Development of a patient position and tracking system for the head and neck hyperthermia applicator

E. Neuteboom
The Hague University - Electrical Engineering

Februay - June 2012

THE HAGUE UNIVERSITY - ELECTRICAL ENGINEERING

FINAL YEAR PROJECT

Development of a patient position and tracking system for the head and neck hyperthermia applicator

Author:
E. NEUTEBOOM

Company Mentor:
ir. W.C.M Numan
School Mentor:
ir. ing. J.Z.M. Broeders

February - June 2012

Abstract

The recently developed HyperCollar applicator was specifically designed to heat tumors in the head and neck region. The applicator uses twelve antennas to create an EM interference pattern, which applies heat to the target area. Patients' position and movement needs to be tightly monitored and controlled during treatment for optimal result. The purpose of the project is the development of a passive tracking system, which uses a camera to track the infrared light from a number of light sources positioned on the patient.

Several measurements were performed to find the accuracy and precision of the system. A frame holding the three infrared LEDs was moved along separate axes while being tracked. The error of the system is obtained by subtracting the fixed known displacement of the three dimensional frame from the tracked data. The measurement error increases as the object is placed further away from the centre.

A linear calibration method was used to correct measured data for the error. After calibration, the range in which the error stays below 5 mm is extended from ±30 mm to ±50 mm for the centre.

A measurement is performed in a clinical setting to get an initial idea of how much a person moves his head during a treatment. The results have shown that a healthy person moves his head a maximum of 5 mm and 3° from the centre position.

The results have shown that a infrared tracking system can be applied to track the movement of a patient after calibration. Subsequent development and implementation of a clinical setup will have to show how much a patient moves his head during an actual Hyperthermia treatment.

Preface

This internship is commisioned by the Erasmus MC - Daniel den Hoed clinic, performed for the gratuation of the education Electrical engineering at The Hague University.

I have enjoyed working at the unit Hyperthermia with a number of great people. Not only has this been a inspiring atmosphere for gaining experience in working in research, but also has it been a great time during the coffee break. Several interesting and sometimes flaming discussions have been held about belief and society. I want to thank Wouter Numan for helping me with my work, introducing new ideas and finding free time to correct my report. I also want to thank Wouter and Roel for being great roommates.

I want to thank my school mentor Harry Broeders for giving a new perspective to the work I have done and coaching me during the project.

Contents

List of Figures

List of Code

Chapter 1 Project

1.1 Discription

This internship was commissioned by the Radiotherapy department, Hyperthermia[1] unit of the Erasmus MC, Daniel den Hoed Cancer Centre.

Hyperthermia is a form of cancer treatment during which tumor temperatures are raised to the range of 40°C-45°C. Within the Hyperthermia Unit of the Daniel den Hoed Cancer Center, around 160 new patients are treated each year to improve the clinical outcome from radio- or chemotherapy.

1.2 Problem statement

The recently developed HyperCollar (HC) applicator[2], was specifically designed to heat a tumor in the head and neck region. The HyperCollar has twelve antennas operating at 434 MHz and uses EM interference for focusing EM energy to heat the tumor region. Interference patterns are controlled through adaption of the antannas' phase and amplitude. Phase and amplitude settings are obtained before treatment through a process called Hyperthermia treatment planning (HTP). Recent research[3] has shown that a movement of >10 mm of the head in the left or right direction lowers the efficiency of the power transfer, which reduces the quality of treatment. This requires that the patients position is tightly monitored and controlled during treatment. Although specific positioning measures have been installed, patient position is currently measured by hand at the start of the treatment. It is our goal to monitor patient position continuously during treatment and eventually use this data to adapt the planning in real-time to the changing situation. High cost (10.000 euro) laser tracking applications exist. The purpose is to improve treatment quality in addition to allowing the patient some freedom to move, thus improving comfort. However, it is currently not known how much a patient moves during a treatment. Developing a low cost application can provide some insight into the problem and whether the use of a high accuracy system is required.

1.2.1 IR tracking system

The purpose of the project is the development of a low cost system to track a patients' position and movement during a HT treatment. A low cost solution for this is infrared-tracking, which uses a camera to track infrared LEDs, at a fixed position on the patients' head. For clinical use, it is not preferred that cables will be attached to the patients' head, thus reflectors will be used to reflect the infrared light. The movement of a patient can be tracked in six different axes: the rotational axes; yaw, pitch and roll and the translational axes; X, Y and Z, which together are called the six degrees of freedom (6DOF). The non-commercial, open-source Delphi application FreeTrack[4] can calculate the relative position and movement of the head,using as input, three dots of infrared light recorded by a webcam. The algorithm used to calculate the 6DOF, is based on the Alter's algorithm[5]. FreeTrack requires isolation of the three infrared sources. It is therefore necessary to remove the infrared filter from the webcam and install a visual light filer instead. Thus only direct IR sources or special IR reflectors will be visable.

1.2.2 Terminology

Object tracking can be divided into two aspects: **position** and **movement**. The **position** of the object is the distance between the center point of the object and the centre point of the camera. In FreeTrack, it is possible to clarify the **position** of the object as the starting (centre) position of the object, with all the parameters set to zero. From this **position**, all **movement** is measured. The **movement** of the object is the distance or angle between the starting position and the changed **position** of the object.

When **position** and **movement** are measured, these measurements have an **accuracy** and a **precision**. The **accuracy** of the measurements indicates how close the measured value is to the true value. The **precision** of the measurements indicates how close several measured values are to each other for the same movement. As long as the **precision** is good enough, the **accuracy** can be compensated through calibration.

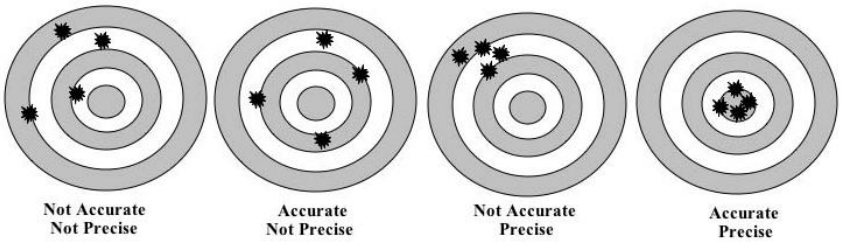


Figure 1.1: Accuracy versus precision[6]

Position and movement of a person are measured in the 6DOF. In the HyperCollar, the directions of the axes are: (See figure1.2).

- X: left(+)/right(-)
- Y: backward(+)/forward(-)
- Z: up(+)/down(-)
- Pitch: Rotate around X axis
- Roll: Rotate around Y axis
- Yaw: Rotate around Z axis

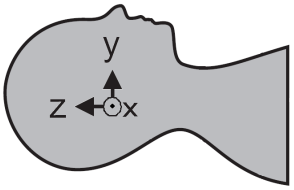


Figure 1.2: Axes used in the HyperCollar

1.2.3 Requirements

The purpose of the project, as discussed above, is to develop a low cost tracking system. This limits the budget for the project to about 100-200 euro.

According to Lagendijk (1998)[7], a patient must be positioned with an accuracy of 2 cm for regional HT quality assurance. Assuming the patient is at the right position, the applicator can heat up a target with an accuracy of about 2-5 mm[8]. Recent research[3] shows that, with a fault of <10 mm (x) or <20 mm (y,z), the reduction of the SAR distribution is too high for quality assurance in H&N HT treatment. In conclusion, to assure treatment quality, a patients position and movement must remain within 5 mm of the planning position. The accuracy for rotation is aquired by moving the point farthest away from the origin of the axes. For instance, the distance between the origin and the end of the nose is 10 cm. When turning the nose 5 mm left, the rotation around the Z axis is: $\tan(5\text{ mm}/100\text{ mm}) \approx 3°$.

When the system will be applied in the clinic, as few material as possible must be attached to the patients' head. This means that no hat or cap can be used and no wires must lead to the head, so no LEDs

can be attached on the head. A solution to this is to use reflective material and a infra red floodlight to light the reflectors.

Knowledge of the patients position offers the prospect of direct adaption of the treatment plan to the changing situation. For this application, the data has to be available in real time. Assuming a person might move 1 mm/ms (put in perspective, 1 mm/ms = 1 m/s = 3.6 km/h), a frame rate of 1/5 ms = 200 fps is aquired to obtain the 5 mm accuracy.

Measurement accuracy requires the position of the camera to remain constant, so the hardware must robust enough to remain fixed in position during the whole treatment time, which is 75 minutes.

Lastly, due to compatibility requirements, data processing and visualization has to be done in Matlab.

1.2.4 Outline of the thesis

Chapter 2 describes measurements performed with the use of a laboratory prototype.

Chapter 3 shows the first clinical implementation of the system. It provides the results of the measurement done in a treatment like setting.

Chapters 4 and 5 provide a conclusion and future perspectives.

Chapter 2 Prototype

With the use of FreeTrack, an infrared tracking system is built to test if the requirements can be made in the laboratory. If the requirements can me made, then the system can be applied in the clinic for a measurement during a treatment like setup.

The system will be tested primarily on accuracy and precision. Constant errors in accuracy can be redused through calibration.

Secondary test objectives are:

- Optimization of FreeTrack settings.
- Usage of different cameras
- Usage of reflectors

2.1 Setup

The setup of the system consists of a webcam, three small infrared light sources with power supply, mounted on the corners of a triangular frame and the FreeTrack software. This system is supplemented with software writen in Delphi and MATLAB to log and process measured data. Figure 2.1 shows the schematical setup of the system.

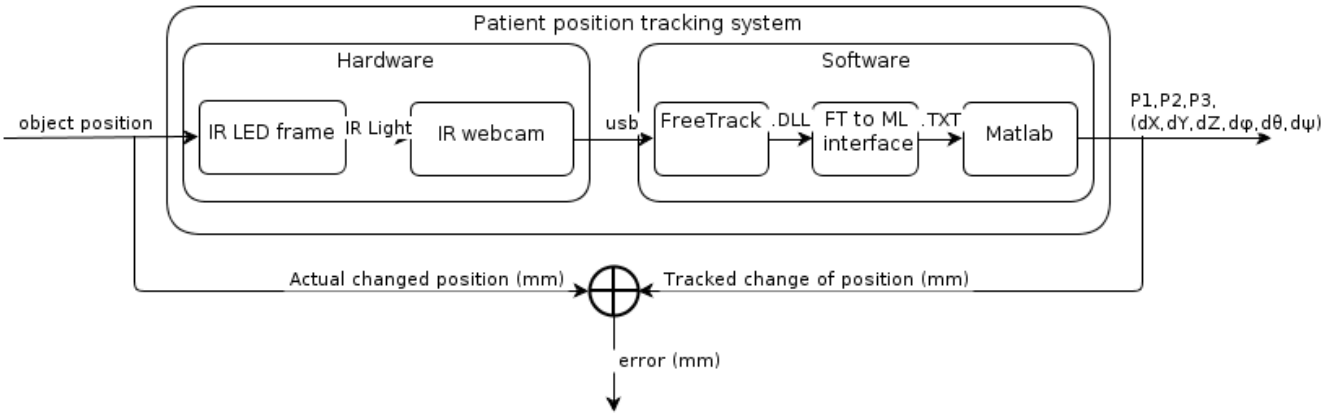


Figure 2.1: A schematic overview of the setup of the system.

The FreeTrack software is the most important part of the system. It reads and interprets the image, made by the webcam. It uses the Alter algorithm to calculate the movement in the 6DOF from three corresponding ponts in the 2D image.

Camera

The Microsoft LifeCam VX-800[9] is chosen as the first webcam to use because of its low price and the ease of removing its infrared filter. To filter visual light, a piece of a floppy disk is taped in front of the camera. In FreeTrack, a treshold can be set to filter unwanted background infrared light.

Infrared LED frame

FreeTrack alows 4 different models for the LEDs: single point, three point clip, three point cap and four point cap. The only suitable model for this application is the three point cap model, because of the position of the HyperCollar around the head.For the accuracy and precision measurements, the LEDs are constrained on a LEGO frame. LEGO is used because it has very accurate dimensions[10], fits very tightly together and is easy to work with. Figure 2.2 shows the model designed in LEGO Digital Designer[11]. The white dots indicate the position of the LEDs. The model dimensions (distances between LEDs), which are required by FreeTrack, are measured directly from the model.

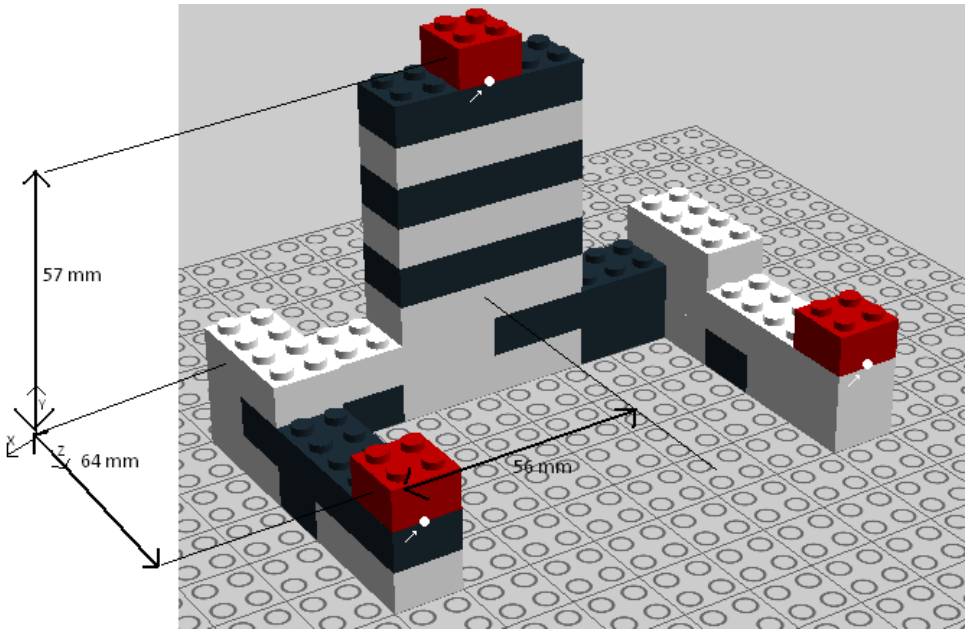


Figure 2.2: Designed LEGO model with its dimensions. The white dots indicate the position of the LEDs.

The model position can also be set. As FreeTrack states, this is the ”Distance from head pivot (center of head, just below ear) to model translation reference point R” and ”The model’s position relative to a center of rotation can be specified to minimise unwanted translation during rotation.” When the head is yawed, it results in a change in X and Y direction, when pitched, it results in a change in Y and Z direction and when rolled a change in X and Z direction. This would mean that, when starting to track the movement of a patient in the HyperCollar, the exact location of the center of rotation of the patients’ head must be known, which is not accurate when estimated. This would be an uncertain setting when this is changed. For the performed measurements, this setting is set to 0. To verify this settings influence on the results, the model position was set to different values, which did not change the error much. However, this has not been tested yet during a treatment.

Figure 2.3 shows the settings in FreeTrack.

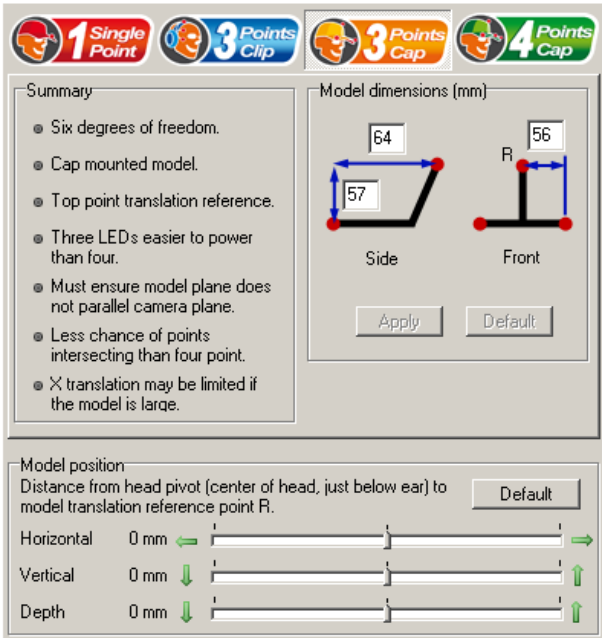


Figure 2.3: Model dimension settings in FreeTrack.

With FreeTrack, the camera settings can be set to improve the visibility of the three points. This is mostly set to default. Other settings are set as follows:

- Actual webcam FPS: 30
- point size

  min: 2 pixels

  max: 30 pixels
- calibration

  Sensor width: 3.0 mm

  Focal length: 6.5 mm

Interface

FreeTrack provides three different software development kits to access the data via a DLL file, which contains the function to return the FreeTrack data. The interfaces are based on the languages MATLAB, C and Delphi. The MATLAB code makes use of MEX (MATLAB Executable[12]) files. MEX files are a way to call custon C routines directly from MATLAB as if they were MATLAB built-in functions. In the MEX file, the C code reads the FreeTrack data from the DLL. Since data processing and visualization is done in MATLAB, this is a very usefull way to interface the data from FreeTrack to MATLAB. The C and Delphi code directly access the DLL to aquire the data. When setting up this interface, both the MEX files and the C interface gave problems with reading the data from the DLL. Delphi gave no problems with this and is therefor used. Altough Delphi is not used very often, it is easy to extend a form with several interace options such as textboxes and buttons.

Figure 2.4 shows the Delphi interface. Initially, the interface did not have any buttons or textboxes. It also showed information not relevant for the measurement. The interface has been extended with buttons to display data, save the data, select data at a certain moment and two textboxes to enter the save path. At the bottom, the COM port at which handles the communication with MATLAB. The virtual pose is the movement of the object relative to its center point. The raw pose is the movement relative to the camera, which is stored. Tranfer to MATLAB can either be done by logging the data to a file in MATLAB and reading it after the measurement or communicating it directly to a running MATLAB application. Communicating it directly has the advantage that visualization can be done during the measurement, but for further usage, the data has also be logged. For this setup, the data is only logged during the measurements and processed and visualized after the measurements are done. (If communication is wanted, this could be done with the use of the COM port, also used for the synchronization with MATLAB for the automated measurements in section 2.3.1). The data is saved continually in a text file with the format of a table, with values seperated by tabs. This way, the data can easily be read by MATLAB. Saving the data happens every 50 ms when the 'start measurement' and the 'save data' buttons are pressed. The 'Select data now' can be used to select a singe value at a specific moment. This is used for accuracy and precision measurements. When the object is placed at a fixed position, the data at that moment can be selected and will be saved in another text file. This way, it is not necessary to watch the whole movement of the object during the whole measurement and it is easy to compare only the important data.

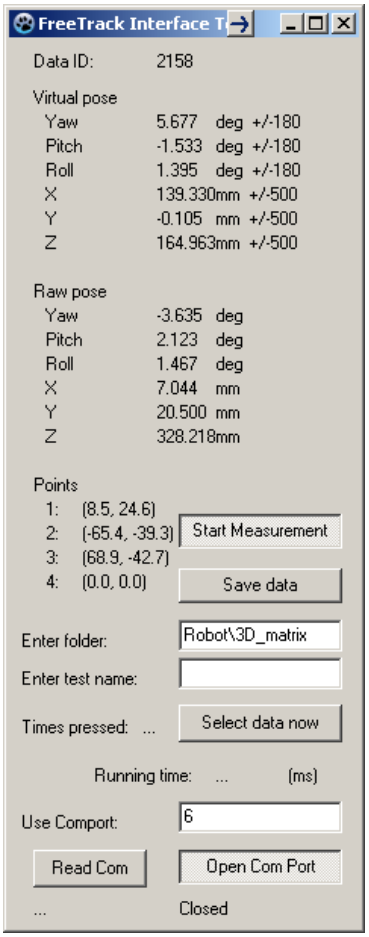


Figure 2.4: The FreeTrack interface.

MATLAB

Two different MATLAB scripts are used to read a specific text file and visualize the data. The data, writen continually, is read and directly plotted, with movement on the vertical axis and time on the horizontal axis. Figure 2.5 shows an example of a figure showing the movement of an object along the X axis.

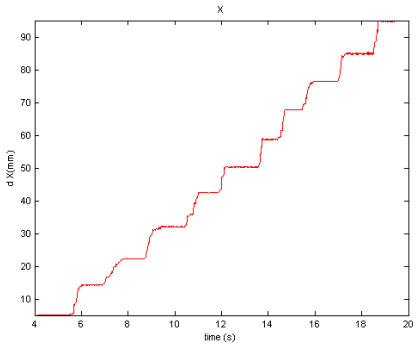


Figure 2.5: The movement of the object along the X axis.

When the selected data is read from the text file, the movement is first made relative to the position around which the object moves. Since the real movement of the object is fixed and known, the tracked data can be compared with the real movement, which results in an error. This script is used for the measurements performed in section 2.2.

2.2 Measurements

Several tests are done with a prototype system to aquire information about the accuracy and precision of the system. With these measurements, the relative translation and rotation of the model have been measured. For every axis seperately, the model was moved a specific distance or angle. The translations and rotations are tracked by FreeTrack. The results of the FreeTrack data is compared to the set translations and rotations to assess accuracy, or error.

To move the object, several different methods are used to compair results and to improve the accuracy of the displacement and measurement of displacement.

After movement recording is started in the FreeTrackClient, the model is moved a certain distance and then hold still in this position for about 4 seconds. Then the object is moved back to its initial position and every measurement is repeated a number of times.

2.2.1 Methods

Three different methods and tools were used to move and measure for the according axes. Different methods are used to compair results for varification. Also, the accuracy of the displacement and the measurement of displacement are improved. The first method only has an approximate displacement. The displacement is done without a fixed ruler. for the second method, axes are drawn on tape on the table and the object is fixed to a heavy elevator table. This way, the displacement is more accurate. for the third method, a LEGO base plate is fixed to the table and the object is displaced over the base plate. The displacement is measured with the known LEGO dimensions. The following methods and tools are used:

1) Approximate

This method is used to get an initial assessment of the accuracy and precision, the procedure for measurements and the complications which could appear during a measurement. At the beginning of a measurement, the object is positiond with the variable to be measured close to zero, except for the Y axis, since the zero position for the Y axis is in the camera. The Y position is kept in a range of 400-700 mm. The starting point for every axis is noted for future reference. Every measurement is performed three times for information about precision.

X/Y direction:

- A calliper is placed at the side or the back, for the X and Y axis respectively.
- The calliper is extended to 10 mm ±2 mm for every step, which moves the model in the direction of the axis.

Z direction:

- The object and the camera are placed on seperate elevator tables.
- A calliper is used to measure the initial height of the model.
- The elevator table with the model is lifted by turning the rotary.

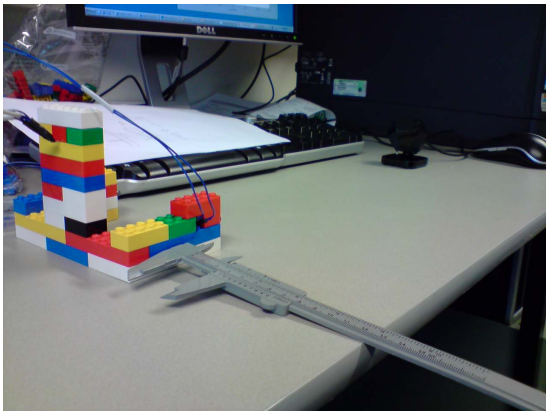
Yaw:

- One protractor is taped unto the table.
- Another protractor is taped under the model.
- The object is placed at -25° by aligning the 0° line of the above protractor aligned with the 25° line of the bottom protractor.

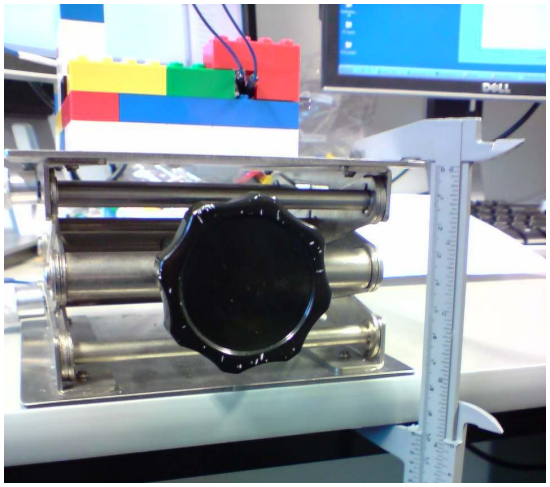
- Model is turned with steps of 5° until 25°.

Pitch/Roll:

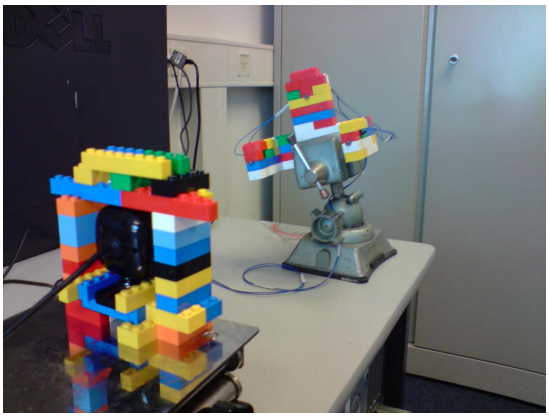
- The object is tightened in a small vice, which has a ball and socket joint. The camera is placed on an elevator table and lifted to about the same height as the object.
- The object is turned and the angle measured with a digital angle meter.



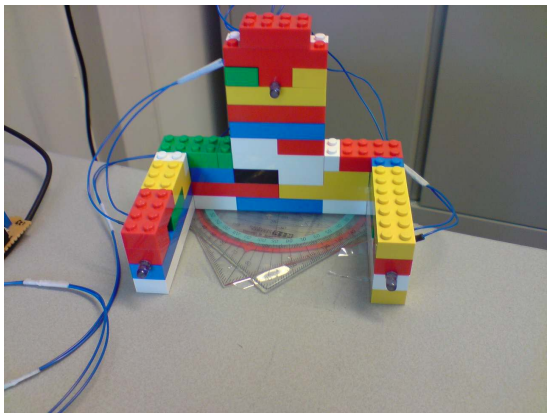
(a) X/Y axes



(b) Z axis



(c) Pitch/Roll



(d) Yaw

Figure 2.6: Methods for the different axes.

2) Taped

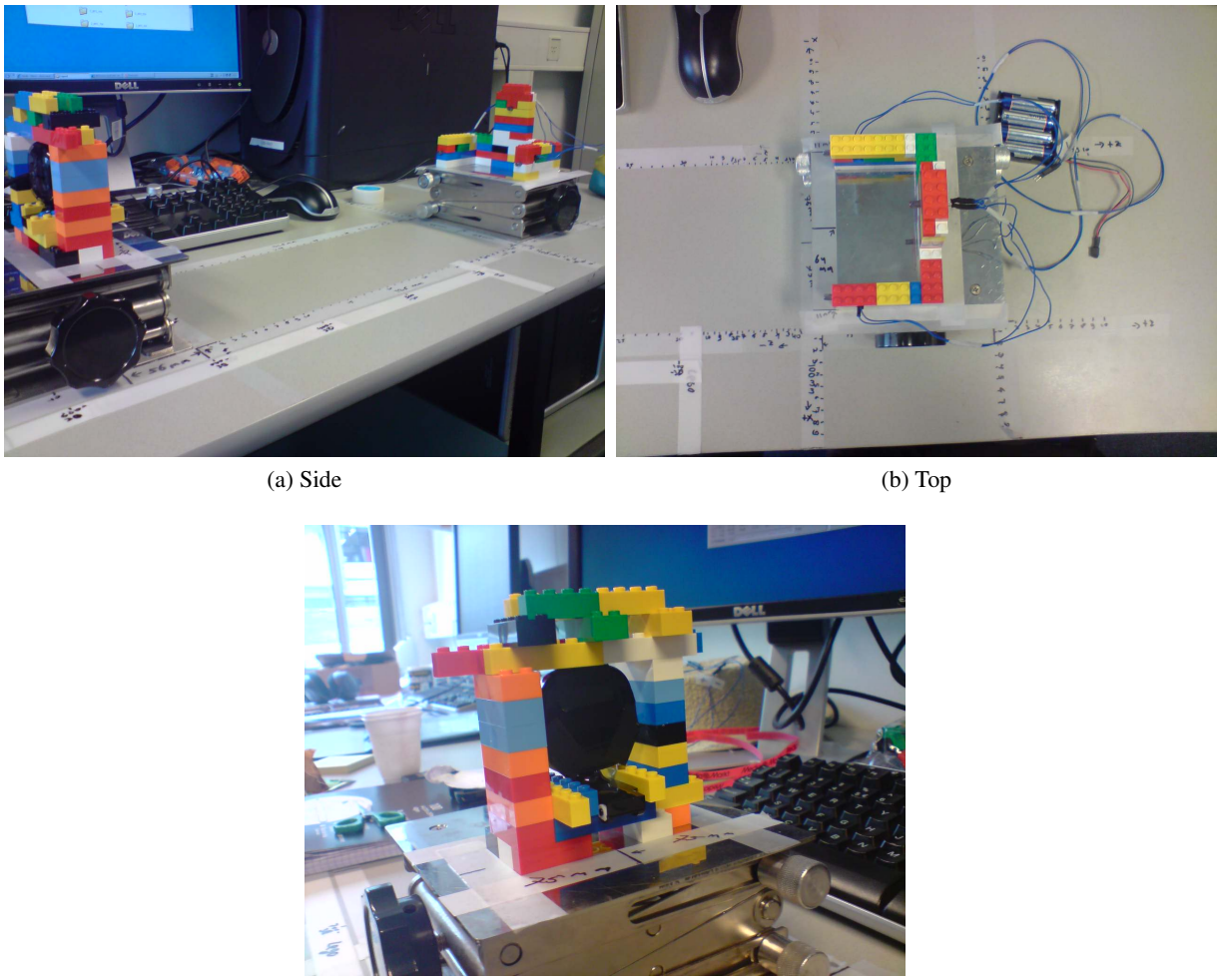
This method is only used for the three translational axes. Around the camera, a LEGO shell is built to keep the cameras orientation fixed. The camera and the object are placed on elevator tables. The initial position of the elevator table and the camera and the object on them, is marked by repositionable tape. On the tape, a division is made with steps of 10 mm. This way, the object and the camera are placed at the same position for every other measurement and the object is moved consistently. At the beginning of a meaurement, the elevator tables are placed at their initial position and the object and the camera on top of the tables. Same measurements are performed for different distances between the camera and the object.

X/Y direction:

- The elvetator table with the object on it, is moved from its initial position to a multiple of 10 mm.
- After waiting for about 4 seconds, the elevator table is placed back to its initial position.

Z direction:

- The same method as the first is used for the Z direction.



(a) Side

(b) Top

(c) Camera in LEGO shell

Figure 2.7: The surface is taped to have fixed initial positions and fixed distances.

3) LEGO base plate

For this method, the object is placed at a fixed position on a LEGO base plate. The base plate has a fixed position marked with tape on the table. Same measurements are performed with different distances between the camera and the object. At the beginning of every measurement, the object is placed at its largest negative position. After the object is moved, the currect data is selected by pressing the ”select data” button in the FreeTrackClient. Every measurement is performed three times.

X/Y direction:

- The object is moved with steps of 1 block from one side to the other of the base plate.
- The number of steps is counted and multiplied by 8 mm to get the distance moved.

Z direction:

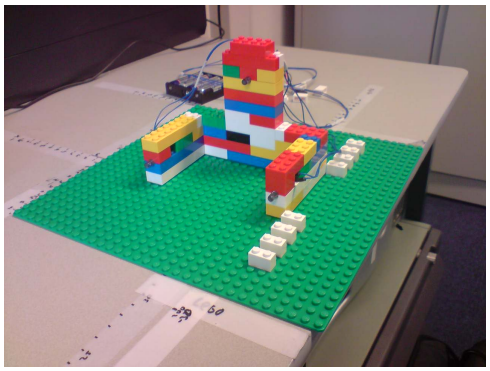
- The camera is placed 6 LEGO bocks higher in order to ensure that a negative value for Z can also be reached.
- The object is moved with steps of 1 block from its lowest to its highest position by placing a LEGO block under the four corners of the object.
- The number of LEGO blocks are counted and multiplied by 9.6 mm to get the distance moved.

Yaw:

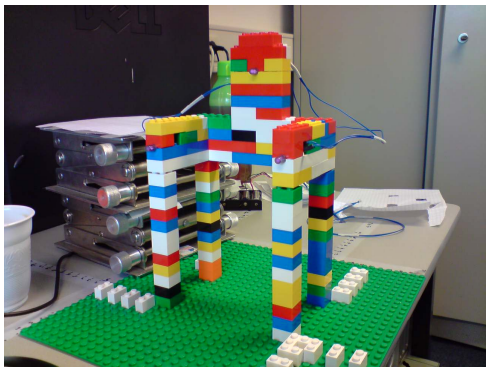
- The same method as the first is used for Yaw only for this method, the distance between the camera and the object differs.

Pitch/Roll:

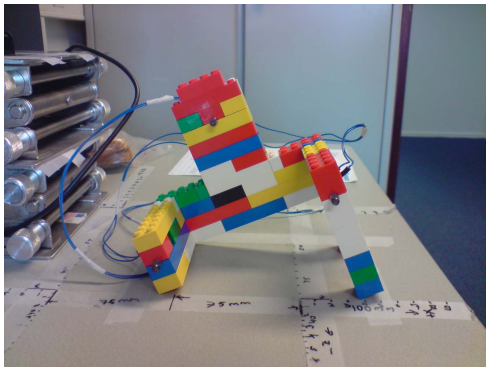
- The object is turned by placing LEGO blocks under one side of the object.
- The angle that is made, is calculated by taking the inverse tangent of the opposite right-angle side divided by the adjacent right-angle side.



(a) LEGO base plate



(b) Change in Z axis



(c) Change in Pitch

Figure 2.8: A LEGO base plate is used to move the object with fixed distances.

2.2.2 Results

A summary of the measurements with the LEGO base plate can be seen in figure 2.9. (The results of all the measurements can be seen in appendix 6.1). The figures (error plots) show on the horizontal axis the distance moved and on the vertical axis the absolute difference (error) between the real measured movement and the movement tracked by FreeTrack. The single markers indicate the measured values, the solid line the average value of three measurements and the dotted line the linear regression line of the average values. The color of the markers and the lines indicate the distance between the camera and the object.

The figures show a rising error for a larger distance moved. The figures also show that, for the lineair fit, the bigger the distance is, the smaller the error, except for the Y axis.

The Y and pitch plots show some inconsistencies. In the +Y direction, the deviation from the average curve has a maximum of 4 mm. For an angle >20° for pitch, the values have a deviation of a maximum of 6 mm, which is caused by interpreting the order of the image points wrong by FreeTrack. This limits the angle of pitch to 20°.

Table 2.1 shows the range in which the required accuracy can be obtained and the precision for a distance of approximately 500 mm between the camera and the object.

Table 2.1: Observations from figure 2.9

Axis	X (mm)	Y (mm)	Z (mm)	Yaw (°)	Pitch (°)	Roll (°)
Accuracy	-32 to 47	-42 to 48	-57 to 57	-25 to 25	-12 to 13	-24 to 24
Precision	1	4	1	1	2 [-20 30]	1

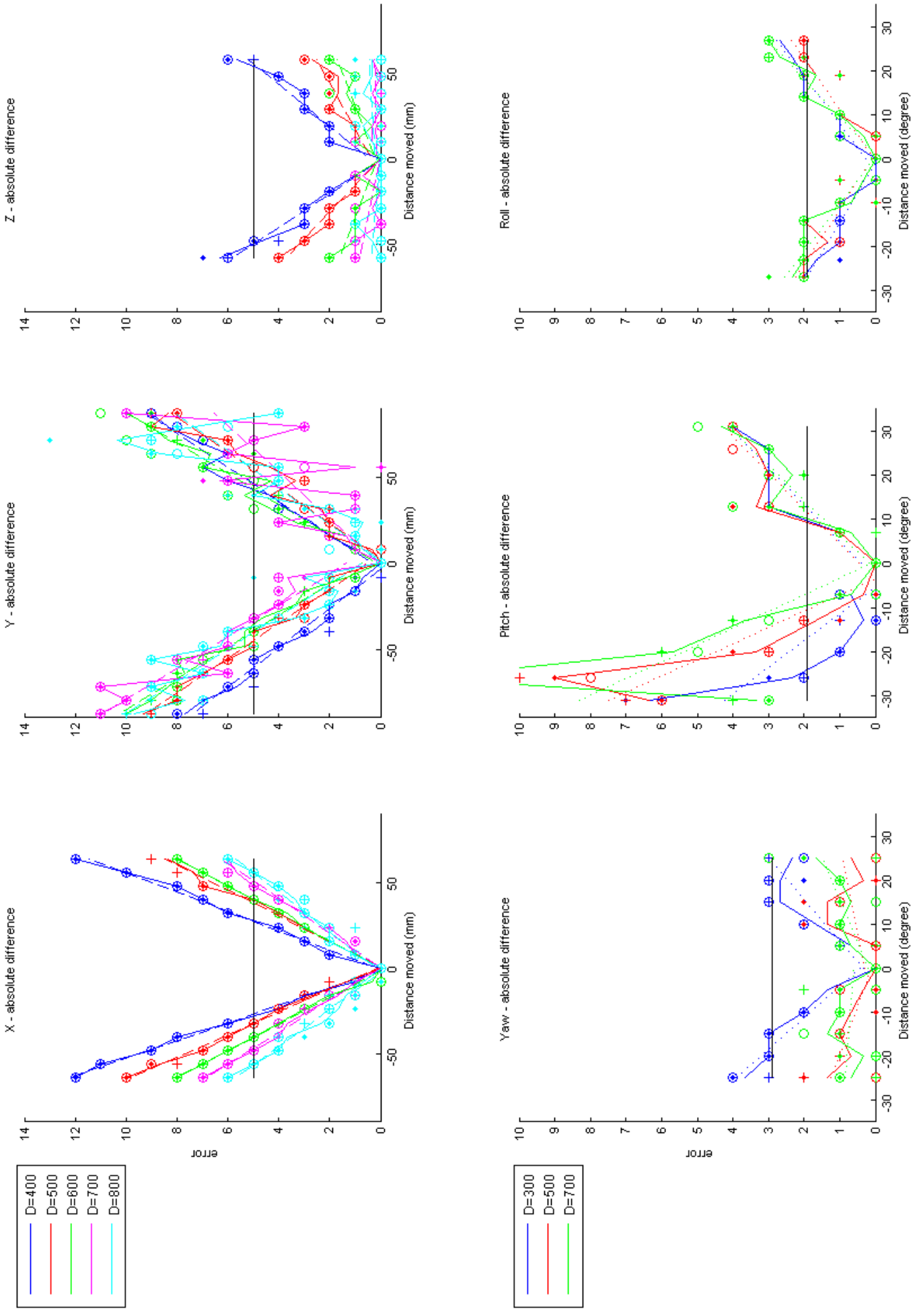


Figure 2.9: Difference between the real movement and the measured movement. The blue horizontal line indicates the required accuracy.

2.2.3 Conclusion

The measurements have shown that the required accuracy can be obtained using FreeTrack, as long as the movement of an object is limited to the values given in the previous section. The further the object is moved from its zero position, the larger the error becomes. The measurements have also shown that this limit varies for different distances between the camera and the object. A distance of 500 mm gives for all the axes the most reliable values and has the biggest range of measured values which stay below the required accuracy.

The linear fit, based on the average error of three measurements shows that the difference between the average measured value and the linear function stays below 1 mm and 1° for all the axes. When correcting the measured values with this linear function, an accuracy of 1 mm and 1° can be obtained, considering the different linear functions for the different distances between the camera and the object. When more and accurate measurements are done, the right calibration can be applied to the final system. With the use of a robot arm, the measurements can be automated and done in a large range and quantity so that statistical conclusions can be taken about the precision and accuracy of the system and a 3D calibration matrix can be obtained to compensate the accuracy.

2.3 Automated measurements

A better accuracy can be obtained by calibrating the error with reliable data. To obtain the right calibration values, a large number of measurements must be done. After calibrating a singular measurement, only the precision of the system influences how close the measurement is to the target. With the large number of measurements, the riliability of the system can be found.

To perform a large number of measurements, a cubical 3D scanner is available. The 3D scanner can move an attached object in the 3 different axes. By letting the robot move the LEGO object serveral patterns for multiple times while tracking with FreeTrack, the data for the calibration can be obtained.

First, error measurements are done to compare the results with the previous performed measurements. After that, a calibration matrix will be made along a cubicle grid which will be used to calibrate singular measurement points.

In the 3D scanner, only the three translational axes are measured since rotation is not possible. Calibration of the rotational axes will be discussed later.

2.3.1 Setup

As in the first measurements performed, the system contains the tracking software FreeTrack and the Delphi-to-MATLAB FreeTrack interface. As hardware, also the same LEGO object with LEDs attached and webcam are used. However, other than the first measurements, which were performed by hand, the measurements in the 3D scanner are automated, which requires the FreeTrack interface to be automatic also. Figure 2.10 shows the schematic overview of the setup in the 3D scanner.

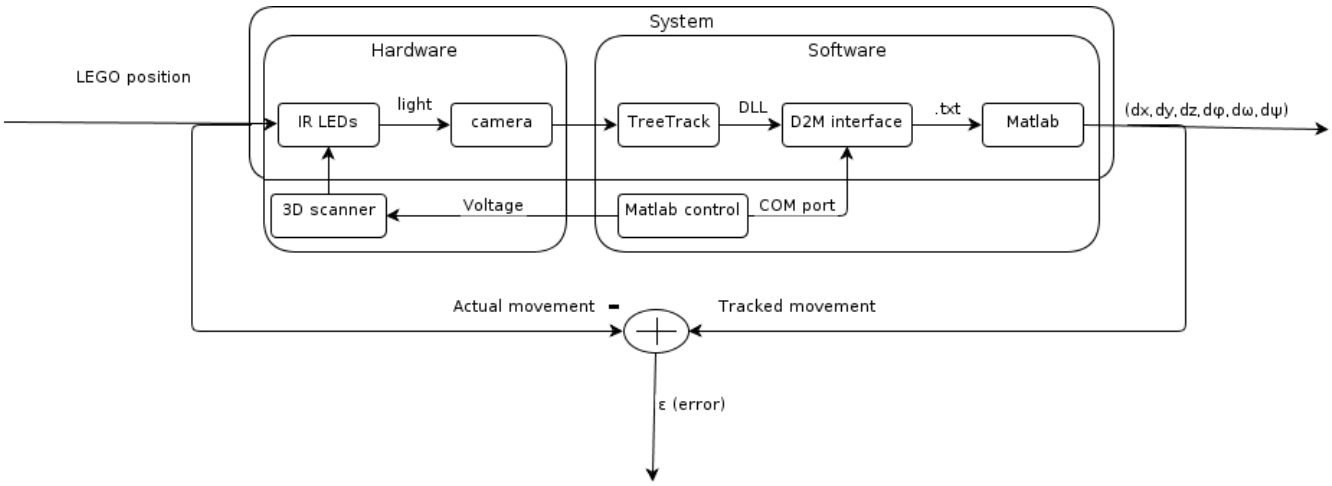


Figure 2.10: The setup in the 3D scanner.

Control of the 3D sanner

The 3D scanner can be controlled to a certain coordinate by setting an analog voltage between 0 and 10 volt on the three seperate channels for the three different axes. The range of the axes is 50 cm, which results in a resolution of 5 cm/V. When putting a values from 0 V to 10 V on A, B and C for the vector [A B C], the object moves as follows, as seen from the camera (with the according axis, used in the HyperCollar, between round brackets):

- A: right to left(X axis)
- B: front to back (Y axis)
- C: bottom to top (Z axis)

MATLAB is used to control the analog outputs, since there is experience within the team with this. If this should be done in Delphi, research has to be done on how to control the analog outputs of the PC. Three different patterns for the three translational axes are scipted in MATLAB. For every measurement, the distance D, between the object and the camera, is varied from 400 mm to 800 mm with steps of 50 mm. Around this center position the object is moved from -50 mm to 50 mm with steps of 5 mm. The measurements are repeated 10 times.

```
1  for n = 1:10
2          pause(10);
3          for D = 1:1:9
4          ao_data = [4 D 4.3];
5          putsample(ao_device, ao_data);pause(5);%go to start position
6              for X = 4:0.1:6
7                  ao_data = [X D 4.3];
8                  putsample(ao_device, ao_data);pause(1);%go to next step
9              end
10         end
11     end
```

Code 2.1: Matlab script to control the 3D scanner

In the matlab code in 2.1 can be seen how a sequence of commando's handles the movement. n goes from 1 to 10 andis the number of repetitions, D the distance between the camera and the object and X the position along the X axis. Steps of 0.1 V result in steps of 5mm along the axes. With the function 'putsample', the voltage values stated in the vector 'ao_data' are set on the analog output. Between every movement, a pause is inserted to give the 3D scanner time to move.

Synchronization between MATLAB and Delphi

In the FreeTrack interface, the data must also be selected automatically. This must done be right after the the movement of the 3D scanner. Either MATLAB or Delphi can have the lead in synchronizing this. If MATLAB has the lead, MATLAB can start the measurement, move the object to its destination and, when the object has arrived, tell Delphi to select the data. The MATLAB script can easily be extended with commands to send to Delphi. If Delphi has the lead, Delphi can tell MATLAB to move the object to the next position and select the data when it has arrived. This requires of MATLAB to communicate back to Delphi that the object has arrived. Since the Delphi application is designed to be controlled by a user, the synchronization is handled by MATLAB. MATLAB can then control the movement of the object and Delphi will display the data, but only select it when MATLAB wants it to. Different methods can be used for the communication between the two different applications.

Since the Delphi application is controlled with the mouse by the user, it is possible to simulate mouse movement and clicks by MATLAB. The cursor can be placed at a certain position and ordered to click at that location. This is an easy action and does not require any changes in the Delphi aplication. This requires the Delphi application to be at a position every time the measurement is run. The possibility that this is not the case and that this method fails is too large.

Another option is to execute MATLAB statements from the Delphi application with the use of Object Linking and Embedding (OLE). By previous tries with the interface from Delphi to MATLAB for the FreeTrack data, this was not successful and is therefore not applied in this case. This might prove to be a valid replacement for the chosen solution, discussed below.

There is also the option of communication between the two applications, either with COM port or with TCP/IP port. When using COM ports, two virtual COM ports have to be created and connected. When each application is connected to one of the two virtual COM ports, strings can easily be writen or read from it. This is an easy way of communication, however the virtual COM port pair has to be set up by a seperate program. When using TCP/IP, a port has to be assigned over which the applications communicate. It also provides the possibility of communication between two applications running on different PC's. The usage of a COM port is chosen over the use of TCP/IP because of the ease of setting up the communication.

With the usage of the Null-modem emulator com0com[13],a virtual COM port pair is created over which the applications can communicate. Figure 2.11 shows the sequencial diagram of the communication from MATLAB to Delphi. (The code used for both applications can be seen in appendix 6.2.

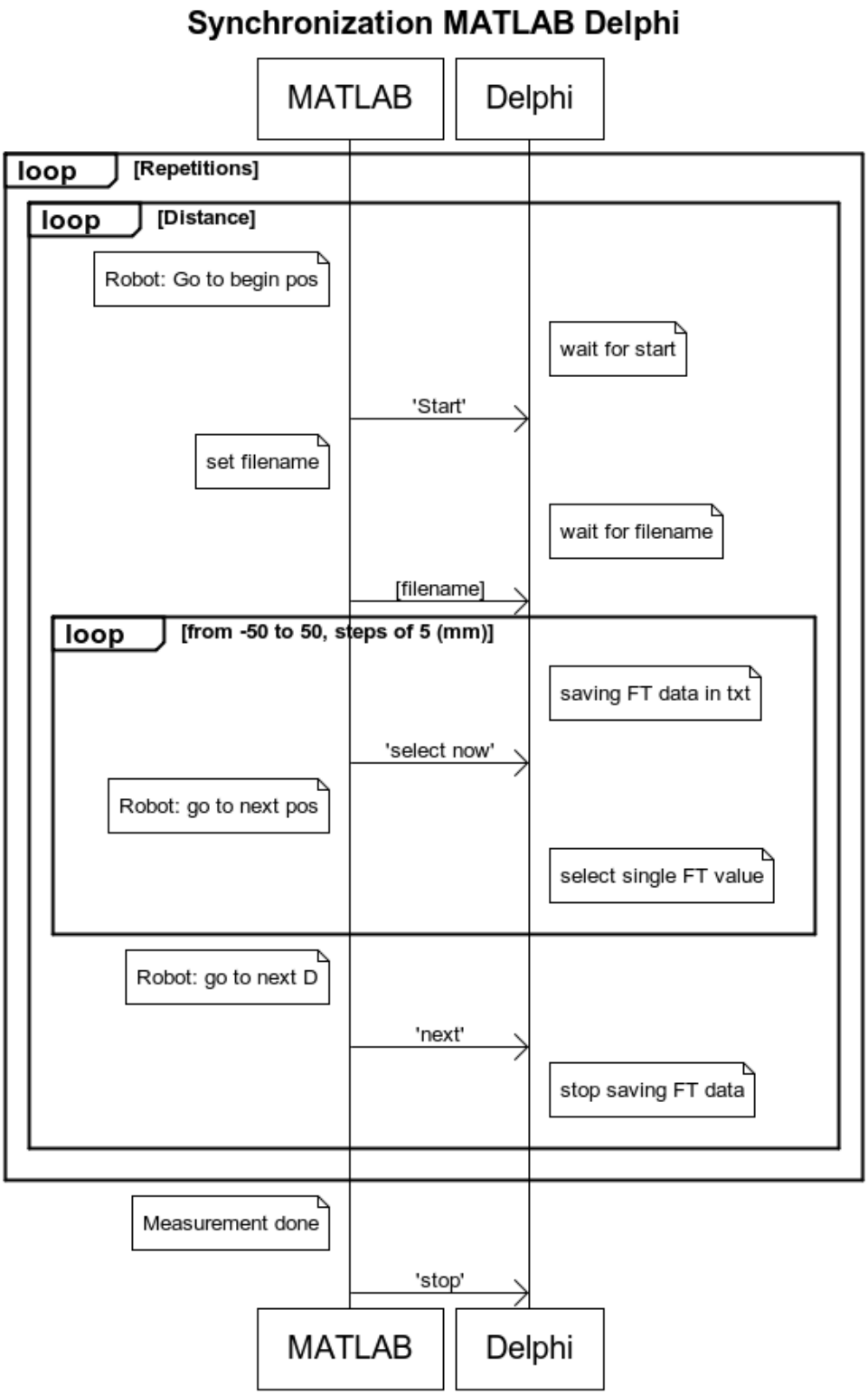


Figure 2.11: The sequence diagram of the communication from MATLAB to Delphi. On the MATLAB side, the 3D scanner is controlled and the command is sent to Delphi. On the Delphi side the command is received and the FreeTrack data is saved in the text file.

For every set of measurement, MATLAB controls the 3D scanner to the initial position and sends 'start' when it has arived. Delphi then waits for any incomming string, which is set as the filename, generated by MATLAB, based on what repetition, the distance, the kind measurement and/or the axis along which is measured. Delphi then starts the recording of the FreeTrack data. MATLAB sends 'select_now' after the 3D scanner moved the object to the next position. When Delphi receives 'select_now', the data is selected. The object is moved over the full range along the axis and when this is done, it is moved to the next distance. This process is repeated by the number of repetitions. Figure 2.12 shows the setup in the 3D scanner.

To keep the communication protocol simple, no acknowledgement messages are sent back from Delphi to MATLAB. This requires the sequence of commands send by MATLAB to be in the right order. Every time, after a command is sent, MATLAB pauses one second for transmission and to allow Delphi to process the command. The time taken by Delphi is always smaller than one second. Any noise is filtered in Delphi by only responding to certain commands. The filename is sent one second after 'start' is sent to reduce the change of receiving wrong data.

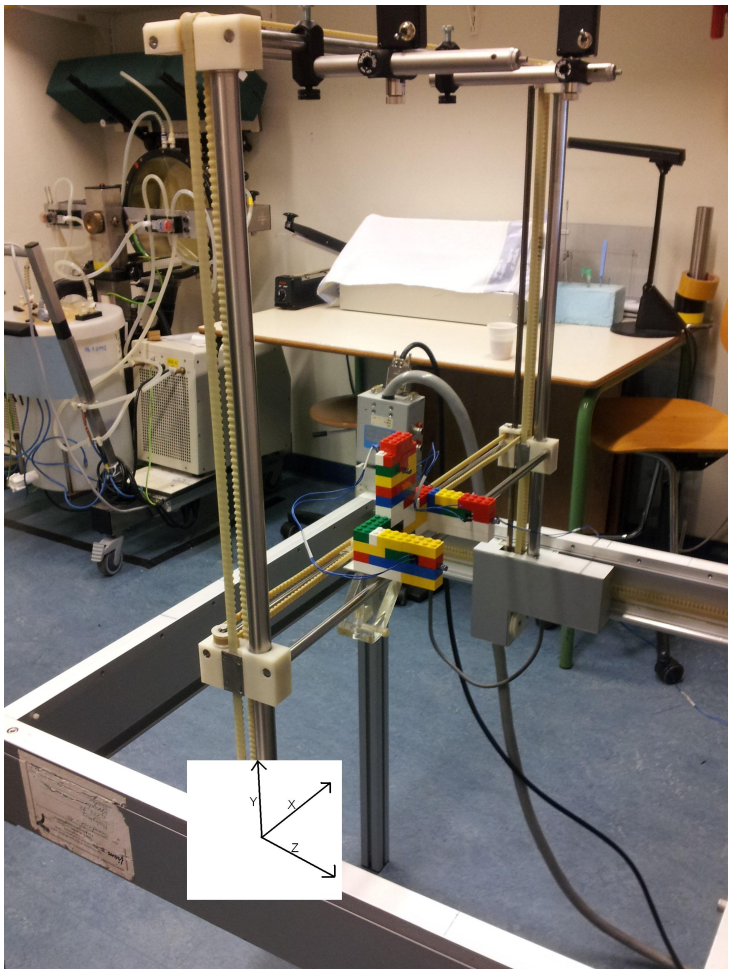


Figure 2.12: The LEGO object in the 3D scanner.

2.3.2 Results

The figures in figure 2.13 show the errors for the three translational axes. The different colors indicate the the different distances from 400 to 800. The different markers indicate the 10 repetitions.

Overall, the results look the same as the results of the measurements performed on the LEGO base plate. There is a difference of 2 mm in accuracy for the X axis. See the comment below for more information about this.

The error for the Y axis at -40 mm from the centre point, for the measurement at 400 mm distance from the camera is a wrong measurement and can be ignored.

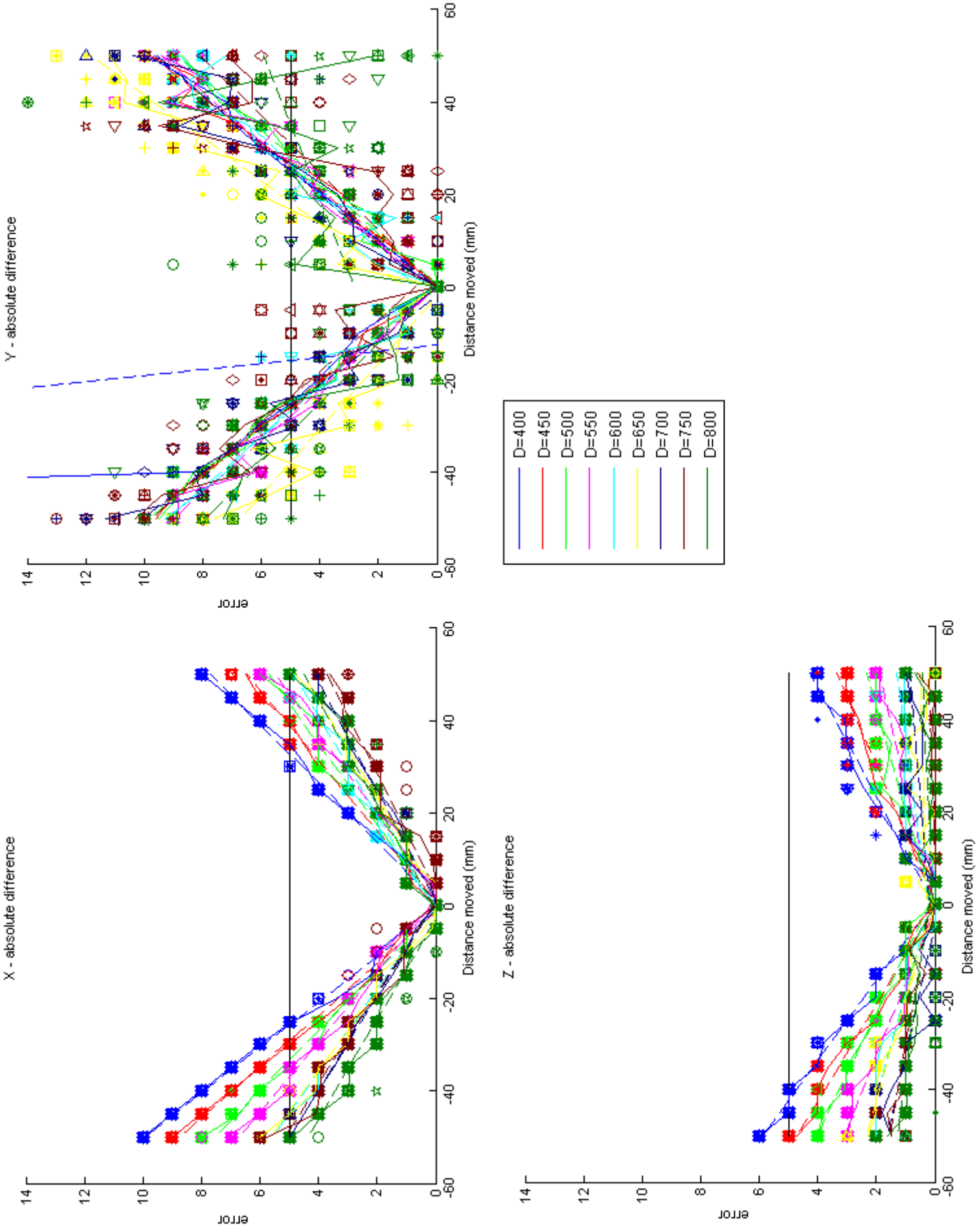


Figure 2.13: Error plots for automated measurements in the 3D scanner.

Comment

The difference in errors between the automated measurements and the measurements on the LEGO base plate is explained by the different setting of sensor width and focal length in FreeTrack. For the measurements on the LEGO base plate, the setting were untouched and were, very likely, 3.0 mm for sensor width and 3.7 mm for focal length. During the automated measurements, it was noticed that these settings influenced the measured data and were set to 4.8 mm for sensor width and 6.5 mm for focal length. 4.8 mm is a standard width for CCD sensors. The focal length is found by placing the object at 500 mm from the camera and setting the focal length so that the result in FreeTrack for the Y axis was close to 500 mm.

2.3.3 Calibration

With the use of the 3D scanner, it is possible to let the object move across a cubicle grid while tracking its movement. The data will then be selected at certain fixed points in the grid. The resulting data will be the calibration matrix. While measuring the movement of a patient, the FreeTrack data can be compaired with the calibration matrix and corrected.

Taken into account that a patient only has the possibility of moving 5 cm from its position in the HyperCollar, the size of the cube is 10*10*10 cm. Divided in steps of 5 mm, the grid has 21*21*21 = 9261 points.

The coordinate [5 3 4.3] is used as the center for the measurement. The object is moved around the center in the order of X-, Z-, Y-axis: from left to right, bottom to top and front to back, as seen from the camera. The measured X, Y and Z values at the certain locations are put in a 21x21x21 matrix in MATLAB. In the same order, a 21x21x21 matrix is filled with the real coordinates of the grid. As a result, there are two matices, one with the real coordinates and a calibration matrix, with the same size and with the real and measured coordinates at the same location. When subtracting the calibration matrix from the real coordinates matrix, an error matrix is aquired. Now, when FreeTrack measures a movement X, Y and Z of the patient, there are the following two methods of calibrating the measured data and compensating for the error:

Nearest

With the three coordinates of a measured position of a patient, the nearest point in the calibration matrix is searched. When the location of that point in the calibration matrix is known, the values, at the same location in the error matrix is taken and added to the measured position to aquire the real position.

The nearest point is found by equation 2.1, with C as the 21x21x21 calibration matrix and p the measured coordinates.

$$nearest = min(sum((|C - p|^2)), 1) \tag{2.1}$$

The measured coordinates are subtracted from every coordinate stored in C. This is then squared and the sum of the three coordinates is taken. The minimum resulting value is the nearest point. From this point, the location in C is taken.

Interpolation

For this method, trilinear interpolation is applied[14]. It is used to interpolate the value of the error in the error matrix by weigthing the value measured value to the eight surrounding points in the calibration matrix and applying this weigth to the points at the same locations in the error matrix. Appendix 6.3 shows the implementation in MATLAB.

When a measurement point has the three coordinates X, Y and Z, the coordinates are weighted to its closest and second closest points in the calibration matrix along their seperate axes. This is done by first

applying nearest and then, considering if the value is higher or lower than the value in the calibration matrix, picking the next or previous point in the matrix. The values are then weighted along their axis to the two closest points by:

$$Yd = (Y - Y_1)/(Y_2 - Y_1) \tag{2.2}$$

where Y_1 stands for the closest point and Y_2 the second closest point along the X axis. And for X and Z the same.

The weight is then applied to the real coordinates matrix, at the same location as the points in the calibration matrix. This is done in three steps. First the Y axis is interpolated by:

$$c_{00} = errorMatrix[X_1, Y_1, Z_1](1 - Y_d) + errorMatrix[X_2, Y_1, Z_1](y_d) \tag{2.3}$$
$$c_{10} = errorMatrix[X_1, Y_2, Z_1](1 - Y_d) + errorMatrix[X_2, Y_2, Z_1](y_d) \tag{2.4}$$
$$c_{01} = errorMatrix[X_1, Y_1, Z_2](1 - Y_d) + errorMatrix[X_2, Y_1, Z_2](y_d) \tag{2.5}$$
$$c_{11} = errorMatrix[X_1, Y_2, Z_2](1 - Y_d) + errorMatrix[X_2, Y_2, Z_2](y_d) \tag{2.6}$$

Then the X axis is interpolated by:

$$c_0 = c_{00}(1 - X_d) + c_{10}(X_d) \tag{2.7}$$
$$c_1 = c_{01}(1 - X_d) + c_{11}(X_d) \tag{2.8}$$

And, lastly, along the Z axis by:

$$c = c_0(1 - Z_d) + c_1(Z_d) \tag{2.9}$$

Figure 2.14 shows the schematical approach of trilinear interpolation. When a measured position is outside of the grid and there are not 8 surrouding points available, nearest is applied.

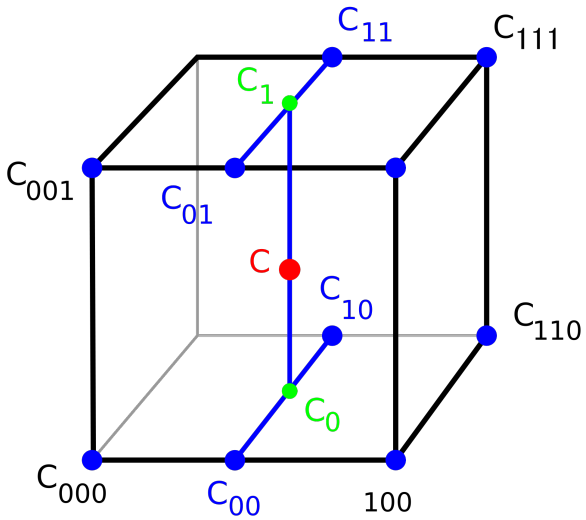


Figure 2.14: Trilinear interpolation[15].

2.3.4 Verification

For verifying the calibration matrix and both the methods used for calibration, the 3D scanner is used to bring the object to a large number fixed positions within the grid. The real coordinates of the positions are known to calculate the error.

The object is moved along the same 3D grid as the calibration matrix, but shifted 2.5 mm, which is half of the distance between two points in the calibration grid. Movement is measured relative to the middle point of the grid. Figure 2.15 shows a histogram of the number of error values before calibration.

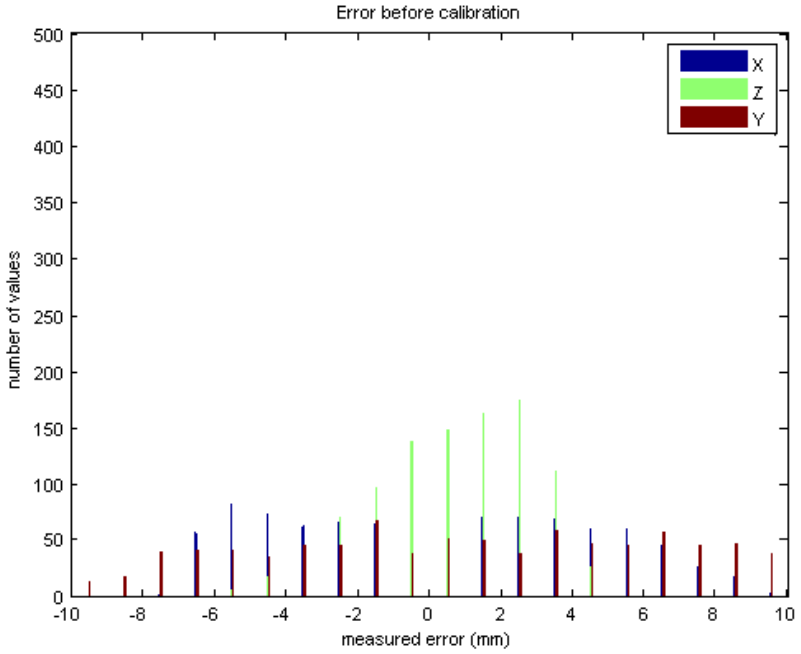
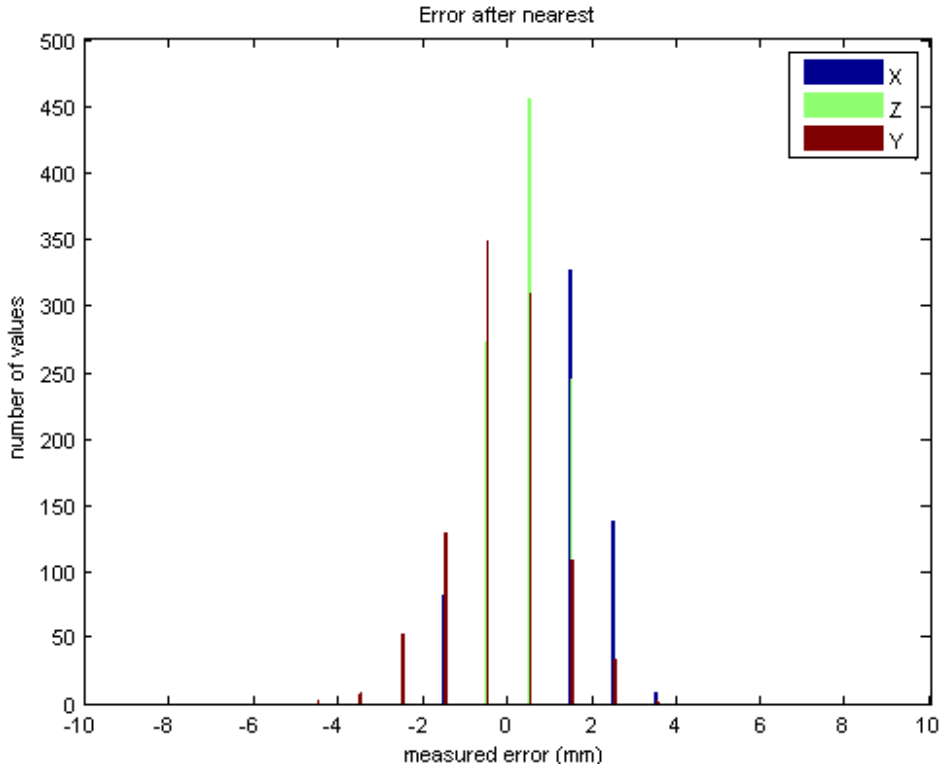


Figure 2.15: The histogram shows the number of error values before calibration with bars with a size of 0.1 mm.

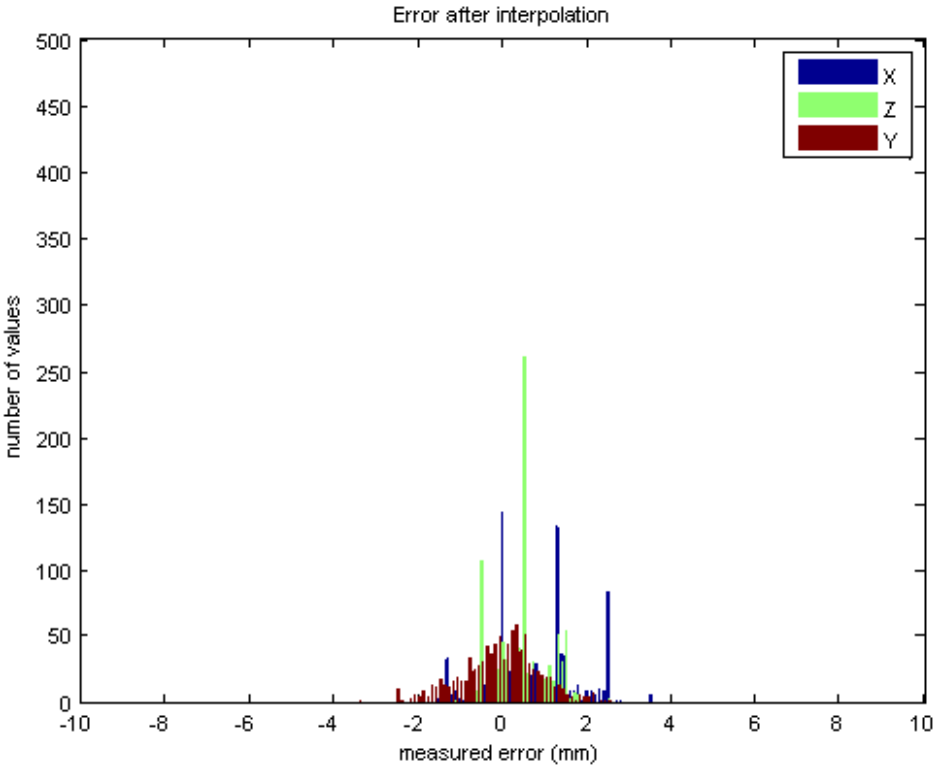
Both nearest and interpolation are applied to the measured values, which error histograms are shown in figure 2.16. After calibrating the measured values with both nearest and interpolation, the error stays within the required 5 mm for the full range of the calibration matrix. Table 2.2 shows the range in which the required accuracy can be made before and after calibration.

Table 2.2: The range in which the required accuracy can be made before and after calibration. The range of the translational axes is based on the measurements in the 3D scanner.

Axis	Required accuracy	Before calibration	After calibration
X	5 mm	-30 - 45 mm	-50 - 50 mm
Y	5 mm	-25 - 30 mm	-50 - 50 mm
Z	5 mm	-50 - 50 mm	-50 - 50 mm
Yaw	2.9°	-25 - 25°	not calibrated
Pitch	1.9°	-10 - 10°	not calibrated
Roll	1.9°	-25 - 20°	not calibrated



(a) Error histogram for nearest



(b) Error histogram for interpolation

Figure 2.16: After calibration, the error values stay below the required 5 mm.

Chapter 3 Clinical setup

To get an initial result of the system working in the clinic, a measurement is done in the HyperCollar during a clinical setting. A person (in this case me) is positioned in the HyperCollar with a simpel farbricated cap on the head, with approximately the same dimensions as the LEGO object. Figure 3.1 shows a picture of the setting. For an hour, the person lay in the HyperCollar trying to hold still. In the beginning of the treatment, this is not very difficult, but as laying in the HyperCollar gets more uncomfortable, this might result in some movement.

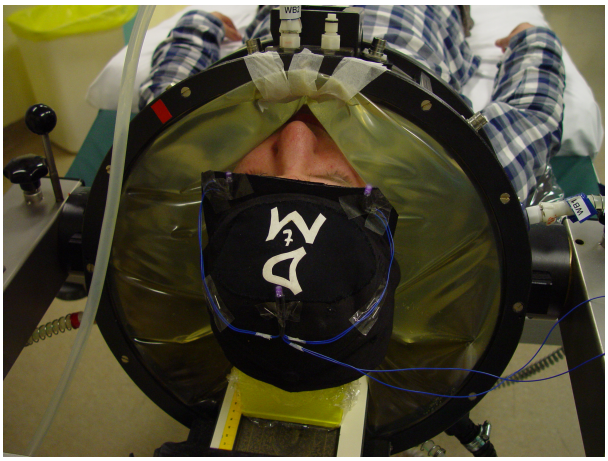


Figure 3.1: An initial measurement in a treatment like setting.

3.1 Results

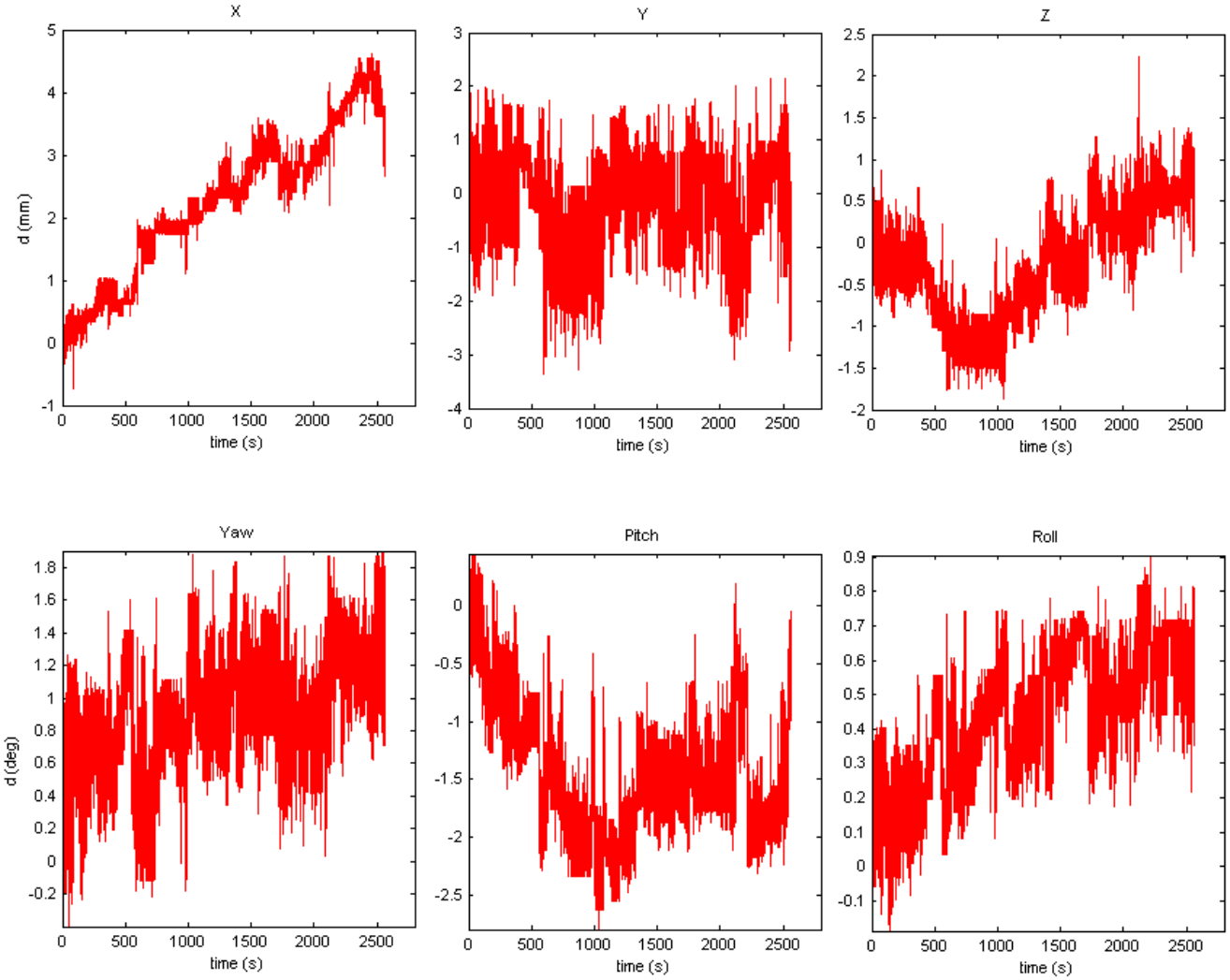


Figure 3.2: The movement along the translational axes during the treatment like setting.

During the treatment, the FreeTrack interface is run continuous so the data is saved every 50 ms in the text file. The data in the text file is read by matlab and is calibrated with the acquired calibration. Both nearest and linear interpolation are applied on the data and show the same results. The tracked movement after linear interpolation can be seen in figure 3.2. It shows that during the treatment, the head is moved 5 mm to the left, moves about 5 mm continuously backwards and forwards and 2 mm up and down. The head is also yawed up to 1.8°, pitched up to -2.5° and rolled up to 0.9°.

3.2 Discussion

When taking conclusions about the shown data, it should be considered that I am a young healthy person and during the measurement, the HyperCollar was not active. When a patient undergoes a treatment and feels discomfort because of his/her illness and the head and neck region is heated, the discomfort is much larger then it was for me. I did not experience the treatment as very uncomfortable and was easily capable of holding my head within a range of 5 mm. The discomfort of a patient undergoing treatment is greater and the patient is less capable of holding the head still.

As mentioned in section 2.1, the model position is set to 0. This means that FreeTrack does not compensate the translation caused by the rotating the head. The tracked movement is not the movement of the head, but the movement of the cap. In figure 3.2, it can be seen that X and roll change the same way, as also for Z and pitch. Besides this, the values of the rotation are not calibrated since the 3D scanner is only used to measure translation.

Table 3.1 shows how much a healthy person moves during the full length of a Hyperthermia treatment in the head and neck region without activation of the applicator.

Table 3.1: The range of the movement during the measurement in a treatment like setting.

Axis	Range ±1
X	0 - 5 mm
Y	-3 - 2 mm
Z	-2 - 2 mm
Yaw	0 - 2°
Pitch	-3 - 0°
Roll	0 - 1°

Chapter 4 Conclusion

The goal of this project was to build a low cost patient position tracking system for Head and Neck Hyperthermia. An infrared head tracking system was built with the use of the free, open source software FreeTrack.

Accuracy and precision measurements have shown that the system has a rising error when displacing the object away from the centre point. This error can be reduced with the use of linear calibration. After calibration, the required accuracy can be achieved in the full range of the calibration matrix, which means any point allong a ±50 mm axis from the centre point. This system is accurate enough to be applied during a Hyperthermia treatment in the HyperCollar.

The system has been tested in a clinical setting. This showed that the movement of a test person inside the HyperCollar, without applying Hyperthermia, is below 6 mm and 4°from the centre point within an hour. This shows that the movement of a person during a treatment in the HyperCollar is not significant, however subsequent development and implementation of a clinical setup will have to show how much a patient moves his head during an actual Hyperthermia treatment.

Chapter 5 Future work

During this project, the main focus has been the accuracy and precision measurements and error correction through calibration. Below are listed what aspects of the system still have to be tested and what development is required before the system can be applied during a treatment with a patient.

- One of the stated requirements was that active light sources could be attached on the head of a patient. The patient should not need to wear a cap. As a possible solution, Passive lighting with the use of infrared floodlight and reflective material like reflective tape and silver beads is tried. These attempts to reflect enough light to distinguish three dots were not succesfull. An overexposed negative film has been suggested[16] as an improvement over the currently used filter, created from a piece of floppy disk, since it filters more of the visual light and allows more infrared light to pass. Different cameras, specially designed for infrared light, can also be used to try to improve the visability of the reflectors.
- The effect of setting the model position to the correct distances has not been investigated. If the dimensions of a patients' head are known and are set in FreeTrack, FreeTrack can compensate for the unwanted translation and rotation, but more research has to be done on the effect of the settings on the measured data.

Bibliography

[1] Erasmus MC. Research projects. http://www.erasmusmc.nl/radiotherapie/research/hyperthermia/projects/. Accessed Febuary, 2012.

[2] Erasmus MC. Development of a head and neck ht applicator. http://www.erasmusmc.nl/radiotherapie/research/hyperthermia/projects/2467076/. Accessed Febuary, 2012.

[3] P. Togni. Patient positioning in head neck hyperthermia: relevance, assessment and reduction, 2009.

[4] FreeTrack. Freetrack website. www.free-track.net. Accessed Febuary, 2012.

[5] T. D. Alter. 3d pose from 3 corresponding points under weak-perspective projection. *A.I. Memo*, 1378, 1992.

[6] NOAA. Accuracy vs precision illustration. http://celebrating200years.noaa.gov/magazine/tct/accuracy_vs_precision.html. Accessed at 2-05-2012.

[7] G. C van Rhoon W.J.W. Lagendijk. Asho quality assurance guidelines for regional hyperthermia. *J. hyperthermia*, 14:125–133, 1998.

[8] M. Paulides. *Development of a clinical head and neck hyperthermia applicator*. PhD thesis, Erasmus Universiteit Rotterdam, 2009.

[9] Microsoft. Microsoft hardware. http://www.microsoft.com/hardware/nl-nl/p/lifecam-vx-800/JSD-00009. Accessed at 8-02-2012.

[10] OrianRobots. Lego specifications. http://orionrobots.co.uk/Lego+Specifications. Accessed at 9-03-2012.

[11] LEGO. Digital designer virtual builder software. http://ldd.lego.com/. Accessed at 9-03-2012.

[12] MathWorks. Mex-files guide. http://www.mathworks.nl/support/tech-notes/1600/1605.html#intro. Accessed at 15-02-2012.

[13] com0com. Virtual serial port driver for windows. http://com0com.sourceforge.net/. Accessed at 7-05-2012.

[14] Paul Bourke. Interpolation methods. http://paulbourke.net/miscellaneous/interpolation/. Accessed at 16-05-2012.

[15] Marmelad. Depiction of three dimensional interpolation. http://en.wikipedia.org/wiki/File:3D_interpolation2.svg. Accessed at 15-02-2012.

[16] Madian. Nui group community forums. http://nuigroup.com/forums/viewthread/6458/. Accessed at 21-05-2012.

Chapter 6 Appendix

6.1 Appendix A - Error plots

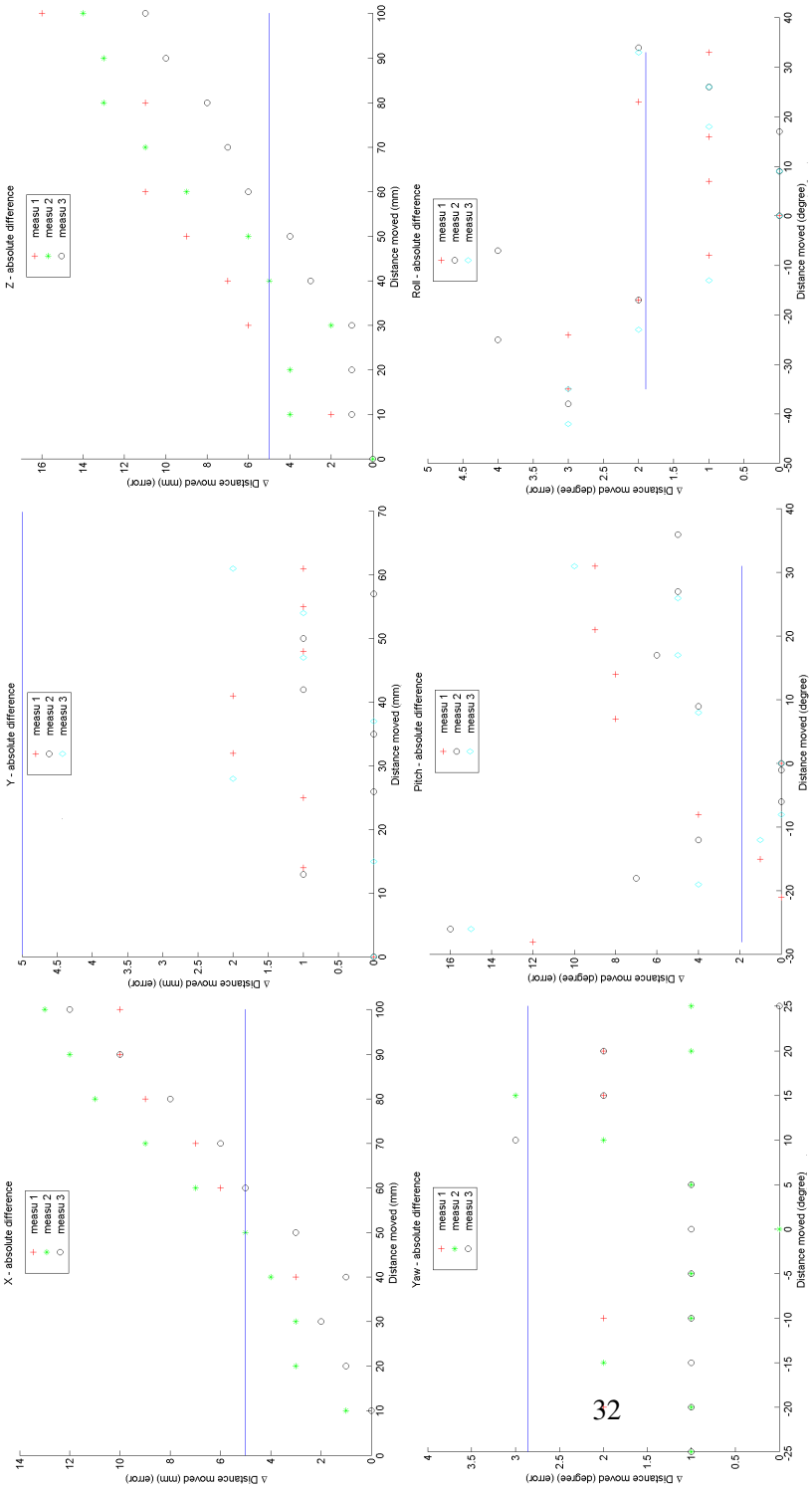


Figure 6.1: Difference between the real movement and the measured movement. The blue horizontal line indicates the required accuracy.

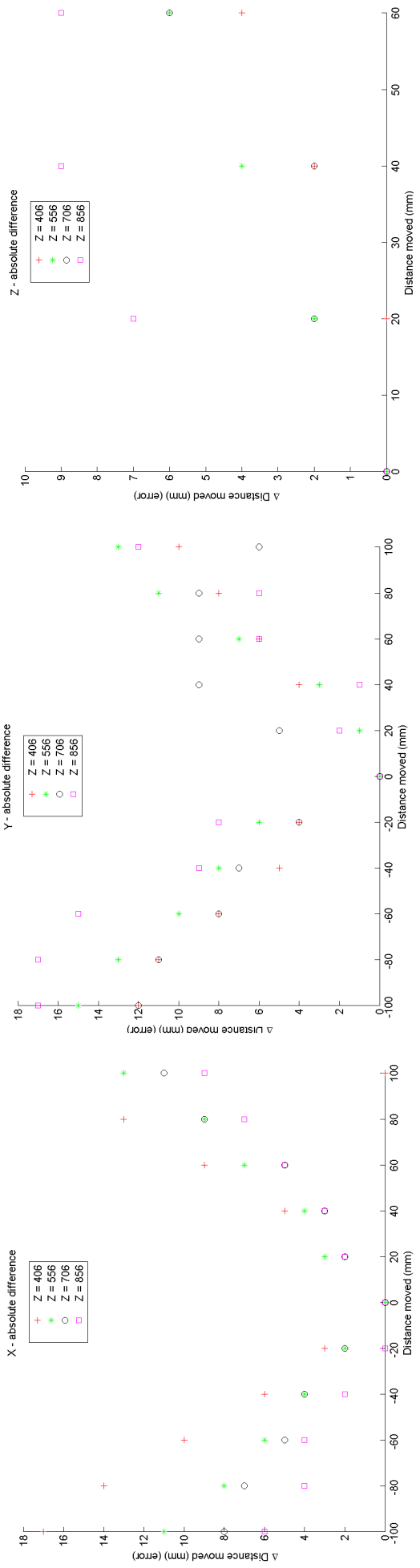


Figure 6.2: Difference between the real movement and the measured movement. The blue horizontal line indicates the required accuracy.

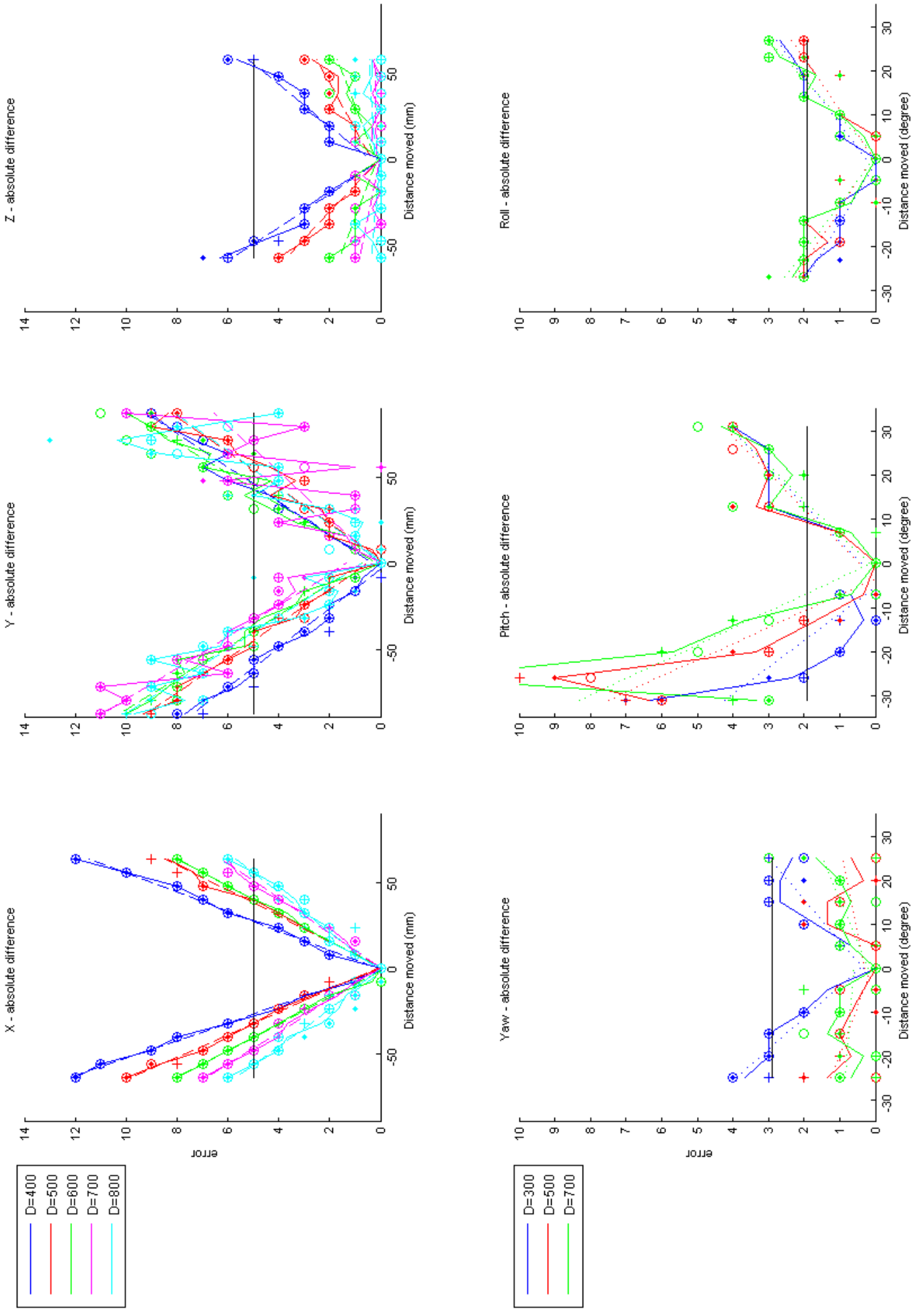


Figure 6.3: Difference between the real movement and the measured movement. The blue horizontal line indicates the required accuracy.

6.2 Appendix B - Communication from MATLAB to Delphi

```
com = serial('COM5');
fopen(com);
for n = 1:10
    pause(10);
    for D = 1:1:9
%go to start position
        fwrite(com,'start');pause(1);%start with new measurement
        file = [axis,'_m',int2str(n),'_D',int2str(D*50+350)]; %filename
        fwrite(com,file); pause(1);
        for X = 4:0.1:6
%go to next step
            fwrite(com,'selectnow');pause(1);%select data
        end
        fwrite(com,'next'); pause(1);%next measurement
    end
end
fclose(com);
```

Code 6.1: Matlab script for synchronization with Delphi. The same matlab code is used as in 2.1, only the 3D scanner code is left out for clarity

```
    readBuf := '';
    readBuf := ComPort_.Read();
     if AnsiCompareStr(readBuf,'stop') = 0 then begin
        SaveToggleBox.checked := false;
     end else if AnsiCompareStr(readBuf,'start') = 0 then begin {*start with ...
         new measurement*}
        readBuf := '';
        readBuf := ComPort_.Read();
        while not(readbuf <> '') do begin
          readBuf := ComPort_.Read();
        end;
        fileName := readBuf; {*filename*}
        SaveToggleBox.checked := true;
     end else if AnsiCompareStr(readBuf,'next') = 0 then begin{*next ...
         measurement*}
        SaveToggleBox.checked := false;
     end else if AnsiCompareStr(readBuf,'selectnow') = 0 then begin {*select ...
         data*}
        SelectDataButtonClick(Sender);
     end else if not(readbuf <> '') then begin
       if SaveToggleBox.checked then begin
          WriteData(Sender,FileName);
       end;
     end;
```

Code 6.2: Delphi script for synchronization with Matlab

6.3 Appendix C -Trilinear interpolation in MATLAB

```
%%Trilinear Interpolation
function [position,fail] = TriLinearInterpolation (calMatrix, grid, p, m1, n1...
    , o1)
fail=0;
if p(2) < calMatrix(2,m1,n1,o1)  %change in m gives change in CT(Z)
    m2 = m1+1;
elseif p(2) > calMatrix(2,m1,n1,o1)
    m2 = m1-1;
else
    p(2) = p(2) + 0.01;
    m2 = m1 - 1;
end
if  p(1) < calMatrix(1,m1,n1,o1)  %change in n gives change in CT(X)
    n2 = n1-1;
elseif p(1) > calMatrix(1,m1,n1,o1)
    n2 = n1+1;
else
    p(1) = p(1) + 0.01;
    n2 = n1 + 1;
end
if p(3) < calMatrix(3,m1,n1,o1)  %change in o gives change in CT(Y)
    o2 = o1-1;
elseif p(3) > calMatrix(3,m1,n1,o1)
    o2 = o1+1;
else
    p(3) = p(3) + 0.01;
    o2 = o1 + 1;
end
if m2 > 21 | n2 > 21 | o2 > 21 | m2 < 1 | n2 < 1 | o2 < 1
    position = [0,0,0];fail = 1;return;
end
    yd = (p(2)- calMatrix(2,m1,n1,o1))...
        /(calMatrix(2,m2,n1,o1)...
        -calMatrix(2,m1,n1,o1));
    xd = (p(1)- calMatrix(1,m1,n1,o1))...
        /(calMatrix(1,m1,n2,o1)...
        -calMatrix(1,m1,n1,o1));
    zd = (p(3)- calMatrix(3,m1,n1,o1))...
        /(calMatrix(3,m1,n1,o2)...
        -calMatrix(3,m1,n1,o1));
    c00 = grid(:,m1,n1,o1)*(1-yd)+...
        grid(:,m2,n1,o1)*yd;
    c10 = grid(:,m1,n2,o1)*(1-yd)+...
        grid(:,m2,n2,o1)*yd;
    c01 = grid(:,m1,n1,o2)*(1-yd)+...
        grid(:,m2,n1,o2)*yd;
    c11 = grid(:,m1,n2,o2)*(1-yd)+...
        grid(:,m2,n2,o2)*yd;
    c0 = c00*(1-xd)+c10*xd;
    c1 = c01*(1-xd)+c11*xd;
    position = (c0*(1-zd)+c1*zd)';
```

Code 6.3: MATLAB script for applying trilinear interpolation