Chapter 1.1.2 are resolved.

For examples of computer assisted methods, see the middle and bottom flowcharts of Figure 3.

1.2 Aim and objective

The aim of the thesis was to develop, implement and evaluate a method for computer-aided insole modeling for an orthopedic clinic in Gothenburg, currently using the traditional plaster casting method. The underlying purpose of this aim was to resolve issues present in the method such as high costs, labour intensive work, extensive delivery times and to bring new features into the design and production of insoles.

The following objectives were identified:

1. Investigation of existing products and research in insole production and determine the details of a computer assisted production system using 3D scanning.

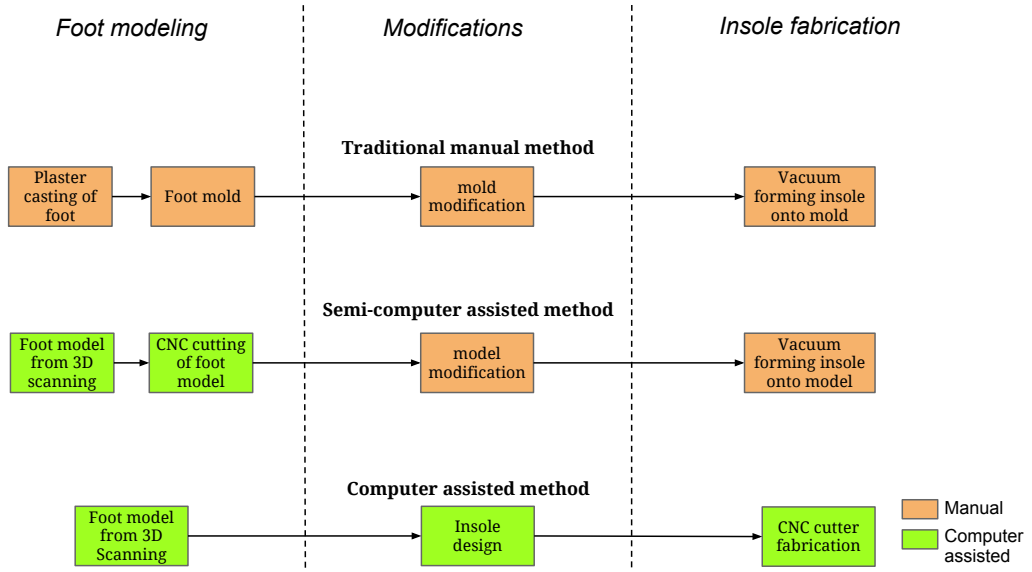


Figure 3: Process overview of traditional manual method (top), semi-computer assisted method (center) and computer assisted method (bottom). The orange blocks represent manual steps in the methods and the green blocks represent computer assisted steps.

2. Evaluation of the Kinect 3D sensor as scanning device to determine if the sensor's accuracy and properties were sufficient for the task of foot modeling.

3. Creation of a foot model from multi-view captures, acting as the base for insole design.

4. Implementation of modification tools, given a foot model, for insole design.

5. Testing of fabrication of an insole model, directly through CNC cutting or 3D printing.

1.3 Scope of thesis

The proposed system is aimed as a proof of concept, and only the immediate problems are considered. The proposed system covers foot scanning of one of the patient's feet with the Kinect sensors from three angles. The multi-view 3D registration of the captures, performed to create a foot model, uses external reference markers.

The design tools were basic implementations, providing pelott imprint design and pronation support. Extended development of more advanced tools was not prioritized.

The proposed system for insole modeling does not include fabrication, even though some fabrication methods have been tested.

To relate the proposed system to existing methods, a study between the insole base and existing insoles has been quantitatively compared with respect to local and shape differences.

1.4 Structure of thesis

The thesis is organized as follows

Chapter 2: Discusses existing products from Ortolab and Delcam, using computer assisted methods of insole production. Some research in insole production is described and finally, some requirements of the proposed system are listed.

Chapter 3: Describes some theory regarding surface construction, 3D registration and point cloud approximation.

Chapter 4: Discusses the Kinect sensor and its properties.

Chapter 5: Evaluates the Kinect sensor as a scanning device at 600–1000 mm.

Chapter 6: Proposes a system for foot scanning, foot modeling and insole design.

Chapter 7: Discusses remarks of the proposed system from Chapter 6.

Chapter 8: Discusses a case study which provides a quantitative comparison between foot models from existing companies and the proposed system.

Chapter 9: Discusses methods for fabrication of an insole from the model described in Chapter 6. CNC cutting and 3D printing were tested.

Chapter 10 Summarizes the work of the thesis, states the key findings and the limitations. This chapter also provides suggestions for future work.

2 Existing products and research in computer assisted insole production

This chapter includes a sample overview of the market situation of computer assisted insole production. Two companies are selected; Ortolab and Delcam. Also, some research is presented for the purpose of broadening the readers scope regarding alternative approaches to the digitization of insole production. Finally, some requirements of the desired system are pointed out.

2.1 Ortolab

Ortolab [14] is a Swedish custom foot insole manufacturer which is a large actor on the Swedish market [15].

Ortolab uses a sweeping flatbed laser scanner of brand Envisic Veriscan [16] for data acquisition [17]. The patients foot is placed with the heel cap on a stand with the toes pointing upwards. A full foot scan takes about 3.3 s with a resolution of about 1.6 mm [17, 18]. The price of a scanner with software is relatively high [17].

After capture, the scan is forwarded to another facility over Internet connection for analyze on a computer. After this step, the data is sent to a cutter that cuts the foot mold out of a Poly Urethane (PU) foam block. This block is then used by the orthopedist in a traditional fashion, using the model for pelott placement and vacuum sucking the final insole fabric onto the foot model [14].

In some settings, the Ortolab system acquire the foot scan when the patient is standing, i.e. in a weight bearing position, which is unwanted [10, 19]. The steps of the method can be seen in the middle of Figure 3.

2.2 Delcam

Delcam [20] is a worldwide company, delivering a complete computer aided design/-computer aided manufacturing (CAD/CAM) system from foot data acquisition to fabricated custom insole [21, 22].

Delcam provides the laser scanner iQube, which is able to scan the foot from a full, semi and non weight bearing position. The scan of one foot takes about 3 s with an accuracy of 0.4 mm [23]. It is stated that the foot model is represented in full color [21].

The company offers their own CAD software, called OrthoModel. The software is extensive, providing several features for insole design [22, 24] such as

- Possibility to work with one or two foot models at the same time.

- Place key points such as for marking the heel, the foot pads and/or metatarsals for measurements.
- Alter the orientation of the insole for extended support e.g. when pronating.
- Comparison of insole and scanned foot model to validate the fit.
- Free hand modification and thickness control of the insole is possible, with free design of the pelott.

Delcam also supplies fabrication products [22, 25] for cutting the insoles from a digital design. Cast Medical is the Delcam partner in Sweden [22, 26].

The Delcam system is expensive being one of the most extensive on the market, providing a complete solution from scanning to manufacturing [22]. The steps of the method can be seen at the bottom of Figure 3.

2.3 Related research in insole production

As described in Chapter 2.1 and 2.2, both semi-computer assisted and fully computer assisted methods are available on the market. Research in the field of insole production has several focuses; Sathish [27] (2012) proposed a system for rapid prototyping of custom orthotics for plantar ulcer. Using Computed Tomography (CT) scans to acquire the bone structure of the limb, the structure was exported to a CAD software for modeling purpose. The modeled foot was then used for fabrication of the insole using Plaster of Paris Powder (POP). Sathish concluded that the given method makes the acquisition of the foot imprint easier than traditional methods and in a more cost effective way.

Huppin (2009) [13] concluded that among existing imaging options, only the ones giving a true 3D image has the potential of fulfilling necessary criteria to act as an alternative to traditional casting methods. This would exclude solutions using for example flatbed scanners from one angle.

Mavroidis (2011) [28] has used a 3D laser scanner with a novel approach for insole design. The foot was scanned and the digitized foot surface was edited to an optimal form using CAD software. The output from the software was forwarded to a rapid prototyping machine for fabrication. A gait analysis of patients wearing the rapid prototyped orthoses showed a comparable performance to prefabricated polypropylene design.

As stated, research deals both with imaging and fabricating challenges of insole production. Another challenge is lowering costs of custom insoles at the same time as introducing better technology.

2.4 Requirements of a novel system

Described above are several techniques for insole production, both by existing companies and research based. Considering a new system, some requirements are established

1. The system described in Chapter 2.1 and 2.2 are high cost systems. A novel system should be of lower cost. This can be achieved by the use of low cost scanners, such as the Kinect sensor.

2. Most systems today do not provide color/texture information of the patient's foot in their scans. A novel system with color/texture information associated with the scans could indicate e.g. sole condition and areas of interest.

3. Most systems today as described in Chapter 2, image the bottom of the foot. A system providing information regarding the Achilles tendon would yield more information in the case of pronation and supination.

3 Theory

This chapter presents the theory used in the reminder of the thesis. This includes theory used for creating surfaces from point clouds, the registration of different 3D data sets to a shared coordinate system and finally how an arbitrary point cloud can be represented on a systematic mesh grid.

3.1 Surface representation of point cloud

A point cloud V is a collection of points in a coordinate system, in this case, the Cartesian (x,y,z) coordinate system. Let a point cloud represent a view of an object. To create a surface representation, a polygonal mesh is used. A polygonal mesh O is defined as

$$O = \{P,V\} \tag{1}$$

where $V = \{v_1,...,v_{N_v}\}$ is a set of N_v vertices where $v_i = (x_i,y_i,z_i)$ and $P = \{p_1,...,p_{N_p}\}$ is a list of N_p polygons where $p_i = \{v_{i,1},...,v_{i,nv_i}\}$. The vertices in p_i forms a polygon with nv_i vertices. If $nv_i = 3$ for all polygons, there will be only triangles in the mesh [29].

The point cloud V, holds all the vertices. Connecting the vertices to triangles, a triangular mesh can be defined. One method for creating the triangular mesh is by Delaunay triangulation [30].

In 2D, Delaunay triangulation is based on connecting points in the (x,y)-plane with the restriction that the circumcircles of the triangles created by the connected points, do not contain any other points. This is visualized in Figure 4, where 10 points are connected through Delaunay triangulation. In can be seen that no circumcircle contains any points.

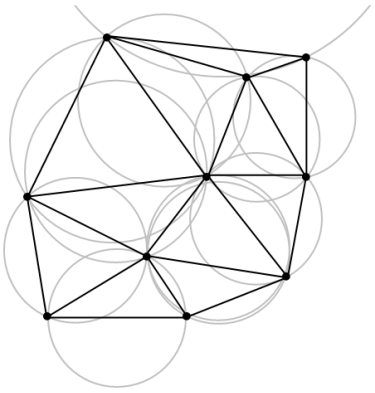


Figure 4: Delaunay triangulation of 2D points, courtesy of *Wikipedia* [31].

Figure 4 shows the 2D case of Delaunay triangulation, which can be applied for 3D data as well to create a 3D surface. To create the surface from a point cloud, the 3D coordinates are projected into 2D space where the Delaunay triangulation is applied, connecting the points. Finally the coordinates are mapped back to 3D. An example case of this, is applying the Delaunay triangulation in the (x,y)-plane of point cloud to connect the points and create the surface. Then, adding the z-component, takes the surface to 3D.

3.2 Absolute orientation of 3D data sets

This chapter aims to describe how a point cloud observed in different coordinate systems can be registered to a shared coordinate system using the method supplied by Horn (1986) [32]. The parameters for rotation R, translation t and scale s are determined for transformation of the point cloud in one system to the best fit in the other system, as seen in Figure 5.

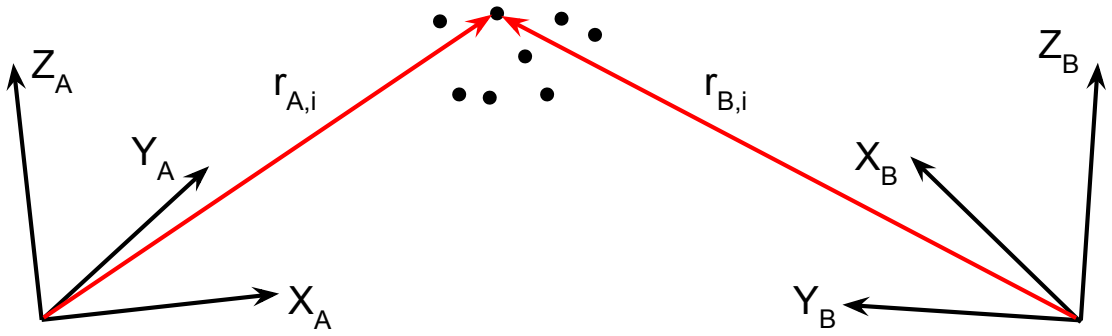


Figure 5: Two views of the same object from different origins. Let $r_{A,i}$ be the position of point i in system A and $r_{B,i}$ the position of point i in system B.

To register a set of points to another corresponding set of points in space, an algorithm minimizing the difference between the point sets through transformation is used. The algorithm finds the closed-form solution of absolute orientation using unit quaternions. Choosing a closed form algorithm ensures speed since the best transformation is calculated directly in one iteration.

The quaternion representation is a number system that extends the complex numbers, and provides a powerful tool for their ability to describe spatial transformations. Using quaternions in the implementation also avoids the possibility of Gimbal lock [33, 34] due to a fourth axis.

Problem description Consider a point cloud as in Figure 5 where r_A and r_B contain corresponding points in space observed from different views.

Calculating the best fit between the point clouds through transformation is done by minimizing

$$e = \sum_{i=1}^{n} ||r_{B,i} - sR(r_{A,i}) - t||^2 \tag{2}$$

where and $r_{B,i}$ and $r_{A,i}$ are corresponding points, s, R and t are transformation parameters and n are the number of corresponding points observed.

The quaternion based method finds the best transformation with respect to equation (2) and explicit expressions for s, R and t are

$$t = \bar{r}_B - sR(\bar{r}_A) \tag{3}$$

where $\bar{r}_A$ and $\bar{r}_B$ are the centroids of the observed points,

$$s = \frac{\sum_{i=1}^{n} r'_{B,i} R(r'_{A,i})}{\sum_{i=1}^{n} ||r'_{A,i}||^2} \tag{4}$$

where $r'_{A,i}$ and $r'_{B,i}$ are the observed points, relative to the centroids $\bar{r}_A$ and $\bar{r}_B$. In the case of $s = 1$, the transformation is a rigid body transformation.

R is found by retrieving the lower right 3×3 sub matrix of the 4×4 rotation matrix $\mathring{q}^T \mathring{q}$, where $\mathring{q}$ is the quaternion that maximizes

$$\sum_{i=1}^{n} (\mathring{q}\mathring{r}'_{A,i}) \cdot (\mathring{r}'_{B,i}\mathring{q}) \tag{5}$$

For a more detailed description of the method, see Appendix A.

3.2.1 Registering multi-view captures

The method described in Chapter 3.2 registers one view of an object to another view of the same object. In the process of multi-view registration, the aim is to register several views of an object to a shared coordinate system, placed arbitrary.

Let r_A and r_B be corresponding subsets of points observed in point cloud P_A and P_B respectively. This can be seen as two cameras, capturing one object, as seen in Figure 6. Attempting to register the point cloud P_A to P_B is then performed by

$$P_{A \to B} = sR(P_A) + t \tag{6}$$

where s, R and T minimizes the least square error e from equation (2).

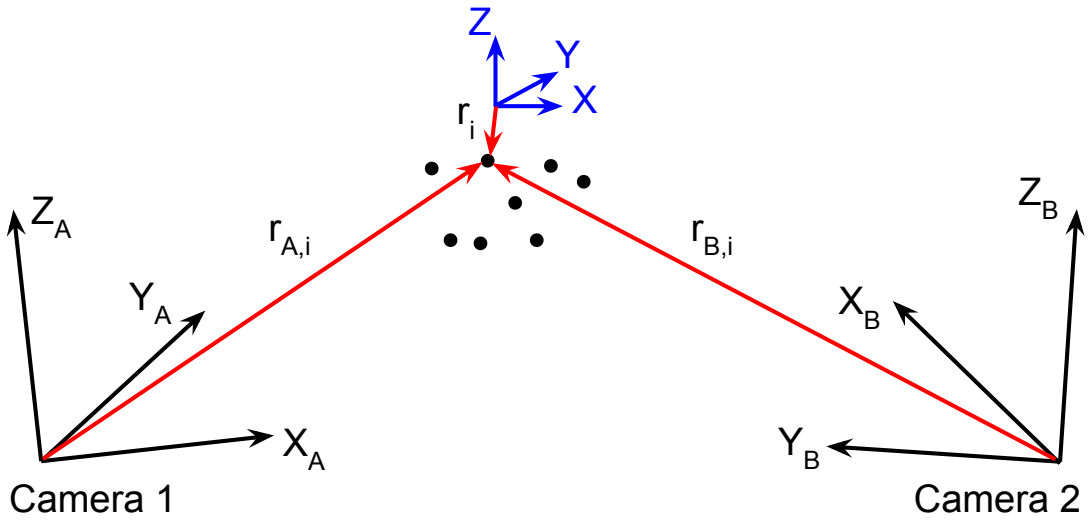


Figure 6: Two camera views of one object. Let $r_{A,i}$ be the position of point i in system A and $r_{B,i}$ the position of point i in system B. Furthermore, let r_i be the position of point i in the new coordinate system (X,Y,Z).

Furthermore, Figure 6 shows the point cloud observed from a third angle, situated in coordinate system (X,Y,Z). Let this point cloud be denoted P and let r be the subset of P corresponding to the points in r_A and r_B. Using the method in Chapter 3.2, the P_A and P_B can be sequentially registered to system (X,Y,Z) using r_A, r_B and r. After transformation, the two views of the point cloud P_A and P_B will be observed from the same coordinate system, (X,Y,Z).

The setup can be extended to any number of different capture angles and any objects, given at least 3 corresponding points observed from all captures to be registered [32].

3.3 Point cloud approximation using rectangular mesh grid

A point cloud with scattered (unevenly spaced) values can be represented on a rectangular mesh grid. Let a point cloud with (x,y,z)-coordinates be projected into the (x,y)-plane. A mesh grid is created with grid points as the new (x,y)-values of the approximation, as seen in Figure 7. It can be seen that the projected points are not evenly distributed in the (x,y)-plane.

In order to approximate a new point cloud, with evenly spaced points in the (x,y)-plane, the projected points (black) are snapped to the nearest neighboring mesh grid point (red). The approximated z-value of the grid points are the mean of snapped point's original z-values.

The described method can be used for example in mesh merging, where the

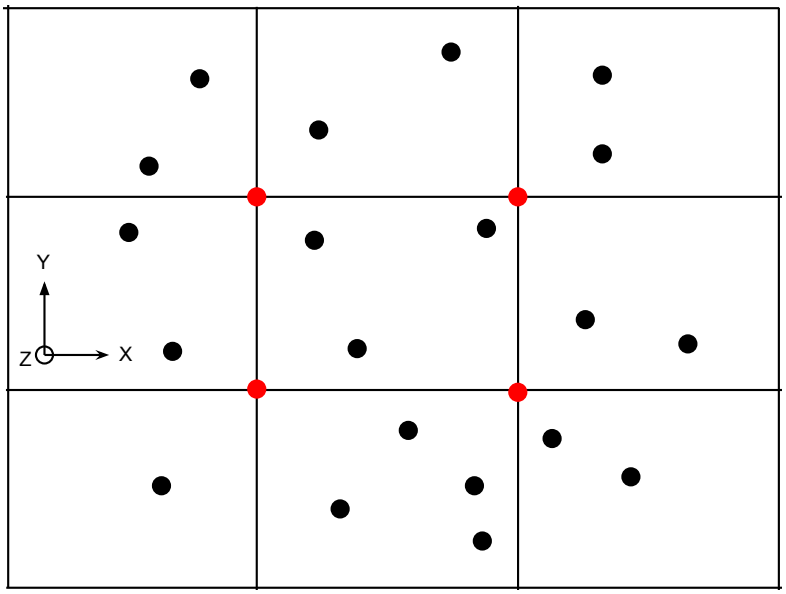


Figure 7: Approximating point cloud to mesh. 2D projected 3D points (black) imposed by a mesh grid with mesh grid points (red). All 2D projected points will be moved to respective closest mesh grid point. The z-value in each resulting mesh grid point will be the mean of the 2D projected 3D points moved to respective mesh grid point.

vertices of two overlapping meshes can be merged and represented by a single representation. A triangular mesh can then be created for the representation, using the Delaunay triangulation method described in Chapter 3.1.

4 The Kinect sensor and tools

This chapter covers Kinect specifications such as depth map resolution and hardware. The chapter also reviews imaging techniques of the Kinect such as data capture and alignment of the RGB and depth data. An overview of tools used to communicate with the Kinect will also be presented.

The Kinect is a measurement/motion sensing device for 3D space, seen in Figure 8. After the introduction in 2010, reengineering of the Kinect device unveiled capabilities that could be used in computer vision applications such as 3D scanning, tracking applications, indoor mapping etc. [35]. This was first exploited by the open source community openkinect.org [36] and later on by e.g. OpenNI [37] and ultimately by Microsoft with the introduction of their own software development kit (SDK) providing tools to use the Kinect for 3D sensing [39]. The Kinect costs around 100 EUR (2013-03-01).

Figure 8: The Kinect sensor with, from left to right, laser emitter, RGB camera and IR camera, courtesy of *Wikipedia* [38].

The device uses an IR laser emitter, RGB camera and an IR camera . Utilizing these, the Kinect sensor is able to deliver a depth map/range map of a scene with an aligned RGB image [39].

The Kinect RGB camera has a resolution of 1280×1024 pixels and delivers images at 10 frames per second (FPS). The IR camera has the functional resolution 640×480 pixels and the IR image is used to calculate the depth map. The depth data streams at 30 FPS and has a resolution of 640×480 pixels. The depth map has a dynamic resolution of 11 bits (2048 levels) and is transferred over USB 2.0 [39].

4.1 Remarks on the Kinect as 3D sensor

The Kinect uses the IR laser pattern emitter and the IR camera for depth measurements. Like any similar device, it is light sensitive and accurate readings are

impossible in direct sunlight, which contains IR wavelengths. Furthermore the depth map delivered, provides inaccurate reading around sharp edges. This was also observed by Anderssen (2012) [40] and is due to the spatial precision of the Kinect IR camera.

The sensor readings regarding the depth data is somewhat noisy and experiments have been done by Andersen (2012) that concludes that the noise on the depth values provided, are normally distributed over time [40].

The minimum scanning distance of the Kinect is 800 mm according to the manufacturer [39]. It is possible to acquire images at nearer distances, but the images will be distorted.

4.2 Real world depth map calculations

The Kinect reads the real world scene with an imposed, structured laser pattern with the IR camera [39]. Converting these readings to real world mm depth is done through optical triangulation. The setup is an IR camera, an IR laser emitter and an object as can be seen in Figure 9. Triangulation in short utilizes known geometry of the setup and a reference setup to calculate depth values of the scene [35].

Let the (x,y)-plane be the image plane and z be the depth axis of the image. The laser emitter covers the scene in a structured laser speckle pattern, which are read by the IR camera and compared to a reference pattern stored in the Kinect. The reference pattern is a measurement of a plane at reference distance Z_0. An object at a different distance than the reference distance will cause the laser marker to shift sideways creating a disparity d between the reference pattern and the observed position of the laser marker. An image correlation procedure inside the Kinect calculates this disparity for all pixels, which creates the depth map.

Consider the object k, then the following known parameters can be used for calculating the corresponding depth in mm from a given disparity [35]

- b: The distance between the IR laser and the IR camera.
- f: The lens to sensor distance of the IR camera.
- d: The measured disparity.
- Z_0: The reference distance for depth map calculation.

To calculate the depth Z_k of an object in mm, let D be the distance between the reference point and the object point. From trigonometry, the ratio

$$\frac{D}{b} = \frac{Z_0 - Z_k}{Z_0} \tag{7}$$

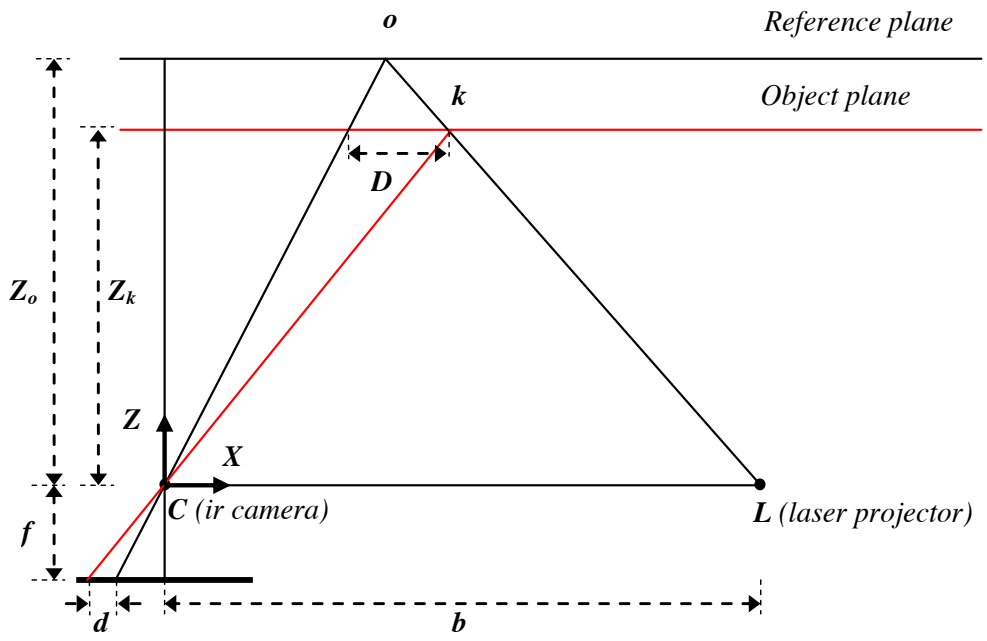


Figure 9: Triangulation technique of the Kinect. The figure shows the IR camera (C), the object (o and k) and the IR laser emitter. The disparity D caused by the position change in the object from o to k, is observed in the IR camera as disparity d. Image courtesy of Khoshelham and Elberink (2012) [35].

can be extracted, where

$$\frac{d}{f_{depth}} = \frac{D}{Z_k} \tag{8}$$

Expressing Z_k as a function of the variables Z_0, f, b and d in equation (8) yields for conversion of disparity values to mm depth values

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{bf}d} \tag{9}$$

Note that Z_k in equation (9) has the dependence of an $\frac{1}{d}$-function where d is the measured disparity value. The variables Z_0, f, b are fixed parameters of the Kinect and can be decided through calibration.

Calculating the real world (x,y)-coordinates of an image pixel can be done from Z_k. Using trigonometry and Figure 9 it is concluded that

$$X_k = -\frac{Z_k}{f_x}(x_k - x_0) \tag{10}$$

$$Y_k = -\frac{Z_k}{f_y}(y_k - y_0) \tag{11}$$

where f_x and f_y are the focal lengths of the IR camera and the variables x_0 and y_0 are the principal points of the IR camera.

4.2.1 Depth formula correction

The expression in equation (9) gives the depth value in mm as a function of disparity. An alternative formula for disparity to depth in mm also exists, stating that equation (9) is not correct. The dependence in reality differs from the ideal case and looks slightly more like a tangent function [36]

$$Z_k = k_1 \tan\left(\frac{d}{k_2} + k_3\right) \tag{12}$$

where k_1, k_2, and k_3 are determined through calibration.

4.3 RGB-Depth frame alignment

The generated depth map can be aligned to the frame of the RGB image, creating a pixel to pixel correspondence, RGB-Depth (RGB-D) data. This is done via a rigid transformation of each real world depth value to the corresponding RGB pixel. Using the intrinsic parameters, being the internal camera characteristics of

the RGB and IR camera respectively, and the extrinsic parameters, describing the relative positions between the RGB and the IR camera, an alignment is possible.

The rotation matrix and translation vector of the transformation describes the relative transform between the depth map and the RGB image. An individual factory calibration of the Kinect contains the transformation for assigning the depth values to the RGB image [41, 42].

Let $R_{d\to rgb}$ and $T_{d\to rgb}$ be the transformation from the depth frame to the RGB frame and apply it to the depth map's real world (x,y,z)-coordinates

$$(x,y,z)_{rgb} = R_{d\to rgb} \cdot (x,y,z)_{depth} + T_{d\to rgb} \tag{13}$$

After transformation of the coordinates in equation (13), the corresponding pixels (i,j) in the RGB frame can be identified as

$$j = \frac{X_k f_{(x,rgb)}}{Z_k} + x_{(0,rgb)} \tag{14}$$

$$i = \frac{Y_k f_{(y,rgb)}}{Z_k} + y_{(0,rgb)} \tag{15}$$

where $f_{(x,rgb)}$, $f_{(y,rgb)}$, $x_{(0,rgb)}$ and $y_{(0,rgb)}$ are the focal length and principal point offsets respectively of the RGB camera. Given equation (14) and (15) the depth map can be rebuilt, aligned to the RGB image.

4.4 Libfreenect framework

In order to access data from the Kinect the framework libfreenect was used [36]. Libfreenect is developed by the open source community at OpenKinect.org and provides access to depth data in disparity or RGB aligned mm format. Furthermore the RGB data can be accessed in different formats and at different rates.

Additionally, some useful features such as providing the option to tilt and level the Kinect using the built in accelerometer was provided by libfreenect [36].

The library also provided access to multiple Kinect devices, enabling simultaneous multi-view capture [36].

4.4.1 Matlab Kinect wrapper

Using MATLAB [43] for calculations and analysis, a wrapper for the libfreenect C programming language code was needed. The implementation by Berg [44] was used. The wrapper in short provides a .mex file which acts as a wrapper of C-code, executable from MATLAB. Using a wrapper made it possible to access Kinect depth and RGB data for storage directly into MATLAB variables. Using the wrapper live capture and analysis was made possible.

The wrapper was partially modified to work with more than one device and to deliver data in the desired format; registered mm values.

5 Evaluation of the Kinect as 3D sensor

As mentioned in Chapter 4.1, the Kinect has some limitations. The following chapter analyzed in detail the properties of the Kinect and validated the usability of the Kinect for foot scanning.

The computer used throughout the project has been Linux based running 64-bit Ubuntu 12.04 LTS with 8 GiB RAM memory, 8 × Intel® Core™ i7 CPU 870 at 2.93 GHz and with a NVIDIA GeForce GTS 250 (rev a2) graphics card.

MATLAB 8.1 (2013a) 64-bit has been used with access to the Image Processing toolbox.

5.1 RGB camera calibration

Calibration of the RGB camera was necessary to determine the (x,y)-coordinates of each RGB pixel accurately. Performing the calibration, the intrinsic parameters of the RGB camera were determined. The intrinsic parameters refers to the internal characteristics of the camera.

The calibration was performed using the camera calibration toolbox for MATLAB by Bouguet (2010) [42]. Using a checkerboard with known dimensions, several angles as seen in Figure C.2 were captured. From these captures, the RGB camera characteristics were determined. The calibration toolbox uses a camera model with parameters regarding focal length, principal point offset, skew and radial and tangential distortions of the $4th$ order [42]. The parameters for the camera model were

- *Focal length:* The magnitude of which the system converges or diverges light along the x,y axes, denoted $f_c = [f_x, f_y]^T$.
- *Principal point:* The location of the cameras true center pixel in the captured image, denoted $cp = [x_0, y_0]^T$.
- *Skew coefficient:* α defining the angle between the (x,y) image axes.
- *Distortions:* The radial and tangential image distortion coefficients, stored in the 5×1 vector δ.

The calibrated focal lengths and principal points were used to determine (X_k, Y_k) from Z_k as in equation (10) and (11). For a complete camera model description, see Appendix B.

5.1.1 Calibration results

The results of the calibration can be seen in Table 1, including a list of calibrated intrinsic parameters acquired through the camera calibrations for two Kinect devices. For a complete presentation of the calibration results, see Appendix C.

Table 1: Calibration results for two Kinect devices (K 1 and K 2 respectively)

Intrinsic Parameters RGB			
Focal length (f_x, f_y)	K 1	$[1034.42, 1034.26] \pm [6.23, 6.09]$mm	
	K 2	$[1038.72, 1038.44] \pm [3.59, 3.59]$mm	
Principal Point (x_0, y_0)	K 1	$[640.19, 527.98] \pm [5.98, 6.29]$pixel	
	K 2	$[624.285, 510.199] \pm [2.340, 2.647]$pixel	
Distortion coefficient (δ)	K 1	$[0.14, -0.24, 0.00, 0.00, 0.00]$	$\pm$
		$[0.01, 0.02, 0.00, 0.00, 0.00]$	
	K 2	$[0.13, -0.23, -0.00, -0.00, 0.00]$	$\pm$
		$[0.01, 0.01, 0.00, 0.00, 0.00]$	

5.1.2 Calibration discussion

In Table 1, it can be seen that the calibration parameters of two Kinect sensors had similar focal lengths and principal points, which indicated that the calibration results were trustworthy. The Kinect parameters may vary in a small interval, but large differences between cameras are unexpected.

5.2 Depth map distortion correction

Analyzing the depth map taken of a flat wall, as in Figure 10, a distortion could be observed. This effect's origin was not fully determined in re-engineering of the Kinect [45]. This effect was mitigated, using a correction matrix $\boldsymbol{CM}$, which was calculated for the 640×480 depth map.

To analyze the distortion, N depth maps at different distances were captured. Capturing depth maps perpendicular to a flat wall at distances $z \in 800 - 1000\,\text{mm}$, with 10 mm steps, the distance dependence of the distortion could be studied. The expected distance z to the wall was determined by the distance read by the Kinect at the image center. At each of the N distances, k frames were captured to reduce the normally distributed noise, and the median value of the k images was used as the true capture. Two attempts to reduce the distortion were conducted. In the experiments below, $N = 21$ and $k = 100$.

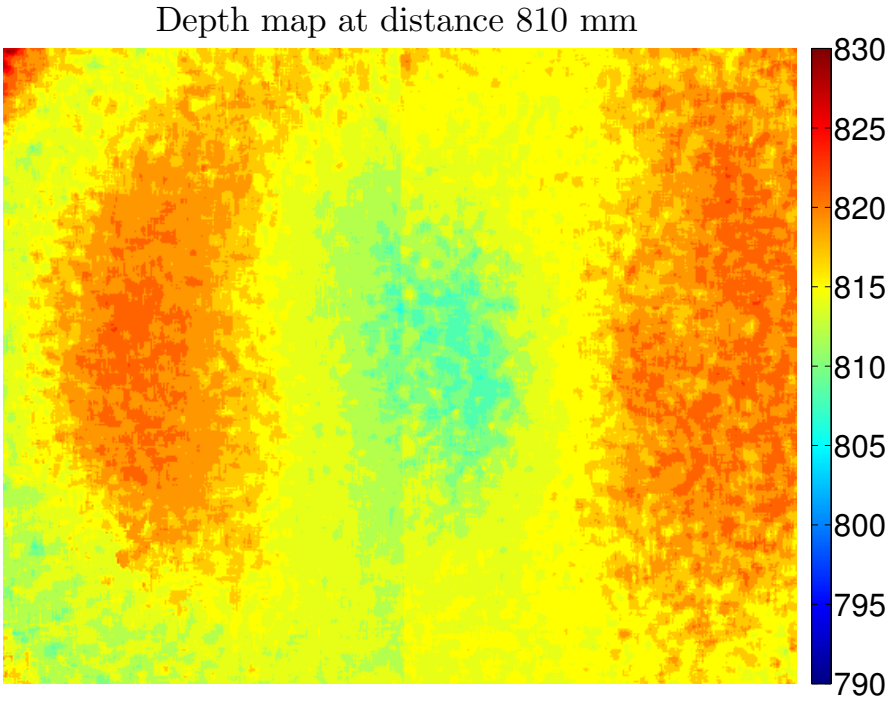


Figure 10: Depth map in mm constructed from 100 median averaged captures of a flat wall, at expected distance $z = 810\,\text{mm}$. The expected distance z to the wall is located in the image center. The vertical line around the center of the image is of unknown origin, but is persistent throughout all measurements.

Method 1 The first method, described in algorithm 1, is a method described by Smisek (2011) [46]. This distortion correction is based on the mean distortion pattern in disparity over distance,

$$\boldsymbol{CM} = \sum_{i=1}^{N} \frac{\hat{\boldsymbol{D}}_i - d_i}{N} \tag{16}$$

where d_i is the expected disparity value, $\hat{\boldsymbol{D}}_i$ is the measured disparity map of a measurement and N is the number of depth map sets measured. Converting a captured depth map mm values $\hat{\boldsymbol{Z}}_i$ to disparity $\hat{\boldsymbol{D}}_i$ was done through

$$\hat{\boldsymbol{D}}_i = \frac{-1}{0.00307(\hat{\boldsymbol{Z}}_i - 3.33)} \tag{17}$$

The values in equation (17) were obtained by Burrus (2011) [41].

Algorithm 1 Calculation of the disparity correction matrix $\boldsymbol{CM}$

1: **for** i=1:N **do**
2: Convert measured depth $\hat{\boldsymbol{Z}}_i$ to disparity $\hat{\boldsymbol{D}}_i$
3: Convert expected depth z_i to disparity d_i
4: Calculate difference map $\boldsymbol{dM}_i = \hat{\boldsymbol{D}}_i - d_i$
5: **end for**
6: Calculate correction matrix as the mean, i.e. $\boldsymbol{CM} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{dM}_i$

The expected depth was taken as the mean of a neighborhood around the image center using a kernel of 5×5 pixels.

Given a distorted depth map, distortion correction was performed by

1. Converting depth to disparity.
2. Correcting distortion in disparity space $\tilde{\boldsymbol{D}} = \hat{\boldsymbol{D}} - \boldsymbol{CM}$.
3. Converting back to mm.

Applying $\boldsymbol{CM}$ on a captured flat surface with expected depth z, yielded the mean absolute error per pixel and distance in disparity

$$e_{r,i} = \frac{|\hat{\boldsymbol{D}}_i - d|}{P} \tag{18}$$

and in mm

$$e_{d,i} = \frac{\left|\hat{\boldsymbol{Z}}_i - z\right|}{P} \tag{19}$$

respectively where P was the total number of pixels in the image (640×480).

Method 2 The first method used the mean error in disparity space to correct the depth map. The second method attempted to fit polynomials to the error over distance, according to algorithm 2.

Algorithm 2 Calculation of the disparity correction matrix $\boldsymbol{CM}$

1: **for** p=1:P **do**
2: **for** i=1:N **do**
3: Calculate difference $dM_{i,p} = \hat{z}_{i,p} - z_i$
4: **end for**
5: Fit $3rd$ order polynomial $\boldsymbol{CM}_p$ to $\boldsymbol{dM}_p$
6: **end for**

Line 5 of algorithm 2 fitted the errors to a polynomial on the form

$$\boldsymbol{CM}_p(x) = ax^3 + bx^2 + cx + d \tag{20}$$

where a, b, c and d were the polynomial constant outputs at line 5 in algorithm 2. The correction was applied by evaluating the polynomial at given expected distance z according to

$$\tilde{\boldsymbol{Z}} = \hat{\boldsymbol{Z}} - \boldsymbol{CM}_p(z) \tag{21}$$

where $\tilde{\boldsymbol{Z}}$ is the corrected depth map.

The mean absolute errors in mm for different distances was calculated using equation (19).

5.2.1 Distortion correction results

Distortion correction was attempted with two methods. The first method assumed a constant disparity pattern error, the second method fitted individual correction polynomials to each pixel to correct distortions depending on distance. The errors before and after applying algorithm 1 and 2 can be seen in Figure 11.

Figure 11 shows mm errors as a function of distance. For algorithm 1, the mean error were reduced from 6.84 mm to 0.95 mm, which is roughly a 7 times reduction. By applying algorithm 2, the mean error of the distorted images was reduced from 6.84 mm to 1.32 mm, which is roughly a 5 times reduction.

An example of a depth map correction through algorithm 1 can be seen in Figure 12.

5.2.2 Distortion correction discussion

Regarding the distortion described in Chapter 5.2, the effect was examined and mitigated. Two alternative distortion reduction algorithms were investigated where

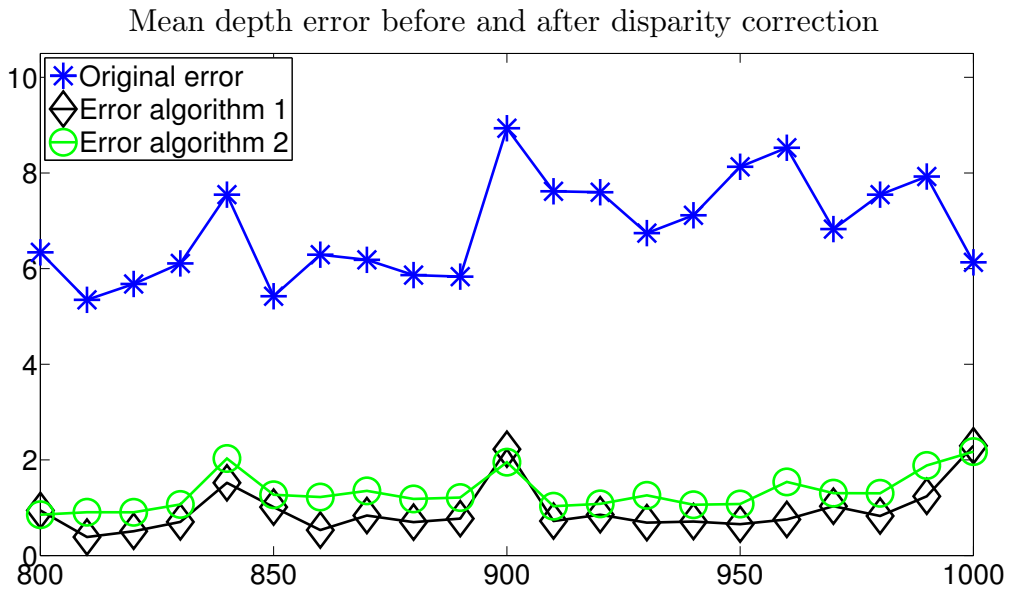


Figure 11: The figure displays the mean error before and after applying algorithm 1 and 2. The resulting error is 6.84 mm, 0.95 mm and 1.32 mm respectively.

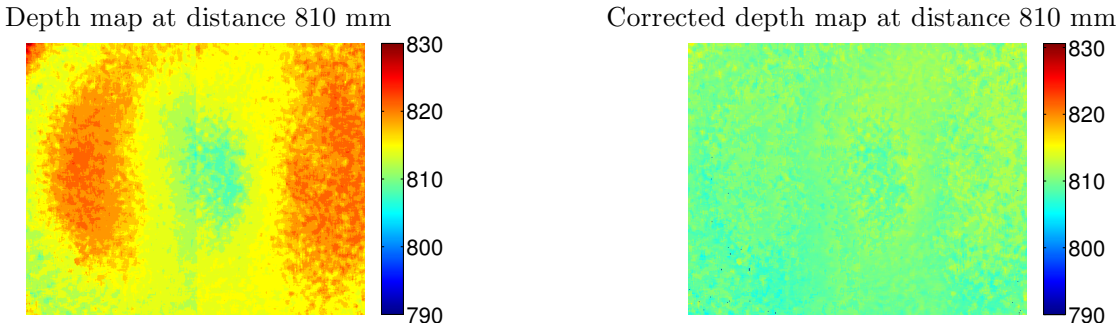


Figure 12: Uncorrected depth map (left) and corrected depth map (right) at distance 810 mm. The correction is performed using algorithm 1.

the first used disparity value correction and the second used polynomial correction. The results, seen in Figure 11, indicated that algorithm 1 was the most effective, providing a lower mean error after correction than algorithm 2. The second method of fitted polynomial would probably be more effective if the correction model was to be used in a larger span of distances and fitted to the disparity values instead of the mm values.

An alternative method is suggested by Herrera (2012) [45], that proposes exponential weighting of the disparity correction matrix $\boldsymbol{CM}$. Herrera's results were promising, measured over a larger distance span (0.8–4 m) than in this project. The project was only concerned with distances in a small interval, below 1000 mm. In this interval, the distortion was found to be relatively constant, yielding good results for algorithm 1.

5.3 Depth resolution

The Kinect depth map has a resolution of 640×480 pixels which provides a fixed spatial resolution given the distance to the object. This spatial resolution was determined by placing the sensor at different distances from an object and noting the metric spacing between the x and y points in the mesh grid spanned. The real world coordinates were calculated as described in Chapter 4.2. The object chosen for scanning was a flat book. The measurement distances d were 600–1000 mm with 10 mm steps.

Theoretically, the IR camera has an approximate horizontal field of view (FOV) of 58 ° and a vertical FOV of 45 ° [39]. Considering the area spanned by the FOV at a given distance d and the fixed pixel resolution of 640×480 pixels, a theoretical spatial resolution was obtained through

$$Res_{x,y} = \frac{2d \tan(\alpha/2)}{S} \tag{22}$$

where α is the FOV in the x and y-direction (horizontal and vertical) and S is the pixel span in the x and y-direction.

Studies regarding the depth resolution required the same test as for the spatial resolution to be executed; capture of a book at distances between 600–1000 mm with 10 mm steps. Taking the mean of the unique depth differences of the book surface, provided an indication to the depth resolution.

Equation (12) was used to provide a theoretical resolution for comparison to the measured resolution.

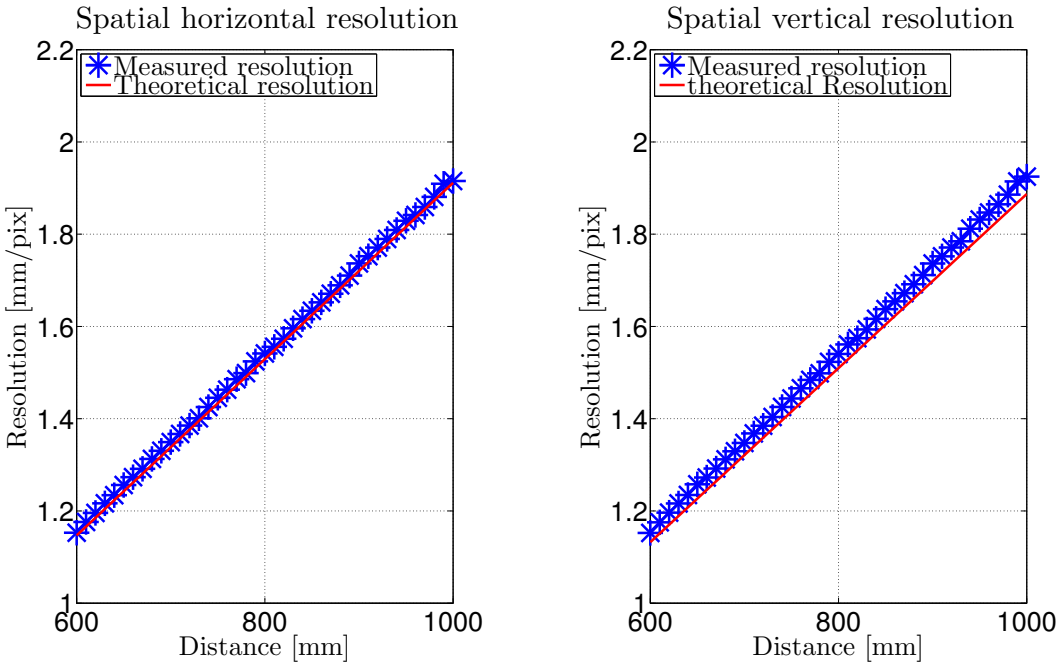


Figure 13: Spatial (x,y)-resolution of the Kinect depth map.

5.3.1 Resolution results

The results of the spatial resolution can be seen in Figure 13. From this figure the spatial resolution at distances between 600 and 1000 mm were determined to be below 2 mm both in x and y-direction [39].

The depth resolution measured can be seen in Figure 14. From this figure it can be concluded that the depth resolution at distances around 600–1000 mm is around 1–3 $^{\text{mm}}/_{\text{pixel}}$.

5.3.2 Resolution discussion

In Chapter 5.3, the resolution of the Kinect was established and validated. The results were in good agreement with the theoretical values both in spatial and depth resolution. As seen in the right plot of Figure 13, the y-resolution compared to the theoretical resolution of a 45 ° FOV camera was compared. There existed some minor error of 0.1 mm which might originate from the FOV of the IR camera not being exactly 45 °, degrading the theoretical resolution.

As observed by Andersen (2012) [40] the depth resolution varies with the distance from the sensor. These results are in agreement with the findings of this report.

In the current setup, depth values in rounded mm units were delivered by

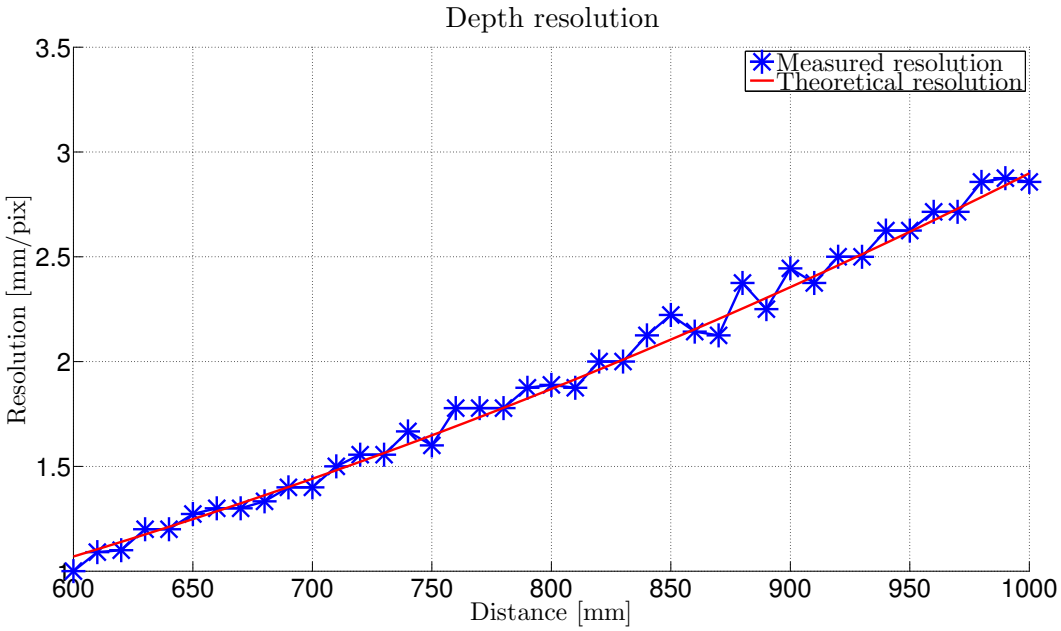


Figure 14: Depth resolution (i.e. z-resolution) of the Kinect depth map.

the libfreenect library [36], degrading the measured resolution. A more accurate resolution would have been achieved using the disparity values and performing the conversion to mm by equation (9).

The observed resolution was deemed sufficient by the orthopedist specialist at FotAnatomi [10]. However, several products exists on the market today with better resolution scanners.

5.4 RGB-Depth alignment validation

The RGB image supplied by the Kinect of 1280×1024 pixels was cropped to 1280×960 pixels by removing the bottom band of 1280×64 pixels. The RGB image is observed to be a scaled version of the depth image. In order to have pixel-to-pixel correspondence the depth map was enlarged by a factor of 2, replicating the rows and columns, to a size of 1280×960 pixels.

To determine how well the RGB-D alignment performs, an experiment that validated the alignment of the frames was conducted.

Six elevated post-it notes were on a table and capturing the scene with the Kinect, as can be seen in Figure 15. By manually marking the center of the post-it notes in the RGB image and taking the mean of the four manually selected corners of each post-it note in the the depth image, a comparison between the pixel positions could be done. The experiment was conducted at 800, 900, 1000

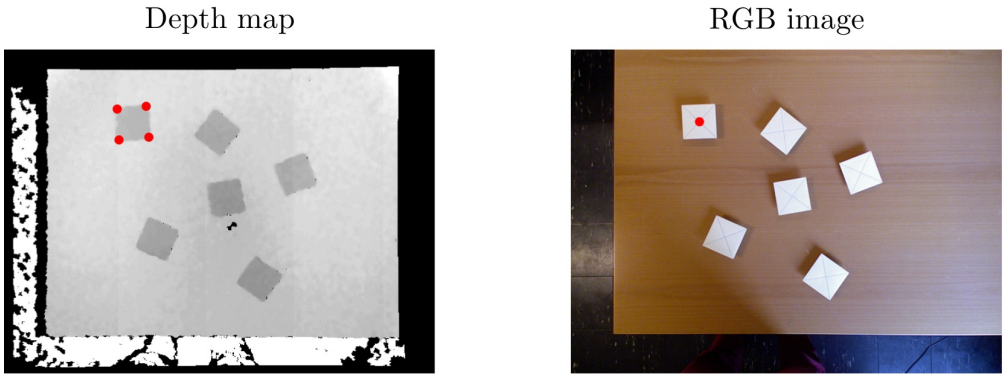


Figure 15: RGB-Depth alignment validation. The figure shows a capture of six post it notes where the alignment between depth map and RGB image was tested. To the left, the four corners of the first post-it in the depth map is marked in red. To the right, in the RGB image, the center of the post-it note is marked in red.

and 1100 mm from the post-it notes for two Kinect devices.

5.4.1 Alignment results

The results of the alignment validation can be seen in Table 2. Here, the offsets between the depth and the RGB image for four different distances are shown.

Table 2: Alignment offset between depth and RGB images from two Kinect devices. The offsets listed are mean values of six measurement points.

Distance	Kinect 1 offset (pixels)		Kinect 2 offset (pixels)	
mm	x	y	x	y
800	-0.17	4.91	-0.05	6.25
900	0.58	5.33	-2.08	5.25
1000	0.58	5.83	0.17	4.08
1100	0.50	5.17	-1.92	4.17

5.4.2 Alignment discussion

The results presented in Table 2 shows that the RGB frame is shifted 5 pixels in y, compared to the the depth image. This result is consistent between the devices and over the four distances measured.

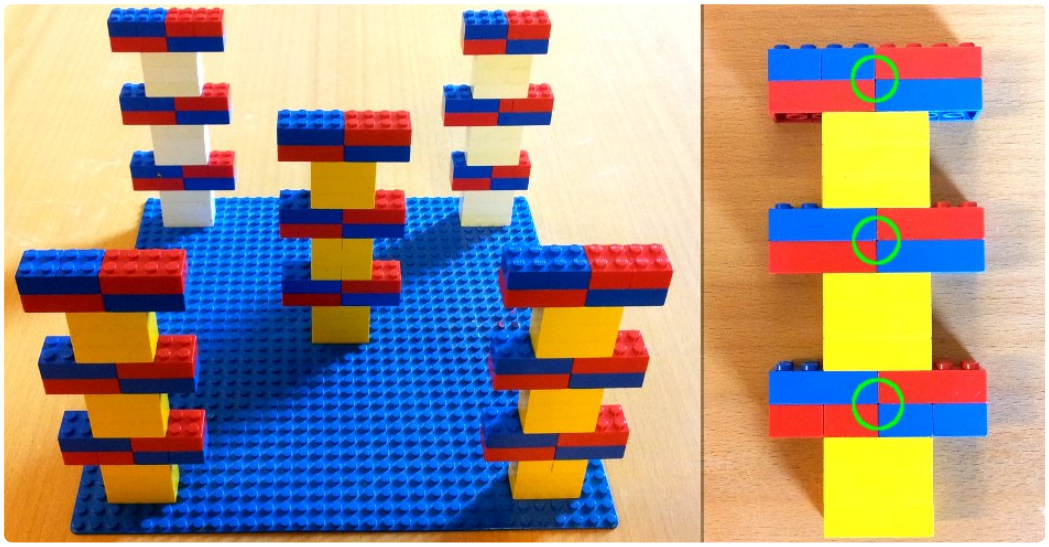


Figure 16: Known object built out of Lego. The Lego structure consisting of 5 towers with 3 levels each, making a total of 15 measurement points. The green circles in the right image shows the intersection used as control point.

The shift in x is negligible since it is a sub pixel shift for Kinect 1, but in Kinect 2 there is in average a shift in x of about 1 pixel.

Correcting these errors were done through a simple shift down in the y-direction of both the RGB images by 5 pixels, and a shift of 1 pixel for Kinect 2 in the x-direction. The cause of this misalignment is assumed to be an effect of the Kinect's intrinsic and extrinsic parameters for both depth and RGB cameras.

5.5 Depth measurement validation using known object

To further investigate the 3D scanning capabilities of the Kinect, a known object was scanned. The object was built from Lego building blocks who's dimensions are well established [47]. Using these established dimensions a ground truth for the object was modeled.

The known object consisted of five towers with three measurement points in each tower, as seen in Figure 16. Distances between these 15 measurement points spanned distances along the Cartesian axes as well as space geometrical distances in 3D. To perform the validation, calculated theoretical distances and measured distances were compared. Testing distances along individual axes was an effective way of identifying errors in x,y and z components separately. A total of 60 distances were measured, as can be seen in Figure 17.

In order to detect the measurement points in the scans, the points were marked

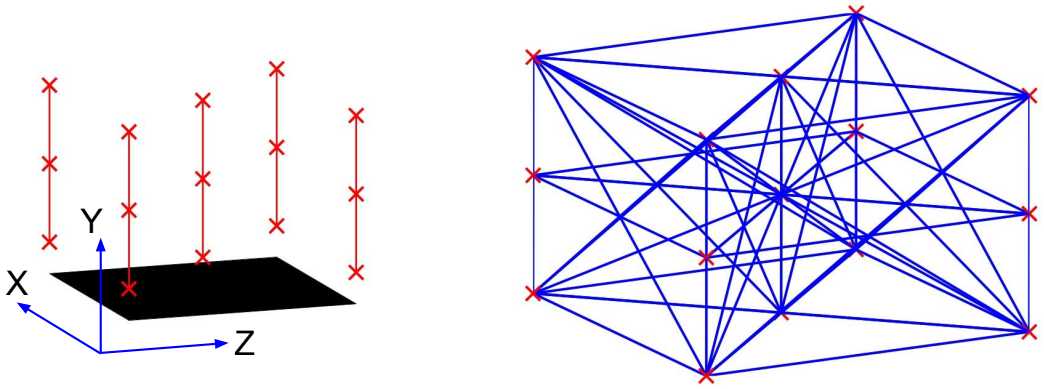


Figure 17: Distance measurements of known object. The five Lego towers (left) with markers are shown resulting in 60 distances between 15 points to be measured (right).

as intersections of colored Lego blocks (red-blue), as seen in Figure 16. Using the RGB image, an accurate manual identification of the measurement points was made.

Furthermore, the scans obtained were median averaged over 10 consecutive images to reduce the random errors introduced by the sensor.

The following scanning distance intervals were examined

- A near scan, below the recommended minimum scanning depth of 800 mm but as closest 640 mm.
- A far scan, evaluating distances over the minimum scanning distance of 800 mm but as farthest 1000 mm.

Measuring the various distances shown in Figure 17, and comparing them to the theoretical distances, yielded the absolute error in mm. Conversion to relative errors were made according to

$$e_r = \frac{|\hat{l} - l|}{|l|} \tag{23}$$

where l was the theoretical distance and $\hat{l}$ was the measured distance.

5.5.1 Depth measurement results

As described in Chapter 5.5, the Lego towers were scanned at two distances from the sensor. The results of the near scan, where the Lego was under 800 mm from

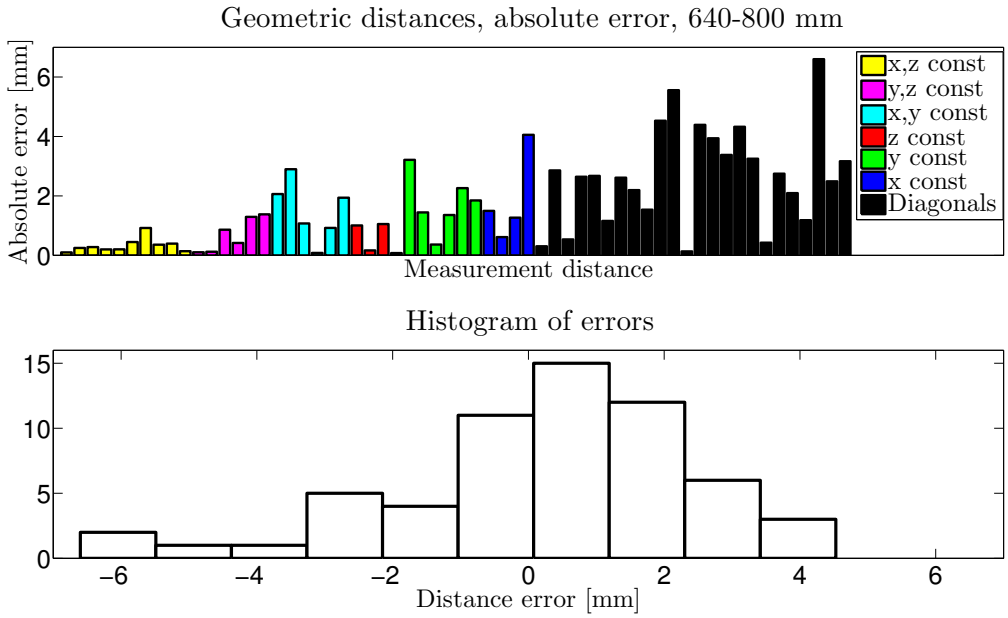


Figure 18: Results of distance measurement validation at distances 640–800 mm. Top: absolute error in mm of 60 distance measurements between Lego tower control points. Bottom: histogram of differences, depicting the distribution of the errors.

the sensor, can be seen in Figure 18. The color of the bars represents a different geometrical type of measurement, e.g. along the x-axis or space geometric diagonals.

As seen in Figure 18, the maximum absolute error was below 6 mm. The mean absolute error was calculated to 1.69 mm with a standard deviation of the absolute error of 1.52 mm. As seen in the histogram, the most frequent errors were centered around 1 mm. Additionally, the mean relative error was calculated to 1.4 %.

Regarding the far scan, the results can be seen in Figure 19. The mean absolute error was 1.24 mm with a standard deviation of 0.99 mm. As seen in the histogram, the most frequent errors were centered around -0.5 mm. The mean relative error was calculated to 1 % and the maximum absolute error was below 4 mm.

5.5.2 Depth measurement discussion

The near scanning distances, as seen in Figure 18, had a greater error than the far scanning distances seen in Figure 19. In the near case, the errors were gathered around a higher mean than in the respective far scan case. Furthermore, as seen in the histograms, numerous errors of 2–4 mm as well as large errors at -6 mm existed in the near scan, indicating some kind of artifact that disturbed the measurements. As seen in the far scan case, errors were accumulated around 0–2 mm without large

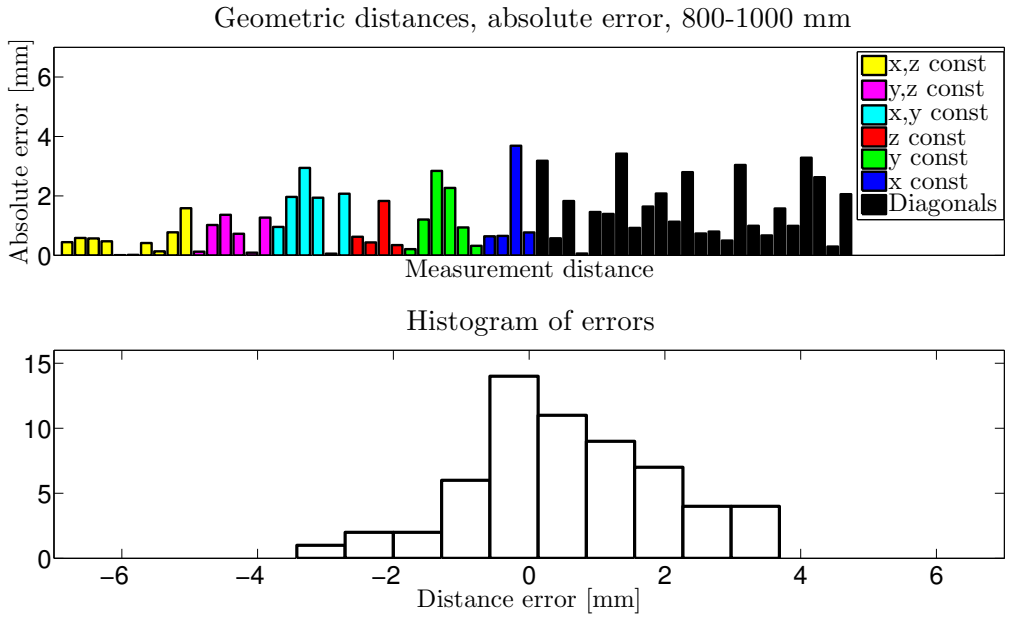


Figure 19: Results of distance measurement validation at distances 800–1000 mm. Top: absolute error in mm of 60 distance measurements between Lego tower control points. Bottom: histogram of differences, depicting the distribution of the errors.

outliers.

Considering the scanning distance just outside the recommended minimum range, the results were promising. At coarser resolution than the near scanning distance, the far scan indicated a lower mean error. It was concluded that scanning was to be conducted above, but close to 800 mm.

6 Proposed system for insole modeling

This chapter describes in detail the proposed system for foot scanning, foot modeling and insole design. The proposed system involved the steps seen in Figure 20. Note that specifications for fabrication are not included in the proposed system.

Figure 20: The proposed insole modeling system involves three steps.

6.1 Foot scanning

Input:	Patient foot and reference system
Output:	Depth maps in mm from 3 angles RGB images from 3 angles

Scanning of the foot was made in a non-weight bearing state, preserving the natural form of the foot. This was accomplished by laying the patient down on his/her chest with the feet hanging out from the bed [19].

A 3D foot model for insole design needed to feature the full bottom of the foot, a few cm of the sides and a clear view of the Achilles tendon [10]. To acquire sufficient information, the foot was captured from three point of views, as seen in Figure 21.

The scanning was conducted above 800 mm from the foot, as described in Chapter 5.5. The distance between the Kinects on the table was 60 cm and the angle between them were about $50\,° - 90\,°$, with centered view on the foot. Furthermore, the table Kinects were placed slightly elevated with regard to the foot, tilted down to center on the foot. The third Kinect was placed at the same scanning distance, angled perpendicular to the Achilles tendon.

To avoid laser interference between the laser emitters of the devices, the emitters were manually blocked sequentially during capture.

A capture interface for foot data was built. This interface allowed for capture from several Kinect sensors simultaneously and can be seen in Figure 22. The data collected was median averaged over consecutive image frames. In order to keep the capture rapid and to minimize foot motion, the capture set sizes were chosen to 10 frames with a total capture time, including emitter blocking, of about 5 s. The depth map and RGB image retrieved, were aligned and the depth map was converted to mm using libfreenect. Also the methods described from

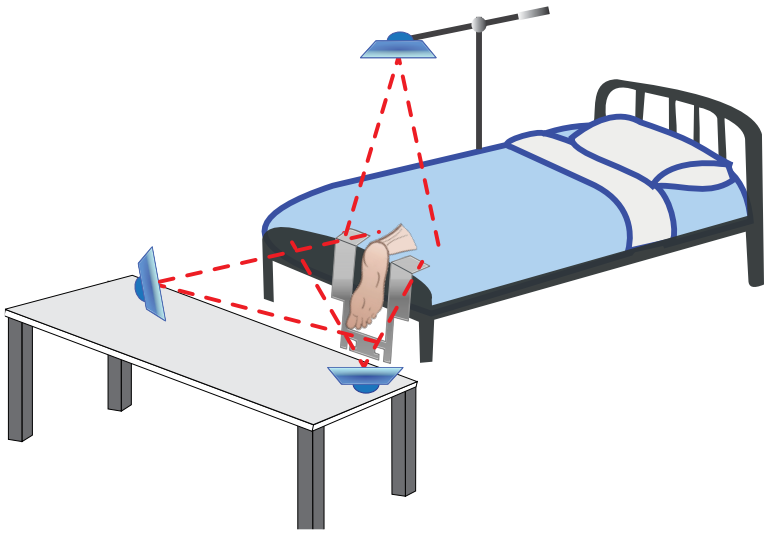


Figure 21: A schematic overview of the system setup including a patient foot and three Kinect sensors. Also notice the gray structure surrounding the foot, which is reference system.

Chapter 5.2 and 5.4 were used for distortion correction and alignment correction respectively.

In Figure 23, the reference system mentioned in Chapter 3.2 can be seen. This system was placed around the foot with reference markers facing the cameras, as can be seen in Figure 22. The reference system is further covered in Chapter 6.2.1.

6.2 Foot modeling

Input:	Depth maps in mm from 3 angles RGB images from 3 angles N theoretical reference markers (x,y,z)
Output:	1 mesh model with color

To use the captured information from different angles, a meshed foot model was constructed that described the shape of the foot as well as the color.

6.2.1 Reference system

The reference systems consisted of blue lego blocks, placed near the scanned foot, as seen in Figure 22. The purpose of the reference system was to provide reference points between the different capture sets for 3D registration.

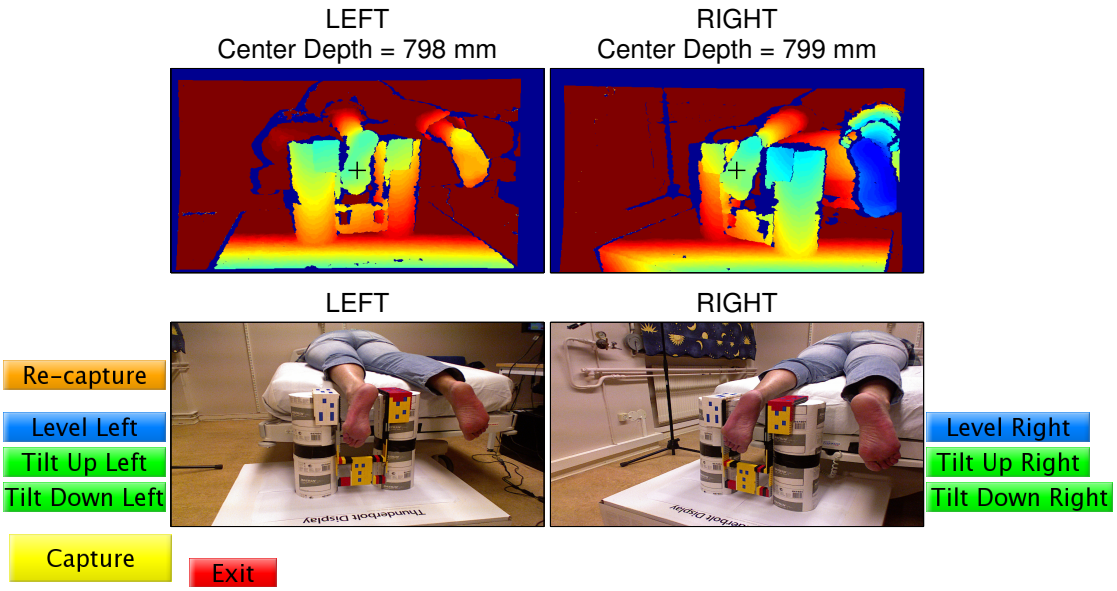


Figure 22: A capture interface example including buttons for tilt, re-capture and data export of two Kinect sensors. The top images show depth maps color coded in mm where the center depth is indicated in the title. The bottom images show the corresponding RGB images.

Various systems were experimented with, including reference markers painted on the foot, different shapes of markers, different colors and systems varying in size and orientation, relative to the foot. Placing reference markers on the foot yielded good results, but from a practical point of view, painting on the patient's feet was unwanted.

The final reference system used can be seen in Figure 23 and was chosen as a set of 20 blue reference markers, where 12 markers lay in the plane of the foot and 8 markers lay in the plane parallel to the Achilles tendon. The system was constructed using Lego blocks, which enabled simple translation to a theoretical model with known distances. The theoretical measurements of the the Lego blocks were provided by Caillau [47].

6.2.2 Detection of reference markers

Input:	Depth maps in mm from 3 angles RGB images from 3 angles
Output:	N captured reference markers (x,y,z) Aligned depth maps in mm from 3 angles RGB images from 3 angles

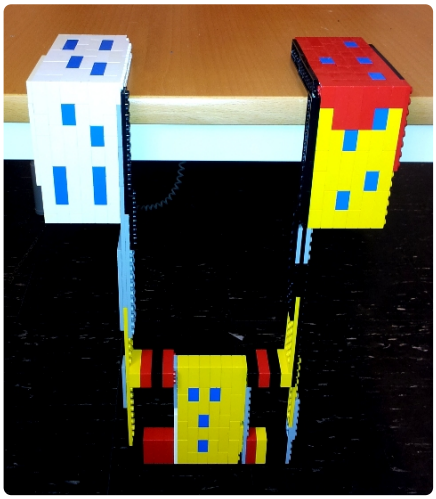


Figure 23: The reference system built out of Lego blocks. The blue blocks represent reference markers. 20 reference markers are present, 8 in the coronal plane and 12 in the transverse plane of the patient's body.

Algorithm 3 gives a brief overview to the detection of reference markers from the RGB image.

Algorithm 3 Marker detection algorithm

1: *Read RGB image*
2: *Enhance strong blue pixels using equation* (24)
3: *Threshold intensity image to binary image*
4: *Select area of interest*
5: *Find centroid, eccentricity and area for marker candidates*
6: *Feature based exclusion of false markers*
7: *Select strongest markers w.r.t. area*
8: *Export marker centroids*

In step 2, algorithm 3, the blue Lego blocks were enhanced from the rest of the RGB image. This was done by creating an intensity image I, containing the difference between the blue component and the largest red/green component for each pixel.

$$I = Blue - max(Red, Green) \tag{24}$$

In step 3-5, image I was thresholded to remove any pixels not containing enough blue color. A region of interest that encapsulated the reference system was manu-

ally selected. Furthermore, connected components of pixels were identified, resulting in marker candidates.

In step 6-8, exclusion of false markers from the marker candidates was done, using eccentricity, removing segments resembling pixel lines more than blocks. Finally, the markers were identified as the largest area components in the image.

An overview of the detection can be seen in Figure 24. The figure shows the marker detection process from a top view capture.

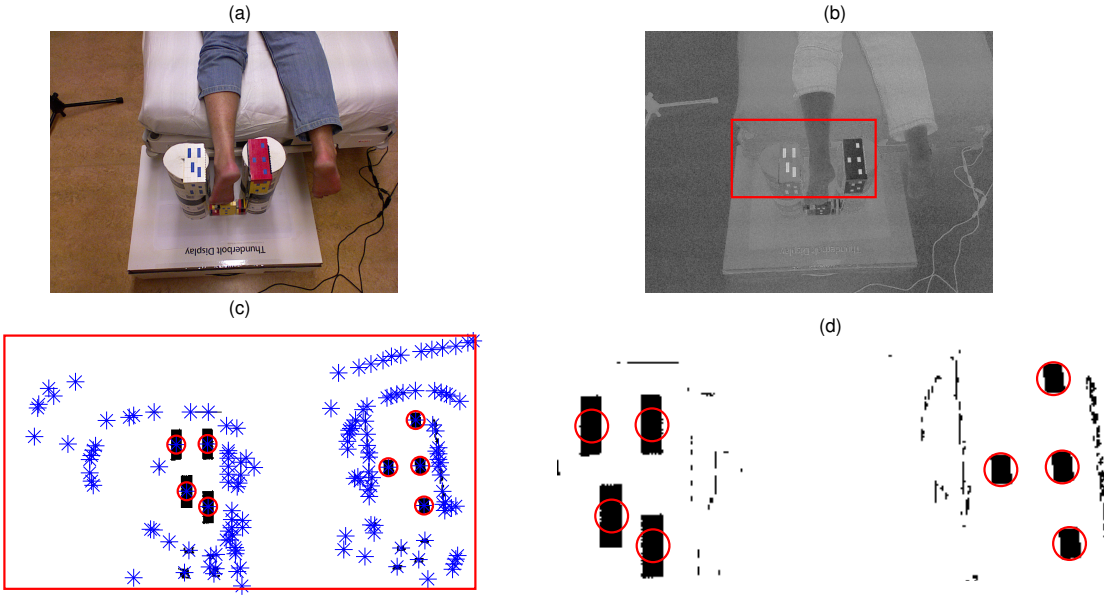


Figure 24: The marker finding process for top view, 8 markers. Sub figure (a) shows the original RGB image from the sensor. The first step (b) involves extracting strong blue pixels from the region of interest, seen in red. The image was thresholded and connected pixel segments were identified for marker candidates, as seen in (c) (i.e. the blue stars). Finally markers were selected, marked as red circles in (d) with respect to eccentricity and area.

When the marker centroids in the RGB images had been determined, the corresponding pixels in the depth map was determined using the alignment between the images described in Chapter 4.3. The depth markers were converted to (x,y,z)-coordinates as described in Chapter 4.2.

6.2.3 Multi-view registration

Input:	N captured reference markers (x,y,z) N theoretical reference system markers (x,y,z) Aligned depth maps in mm from 3 angles RGB images from 3 angles
Output:	3 registered mesh models with color

Proceeding from multi-view scanned data to a mesh model of the foot with color required some data processing. The depth map and RGB images of the captures were processed as described in Chapter 4.2 and 4.3. This converted the depth maps into point clouds with color information for each point. Furthermore, surfaces to the RGB-D data was constructed using 2D Delaunay Triangulation in the image plane of the capture, described in Chapter 3.1. The color of the triangular faces were determined by interpolation between the colors from the triangle vertices. Concluding, a mesh model with color was determined for each capture.

The captures contained data for the whole FOV, foot and room. Segmentation of the foot was done by selecting a volume of interest in space, were the (x,y,z)-coordinates of the foot resided.

To bring the three foot data sets sequentially to one shared coordinate system, they were rotated and translated as a rigid body to the origin of the reference system (Figure 23), located in the centroid of the scanned foot. This was done using the detected marker coordinates and the MATLAB implementation of the absolute orientation quaternions by Wengert and Bianchi (2010) [48], based on Horn's method [32] described in Chapter 3.2. An example of such a registration can be seen in Figure 25.

As seen in Figure 25 a fit of the captured reference markers to the theoretical reference markers was accurate with error residuals per point of:

- *Left set (blue)*: 2.03 mm
- *Right set (black)*: 1.59 mm
- *Top set (green)*: 1.29 mm

6.2.4 Removal of overlapping data in the foot bed

Input:	3 registered mesh models with color
Output:	1 mesh model with color

Having registered the foot captures to one shared coordinate system, overlapping segments existed due to the overlapping field of views of the cameras. In order to

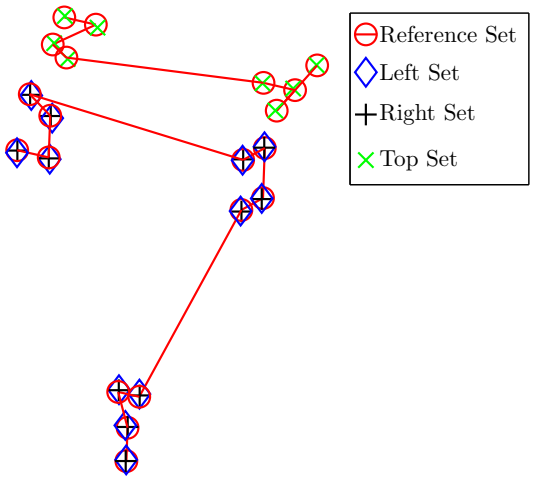


Figure 25: The plot shows 20 markers registered to the reference system. The red set is the reference markers calculated from Lego dimensions. The blue, black and green sets are detected marker sets from the captures. The error residual per point is approximately 1.64 mm.

eliminate the overlaps occurring on the bottom of the foot, segments were removed. This was done by cutting the data sets at the line dividing the bottom of the foot into left and right. This can be seen in the rightmost model of Figure 26, where the left part of the foot sole originate from the leftmost Kinect and the right part of the foot originates from the rightmost Kinect.

6.3 Insole design

Input:	1 mesh model with color
Output:	Insole model

The following section will describe the available tools in the graphical user interface (GUI) for insole design. The aim of the GUI was to provide basic tools for insole design. The GUI was built in MATLAB and can be seen in Figure 27.

6.3.1 Pronation/supination correction tool

Given the model described in Chapter 6.2 eventual pronation/supination can be determined. By establishing the misalignment between perpendicularity between the Achilles tendon and the natural plane of the foot. Having established the angle

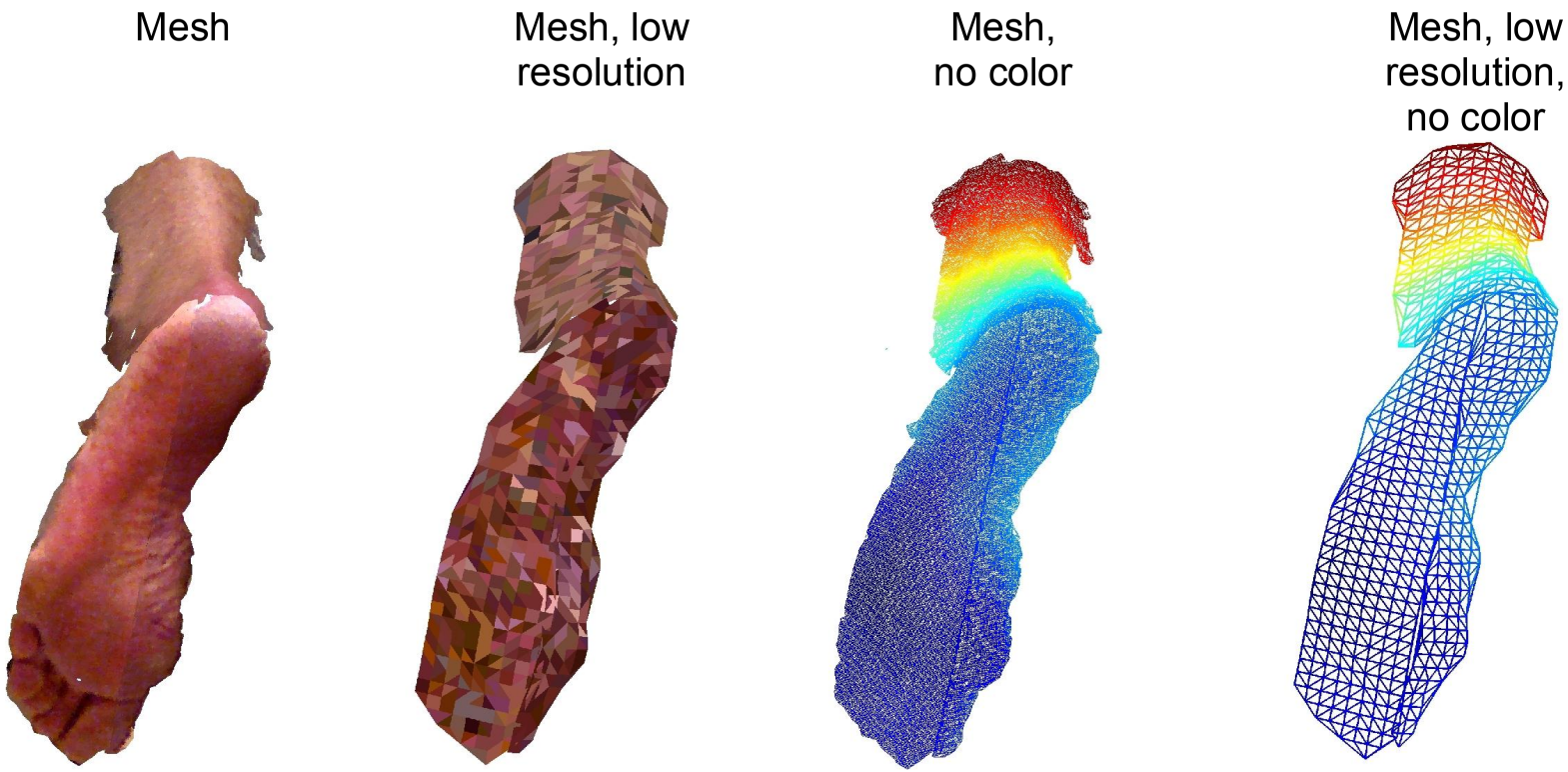


Figure 26: Mesh model of a foot created from three captures. The first model is used by the project and the mesh is colored by interpolating the colors of the vertices defining each triangle. The second model is a low resolution version of the first model where the triangle colors are the mean of the vertices defining each triangle. The third model shows the mesh that the models are based on and the fourth model shows a low resolution version of the mesh, for easier visualization.

of misalignment, the bottom of the insole is placed perpendicular to the Achilles tendon. This will keep the Achilles tendon in a neutral position during standing and gait.

This is done by rigid body transformation of the foot model, to achieve the proper orientation with regard to the bottom of the insole.

6.3.2 Pelott tool

To design a smooth pelott elevation in the foot, a neighborhood function was used

$$g(x,y) = Ae^{-[\frac{(x-\mu_x)}{2\sigma_x}+\frac{(y-\mu_y)}{2\sigma_y}]} \tag{25}$$

where A was the peak amplitude, (μ_x,μ_y) is the point where the function is applied and σ_x, σ_y were the standard deviations in x and y respectively. When applied at a point (μ_x,μ_y), a neighborhood will be elevated.

In order to retain the elevation to a closed area, the choice between a free hand encapsulation and a template form in the shape of an egg was used. The shape of the form was determined by

$$x = \pm\frac{\sqrt{(a-b)-2y+\sqrt{4by+(a-b)^2}}}{\sqrt{2}}\sqrt{y} \qquad for\ a \geq c \geq 0 \tag{26}$$

where $c = a - b$, and $b = ra$, where a and r are user defined ratio constants. The constant a controls the height of the template.

The template form could be placed freely in the (x,y)-plane, and the enclosed area was altered with the use of the neighborhood function from equation (25). Areas outside this form was not altered.

6.3.3 Graphical user interface layout

The tools described above are implemented as a GUI environment. The interface was divided in the following four main panels:

1. *Left:* Viewing and editing patient data.
2. *Right:* Foot mesh model with color. Apply foot modifications here.
3. *Bottom:* Fields for specifying modification tool parameters.
4. *Center:* Foot model for viewing impact of changes, also directly editable.

Additional to the panels, a toolbar located in the top of the GUI holds all the tools available for the user

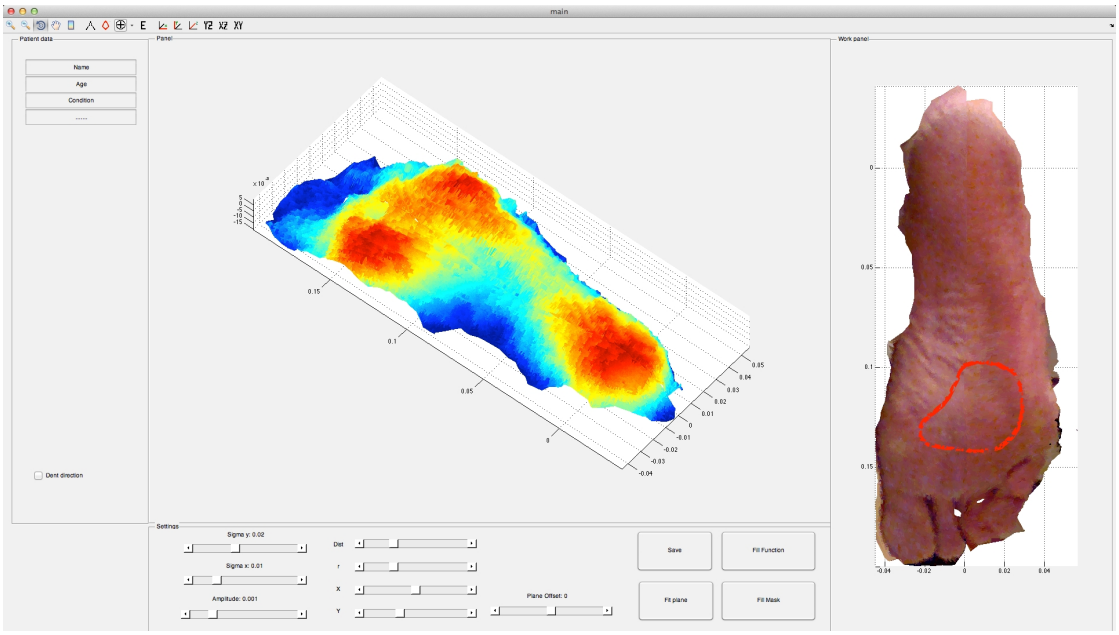


Figure 27: GUI for insole design. The figure shows two different views of the foot model. The left mesh model is seen without color and the mesh model to the right is seen with color and an area drawn in red, marking the placement of the pelott.

- Free hand pelott using neighborhood function.
- Pelott enclosing form.
- Place key points, e.g. at heel and footpad for measurements.
- Add support for counteracting pronation/supination.
- Freely place and rotate reference plane for visualization.

6.3.4 Insole model

After designing the insole, the method for rectangular mesh grid approximation, described in Chapter 3.3, was used to create the contact area of the insole. The simplified point cloud was then meshed as described in Chapter 3.1.

The contact area of the insole was placed in a 3D block which was converted into an STereoLithography (.stl) file, a common CAD format compatible with numerous softwares [49] for CNC cutters and 3D printers. The mesh model of the insole can be seen in Figure 28.

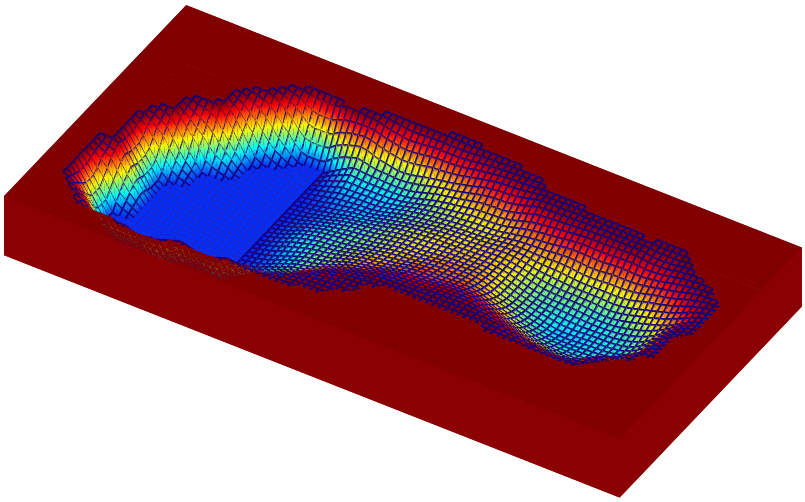


Figure 28: The insole model after design. Notice that only the part of the insole that faces the foot has been designed, i.e. the side and bottom of the insole is untouched. Also notice that no pelott is included in this model.

6.4 Block diagram system overview

The proposed system described in Chapter 6 can be summarized in a block diagram, as shown in Figure 29.

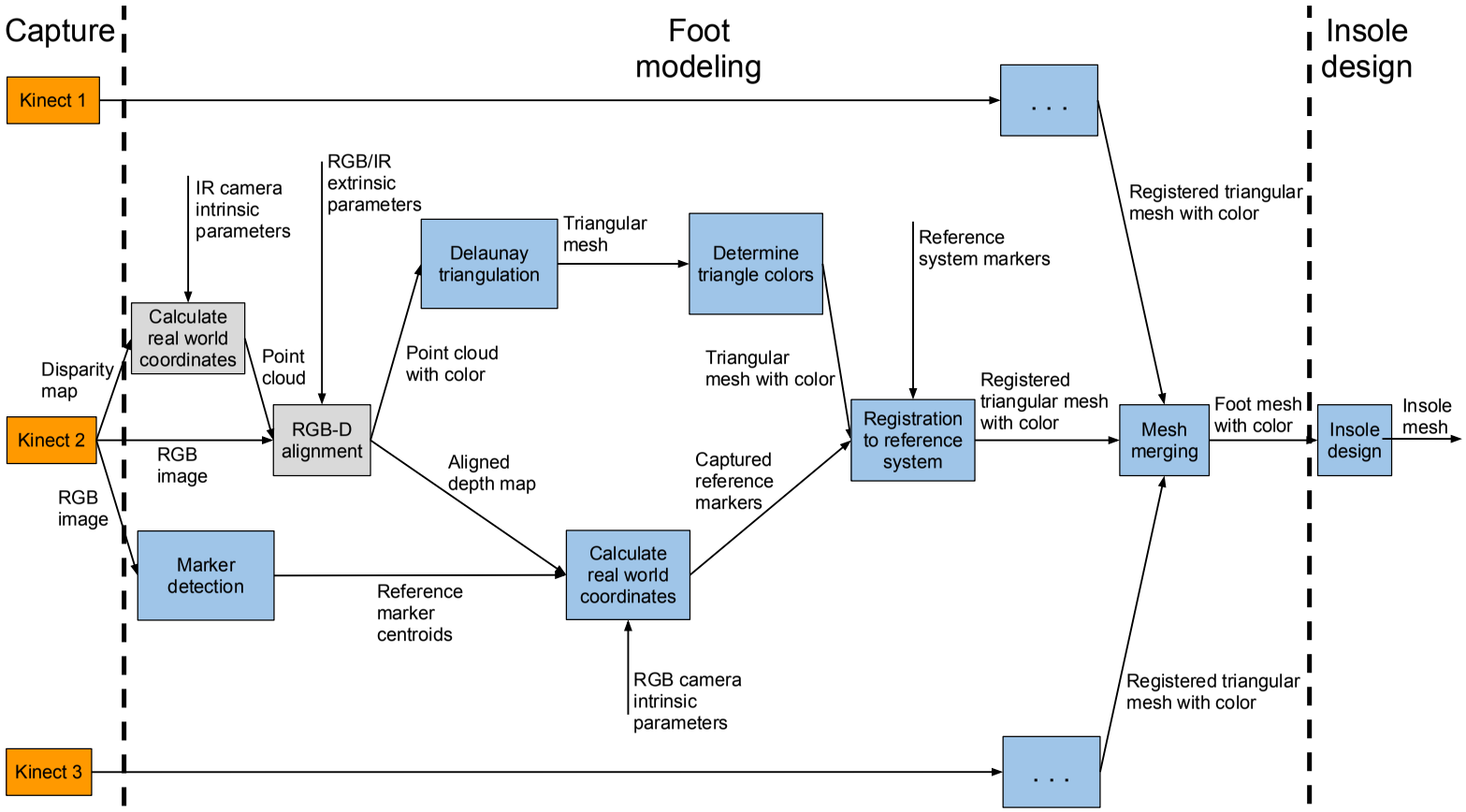


Figure 29: Block diagram of inputs and outputs in the proposed system for computer-aided insole design. The original outputs from the Kinects; the disparity maps and RGB images, are used to create an insole mesh using an external reference markers system. The figure shows the data flow from three Kinects, even though the details are only displayed for Kinect 2. The orange blocks shows the hardware. The gray blocks shows the usage of pre-implemented functions and the blue blocks shows the steps developed for the proposed method.

7 Proposed system remarks

This chapter discusses the system proposed in Chapter 6, which covered foot scanning, foot modeling and insole design.

7.1 Foot scanning remarks

Two devices For capture, three Kinect sensors were proposed. The project only had access to two devices. This in effect caused one camera to capture multiple views. This was both unwanted and unpractical since the setup could not be held rigid and foot movement between captures was possible. Given a third Kinect, the results might have been more stable. The manual moving of one camera to capture the top view could have affected the results, even though the exact placement of the top camera was not important.

Capture interface The capture interface in Figure 22 had simple purposes, allowing recapture and tilting of the Kinects. Development of an extensive capture interface was not a priority. The requirements were to allow simple capture and data verification of several Kinects. Regarding the median filtered captures, the 10 images elongated each total capture time to about 1 s and mitigated the normally distributed noise in the depth readings. In the aspect of foot movement during scanning, times above 5 s were unwanted and the best results were obtained for shorter rather than longer capture sets.

Sensor problems An issue with simultaneous capture was the laser interference between the Kinect devices. The laser emitter was constructed for continuous use and can not be switched off with the current framework. To simply cover the emitters functioned as a working solution but elongated the total capture time considerably.

The nature of the Kinect hardware, with one IR emitter located a distance from the IR camera, gave rise to laser shadows at edges. These shadows were created when a location in the IR image was not hit by the laser emitter. This was noticed when taking data captures of the foot from wide angles, as the side of the foot disrupted the laser emitter from reaching particular areas.

A larger angle between the Kinect and the surface normal lowered the resolution. Therefore, the highest resolution was achieved when taking the data capture perpendicular to the object.

More than 3 Kinects Using more than three kinect angles could improve the resolution in areas with a wide angle to the cameras. Also, using for example four

Kinects where the top Kinect is interchanged for two Kinects, a more detailed model of the Achilles tendon and the lower leg could be achieved. Weather to use more Kinects or the proposed number of three, depends on the information needed regarding the surroundings of the foot.

7.2 Foot modeling remarks

Reference system The reference system seen in Figure 23, was the product of some experimenting. Placing the markers close to the foot, but not on the foot required a custom structure. The lego block provided a good alternative, due to their building capabilities and known dimensions.

The reference system of Lego was used with success in the project and provided a theoretical ground truth for the registration. Attempting to place markers that spanned the entire area of the foot ensured that the reference markers represented the whole foot surface.

Marker detection Detection of the markers was performed through operations such as thresholding and using region properties such as area and eccentricity. The now semiautomatic marker detection would be improved in case of a rigid setup. As some steps in the detection such as selecting a region of interest could be automated. The resolution of the RGB image, being twice the resolution of the original depth image, provided accurate marker detection as seen in the rightmost images of Figure 24.

Registration Regarding the registration performed in Chapter 6.2.3 and seen in Figure 25, the results were promising with residuals of about 1–2 mm per fitted point. Keeping in mind the spatial resolution of 1–2 mm and the depth resolution of 2–3 mm, the errors were low.

7.3 Insole design remarks

The specific insole design tools described provided insole design features such as supination/pronation counteraction and pelott placement. The design interface was sufficient for the task of applying extra support in the case of pronation and supination and creating a pelott for local elevation. In the interface, seen in Figure 27, the mesh model with color provided accurate information regarding the condition of the sole and enhanced the impression of working with an actual foot.

8 System evaluation through case study

Comparison of the proposed system from Chapter 6 to existing products was made by a quantitative case evaluation. The foot mesh model from the proposed system was compared to two external company insoles, all modeled from the same foot. The products compared were

1. Proposed digital system.
2. FotAnatomi [10], traditional plaster cast production method.
3. Ortolab, semi-digital method.

To perform the comparison, a patient experienced with both the traditional plaster cast method and a semi-computer assisted method was chosen as case subject. The patient was scanned with the Kinect sensors and foot modeling and insole design were performed.

The traditionally casted replica was created 2013-02-20 by FotAnatomi. The plaster cast mold is also modified with a pelott imprint, done while casting the plaster shell. The resulting mold can be seen in Figure 2 with a pelott imprint marked in red [10].

The computer assisted insole was made by Ortolab. The replica was cut out of PU-foam from a digital model of the patient's foot, in autumn 2012. Furthermore, the foam model was modified with a pelott imprint by an orthopedist from Ortolab. The foam model can be seen in Figure 30 with a pelott imprint marked in red. A comparison between the results of the three systems could then be conducted.

8.1 Subject description

The patient had a forefoot valgus deformation emerging on the outer side of her left foot due to forefoot insufficiency [10]. This caused the patient pain during several years. To remedy the pain, the Ortolab insole alternative was tried. The patient then tried the services of FotAnatomi.

8.2 Scanning patient and molds

The patient was scanned from three angles. The left and right Kinect were placed on a table as depicted in Figure 21, the top Kinect in the figure was hold manually for the last data capture set. A foot model was constructed as described in Chapter 6.2 and an insole proposition was created with a pelott imprint, mimicking the design of the mold from FotAnatomi, which was considered as the reference.

The two replicas from FotAnatomi and Ortolab respectively were scanned with the Kinect sensor.

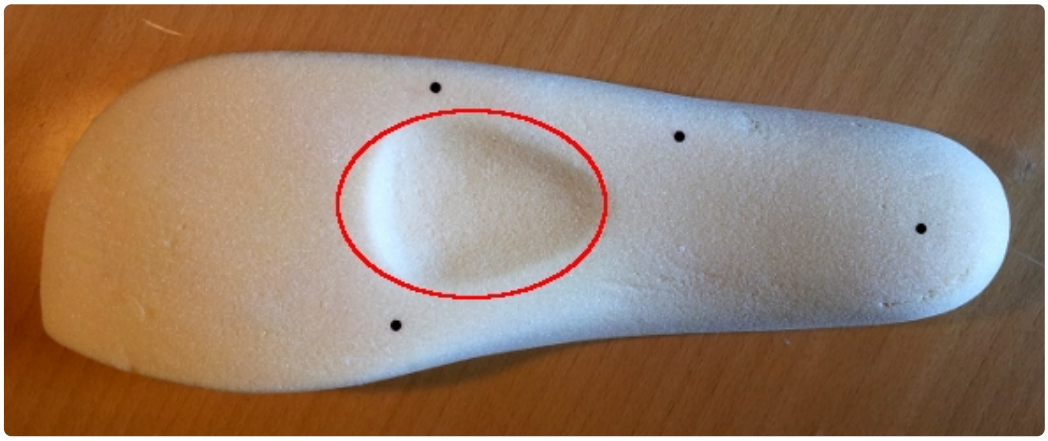


Figure 30: Model of foot in PU-foam made by Ortolab. The pelott carved into the foam is marked in red. The 4 black dots represent reference markers for registration.

8.3 Comparison method

The different foot models from the proposed system, FotAnatomi and Ortolab are hereafter referenced to as model N, model F and model O respectively. The different foot models, now digitized, all had pelott imprints for spreading the pressure on the foot more evenly.

In order to compare the models, registration to a shared coordinate system was conducted for each scan. Markers as seen in Figure 30 (black) were placed in locations where small or no modifications had been done to the models. This was done to maximize the correlation between the models after registration. These markers were identified manually and the method for registration, described in Chapter 3.2, was used. The shared coordinate system for registration was chosen to be situated in model F.

The Ortolab insole mold is a type of crisp foam, which was challenging for the Kinect to get good readings of. Model O was therefore contaminated by larger variations than model N and F. To reduce the noise impact on the comparisons, a spatial median filtering operation was applied to the projected scans. The median filtering was done with a kernel size of 6×6 pixels, representing approximately $36\,\mathrm{mm}^2$.

The model comparison was then made by projecting the mesh models into the (x,y)-plane and taking the absolute difference in mm between pair of images

$$R_{O-F} = |O - F| \tag{27}$$

$$R_{N-F} = |N - F| \tag{28}$$

$$R_{N-O} = |N - O| \tag{29}$$

Model F was used as reference measurement. The third comparison, R_{N-O} was included for completeness.

8.4 Case study results

The different foot models can be seen in Figure 31, where blue indicates nearer and red indicates farther away from the sensor in mm.

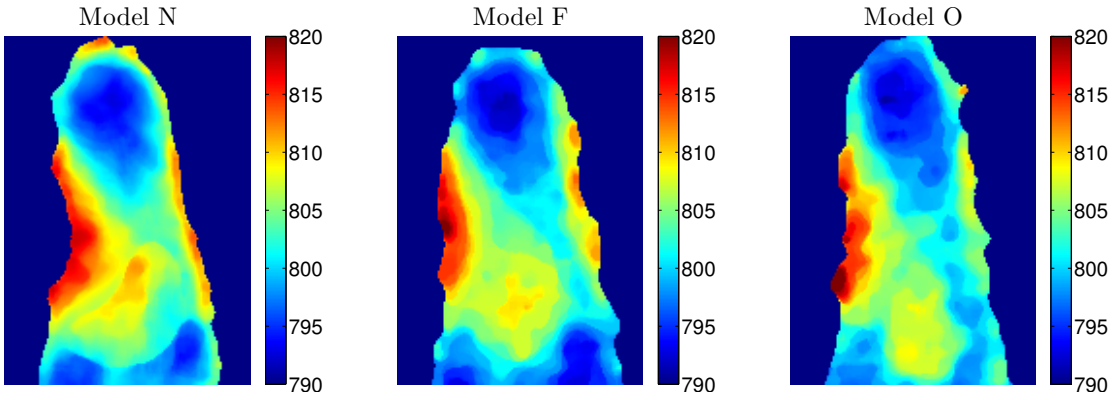


Figure 31: Color coded depth maps of the proposed system, foot model N (left), plaster casted foot replica F (middle) and Ortolab PU-foam replica O (right) as seen from underneath the foot.

The results of geometric comparison between the foot models in Figure 31 can be seen in Table 3.

Table 3: Geometric shape comparison between foot models. All distances are in mm.

	Model N	Model F	Model O
Foot height	1260	1280	1350
Heel width	580	570	600
Forefoot width	570	620	580
Arch height	24	18	21

The resulting comparison after applying equations (27-29) can be seen in Figure 32, where the color map indicates high similarity in blue and low similarity in red.

Model R_{O-F} had a mean difference of 1.26 mm, model R_{N-F} had a mean difference of 1.20 mm and model R_{N-O} had a mean difference of 1.45 mm.

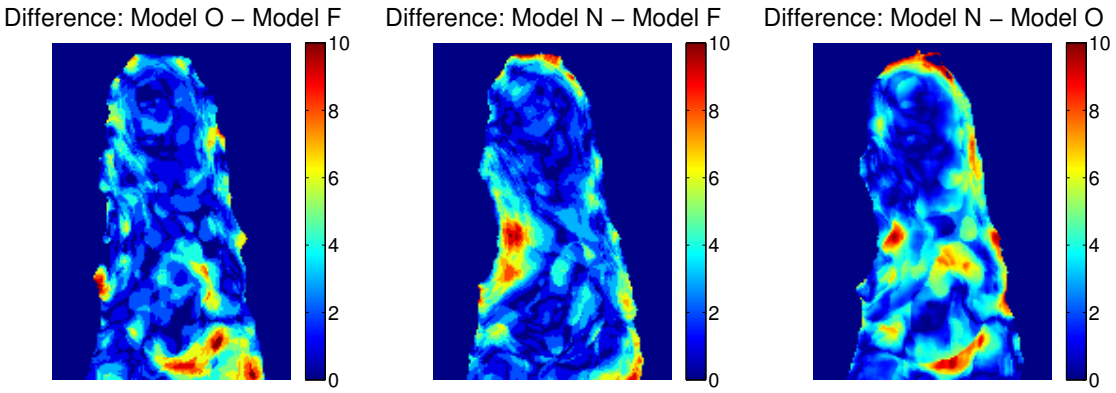


Figure 32: Absolute difference in mm of R_{O-F} (left) and R_{N-F} (middle) and R_{N-O} (right).

8.5 Case study discussion

As seen in Figure 31, the variations were higher in model O than in the other scans, even after filtering. This was noticed by the uneven color map, which were smoother in model N and F. The uneven scan was due to the PU-foam material of which the Ortolab replica was made of. Its crisp surface was very hard to scan accurately with the Kinect.

The comparison in Figure 32 of R_{O-F} equaled to a mean difference of 1.26 mm, and the mean difference R_{N-F} equaled to 1.20 mm. This gave an 0.95 times lower mean difference for the R_{N-F} comparison, that indicated that overall the differences were quite similar. Higher local differences were seen around the arch and pelott areas.

8.5.1 Local differences

As seen in Figure 31 and Table 3, model N had a higher arch than the other models. Explaining this trough biomechanics change in the patient foot, between the company models and the project's scan was unlikely - the scans are only a few months apart. The difference could be explained by the fact that even though the patient's arch in reality was as high as in model N, model O and F are shaped with a lower valve.

An insole with a very high arch height would be impossible to fit inside a shoe, therefore a maximum height for the insole exists [10]. It is possible that this was the case with model O and F and that the arch had been lowered for practical reasons. The true reason of this result however, was not determined.

In Figure 32, it can be noted that there was a large difference in the middle of the forefoot for the R_{O-F} result. This was mainly because of the pelott placement, which in model O, was 5 mm further to the front of the foot than in model F. This was a design factor which affected the insole, but did not dependent on the insole production system used. The same difference was visible in the R_{N-O} result.

Furthermore, in Figure 32, the R_{N-F} result showed less difference in the arch filled area than the R_{O-F}, indicating that it was possible to provide basic pelott design using the GUI described in Chapter 6.3.

The R_{N-O} result also showed a significant difference in the middle of the foot, which was due to the displacement between the two pelott designs.

8.5.2 Shape differences

From Table 3 it was noted that model N and model O were similar concerning width. Model F however was broader than the other models.

Model F, being broader, yielded more space around the patients forefoot than model O. The same broadening would be possible to perform on model N before fabrication if needed.

9 Direct insole fabrication

Fabrication of the insole was performed in purpose of testing fabrication methods. Using the designed insole model from Chapter 6.3.4, a .stl-file was created and sent to fabrication.

The following two options was explored for fabrication

- Cutting the insole using a computerized numerical cutter (CNC). Starting from a block of material, unnecessary material is removed until only the insole is left.
- 3D printing the insole. This method glues thin slices of material together, building the prototype.

Both options were prototyped and the result of the CNC method can be seen in Figure 33 and a small part of the foot's arch, 3D printed, can be seen in Figure 34.

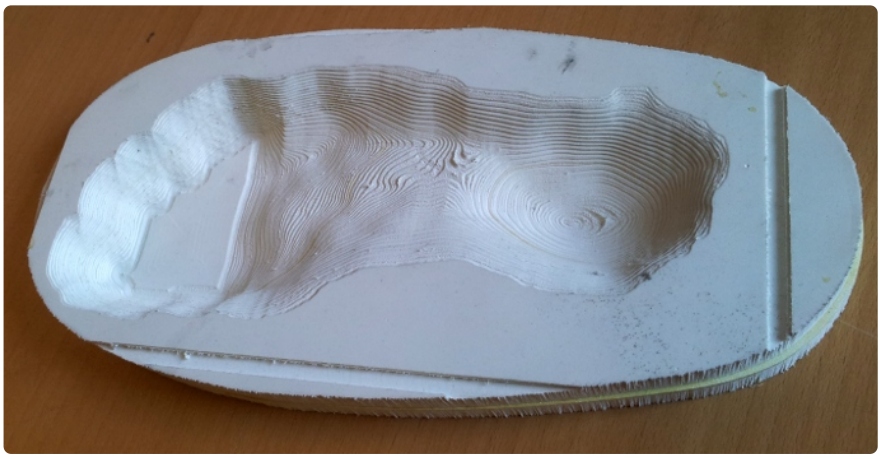


Figure 33: CNC cut insole from insole model. The resolution of the cutting was 0.3 mm [50].

Using a CNC cutter to directly produce the insole from the model as in Figure 33, the steps from foot capture to final insole are kept to a minimum. Regarding the accuracy of the cutting process; the time to fabricate the form in Figure 33 required about 3 h of fine cutting with a home-made CNC cutter. Some features were not implemented when fabricating the prototype, such as side and bottom contours. These features might prove difficult for the cutter to handle. Cutting thin soft materials is hard since no support is given for the cutter.

A 3D printer fabrication was also conducted, as seen in Figure 34. Only a part of the model was fabricated. The surface of the printed version was smoother

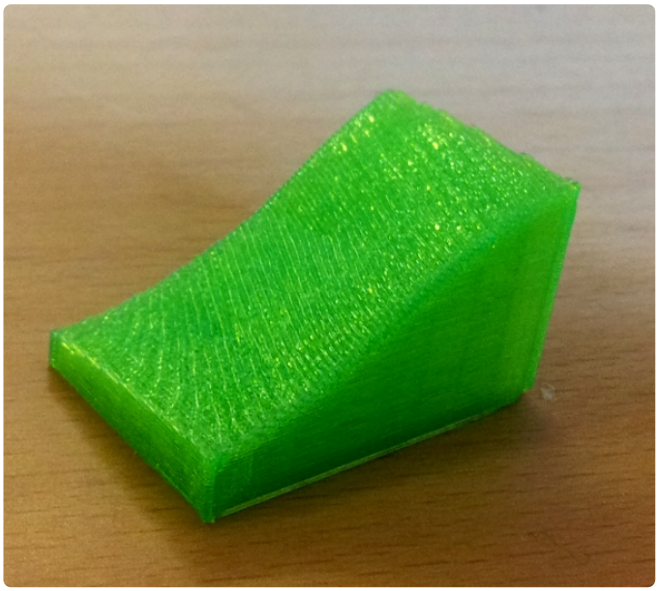


Figure 34: 3D printed insole part of insole model. The resolution of the print was 0.2 mm [51].

than the CNC cut versions for due to the higher fabrication resolution. The problem however, was printing soft material, suitable for insoles. Using the common technology of protrusion 3D printing, soft materials have proven challenging. 3D printing, which's capabilities and implementations have exploded in the last few years, shows a promising but uncertain future.

10 Summary and conclusion

This chapter reviews the work conducted throughout the project. It also discusses the key results and findings, and propose suggestions for future work.

10.1 Thesis summary

Chapter 1: The chapter gave an introduction to limb associated pain and custom insoles as a treatment. The primary aim of this thesis was to:

"Develop, implement and evaluate a system for insole modeling using low cost 3D scanners, foot mesh modeling and computer assisted insole design"

Chapter 2: The chapter discussed the current market situation and relative work of other research related to insole modeling and fabrication.
Chapter 3: The chapter discussed some theory used in the project for creating surfaces from point clouds, registration of 3D data sets and approximating scattered data to a rectangular mesh grid.
Chapter 4: The chapter introduced the Kinect sensor with imaging techniques, limitations and associated software.
Chapter 5: The chapter evaluated the Kinect with regard to distortions, resolution, accuracy, and RGB-D alignment.
Chapter 6: The chapter discussed the proposed system for insole modeling. Scanning of the foot with three Kinect sensors was used for foot modeling. Design of the insole model was done through implemented software.
Chapter 7: The chapter discussed remarks of the proposed system from Chapter 6.
Chapter 8: The chapter evaluated the proposed system by conducted a case study. A quantitative comparison was made using existing products available on the market.
Chapter 9: The chapter described insole fabrication testing, using a CNC cutter and 3D printer.

10.2 Key results

The Kinect evaluation have in summary concluded that the sensor is sufficient for the task of foot scanning with appropriate data processing. Using this device, available at a low cost, accurate foot modeling using multi-view captures was done.

Furthermore the design interface enabled simple design of a basic insole from a foot model, with the use of tools for orientation and modification.

The case study provided a quantitative comparison between the proposed system and existing methods which indicated that the proposed system could achieve similar results as existing methods on the market today.

The proposed system, being low priced, provided a comparable foot modeling system compared to other systems on the market.

10.3 Limitations

Using the factory calibration for the RGB-D alignment, even though it was manually verified and corrected, degraded the results. If a proper stereo calibration would have been made between the Kinect RGB and IR camera, it could have improved the results. This would also solve the limitation of whole mm values delivered from libfreenect's alignment.

Regarding the case study, the project were not able to compare the proposed system to a fully computer assisted system.

Not having a higher precision scanner regarding the comparison in Chapter 8 degraded the results as the scan regarding the Ortolab mold was very coarse. Also, any artifacts produced by the Kinect affected the existing company models.

10.4 Future work

Considering the system proposed, there are improvements to be made as well as new suggestions. There are other low cost 3D sensors today that has a minimum scanning distance of about 350–400 mm such as the Kinect for Windows with SDK [39] and Primesense 1.09 sensor [52]. By scanning at nearer distances, higher resolution can be achieved.

An evaluation of the system with two feet at the same time should be performed. It might be necessary with a more extensive reference system. However, capturing two feet at once is something that is not available on the market and could provide correlated information between the feet and increase the efficiency of patient scanning.

Interfering laser emitters were a problem with multiple Kinect captures. Using software to control the emitters or other solutions, is crucial if true simultaneous data capture is to be made, which would yield the best results with foot capture in a sub second interval.

The fabrication process today is likely to be done by a CNC cutter. In the future however, 3D printing could be an alternative. Given soft, functional 3D printing material, it would be possible to print the insole model with high accuracy.

10.4.1 Future design tools

The design interface described in Chapter 6.3 could be further developed. Extending the design interface with additional tools, required by an orthopedist, the insole design could be greatly improved. Personal communication with FotAnatomi [10] has suggested some future improvements to the developed insole design software:

- A more intuitive tool for pelott design; While the mouse and neighborhood function provides functional local modification, it is at this moment somewhat non-intuitive. Controlling the modification with reliable accuracy is crucial for proper and efficient design. Attempts should be made with e.g touch screen modification, an improved mouse interface or something radically new.
- The ability to mark areas of interest while scanning, giving the orthopedist the ability to mark a location for pelott or a location where other compensation is needed. These markings should then be visible in the computer model of the foot.

References

[1] Dreinhöfer, K.E., Reichel, H., Käfer, W. (2007) Lower limb pain. *Best Practice & Research*, vol. 21, no 1, pp. 135-152.

[2] Dunn, J.E., Link, C.L., Felson, D.T., Grincoli, M.G., Keysor, J.J., McKinlay, J.B. (2004) Prevalence of Foot and Ankle Conditions in a Multiethnic Community Sample of Older Adults. *American Journal of Epidemiology*, vol. 159, issue 5, pp. 491-498.

[3] Hill, C.L., Gill, T., Menz, H.B., Taylor, A.W. (2008) Prevalence and correlates of foot pain in a population-based study: the North West Adelaide Health Study. *Journal of Foot and Ankle Research*, vol. 1, pp. 2.

[4] Landorf, K.B., Keenan, A.M., Herbert, R.D. (2006) Effectiveness of Foot Orthoses to Treat Plantar Fasciitis. *American Medical Association*, vol. 166, no 2006, pp. 1305-1310.

[5] International Standards Organization Document ISO 8549-1:1989.

[6] Kogler, G.F., Solomonidis, S.E., Paul, J.P. (1995) In vitro method for quantifying the effectiveness of the longitudinal arch support mechanism of a foot orthosis. *Clinical Biomechanics*, vol. 10, no 5, pp. 245-252, ISSN 0268-0033, 10.1016/0268-0033(95)99802-9.

[7] Kato, H., Takada, T., Kawamura, T., Hotta, N., Torii, S. (1996) The reduction and redistribution of plantar pressures using foot orthoses in diabetic patients. *Diabetes Research and Clinical Practice*, vol. 31, no 1-3, pp. 115-118, ISSN 0168-8227, 10.1016/0168-8227(96)01214-4.

[8] Kogler, GF., et al. (1995) Biomechanics of longitudinal arch support mechanisms in foot orthoses and their effect on plantar aponeurisis strain. *Clinical Biomechanics*, vol. 11, no 5, pp. 243-252.

[9] Stell, J.F., Buckley, J.G. Controlling excessive pronation: a comparison of casted and non-casted orthoses (1998), *The Foot*, vol. 8, no 4, pp. 210-214, ISSN 0958-2592, 10.1016/S0958-2592(98)90031-1.

[10] Silva Oliviera, F.M.d., personal communication.

[11] Image: Pronation. *Orthopedia Wiki*. http://orthopedia.wikia.com/wiki/Pronation. (2013-05-09).

[12] Pratt, D.J., Sanner, W.H., (1996) Paediatric foot orthoses. *The Foot*, vol. 6, pp. 99-111.

[13] Huppin, L. (2009) Technology: Choosing a digital foot scanner. *Lower Extremity Review*. http://lowerextremityreview.com/article/technology-choosing-a-digital-foot-scanner. (2013-03-06).

[14] *Ortolab*. http://ortolab.se. (2013-03-05).

[15] Personligt utformade fotbäddar. *Lifecenter*. http://www.lifecenter.nu/forforetag/personligt-utformade-fotbaddar. (2013-05-09).

[16] VeriScan, Podiatric Scanner. (2008) *EnvisicVPS*. http://www.envisicvps.com. (2013-03-06).

[17] Jörgen Wiklander (VD, Ortolab AB) interviewed by the authors the 26 February 2013.

[18] Pröckl, E. (2006) På god fot med lasern. *NyTeknik*. http://www.nyteknik.se/nyheter/bioteknik_lakemedel/medicin_teknik/article40220.ece. (2013-05-09).

[19] Pratt, D. J. (1995) Functional foot orthoses. *The Foot*, vol. 5, pp. 101-110.

[20] *Delcam*. http://delcam.com/. (2013-05-28).

[21] Custom Orthotic Insoles. *DelCam Healthcare Solutions*. http://www.orthotics-cadcam.com/orthotics-solution/index.asp. (2013-03-04).

[22] Anders Brask (Sales, CAD/CAM Protech) interviewed by the authors 2013-02-15.

[23] iQube. *Custom Orthotic Insoles*. http://lz.orthotics-cadcam.com/en/iqube.html. (2013-03-04).

[24] Design with OrthoModel. *Custom Orthotic Insoles*. http://lz.orthotics-cadcam.com/en/orthomodel.html. (2013-03-04).

[25] Make with OrthoMill. *Custom Orthotic Insoles*. http://lz.orthotics-cadcam.com/en/orthomill.html. (2013-03-0).

[26] Cast Medical, Health for life. *Cast Medical*. http://www.castmedical.se/Hem/tabid/71/Default.aspx. (2013-03-04).

[27] Sathish K.P., Sudesh, S., Lazar, M. (2012) Customized Foot Orthosis Development by 3D Reconstruction of the CT Images. *Scientific Research*, no 4, pp. 692-695.

[28] Mavroidis, C., Ranky, R.G, Sivak, M.L., et al. (2011) Patient specific ankle-foot orthoses using rapid prototyping. *Journal of NeuroEngineering and Rehabilitation*, vol 8, no 1.

[29] Campbell R.J., Flynn P.J. (2001) A survey of free-form object representation and recognition techniques- *Computer Vision and Image Understanding*, vol 81, pp 166-210.

[30] de Berg, M., Cheong, O., van Kreveld, M., Overmars, M. (2008) Delaunay Triangulations | *Computational Geometry* , pp 191-218. Berlin:Springer Berlin Heidelberg.

[31] Image: Delaunay triangulation. *Wikipeida*. http://en.wikipedia.org/wiki/Delaunay_triangulation (2013-06-15)

[32] Horn, B. K. P. (1986) Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, vol. 4, pp. 629.

[33] Mukundan, R. (2012) Quaternions | *Advanced Methods in Computer Graphics* pp:77 -112. Springer.

[34] Lepetit, V., Fua, P. (2005) Mathematical Tools. | *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Now Publishers Inc, 2005. pp. 12. (Foundations and Trends in Computer Graphics and Vision Series).

[35] Khoshelham, K., Elberink, S.O. (2012) *Accuracy and resolution of Kinect depth data for indoor mapping applications*, Sensors (2012).

[36] OpenKinect. (2012) *OpenKinect*. http://openkinect.org. (2013-03-06).

[37] OpenNI, The standard framework for 3D sensing. *OpenNI*. http://openni.org. (2013-03-05).

[38] Image: Kinect. Wikipedia. http://en.wikipedia.org/wiki/Kinect. (2013-05-13).

[39] Kinect for Windows Sensor Components and Specifications. *Microsoft Developer Network*. http://msdn.microsoft.com/en-us/library/jj131033.aspx. (2013-03-06).

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

[41] Nicholas Burrus (2011), Kinect Calibration, http://burrus.name/index.php/Research/KinectCalibration. (2013-05-14).

[42] Bouguet, JY. (2010) Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html. (2013-04-18).

[43] *MathWorks*. http://www.mathworks.se/. (2013-06-16)

[44] Kinect to Matlab on OSX and linux via mex, Adam berg, http://acberg.com/kinect/. (2013-05-13).

[45] Herrera, D.C., Kannala, J., Heikkila, J. (2012) Joint Depth and Color Camera Calibration with Distortion Correction. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 34, nr 10, pp. 2058-2064.

[46] Smisek, J., Jancosek, M., Pajdla, T. (2011) "3D with Kinect". *Proc. IEEE Int'l Conf. Computer Vision Workshops*.

[47] Cailliau, R. Lego dimensions, the measurements. *RobertCailliau*. http://www.robertcailliau.eu/Lego/Dimensions/zMeasurements-en.xhtml. (2013-04-03).

[48] Wengert, C., Bianchi, G. (2010) Absolute Orientation. *Matlab Central, File Exchange*. http://www.mathworks.com/matlabcentral/fileexchange/22422-absolute-orientation. (2013-02-02).

[49] Stroud, I, Xirouchakis, P.C. (2000) STL and extensions | *Advances in Engineering Software* Volume 31, Issue 2, February 2000, Pages 83–95. (2013-05-30).

[50] 3D cutter, Portal type, built by Håkan Thorén.

[51] 3D printer: Makerboot 2. https://store.makerbot.com/replicator2.html. (2013-05-14).

[52] Primesense. http://www.primesense.com/. (2013-05-17).

[53] Heikkila, J., Silven, O., (1997) A four-step camera calibration procedure with implicit image correction. | *Computer Vision and Pattern Recognition;* 17-19 Jun 1997, San Juan, pp. 1106 - 1112.

A Absolute orientation using closed-form quaternions

The following section is a recreation of Horn (1986) [32].

Let r_A and r_B be two observations of one object as seen in Figure A.1.

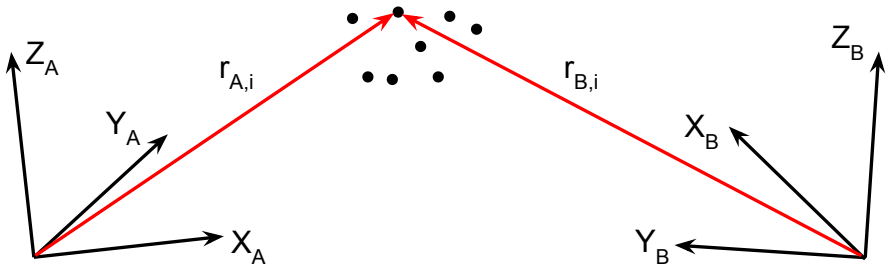


Figure A.1: Two views of the same object from different origins. Let $r_{A,i}$ be the vector to point i in system A and $r_{B,i}$ be the vector to point i in system B.

Finding the transformation that minimizes the least square error between the n observations can be written as

$$e = \sum_{i=1}^{n} ||r_{B,i} - sR(r_{A,i}) - t||^2 \tag{A.1}$$

where R, s and t are the transformation parameters rotation, scale and translation respectively. Determination of these parameters is shown below.

Translation Shifting r_A and r_B to their respective centroids through translation reduces equation (A.1) to

$$e = \sum_{i=1}^{n} ||r'_{B,i} - sR(r'_{A,i}) - t'||^2 \tag{A.2}$$

where

$$t' = t - \bar{r}_B + sR(\bar{r}_A) \tag{A.3}$$

and where $r'_A = r_A - \bar{r}_A$, $r'_B = r_B - \bar{r}_B$ and where $\bar{r}_A$, $\bar{r}_B$ are the centroids of r_A and r_B respectively. By expanding equation (A.2) it is clear that minimization with regard to t occurs when $t' = 0$ or

$$t = \bar{r}_B - sR(\bar{r}_A) \tag{A.4}$$

determining the optimal translation as the vector between r_B and the scaled and rotated r_A.

Scale Expanding equation (A.2) and using the orthonormality of matrix R the expression can be rewritten as

$$S_B - 2sD + s^2 S_A \tag{A.5}$$

where S_a and S_b are the sums of squares of the respective sets and D is the sum of the dot product between set r_B and the rotated set r_A. Completing the square in s and writing the expression for the error, it can be found that the scale factor that minimizes the error in equation (A.5) is determined by

$$s = \frac{\sum_{i=1}^{n} r'_{B,i} R(r'_{A,i})}{\sum_{i=1}^{n} ||r'_{A,i}||^2} \tag{A.6}$$

Rotation Translation and scale are determined by the rotation. It can be found that further minimization of equation (A.5) with regard to R is achieved by maximizing

$$D = \sum_{i=1}^{n} r'_{B,i} R(r'_{A,i}) \tag{A.7}$$

Equation (A.7) can be written with quaternion notation as

$$D = \sum_{i=1}^{n} (\mathring{q} \mathring{r}'_{A,i}) \cdot (\mathring{r}'_{B,i} \mathring{q}) \tag{A.8}$$

where $\mathring{q}$ is a unit quaternion representing rotation and $\mathring{r}'_{A,i}$, $\mathring{r}'_{B,i}$ are quaternion representations of the sets $r'_{A,i}$, $r'_{B,i}$ respectively.

Equation (A.8) can then be written as

$$D = \mathring{q}^T \left(\sum_{i=1}^{n} N_i \right) \mathring{q} \tag{A.9}$$

It is shown in Horn (1986) [32] that the quaternion $\mathring{q}$ maximizing

$$\mathring{q}^T N \mathring{q} \tag{A.10}$$

is the the eigenvector corresponding to the largest eigenvalue of N.

The final rotation is then found by retrieving the lower right 3×3 sub matrix of the 4×4 rotation matrix

$$\mathring{q}^T \mathring{q} \tag{A.11}$$

B Kinect RGB camera model and calibration

The following camera model description is a recreation from Bouquet (2010) [42], which in turn is based on the report by Heikkilä (1997) [53].

Given a point P in space with coordinates (X_c, Y_c, Z_c), let the normalized image projection x_n equal

$$x_n = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{B.1}$$

Let

$$r^2 = x^2 + y^2 \tag{B.2}$$

By including lens distortion δ, the normalized point coordinate x_d is defined by

$$x_d = \begin{bmatrix} x_d(1) \\ x_d(2) \end{bmatrix} = (1 + \delta(1)r^2 + \delta(2)r^4 + \delta(5)r^6)x_n + d_x \tag{B.3}$$

where d_x is the tangential distortion vector, defined as

$$d_x = \begin{bmatrix} 2\delta(3)xy + \delta(4)(r^2 + 2x^2) \\ \delta(3)(r^2 + 2y^2) + 2\delta(4)xy \end{bmatrix} \tag{B.4}$$

where f_c is the focal length, α is the skew coefficient and cp is the principal point offset. The complete camera model is therefore

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_c(1) & \alpha f_c(1) & cp(1) \\ 0 & f_c(2) & cp(2) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d(1) \\ x_d(2) \\ 1 \end{bmatrix} \tag{B.5}$$

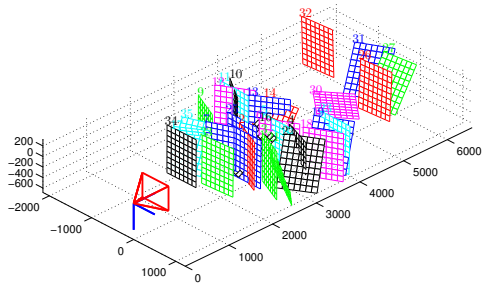
where x_p, y_p is the projection of point P on the image plane.

C RGB camera calibration results

Below follows the complete results of the RGB camera calibration of two Kinects. The calibration was done using the toolbox supplied by Bouguet (2010) [42], which in turn is based on the report by Heikkilä (1997) [53]. The main functions used from the toolbox is **calib_gui** and **visualize_distortions**.

Figure C.2 shows the orientation of the different checkerboard images that was acquired.

Orientation of reference set Kinect 1



Orientation of reference set Kinect 2

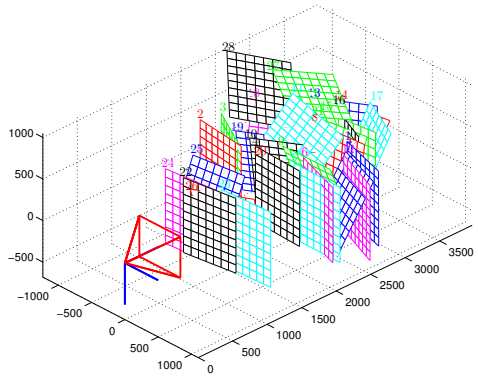


Figure C.2: Captured sets used for the calibration for Kinect 1 and 2. All distances are in mm.

Figure C.3 shows the radial and tangential distortion components for both Kinect 1 and 2.

A complete distortion model for the two Kinect's can be seen in Figure C.4.

The reprojection error is shown in Figure C.5.

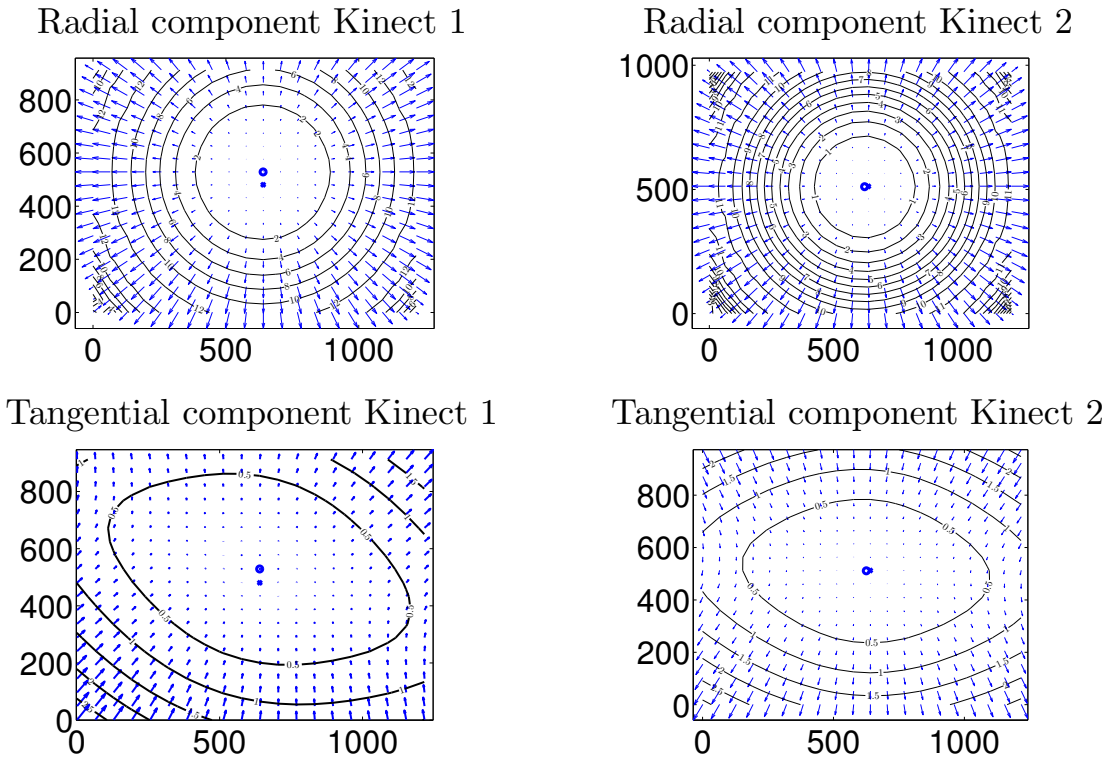


Figure C.3: Radial and tangential distortion components for Kinect 1 and 2.

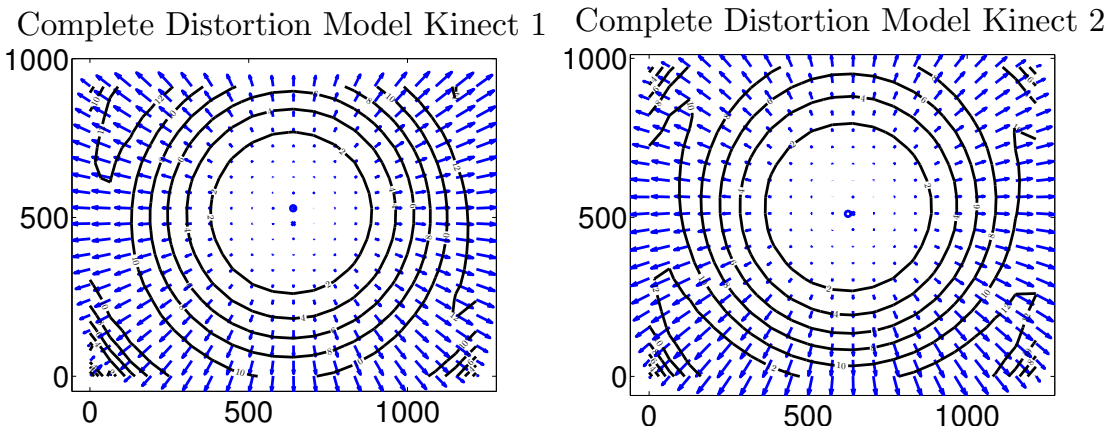


Figure C.4: Complete distortion model for Kinect 1 and 2, with both radial and tangential distortion components.

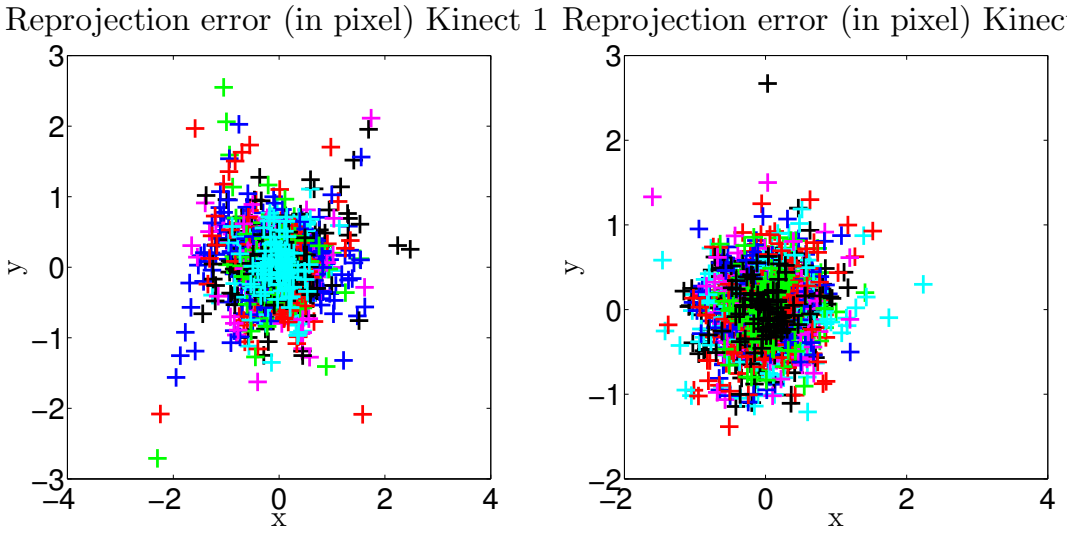


Figure C.5: Reprojection errors of the calibration in pixels for Kinect 1 and 2.