2017•2018
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Calibration and evaluation system for 3D camera systems

PROMOTOR :
Prof. dr. ir. Luc CLAESEN

BEGELEIDER :
De heer Wout SWINKELS

Thomas Aerts

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

2017•2018
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Calibration and evaluation system for 3D camera systems

PROMOTOR :
Prof. dr. ir. Luc CLAESEN

BEGELEIDER :
De heer Wout SWINKELS

Thomas Aerts
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Acknowledgements

This thesis is the pinnacle of the education to Master of Science in Electronics Engineering and Computer Science at the Hasselt University. The realization of this master's thesis was a challenging but gratefull experience. Without the help of others, this work would not have been completed in the given time. Therefore, I would like to thank everyone who contributed in any kind to the different stages of this thesis.

I would like to express my deepest gratitude to my promoter prof. dr. ir. Luc Claesen and co-promoter ing. Wout Swinkels. I was given a lot of freedom to accomplish my goal while they guided me in the needed direction. They always helped me with additional information, excellent guidance and sufficient patience when I encountered a problem.

Also thanks to the co-workers at the Makerspace, in particular Mr. Tom De Weyer. His expertise about the machines and materials I had to use were critically important and helped to bring my design to the final stage. I could even rely on him after his working hours when the makerspace was closed.

I would also like to thank my family, friends and everyone who helped me in one way or another. Especially thanks to my girlfriend Sofie for her unending support and motivation.

Thomas Aerts, January 2018

Table of contents

Inhoudsopgave

List of Tables

List of Figures

List of Abbreviations

1080p	1080 Horizontal Lines Resolution
6DOF	6 Degrees of Freedom
720p	720 Horizontal Lines Resolution
A	Ampère
C	The C Programming Language
C++	The C++ Programming Language
CMOS	Complementary Metal Oxide Semiconductor
DC	Direct-Current
DDR3	Double data rate type three SDRAM
GB	Gigabyte
GUI	Graphical User Interface
HD	High-definition
HDMI	High-Definition Multimedia Interface
HP	Hewlett-Packard
IDE	Integrated Development Environment
JPG/JPEG	Joint Photographic Experts Group
mA	Milliampere
MDF	Medium-density Fiberboard
µm	Micromillimeter
Mm	Millimeter
MOV	QuickTime Movie File Format
MP	Megapixel
PWM	Pulse Width Modulation
SD	Secure Digital Non-Volatile Memory Card
SDRAM	Synchronous Dynamic Random-Access Memory
USB	Universal Serial Bus
V	Voltage
Wi-Fi	Wireless Local Area Networking

Abstract

The Computational Sensor Systems research group, located at the university campus in Diepenbeek, is committed to the development of modern multi-camera video and smart vision systems. It achieves this through novel hardware-software architectures, each designed for the specific camera distribution to optimize resolution, frame rate and latency. Camera calibration is necessary when images from different cameras are combined and objects are measured in the scene. The calibration eliminates the radial distortions of the lens and the tangential distortions of manufacturing errors. The main objective of this master thesis is to realize a setup to perform accurate and repeatable camera calibration to compare different calibration methods and techniques.

The development of the demo setup comprises three parts. First, the hardware is assembled and consists of two major parts, i.e. the stepper motor and stepper driver circuit. Different drivers are tested to compare consistency. Secondly, a custom design made medium-density fiberboard is implemented to attach different calibration patterns to. Thirdly, the software to calibrate the camera is realized using MATLAB and OpenCV and a graphical user interface is made in Java.

Camera calibration is an effective method to improve image and video quality of less qualitative cameras. The stepper motor in combination with micro stepping produces accurate repeatable movement. Test results show that images captured with the setup are consistent and the implemented algorithm provides repeatable measurements.

Samenvatting

De Computational Sensor Systems onderzoeksgroep, gesitueerd aan de universitaire campus UHasselt in Diepenbeek, is toegewijd aan de ontwikkeling van moderne multi-camera video en smart vision systemen. Ze bereikt dit door originele hardware-software architecturen te ontwerpen voor specifieke camera distributies om de resolutie, het aantal beelden per seconde en latentie te optimaliseren. Camerakalibratie is noodzakelijk wanneer beelden van verschillende camera's gecombineerd worden en men objecten in de scene exact wil meten. De kalibratie elimineert de radiale vervormingen van de lens en de tangentiële vervormingen veroorzaakt tijdens het productieproces. Het hoofddoel van deze masterproef is om een opstelling te realiseren om nauwkeurige en herhaalbare camerakalibratie uit te voeren zodat men verschillende kalibratiemethoden en technieken met elkaar kan vergelijken.

De ontwikkeling van de installatie bestaat uit drie delen. Ten eerste wordt de hardware geassembleerd en deze bestaat uit twee hoofdonderdelen, d.w.z. de stappenmotor en het stappenmotor-aandrijfcircuit. Verschillende circuits worden getest om de consistentie te vergelijken. Ten tweede is een aangepast ontwerp, gemaakt van vezelplaat met gemiddelde dichtheid, geïmplementeerd om verschillende kalibratiepatronen aan te bevestigen. Ten derde is de software voor het kalibreren van de camera gerealiseerd met behulp van MATLAB en OpenCV. Een grafische gebruikersinterface is gemaakt in Java.

Camerakalibratie is een effectieve methode om de beeld- en videokwaliteit van camera's te verbeteren. De stappenmotor in combinatie met microstappen levert een nauwkeurige herhaalbare beweging op. Testresultaten laten zien dat afbeeldingen die met de opstelling zijn gemaakt consistent zijn en het geïmplementeerde algoritme herhaalbare metingen biedt.

Chapter 1

Introduction

1.1 Research situation

During the Master program in Engineering at Hasselt University, I had the opportunity to accomplish my thesis at the Computational Sensor Systems research group. The main objective of this master's thesis is to realize a demo setup to perform accurate and repeatable camera calibration to compare different calibration methods and techniques.

Modern digital imaging sensors give rise to numerous innovative applications in multi-camera video and smart vision systems. Multi-camera systems are used in omnidirectional video, 3D reconstruction, visual-interpolation and virtual cameras. Smart vision systems are applied in real-time traffic control, production line control, vehicle and medical applications. The computational and communication requirements for such camera systems are very challenging and require application-based hardware.

The CoSenS (Computational Sensor Systems) research group at the University of Hasselt conducts research and develops new hardware-software architectures for high resolution, high frame rate, low latency, multiple camera distributed computing computational video and vision systems.

1.2 Problem statement

A common method to measure translations and rotations in a 3D space is by use of camera systems. These systems are first calibrated to eliminate lens distortions. Widely used techniques to derive the intrinsic and extrinsic camera parameters are already extensively investigated [1], [2]. These techniques can also be used to derive the position in 6 degrees of freedom (6DOF) by use of a calibrated image and the calibrated camera.

The problem is that there is no general setup to test and evaluate the accuracy of the different available camera pose estimation methods. The goal of this thesis is the development of a pose estimation system based on a single-camera setup. In this setup, the practical accuracy of the camera calibration to estimate the XYZ translation and 3D rotation is researched and compared to the ideal theoretic model. This comparison is made so that it is possible to tell where specifically the practical data deviates from the ideal calculated model. For instance, a medical procedure that uses camera pose estimation needs to be very accurate, repeatable and user-friendly [3]. If the camera pose estimation method is not accurate, the operation risks failure and the patient could be in discomfort afterwards. By measuring the deviation between the practical and ideal model, the deviation can be altered through software implementation that reduce it to a minimum.

1.3 Objectives

This thesis aims to develop a highly accurate camera calibration test setup which will allow the testing of various camera pose estimation methods.

Firstly, the translation across the axis from which the calibration image will move relatively to the camera must be accurate enough so that for a given reference point the calibration is repeatable.

Once the translation movement is proven to be precise, rotation around the translation axis will be added. The rotation movement and the translation must remain stable so that again the calibration is repeatable. Combining these geometric transformations in space will result in a myriad of possible configurations for which the camera can be calibrated with a calibration image. This first step will include the crafting of a test bench and the implementation of a stepper motor circuit. The stepper motor is necessary to provide the movement along the axis and will be controlled through a software interface on a microcontroller.

Secondly, the quantitative data gathered from this calibration must be presented in an orderly graphical user interface so that the user immediately can compare the practical data from the current configuration with a previous configuration, an ideal model or another estimation method. According to this comparison, a specific software algorithm can be called to reduce the deviations between the different models to a minimum and so produce a camera image that is much closer to reality.

Finally, several improvements in processing and filtering can be implemented to boost the overall calibration speed. This includes testing different calibration images, optimizing the used algorithms that provide the ordered data for the user interface and an in depth look at the interface between the microcontroller and the host computer.

1.4 Methodology

First, the camera for the pose estimation will have to be calibrated so that the intrinsic parameters are known. To measure translations and rotations, the barrel and tangential distortions must be removed to produce an undistorted image that can be used in all the measurements. During this first phase, a study of literature will determine if the algorithms that perform the pose estimation are going to be written in Python. This implementation will rely heavily on the use of the Open Source Computer Vision Library (OpenCV), an open source library of functions developed for real-time computer vision. This library includes optimized algorithms for both classic and state-of-the-art computer vision and provides a practical application interface that integrates with a MATLAB or Python interface.

Secondly, once these distortions are documented and the programming choice is made, a procedure of testing the camera pose estimation methods against each other will be implemented. To effectively compare the different methods, a test bench that assures that every measurement will take place in the same condition will be built. This is necessary because the camera pose estimation accuracy relies on the accuracy of the distance from the calibrated image to the calibrated camera. Thirdly, the rotation accuracy around one of the primary axis is measured and finally the standard deviation of the former two parameters will be determined with respect to a fixed position. The deviation will be plotted against other models and this comparison will indicate the proper software algorithm to counter said deviation [1], [2], [4].

1.5 Outline of the thesis

Chapter 1: Introduction

We start by situating the setting of the thesis, along with the problem statement. We set certain objectives that must be completed and we explain the methodology on how we are going to achieve this. We go over the needed materials and the used methods.

Chapter 2: Literature Review

In this chapter, we explain the theory behind different camera models and the methods used to calibrate them. Most of camera theory is derived from the OpenCV guide, which explains models and math in detail. Further we explain the practical model of a stepper motor with the theory behind micro stepping.

Chapter 3: Materials and methods

In the third chapter, we give an overview of the specifications of the hardware and the software implementations for the test setup.

Chapter 4: Design Results

In the fourth chapter, we show the pattern applicator design that is used for the calibrations with the test setup. Further we explain the user interface that controls the movement of the stepper motor.

Chapter 5: Measurements

In the fifth chapter, we present the measurements. These include the accuracy measurements of the stepper motor and the comparison between the calibration by hand and the calibration with the test setup.

Chapter 6: Conclusion

In the last chapter, we present the conclusion and suggest further improvements for future work.

Chapter 2

Literature review

To understand camera calibration, the principles behind the working of a camera and the errors that can occur when capturing an image are explained. The first section explains the theory of a pinhole camera model, a simple camera without a lens. The pinhole camera is a theoretical concept. A practical model involves the implementation of a lens. A lens causes distortions to the acquired image alongside the distortions that are caused by the camera manufacturing process. The purpose of camera calibration is to eliminate or minimalize these lens distortions and map the intrinsic properties.

2.1 Pinhole camera model

Light that illuminates an object is mostly absorbed. The fraction of light that is not absorbed, is reflected and captured by an image sensor or the human eye. The reflected light conveys information about the color of an object. Light is electromagnetic radiation bound with an electromagnetic spectrum and is represented as a wave. The attenuation and wavelength of the light wave determine its intensity and color. Visible light as seen by the human eye is generally defined as ranging from 400 nm to 700 nm.

In theory, a pinhole camera captures an image without distortion. The reflected light from an object goes through an infinitesimal small hole in the front of the camera and is projected on the surface of the back of the camera (see Figure 1 left side). This small hole is also called the optical center of the camera. The ideal pinhole camera model describes the relation between the object point P, the real point in 3-dimensional space, and the image point P', the projection of the object point onto the camera [5].

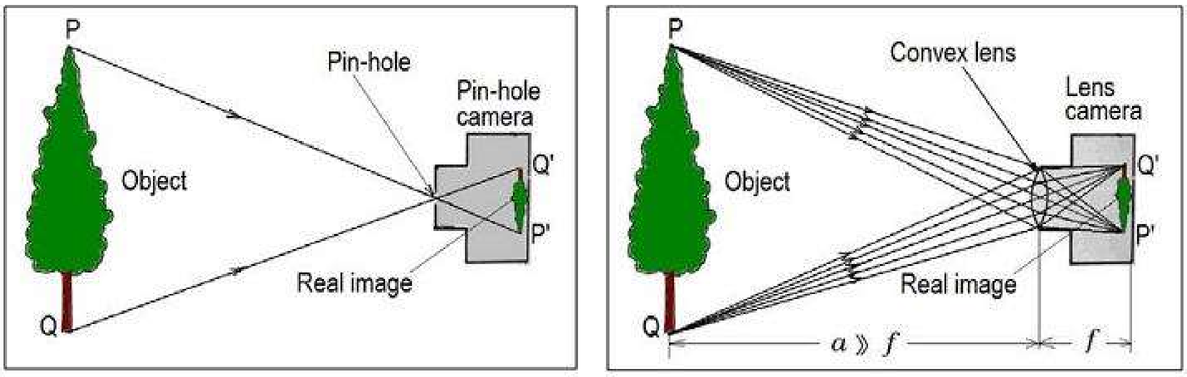


Figure 1: Left the pinhole camera model and right the camera lens model [6].

Light rays reflect in many directions from an object and the number of rays that reach the small hole is very low, therefore an implementation of the ideal pinhole camera model is not very practical. A practical model would involve longtime exposure to light to genuinely capture the image, consequently this model can only be used for very slow imaging. To overcome this problem and create a camera with a higher shutter speed, a lens is introduced. Lenses gather the incoming light by bending the beams from multiple angles to the image sensor (see Figure 1 right side). However, the use of a lens introduces a more complex projection geometry and lens distortions.

In the normal pinhole camera model, the captured scene is inverted because the image plane is placed behind the pinhole. When the image plane is virtually placed in front of the pinhole, the discomfort of the scene appearing upside down is resolved. This change allows a mathematically equivalent non-inverting geometry, making the math easier to write down. The image plane is situated at a distance f (focal length) from the camera center C (see Figure 2). The Z-axis that goes through the camera center perpendicular to the image plane, is called the principal axis or the optical axis. The point of the perpendicular intersection between the principal axis and the image plane is called the principal point p. Figure 2 shows this point to be at the image plane center. The intersection of the line created by the camera center and the actual object point X with the image plane results in the projection of a point in the real world.

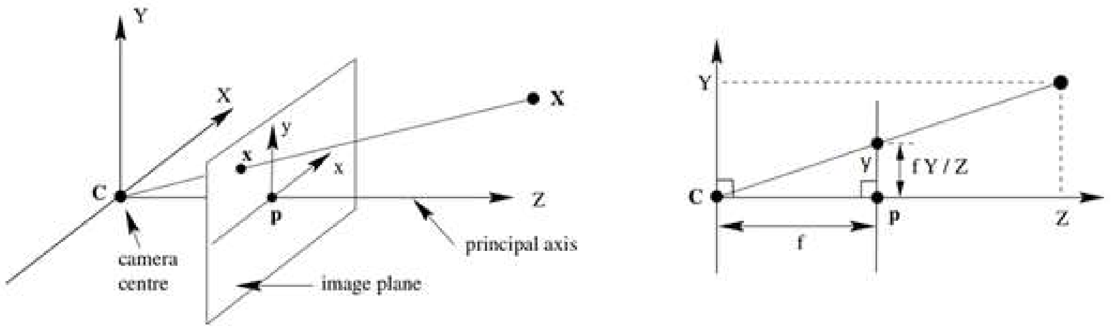


Figure 2: Pinhole camera model with the image plane situated in front of the camera coordinate system [7].

The relation between the 2D coordinates in an undistorted image plane and the 3D coordinates are calculated with basic trigonometry, as shown on the right side of figure 2. For a point p in 3D space with coordinates (X, Y, Z) to a point on the image plane at distance of focal length f, it is possible to establish the following equation [8].

$$\frac{f}{Z} = \frac{y}{Y} \leftrightarrow y = \frac{fY}{Z} \qquad (1)$$

When looking down from the y-axis to the xz plane, one can find the same relation:

$$\frac{f}{Z} = \frac{x}{X} \leftrightarrow x = \frac{fX}{Z} \qquad (2)$$

With these equations, we can map the 3D point p (X, Y, Z) to the 2D xy plane:

$$(\mathrm{X}, \mathrm{Y}, \mathrm{Z}) \mapsto (\mathrm{x}, \mathrm{y}) \leftrightarrow (\frac{fX}{Z}, \frac{fY}{Z}) \qquad (3)$$

In homogenous coordinates and a more formally matrix transform this results in $\vec{x} = P\vec{X}$:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ fZ \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (4)$$

This 3×4 matrix *P* is called the camera projection matrix.

2.1.1 Camera intrinsic properties

The intrinsic parameters of a camera do not depend on the surrounding area of the captured image. Instead these parameters are intrinsic to each specific camera and thus never change. To obtain the camera calibration matrix which contains the parameters, a translation component must be added to the left 3×3 matrix of P. When adjusting matrix P for the reference frame in the focal plane, the camera calibration matrix K is obtained. For further generalization, the matrix is multiplied with a diagonal m-matrix to account for non-square pixels, with m the number of pixels per unit of distance. This value is represented by α. The x-coordinate and y-coordinate are the origin points in the world coordinate system. The last potential distortion in the calibration matrix is the possibility of the axis not being perpendicular. This distortion can occur when the lens is not placed in absolute parallel with the image sensor during the manufacturing process. The angle between the optical axis and the sensor is called skew and is denoted by s. In most applications, the skew is negligible or unnoticeable. The final calibration matrix is denoted as follows [8]:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} m_x & s & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

For further calculations, the coordinate system and superscripts are used to denote the correct plane. When applying this form to the equation $\vec{x} = P\vec{X}$, the result looks like:

$$\vec{x} = P\vec{X} :\leftrightarrow \; P^D = K.P^U = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} .P^U \qquad (6)$$

Here (P^D) and (P^U) represent the distorted and undistorted image coordinates, (f_x, f_y) the distance to the focal point along each axis. The parameters (c_x, c_y) are the offset values of the principal point. Due to inaccurate placement during manufacturing, the center of the image sensor will not necessarily lie on the optical axis. In optimal conditions this offset is zero. Note that now all parameters except the skew are written in pixel units.

2.1.2 Distortion parameters

Optical distortion is a deviation from projection that occurs due the imperfection of lenses and manufacturing. An ideal parabolic lens is mathematically a perfect lens but they are practically not possible to manufacture. Instead spherical lenses are used to approach the ideal parabolic model. These spherical lenses cause radial distortions. This is illustrated in figure 3. Because the lens is not perfectly parabolic, straight lines on an image look deformed. Lines in the center of the image still look straight, but further from the center they are bent. There are two types of radial distortion: negative radial distortion, also referred to as pincushion distortion, and positive radial distortion, also referred to as barrel distortion [9].

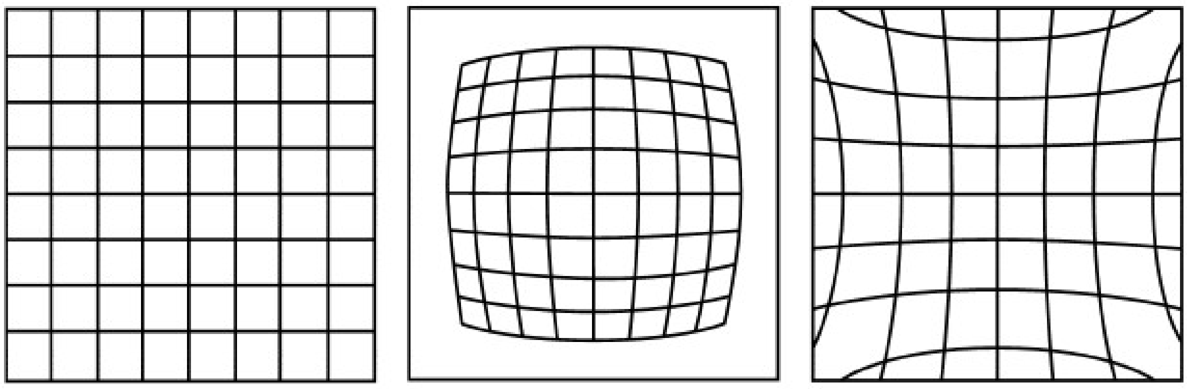


Figure 3: The original grid and the effect of barrel (center) and pincushion distortion (right) [9].

Any distortion can be approximated by Taylor's series expansion. Taylor's expansion approximates the radial distortion coefficients according to the chosen polynomial degree. In theory, one can make a perfect approximation by expanding the series to a polynomial of infinite degree, but for average cameras this is not necessary as it would only consume more computing power and cause numerical instability. For theoretic purpose and later on software implementation, the polynomial is limited to a third degree equation. Highly distorted cameras such as wide-angle or fish-eye lenses can be approximated with a third radial distortion component. The following equations describe the relation between undistorted image coordinates P^U and distorted image coordinates P^D when radial distortion occurs [4]:

$$\begin{aligned} P_x^D &= (1 + k_1 {r_u}^2 + k_2 {r_u}^4 + k_3 {r_u}^6) P_x^U \\ P_y^D &= (1 + k_1 {r_u}^2 + k_2 {r_u}^4 + k_3 {r_u}^6) P_y^U \end{aligned} \quad (7)$$

The factors k_1, k_2 and k_3 are the radial distortion coefficients and r_u the distance between the principal point and the corrected pixel.

Next to radial distortion there is also the possibility of tangential distortion. This type of distortion occurs during manufacturing and is caused by the improper placement of the camera sensor, see figure 4. The adhesive used to connect the sensor with a vertical plane prevents it from being placed exactly parallel to the lens. The tangential distortion equation is as follows, with p_1 and p_2 the tangential distortion coefficients:

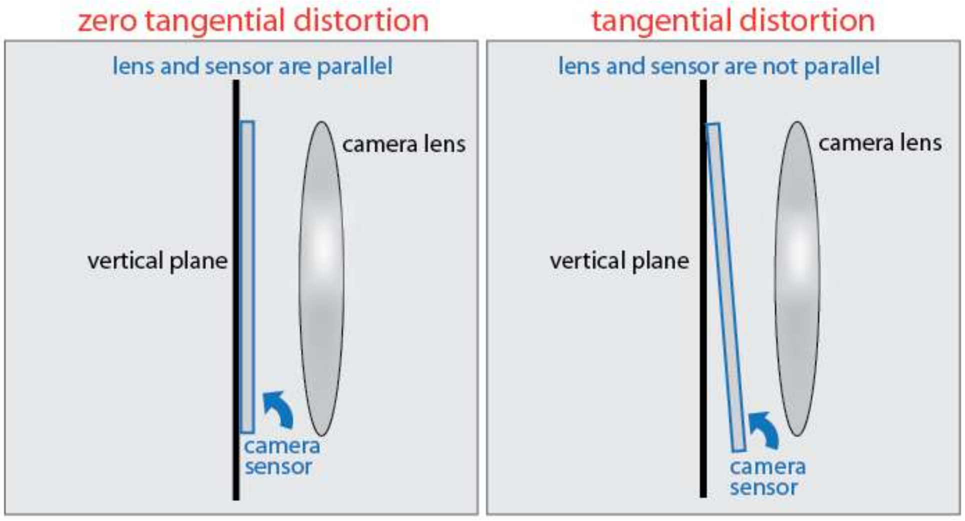


Figure 4: Tangential distortion [10].

$$\left.\begin{aligned} P_x^D &= P_x^U + p_2\left(r^2 + 2P_x^{U^2}\right) + 2p_1P_y^U \\ P_y^D &= P_y^U + p_2\left(r^2 + 2P_y^{U^2}\right) + 2p_1P_x^U \end{aligned}\right| r^2 = (P_x^U)^2 + (P_y^U)^2 \quad (8)$$

The combination of these distortion types of distortion results in the relation between an undistorted point P^U and a distorted point P^D. The total distortion vector is denoted as $[k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$. The following equation is known as Brown's distortion model [11].

$$\begin{aligned} P_x^D &= \left(1 + k_1{r_u}^2 + k_2{r_u}^4 + k_3{r_u}^6\right)P_x^U + p_2\left(r^2 + 2P_x^{U^2}\right) + 2p_1P_y^U \\ P_y^D &= \left(1 + k_1{r_u}^2 + k_2{r_u}^4 + k_3{r_u}^6\right)P_y^U + p_2\left(r^2 + 2P_y^{U^2}\right) + 2p_1P_x^U \end{aligned} \quad (9)$$

2.2 Camera calibration

2.2.1 Rotation matrix and translation vector

To know the pose of an object, the coordinates of this object can be described relative to the camera coordinate system in terms of a rotation matrix and translation vector, as shown in Figure 5.

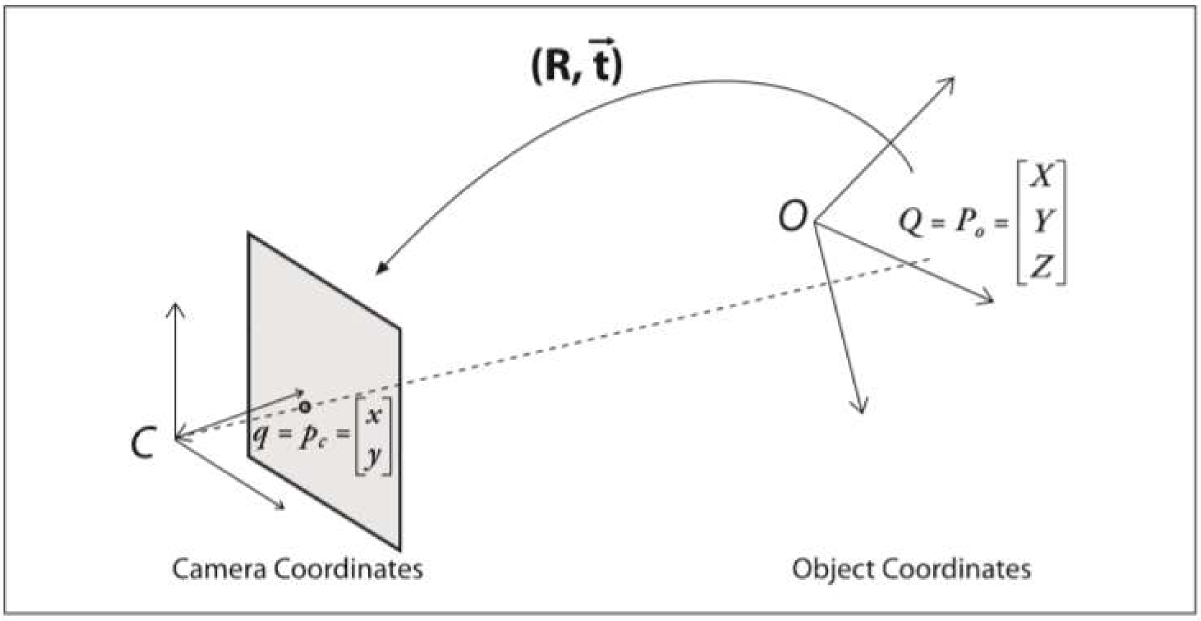


Figure 5: From object coordinates to camera system coordinates: point p is related to point P by applying the combination of a rotation matrix R and a translation vector t [4].

A rotation is equivalent with describing a point's location in another coordinate system. Rotating around a point or an axis is equivalent to rotating the coordinate system by the same angle in the reverse direction. Rotation in the two dimensions is described as a matrix multiplication following the simple trigonometry. Rotation in three dimensions consists of two-dimensional rotation around the main axis while the pivoting axis stays constant. To calculate the total rotation matrix R, the product of the separate rotation matrices $R_x(\psi)$, $R_y(\phi)$, and $R_z(\theta)$ must be performed, each representing the rotation around the x-, y- and z-axes with their respective rotation angles ψ, ϕ, and θ. The total rotation matrix R is denoted as follows [4]:

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}, R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix},$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (10)$$

$$R = R_x(\psi)R_y(\varphi)R_z(\theta)$$

The translation vector is a shift in origin from one coordinate system to another coordinate system so it can be described as an offset from the origin of the first coordinate system to the origin of the new coordinate system. The translation from the world coordinate system to the camera coordinate system is given by this simple equation:

$$\vec{T} = origin_{object} - origin_{camera} \qquad (11)$$

By combining the rotation and translation we get the transformation of a point in the world coordinate system, P^W, to a point in the camera coordinate system, P^C. *R* is the total rotation matrix in the world coordinate system and $\vec{T}$ the translation in the world coordinate system.

$$P^C = R\left(P^W - \vec{T}\right) \leftrightarrow [R\ \vec{T}]P^W \ (12)$$

When substituting the camera intrinsic properties found in (6) in (12), we get the relationship between pixel coordinate points and points in the world coordinate system, with s the scaling factor, $p_z{}^C$:

$$sP^p = AP^C = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[R\ \vec{T}\right] P^W = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} P^W \qquad (13)$$

To solve equation 13, Direct Linear Transform [12] is used. Assume that the used marker is situated in a flat plane. Now for practical calculations a switch is made to a marker coordinate system Ψ_M, instead of the world coordinate system and the z-coordinate in equation 13 is removed. This can be done because all the points are on the marker. The equation is rewritten to:

$$sP^U = [R_M T_M]P^M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} P_X^M \\ P_Y^M \\ P_Z^M \\ 1 \end{bmatrix} \xrightarrow{P_Z^M\ removed} \begin{bmatrix} r_{12} & r_{12} & t_1 \\ r_{22} & r_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_X^M \\ P_Y^M \\ 1 \end{bmatrix} = H \begin{bmatrix} P_X^M \\ P_Y^M \\ 1 \end{bmatrix} (14)$$

The calibration matrix A is estimated during the calibration process, the rotation matrix and translation vector are unknown and must be determined to know the pose of the camera. As the equation above shows, for each known point in the world and their corresponding pixel coordinate, there are two equations with three unknown variables. With six or more points from the real world, there will always be a unique solution. Increasing the number of points will result in more equations and thus a higher accuracy of the estimation [4].

2.2.2 Reprojection error

When validating the quality and the accuracy of the calibration, it is possible to reproject the calculated points onto the marker image. The reprojection error is the distance measured between a saddle-point calculated during calibration and the reprojected saddle-point after calibration. This process is visualized in Figure 6.

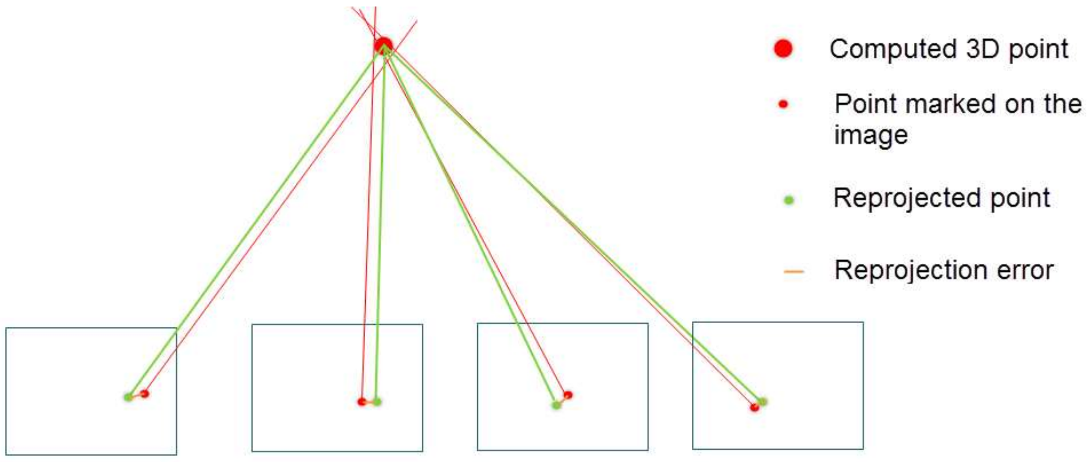


Figure 6: Reprojection error [13].

Each calibration image has its own mean reprojection error, which is the mean reprojection error of all the occurred errors in the image. The overall mean reprojection error is the mean of all the used images combined. The Euclidean distance between a point marked on the image and the reprojected point can be calculated with the following equation:

$$RMS\ error = \sqrt{\frac{\sum_{i=1}^{n}(x_{mpi}-x_{rpi})^2+(y_{mpi}-y_{rpi})^2}{n}} \qquad (15)$$

N is the total amount of detected points, (x_{mpi}, y_{mpi}) the coordinates of the marked points and, (x_{rpi}, y_{rpi}) the coordinates of the reprojected points.

2.2.3 Chessboard pattern and saddle-points

A chessboard is an interesting calibration marker. It has a pattern of alternating squares which all have the same dimension. The transition from two black square corners and two white corners is called a saddle-point. Other patterns, for example a pattern of flat circles, can be used to calibrate a camera if the dimensions are known. Multiple pictures of the chessboard are taken to gather enough equations to solve the calibration matrix. This is represented in Figure 7.

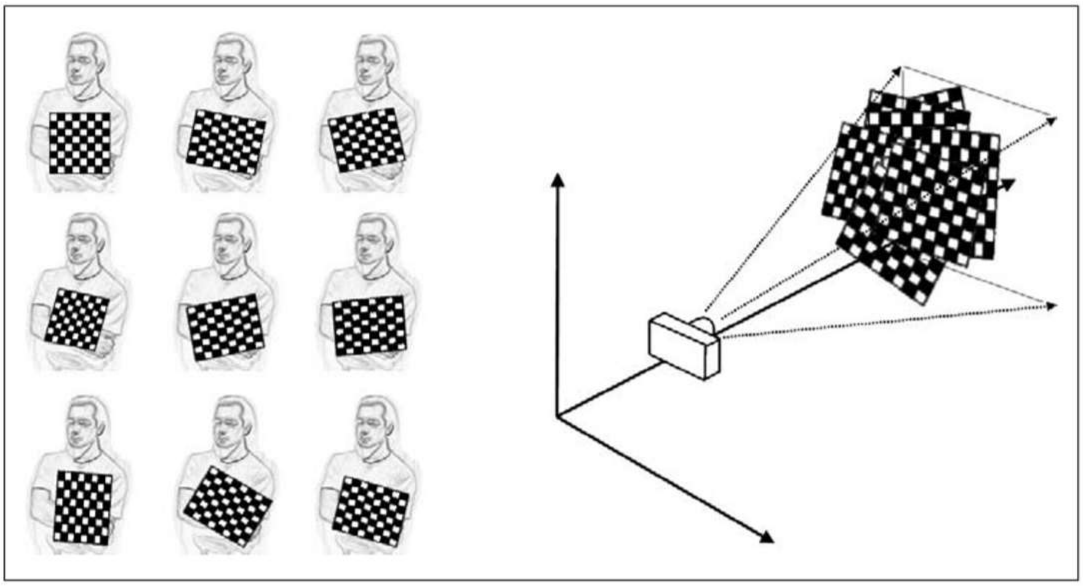


Figure 7: Calibration by hand with a chessboard pattern [4].

A chessboard pattern with x × y squares has (x-1) × (y-1) saddle points. The saddle points of the chessboard can only be approximated as the resolution of the camera is limited. The closer a saddle-point can be determined to a single pixel, the more accurate the camera calibration. Saddle-points are detected as the transition of the corners in the square pattern following the method of Harris or Noble. Sub-pixel positions are calculated by fitting and interpolating the quadratic functions according to the intensity of the surrounding profile [4].

2.2.4 Planar homography

In computer vision science,the planar homography is the definition of the projective mapping from plane to another plane. In case of a camera it is the mapping of two-dimensional planar surface points to the image sensor of the camera. The mapping can be defined as follows:

$$Q = [X \quad Y \quad Z \quad 1]^T$$
$$q = [x \quad y \quad 1]^T \tag{16}$$

The action of homography is then simply expressed as:

$$q = sHQ$$

The coordinates of the 3D world object are described by Q and the mapped points on the image sensor correspond to q as shown in Figure 8. The points are mapped from one plane to another according to the homography matrix H. the factor s is an arbitrary scale factor that is conventionally factored out of matrix H. H can be split up in to two parts: one containing the physical transformation and one containing the projection along with the camera intrinsic properties.
Because the chessboard image is flat, it can be assumed that the plane Z = 0. Now it is possible to observe that H is a 3×3 matrix [4]:

$$q = sHQ = sAW \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [r_1 \quad r_2 \quad t] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{17}$$

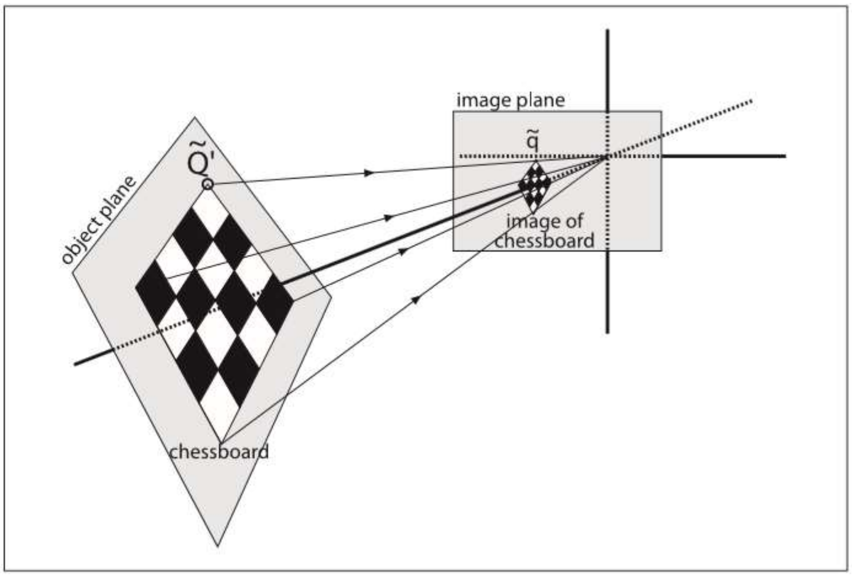


Figure 8: Planar homography [4].

During the calibration process, the homography matrix and intrinsic camera matrix are calculated for each image. The rotation matrix and translation vector are described by six unknown parameters. More images provide more equations to accurately calculate the intrinsic parameters. OpenCV implements different methods to calculate the intrinsic camera parameters and distortion factors. The Zhang method and Brown method are respectively used to calculate these parameters [4].

2.2.5 Rodrigues Transformation

For mathematical purposes, the rotation properties are represented in a 3-by-3 matrix. This way the matrix can be conveniently multiplied or divided with a vector. This representation is not very intuitive and human readable. The Rodrigues transform captures the relation between the magnitude of the vector and the magnitude of the rotation. If r represents the three-dimensional vector r = [r_x r_y r_z], then the magnitude of the rotation θ is calculated as θ = |r|. Moving on from these equations we can transform the matrix representation with the following equation [4]:

$$R = \cos(\theta)\,.I + (1 - \cos(\theta)).rr^T + \sin(\theta)\begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ r_y & r_x & 0 \end{bmatrix} \qquad (18)$$

2.3 Stepper motor

2.3.1 Introduction to stepper motors

A stepper motor is a synchronous, brushless DC motor that converts current into mechanical force and as a result of this movement. A stepper motor divides its full revolution into several discrete steps, hence the name. Most basic stepper motors commonly have a step angle of 1.8°, meaning that a full revolution of 360° degrees takes 200 steps. It moves one step at a time and each step is of the same size with a relative uncertainty according to the step. The inside of the permanent magnet stepper motor is comprised of multiple coils that are organized in phases that drive a permanent magnet. Letting current run through the coil will result in a magnetic field that will rotate the magnet and therefore rotate the motor. The current to control the motor is applied in pulses. The frequency of these pulses will determine the speed of the rotation. Because the speed of rotation and the steps are easily controlled, a stepper motor is an excellent choice for precision motion control applications, both industrial and commercial. Several advantages are the high torque at low speeds, the reliability of operation, the simple robust design and the low cost. The precision and repeatability of each step is determined with a relative accuracy of the full step and this relative accuracy is non-cumulative from one step to the next step.

2.3.2 Types of stepper motors

The three basic types of stepper motors are the permanent magnet stepper, the variable reluctance stepper and the hybrid stepper, the latter being a combination of the first 2 steppers. The differences in construction and operation are explained below.

Permanent magnet stepper motor

Permanent magnet steppers use a permanent magnet in the rotor, called a canstack rotor. Figure 9 shows a representation of this type of stepper on the left, with a closeup of the canstack rotor on the right.

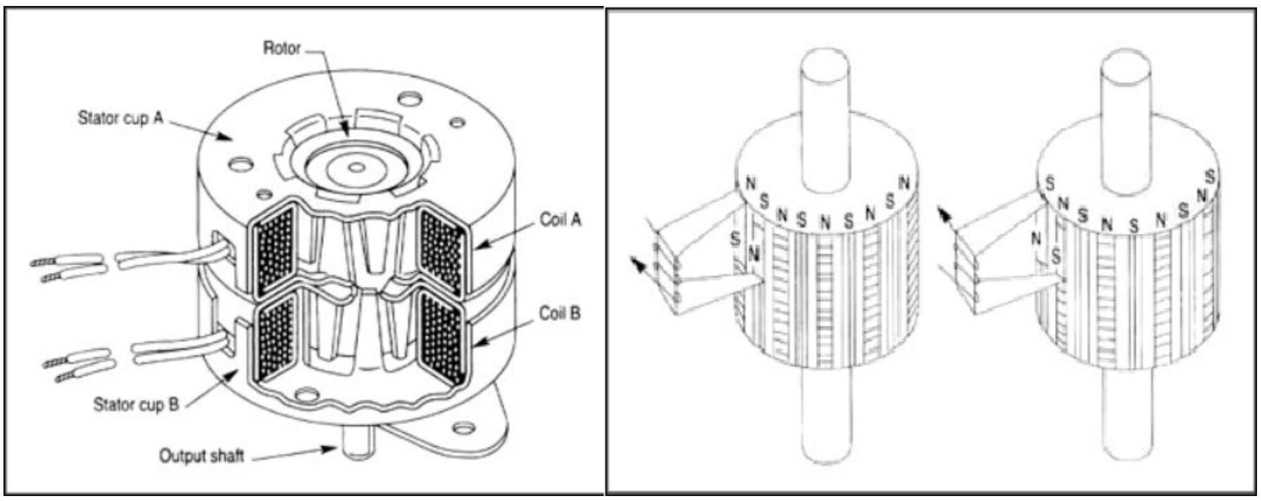


Figure 9: Left the full overview of a permanent stepper, right the canstack rotor shaft [14].

As shown in the figure, a permanent magnet motor can have multiple rotor coils or windings. The full revolution of this type of stepper depends on the number of rotor teeth, the number of stator poles and stator teeth. For example, a basic permanent magnet stepper has 50 rotor teeth and 8 stator poles each with 5 teeth, resulting in 40 stator teeth. With these specifications, the stepper motor will do 200 steps to make a full revolution. A full revolution corresponds to 360° thus dividing the revolution in 200 steps results in a 1.8° step angle. The key feature of this type of stepper is the lack of electrical brushes that are normally present in rotating shafts. These brushes in traditional motors provide the contact between the stationary and rotating parts but are not necessary here because of the magnetic properties. This saves maintenance and replacement costs, as these brushes have a limited lifetime [13].

Variable reluctance stepper motor

A variable reluctance stepper motor does not involve a permanent magnet but it induces non-permanent magnetic poles on the ferromagnetic rotor. Because the magnet is not permanent, the induced magnetic field strength can vary. Modern reluctance motors can deliver high power and torque but usually at the disadvantage of a high torque ripple and high noise at lower application speeds. Torque ripple is the variation between the maximum value of torque and the minimum value of torque generated during one full revolution. Generally, variable reluctance motors are optimized for high speed and continuous rotation with a minimum amount of torque ripple [3]. Torque ripple can be minimized using microprocessors or integrated controllers that perform the necessary calculations to switch windings at the exact time. The winding current must be monitored and altered in real time to achieve desirable ripple. This gives strict requirements and additional cost, making this type not very suitable for low volume operations due to the complexity.

2.3.3 Unipolar and bipolar

Apart from the distinction in construction, a stepper motor can also be categorized according to number of coils and the winding arrangements. We discuss the difference between unipolar and bipolar motors, the two available options for two phase stepper motors.

Unipolar stepper motors have a lead that is common to both coils, as indicated by the green lead in Figure 10. This center tap will always be negative or ground voltage level in respect to the phase coils. The schematic in Figure 10 shows a 2-phase motor. Each coil can generate a magnetic field which will move the rotor. The magnetic pole which indicates the direction can be easily changed by turning one phase of and the other on. A unipolar driver circuit is relatively easy since it can be accomplished with only transistor logic. Because only one phase is active at a given time, the torque will be less in comparison to a bipolar stepper motor where all the coils are active during the movement of the rotor.

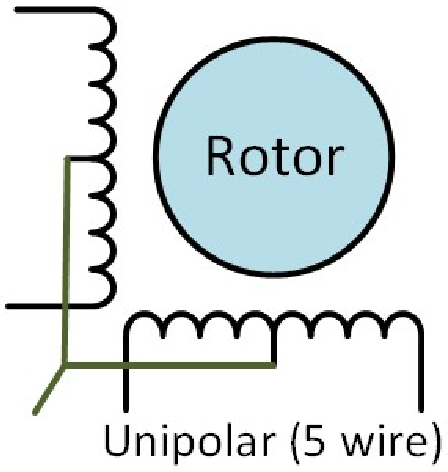


Figure 10: Simple schematic of a unipolar stepper motor [15].

Bipolar stepper motors do not have a common lead wire. They have a single winding per phase. This gives the advantage that all phases are active during rotation and thus the motor can deliver more power. The disadvantage is the need for a more complicated driving circuit to reverse the magnetic pole. This is generally done with a H-bridge configuration to reverse the flow of the current through the phases [15].

2.3.4 Stepper motor linear motion

A stepper motor will often be attached to a belt and pulley to change the rotational movement of the shaft into a linear motion. The belt is often referred to as a timing belt, since the number of teeth and the pitch between them determine the synchronization of the movement. Figure 11 indicates the pitch P, the total height of the belt with teeth H and the height of the teeth T.

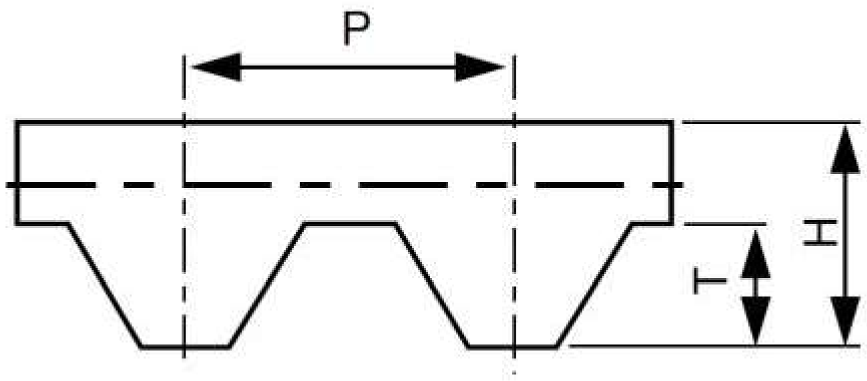


Figure 11: Timing belt dimensions [17].

To determine the linear motion of the stepper we use the following equation. S_{rev} is the number of steps per full revolution of the motor, f_m is the microstepping factor, p is the pitch of the timing belt and N_t is the number of teeth on the pulley that is attach to the motor shaft. Note that this equation does not account for possible mechanical backlash [18].

$$\frac{s_{rev} * f_m}{p * N_t} = \frac{\#\, steps}{mm} \qquad (19)$$

2.4 Microsteppping

Microstepping is a method to move the stepper motor more smoothly and in smaller steps than full- or half-step size. By altering the stator flux of a stepper, whole integer divisions of a full step can be reached. A smaller step size means less vibration caused by the mechanical movement of the stepper. If the vibration is kept at a minimum, the linked audible noise will also be greatly reduced and resonance problems solved. The smaller step size automatically means a smaller step angle and more accurate positioning. For a unipolar stepper motor, microstepping can be achieved by changing the duty cycle of the current in the coils. This is shown in Figure 12. The change of duty cycle corresponds to a sinusoidal signal [19].

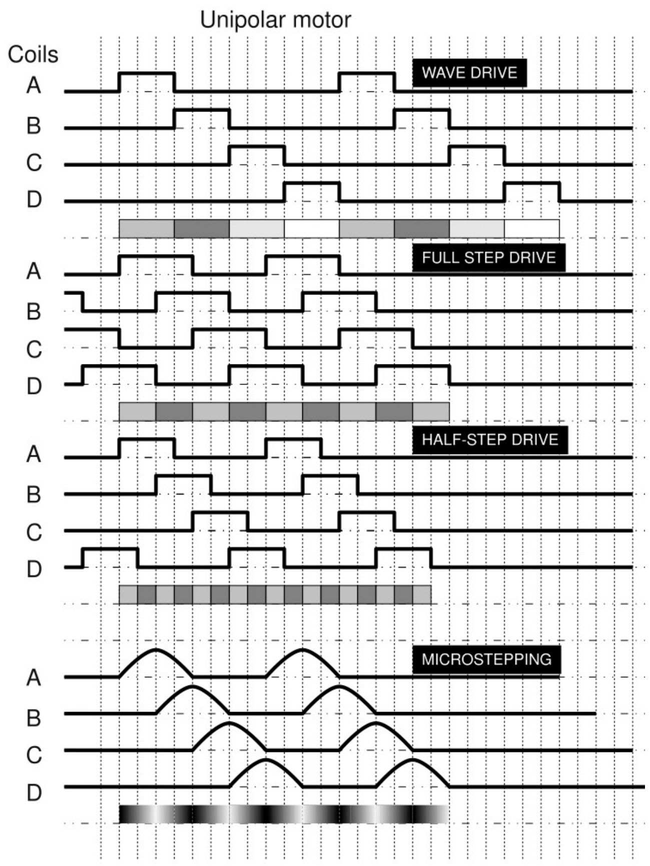


Figure 12: Different drive modes for unipolar stepper [20].

The key aspect of microstepping is the increase of the step resolution, going beyond the inherent specifications of the manufacturer. By using complex driving circuitry, this increase in dynamic range can also be achieved, but this has a greater cost as these systems are often not cheap. However, a major disadvantage of applying this technique is the loss of torque. Each consecutive increase in the number of micro steps results in a loss of holding torque. This loss in torque is the result of the change in duty cycle in the coils. A smaller duty cycle limits the power output to the phases. This is illustrated in Figure 13. After a factor of 16 micro steps per full step, the holding torque is less than 10% of the full power. Certain applications that need to maintain optimal power output may therefore not benefit from microstepping.

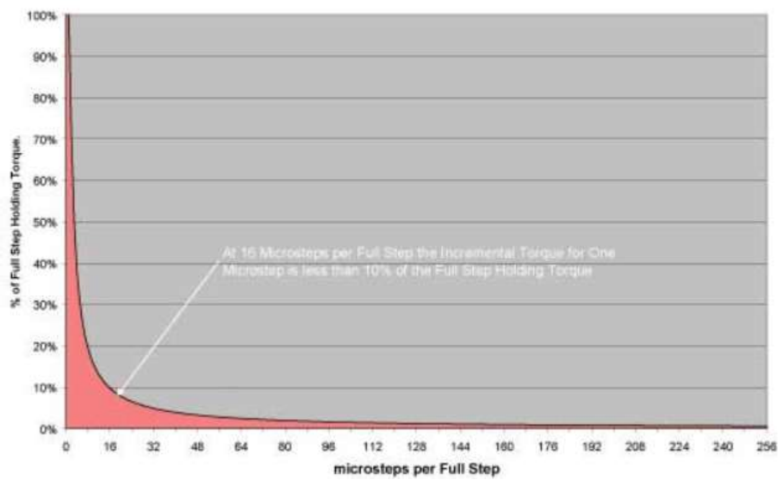


Figure 13: Holding torque loss in microstepping application [17].

Chapter 3

Materials and methods

3.1 Hardware

3.1.1 HP Scanjet 2400

The Scanjet 2400 is used as a base for the test setup. It is an older type of a digital flatbed scanner made by Hewlett-Packard. The scanner is shown intact in Figure 14a and broken apart in Figure 14b. Notice the unipolar stepper motor found in the left corner of Figure 14b. The scanning head is attached with a pulley belt to the stepper motor, which acts as a transducer to perform the movement along the fixed rod. Additional electronics have been removed from the scanner since they provide no additional functions in this case. Exact specifications of the stepper motor are not available, neither by HP or by the manufacturer Epoch Electronics. This setup is chosen because of the small error that can occur the mechanical movement over the fixed rod. The rod is manufactured with high precision.

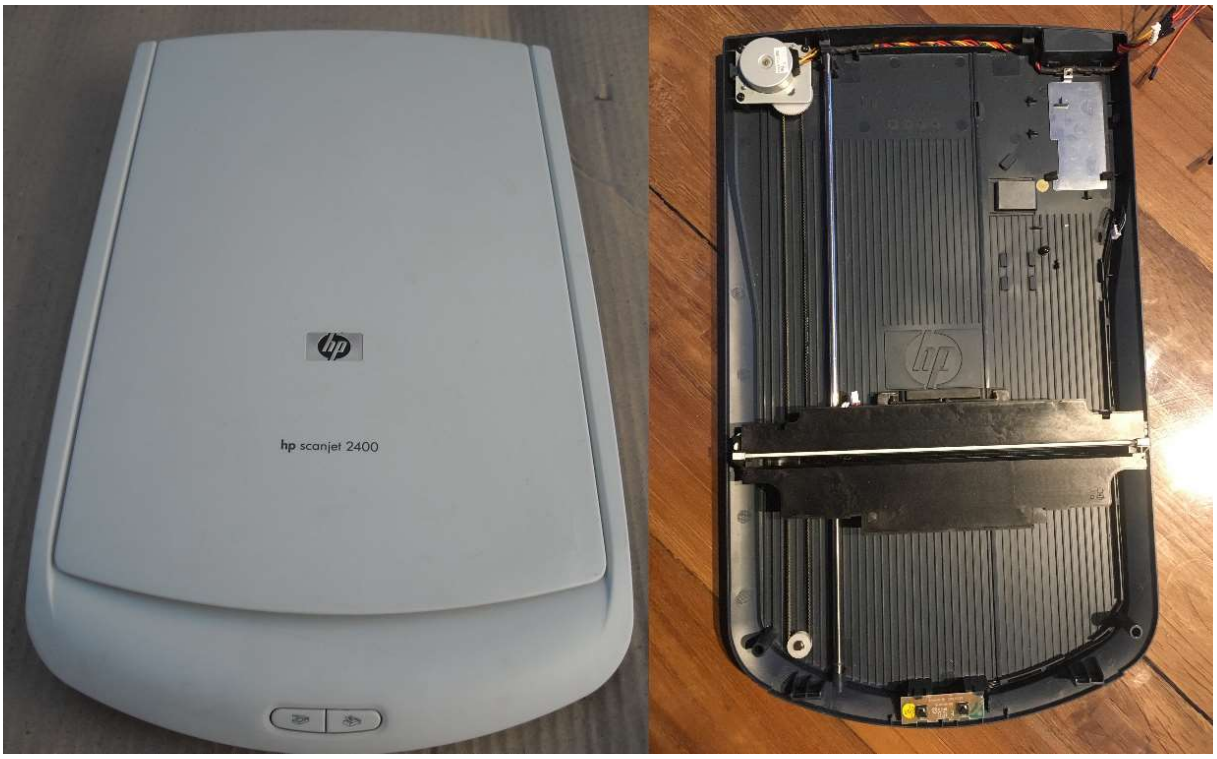


Figure 14: Scanjet 2400 as setup base a) Left side original casing, b) broken apart

3.1.2 Arduino Uno R3

The Arduino Uno is a microcontroller board based on the ATmega328P. It contains 14 digital input/output pins of which 6 can be used as PWM outputs, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button, see Figure 15 [21]. It can be powered over USB or through an AC adapter. Arduino has a widespread community and many projects have already been released using their microcontroller. Additional hardware and features are available at the Arduino store.

Figure 15: The Arduino Uno [21].

The Arduino can be programmed through its own software IDE. The ATmega328 on the Arduino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board.

3.1.3 Stepper motor drivers

ULN2003A Unipolar Driver

The ULN2003A contains seven pairs of high-voltage, high-current Darlington transistor arrays, used to power the coils in the stepper motor, see Figure 16. The Darlington arrays contain outputs with common-cathode clamp diodes for switching inductive loads. A single Darlington pair has a current rating of 500 mA though higher current capability is possible though parallel wiring. The base resistors allow for direct operation from CMOS devices and therefore it has found use in many applications such as relay drivers, display drivers and in this case as a stepper driver. There are 6 inputs, 4 to drive the coils and 2 for the power input. There are 5 outputs, 2 pair for the coil windings and 1 common tap. This driver circuit was used to drive the unipolar stepper motor of the HP Scanjet 2400 and to test the accuracy of micro stepping against another driver circuit. In this case, custom code had to be written to adjust the duty cycle in the coils windings to achieve micro stepping, since no existing libraries were available [22].

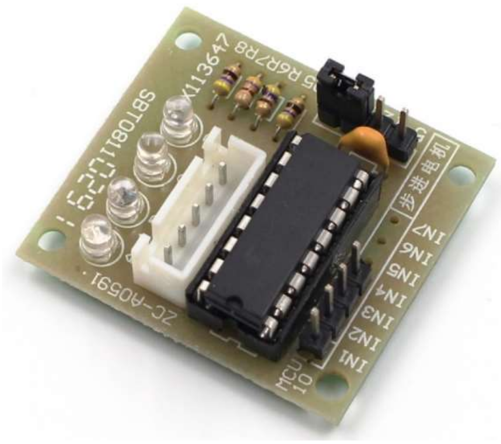


Figure 16: ULN2003A Driver [22].

AdaFruit Arduino Motorshield V1

The Adafruit Motorshield kit is a shield upgrade for the Arduino Uno. It is displayed in Figure 17. It can drive up to 4 bi-directional DC motors, with individual speed selection, or 2 stepper motors, in single coil, double coil, interleaved or micro stepping, mode at a time. The L293D chipset contains quadruple high-current half-H drivers, each capable of providing bidirectional drive of up to 1A current at voltages ranging from 5V to 35V. It can handle inductive loads and other applications that need a high voltage and high current supply. The central IC is a 74HC595. It is an 8-bit serial in, serial/parallel-out shift register with an additional store register that handles the serial-to-parallel conversion of the signals. Preexisting libraries make it easy to control a stepper motor with the shield. By modifying the underlying header files, it is possible to accomplish the needed micro stepping function without further adjusting the Arduino code [23].

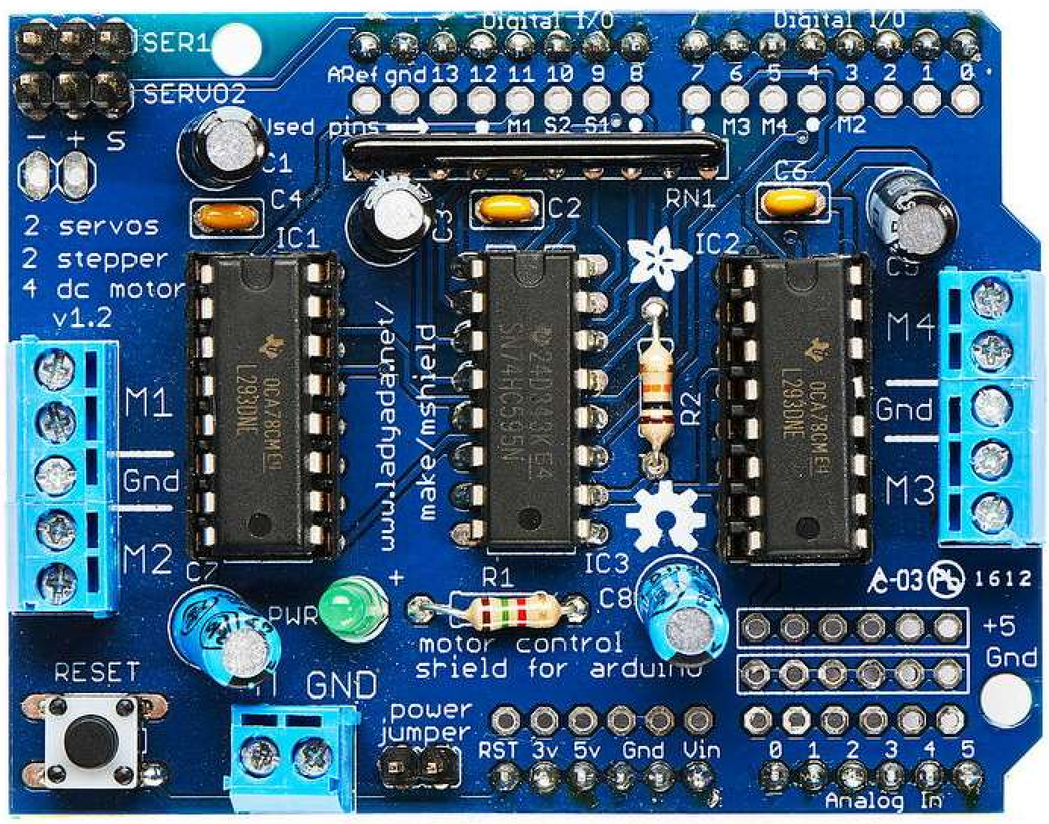


Figure 17: Adafruit Arduino Motorshield V1 [23].

3.1.4 Camera

The Denver ACT-8030W is a wide-angle, full-HD, WIFI action camera. It can film 1080p at 60 frames per second or 720p at 120 frames per second, has an 8 megapixel CMOS-sensor, a wide-angle lens of 170° and Wifi functionality. It has the built-in function to take still pictures with a resolution of 16MP (4640x3480), 12MP(4000x3000), 8MP(3264x2448) or 5MP (2592x1944), features a slow-motion and time-lapse function and an JPG image format. The camera is shown in Figure 18.

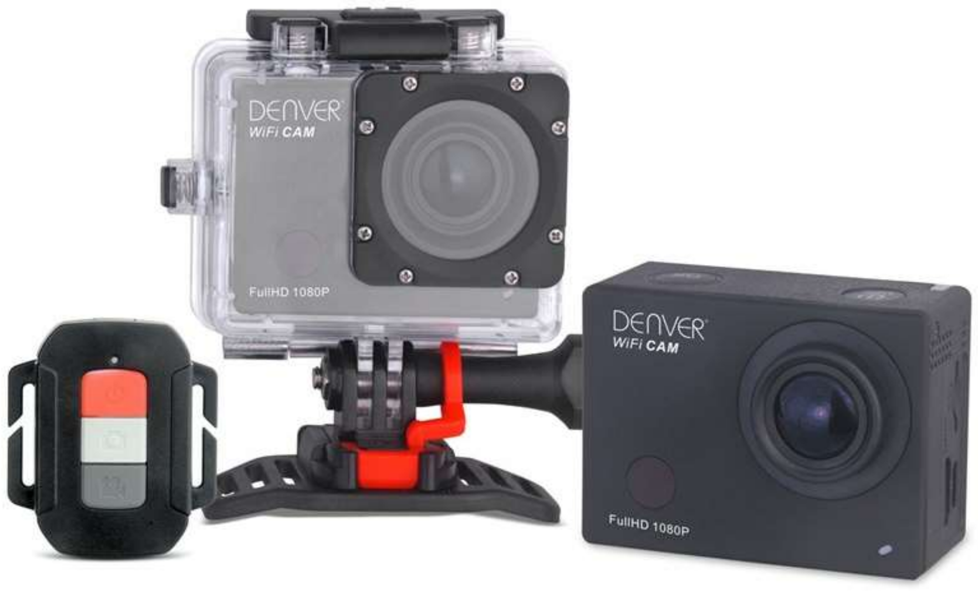


Figure 18: Denver ACT-8030W [23].

This Denver camera is used for its capability to communicate over WIFI. This makes it possibly to take pictures handsfree and without disturbing the mounted camera as taking a picture by pressing the camera button itself would cause a minimal but noticeable position shift. Other benefits are the handlebar and helmet mount which help to position the camera in a sturdy position and allowed for different positioning angles. The specifications are shown in Table 1 [24].

Tabel 1: Specifications of the Denver ACT-8030W [24].

Features	Denver ACT-8030W
Focus	Diagonal: 100° / Horisontal 86° / Vertical 46°
Video Resolution	1080P (1920x1080) @ 60fps/30fps 720P (1280x720) @ 120fps/60fps
Video Format	.MOV (H.264)
Image Resolution	16MP (4640x3480) 12MP (4000x3000) 8MP (3264x2448) 5MP (2592x1944)
Image Format	JPG
Internal memory	1GB DDR3
External memory	Micro SD card up to 32GB
Lithium battery	1200 mAh
Playback mode	HDMI, Wifi or USB

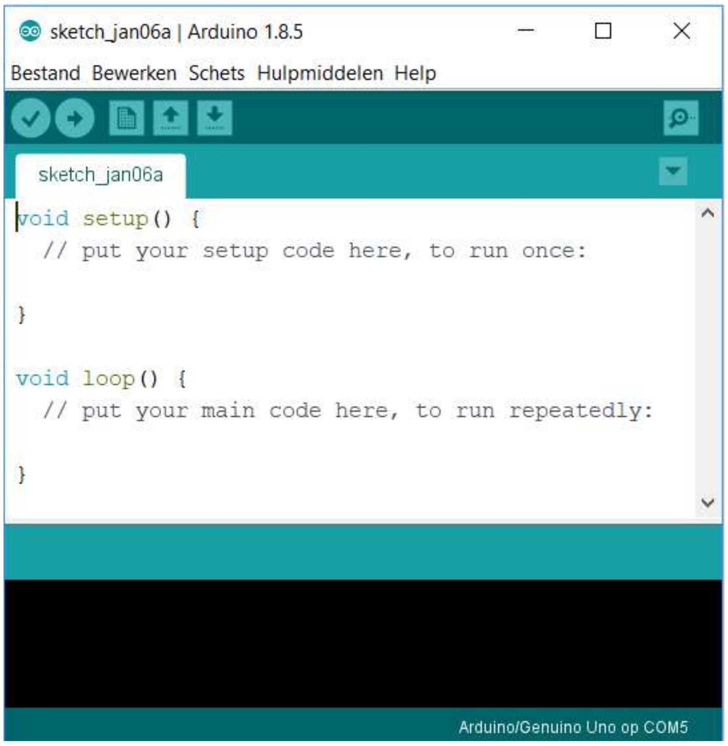
3.2 Software

3.2.1 Arduino IDE

The Arduino integrated development environment is an open-source tool used to develop, maintain and program the code that runs the microcontroller. The Arduino language is a dialect with a lot of features of the programming languages C and C++. The IDE is a cross-platform application written in Java and is based upon the Processing and Wiring open-source languages. The IDE supplies a similar software library as the Wiring project, which provides many common input and output procedures as well as predefined source code examples.

It has the features of a modern code editor such as text cutting and pasting, searching and replacing text, automatic indenting, syntax highlighting, and provides the mechanism to compile and upload programs to the microcontroller with a simple click of a button. Serial input and output can be displayed on the message area located at the bottom. Errors that could arise are also displayed here with their own error code [21].

A program written with the IDE for Arduino is called a sketch. Sketches are stored as text files with the file extension ".ino" and older versions are stored with the file extension ".pde".
A sketch must always contain the two functions, the setup and the loop. The setup function is run one time once the Arduino is powered up or reset. It initializes the variables and pin modes and included libraries. The loop function keeps executing the code inside the function repeatedly until the Arduino is powered off or reset. An example of an empty sketch with both loops is given in Figure 19.


sketch_jan06a | Arduino 1.8.5
Bestand Bewerken Schets Hulpmiddelen Help
sketch_jan06a
Arduino/Genuino Uno op COM5

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 19: Empty Arduino sketch in Arduino IDE

3.2.2 OpenCV

OpenCV stands for Open Source Computer Vision. It has more than 2500 optimized algorithms for computer vision and machine learning with real-time application. It is free to use under the BSD-license and cross-platform, meaning that it can run on almost all modern operating systems. The goal of the OpenCV project is to advance research by providing optimized code and infrastructure. It disseminates knowledge and structure for current and future developers to maintain readable and transferable code.
The libraries are portable and performance optimized but the underlying code remains property of the OpenCV foundation. The libraries are natively written in C++ and the primary interface is in C++. There are bindings and API's for several other languages such as Python, Java and MATLAB. All calibration, image comparison and undistorting algorithm implementations of the OpenCV libraries in this thesis are implemented through the MATLAB binding interfaces [25].

Figure 20: OpenCV Logo [24].

3.2.3 Java

Java is class-based, object-oriented programming language developed by James Gosling at Sun Microsystems, later acquired by Oracle Corporation. As of 2016, it is one of the most used and popular languages for client-server web applications and it serves as the native language for Android development. Java code is compiled into bytecode that can run on any platform with a Java virtual machine and is therefore portable to many architectures. The language is heavily influenced by C and C++.
The GUI that controls the serial communication between the stepper motor and the microcontroller is written in java. Java is chosen because of its cross-platform portability and intuitive GUI designer.

Chapter 4

Design results

4.1 Calibration pattern applicator

The calibration pattern applicator was cut out of a 3mm medium-density fiberboard with a Trotec 100R lasercutter, see Figure 21. It consists of two 224 mm × 67 mm baseplates, each with cutouts over the location of the movement axis. The first plate contains cutouts for an angle positioning of the pattern applicator at ±45° and ±90°. The second base plate has cutouts for the angles ±90° and ±20°. There are two pattern applicators, a small one with a surface of 60 mm × 60 mm and a large one with a surface of 140 mm × 120 mm. Each pattern marker is fortified with a wedge to increase stability. The slots have an area of 9,84 mm × 2,86 mm to tightly fix the 10 mm × 3 mm protruding stubs in place. Figure 22 shows the calibration pattern applicator installed on the test setup.

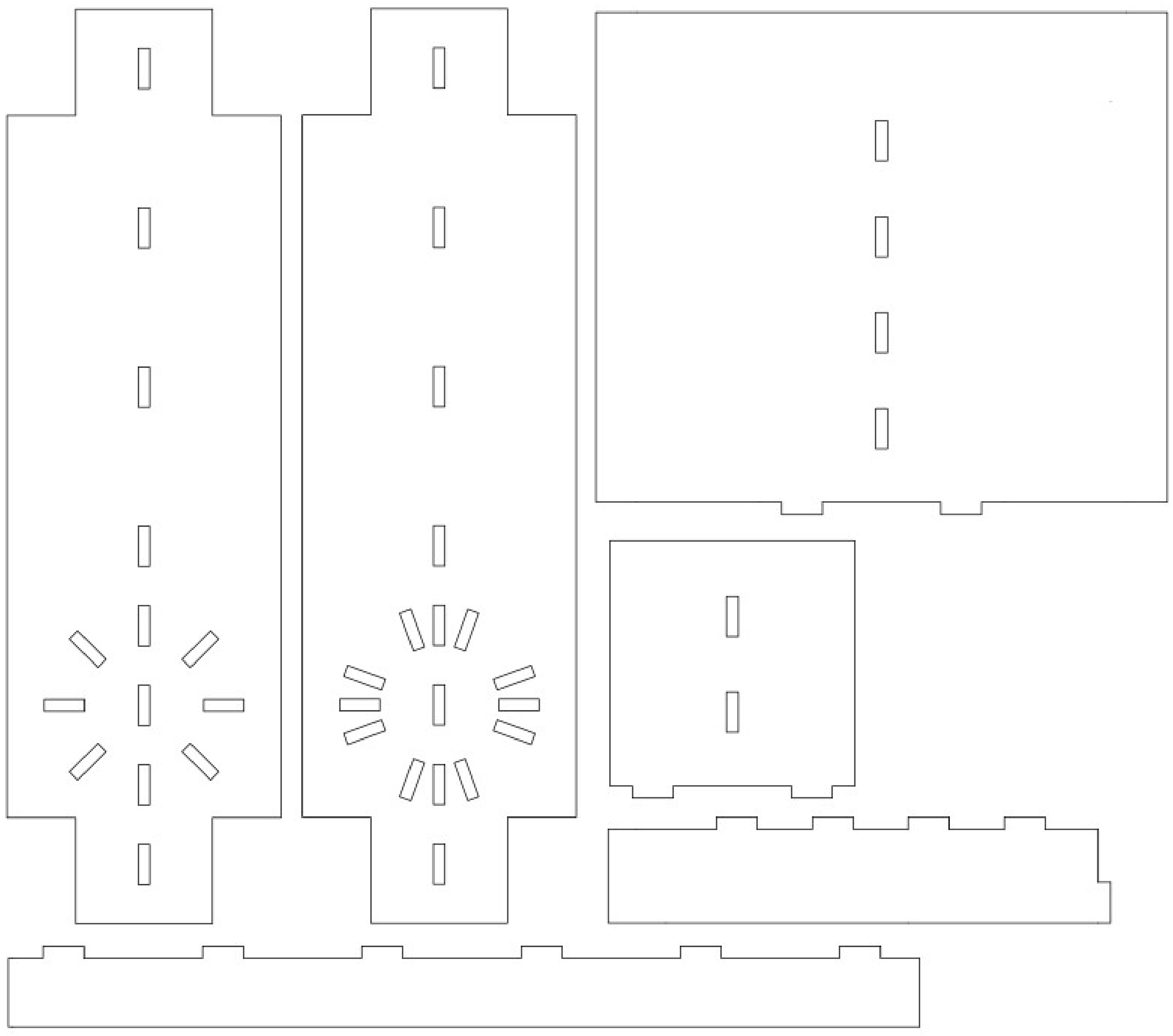


Figure 21: Calibration pattern applicator made from MDF.

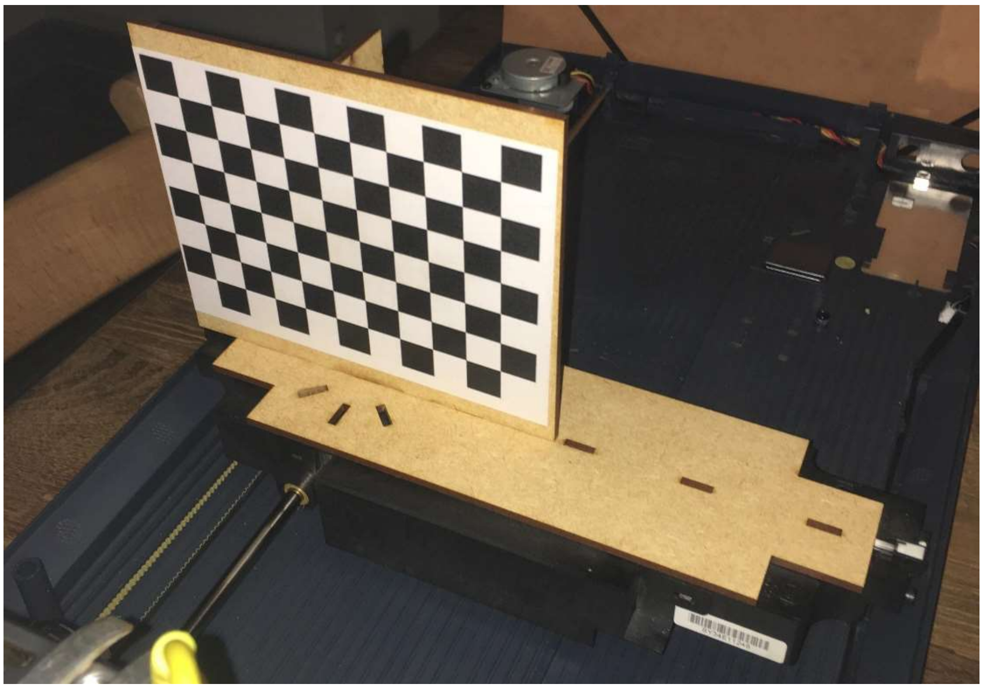
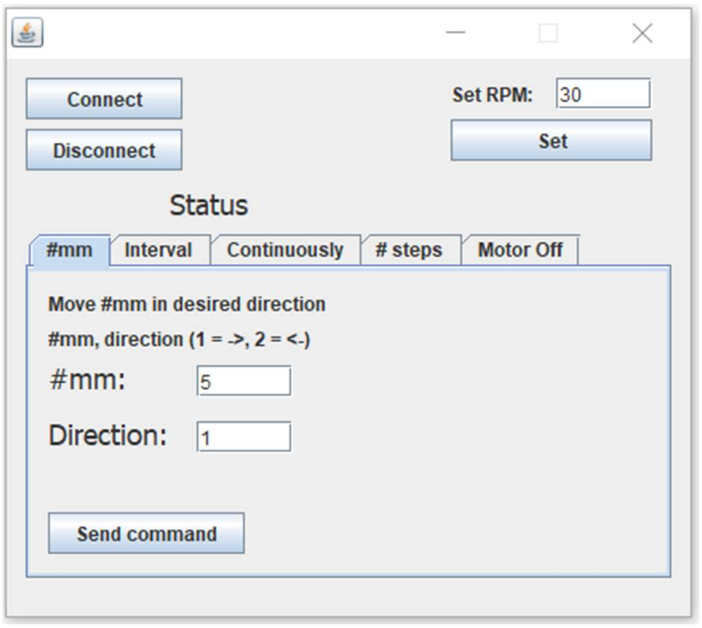


Figure 22: The pattern applicator set on the moving scanner head.

4.2 Java Graphical User Interface

A graphical user interface is designed to control the stepper motor. An Arduino only accepts serial input and output communication in formatted text through its own IDE. The JAVA GUI provides a more efficient and easier way to send the serial commands. There are three general buttons always present on the interface. The first button is to establish the serial connection between the Arduino and the Java GUI. A status label returns if the action is successful or unsuccessful. The second button is the disconnect button to end the connection. The third button will adjust the speed of motion of the stepper motor as it sets the rounds per minute.

The first protocol, displayed in Figure 23, moves the calibration pattern to the entered distance in the chosen direction. The distance is entered in millimeters and the direction is entered as an integer, 1 being the movement towards the stepper motor and 2 being the movement away from the stepper motor.



Set RPM: 30

Move #mm in desired direction

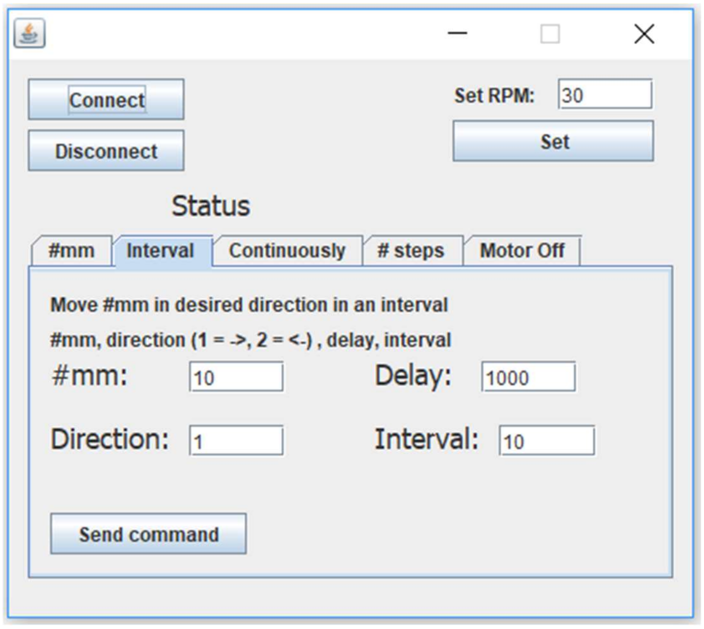
#mm, direction (1 = ->, 2 = <-)

#mm: 5

Direction: 1

Figure 23: Java GUI first panel

The second protocol moves the calibration pattern in an interval with a selected delay time. The pattern will move to the entered distance, wait for the delay to pass and will move again. The number of times pattern should move, is entered in the interval box, Figure 24.



Set RPM: 30

Move #mm in desired direction in an interval

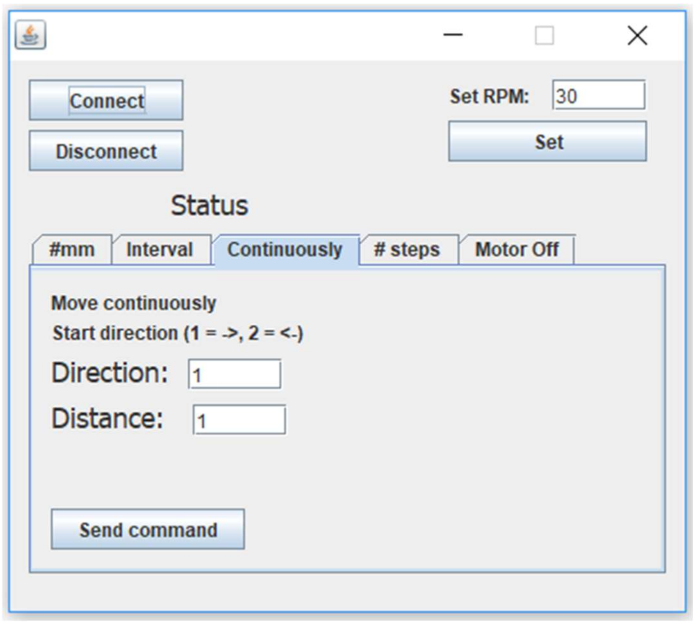
#mm, direction (1 = ->, 2 = <-) , delay, interval

#mm: 10 Delay: 1000

Direction: 1 Interval: 10

Figure 24: Java GUI second panel

The third protocol will continuously move the pattern back and forward for a chosen distance, see Figure 25. The fourth protocol acts the same as the first protocol, except that the movement is done in whole iterations of full step values. When the stepper motor moves in millimeter values, it is possible that the motor will stop its movement in between two micro steps of a full step. Since this is not a natural state for a stepper motor to be placed in, this protocol ensures that only full steps are made. If the micro stepping factor is equal to 16, the smallest amount of movement is 16 micro steps.



Move continuously

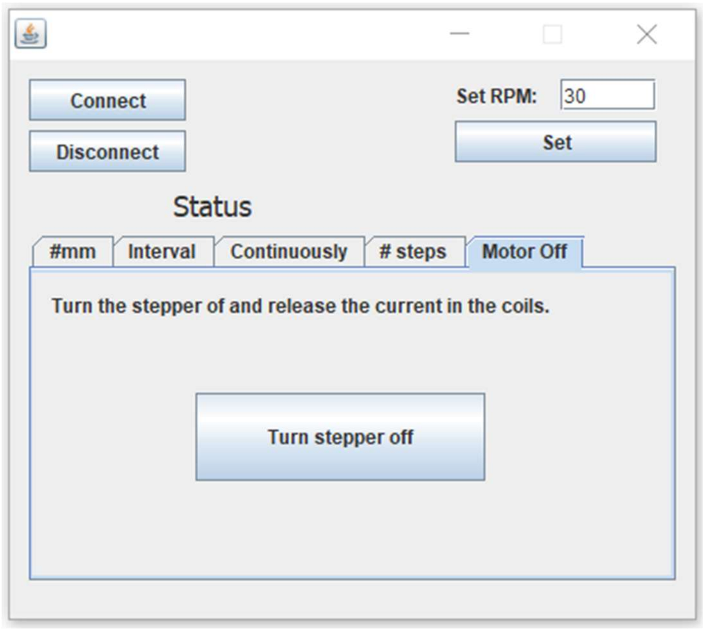
Start direction (1 = ->, 2 = <-)

Direction: 1

Distance: 1

Figure 25: Java GUI third panel

The last protocol is simply there to shut of the motor and de-energize the coils, see Figure 26.



Turn the stepper of and release the current in the coils.

Figure 26: Java GUI last panel

Chapter 5

Measurements

5.1 Stepper motor accuracy

5.1.1 Introduction

The measurements in this chapter are done to determine the most effective way to drive the unipolar stepper motor. Two stepper driver circuits are tested against each other, the Adafruit Arduino Motorshield V1 and the ULN2003A. A bipolar stepper driver, the EasyDriver stepper motor driver, is also tested because the available motor could be driven in bipolar mode. However, the results are non-linear and the bipolar mode caused the motor to overheat, possibly damaging it if exposed for longer periods of time. Measurements done with the EasyDriver are therefore excluded.

5.1.2 Experimental setup

The stepper motor provides the translation motion along a fixed axis of the Scanjet 2400. The accuracy of this movement is measured with the setup shown in Figure 27.

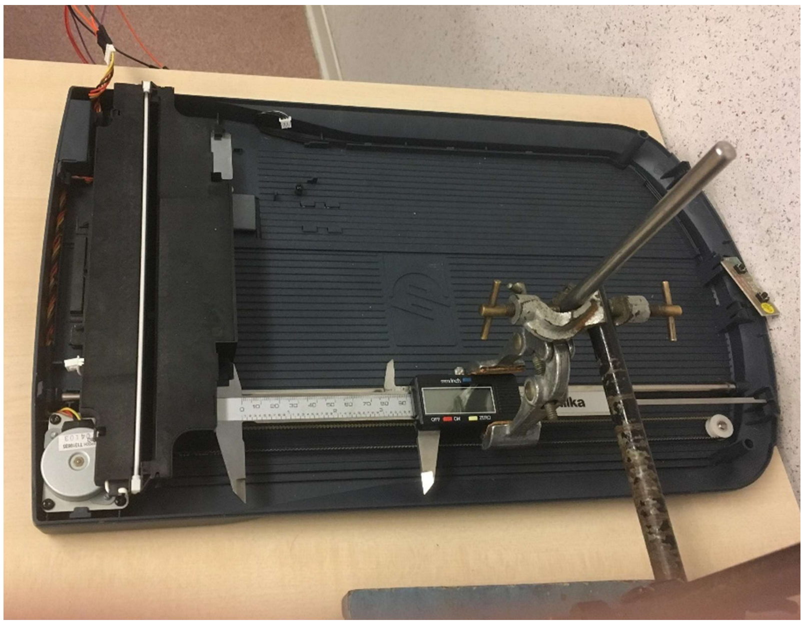


Figure 27: Setup to measure accuracy of movement with a digital caliper.

The stepper driver circuits are powered with the LASCAR DC power source around their voltage maximum to assure optimal working. For the ULN2003A, the maximum input is 12V and for the Adafruit Motorshield, the maximum input is 20V. The digital caliper was placed as perpendicular as possible over the movement axis to minimize the measured error. The caliper has length of 150 mm and is used in the

5.1.3 Test results

A 100 distance measurements are made for each of the micro stepping factors. This is done for a distance of 10 mm and a distance of 100 mm using the ULN2003A driver. The reduction in variance for each consecutive factor is shown in Figure 28 for respectively 10 mm and 100 mm. A full step corresponds to 0,67 mm. For each consecutive micro step, this full step can be multiplied by the micro stepping factor to calculate the size of the micro step.

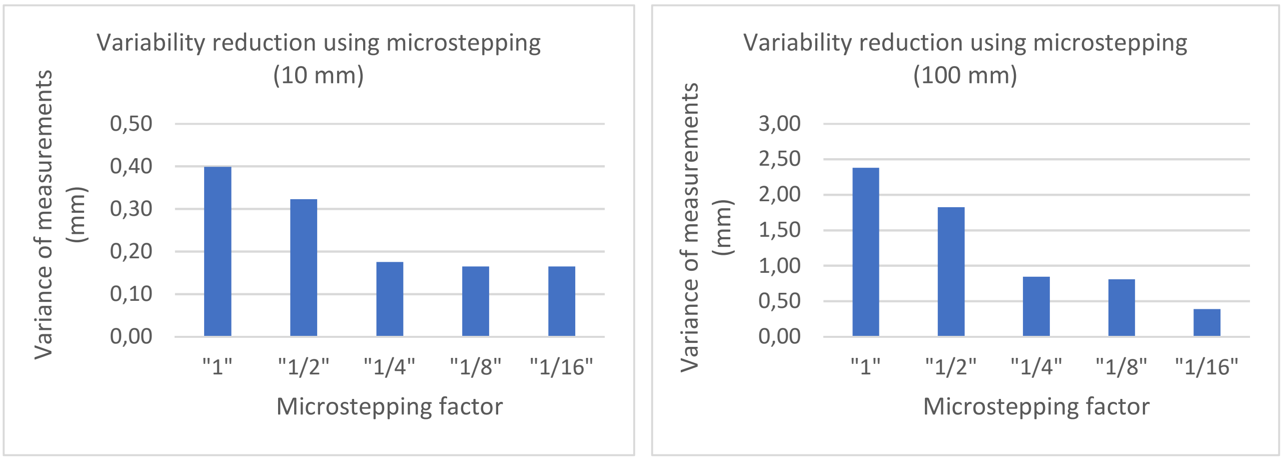


Figure 28: Reduction in variance by increasing the micro steps

The overall measurements of both the ULN2003A and the Adafruit Motorshield for micro stepping factors “1/8” and “1/16” are plotted in Figure 29 for a distance of 10 mm. Both circuits are driven in micro stepping mode with a speed of 30 rounds per minute with a DC power supply of 12V. The Adafruit Motorshield is driven with the available C libraries and for the ULN2003A, custom code is written to perform microstepping. For the ULN2003A, a mean of 9.98 mm with a standard deviation of 0.04 mm for 8 micro steps per full step and a mean of 10.00 mm with a standard deviation of 0.04 mm for 16 micro steps per full step are measured. For the Adafruit Motorshield, a mean of 9.99 mm with a standard deviation of 0.05 mm for 8 micro steps per full step and a mean of 10.00 mm with a standard deviation of 0.03 mm for 16 micro steps per full step are measured.

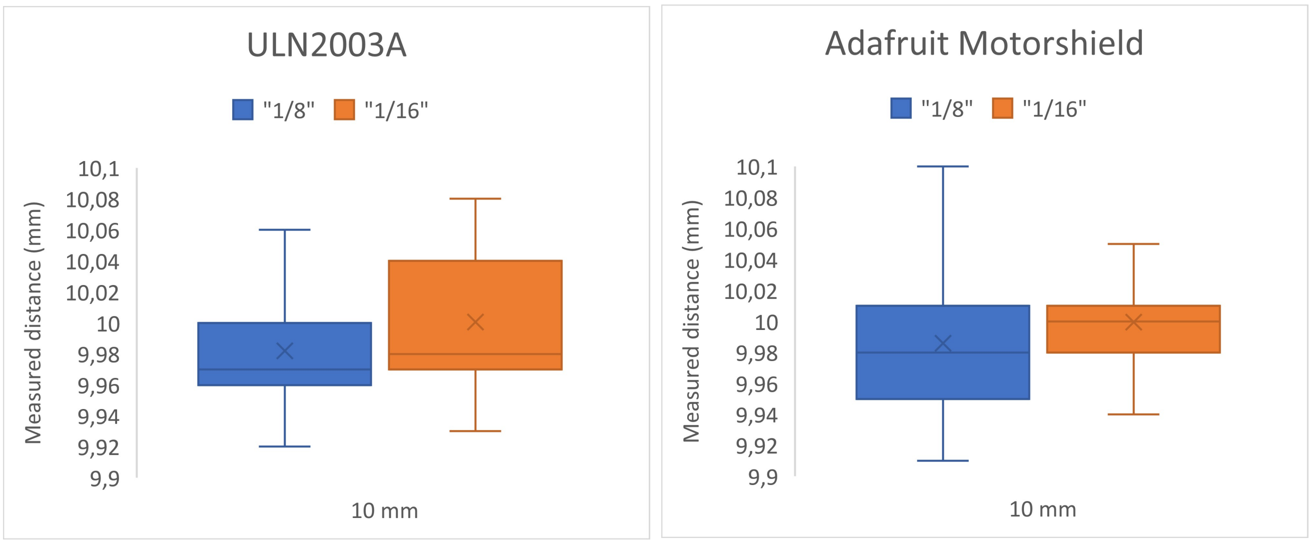


Figure 29: Distance measurements for 10 mm

The overall measurements of both the ULN2003A and the Adafruit Motorshield for micro stepping factors “1/8” and “1/16” are plotted Figure 30 for 100 mm distance. For the ULN2003A, a mean of 100.13 mm with a standard deviation of 0.20 mm for 8 micro steps per full step and a mean of 100.33 mm with a standard deviation of 0.10 mm for 16 micro steps per full step are measured. For the Adafruit Motorshield, a mean of 100.17 mm with a standard deviation of 0.08 mm for 8 micro steps per full step and a mean of 100.21 mm with a standard deviation of 0.06 mm for 16 micro steps per full step are measured.

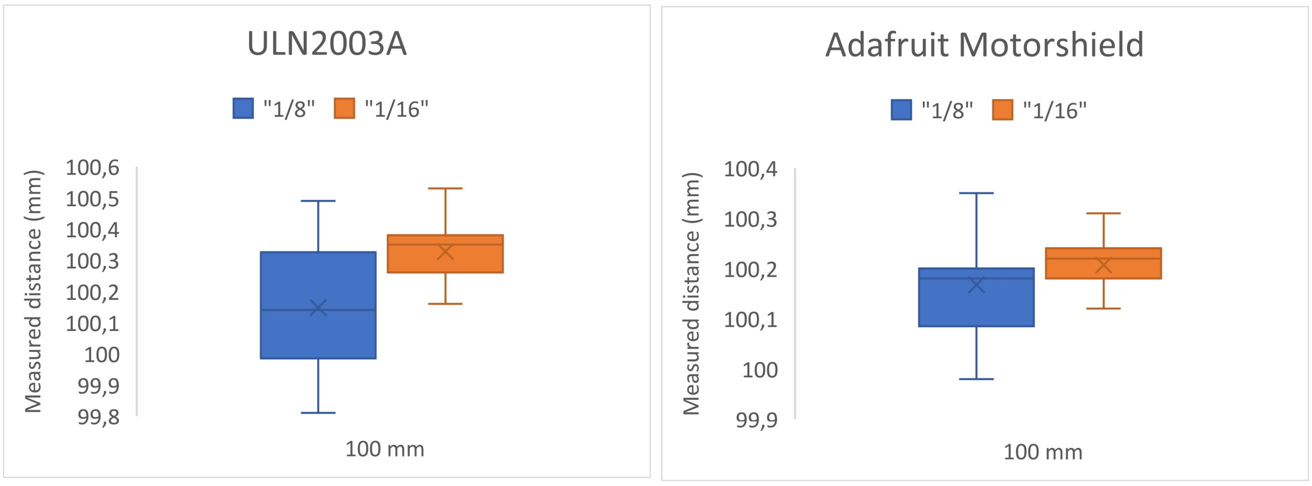


Figure 30: Distance measurements for 100 mm

5.1.4 Conclusion

After analyzing the results, it is clear that increasing the number of micro steps per full step decreases the overall variance of the measurements and therefore increases the accuracy.
While both stepper drivers perform similar for most of the measurements, the Adafruit Motorshield configured with 16 micro steps per full step proves to be the most accurate choice with the least variance over both distances. The difference in accuracy is possibly caused by an error in the code that drives the ULN2003A. The libraries for the Adafruit Motorshield are open source and used in many applications. The error in the code for the ULN2003A can be a miscalculation by the lack of specifications of the stepper motor. Another possibility is a loss in power in the coils. The Adafruit has different power settings to adjust for multiple power supplies [23]. Further measurements are made using this configuration.

5.2 Manual calibration

5.2.1 Camera calibration in MATLAB

The calibration by hand with the camera calibration toolkit in MATLAB designed by Caltech Institute of Technology [25] with25 images placed at different angles and positions from the camera. For calibration, a 7x10 chessboard pattern with 26mm squares is used. A 7x10 pattern will give 6x9 usable saddle-points to represent the world position. The calibration will return the intrinsic, extrinsic and distortion parameters along with the parameter estimation error and the mean reprojection error.
The result of the camera calibration is shown is Table 2. Figure 31 represents all the used positions of the pattern during the calibration. This visual method only provides a quick overview, it is not used to determine accuracy. An analysis of the reprojection error shows that the pixel error is concentric around the center of the image, see Figure 32.

Table 2: Calibration parameters and settings

Camera Intrinsics	
Focal Length	[3330.537 3335.727] ± [4.90447 4.89167]
Principal point	[2217.710 1765.382] ± [6.39445 4.80535]
Skew	[0.00000]
Distortion Vector	
Distortion	[-0.40839 0.17764 0.00144 -0.00005 0.00000] ± [0.00242 0.00478 0.00030 0.00022 0.00000]
Reprojection Error	
Pixel Error	[0.53927 0.53662]
Calibration Settings	
Image Resolution	4640 x 3480
# Patterns	20
World Units	mm
Image Units	Pixels

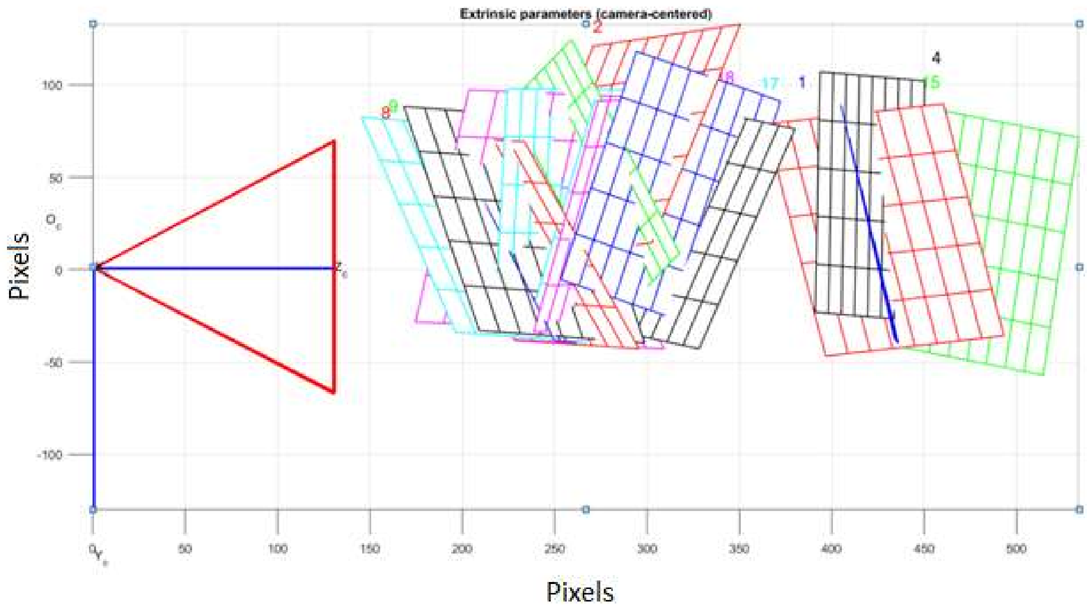


Figure 31: Side view of pattern positions used in the calibration

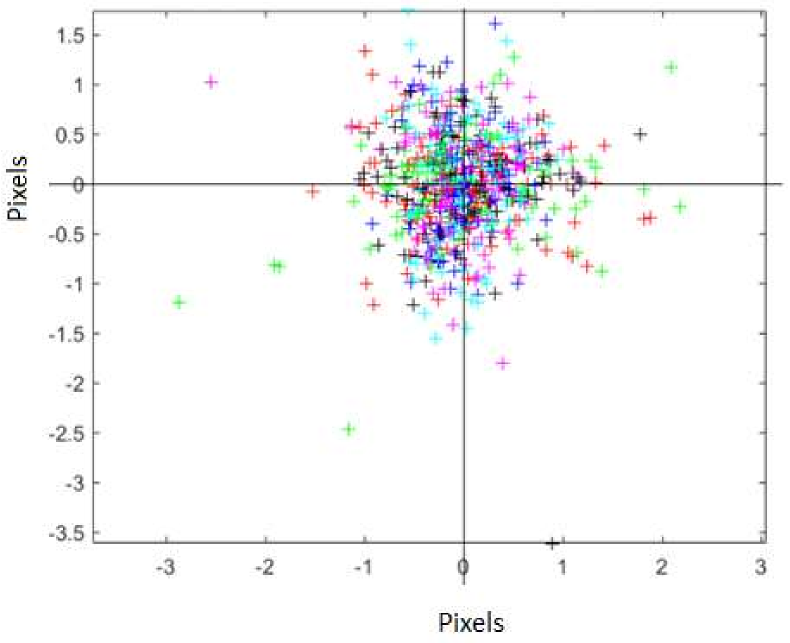


Figure 32: Reprojection error analysis

5.3 Calibration evaluation

5.3.1 Introduction

The evaluation is made of the calibration by hand against the calculations of the calibration done with the test setup. This is done by comparing measurements of the focal length and fixed pattern points, the reprojection error, rotation measurements for known angles, distance measurements for full step advancements and fixed position origin points.

5.3.2 Setup

The measurements use the test setup to evaluate the movement accuracy. The images that are taken all lie on the same axis perpendicular to the camera lens. The pattern is rotated around its vertical axis over angles of -45°, -20°, 0°, 20° and 45°. An example calibration with this method is shown in Figure 33. To further improve the accuracy, the pattern only moves in full step advancements and each movement stops when a full step is completed. This full step is divided into 16 micro steps, so the smallest movement is 16 steps at the time. The pattern is a 11x8 chessboard with a square size of 12mm. The resolution of the images is 4640 x 3480 pixels.

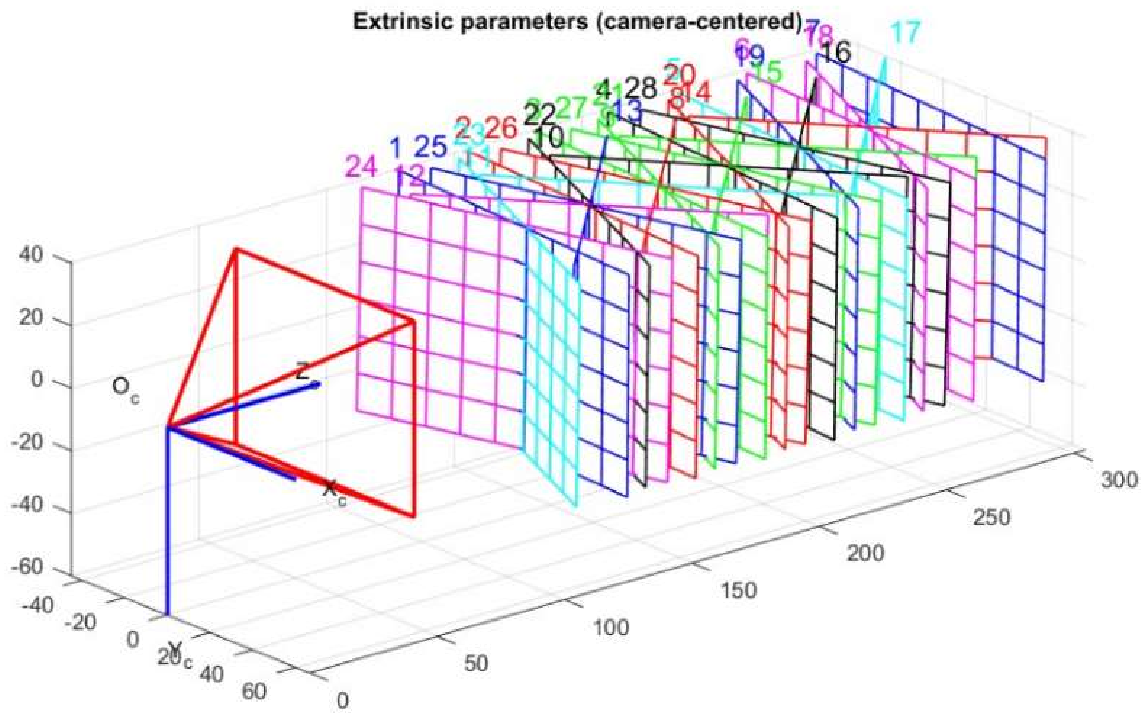


Figure 33: Example calibration with the test setup, distances are in pixel size

5.3.3 Measurements

Focal Length

The focal length is one of the main parameters acquired after calibrating the camera. Figure 34 shows the focal point in pixel coordinates for 15 calibrations. The average focal point x-coordinate has a value of 3332.26 pixels with a deviation of 4.51 pixels and the average focal point y-coordinate has a value of 3334,41 pixels with a deviation of 4.80 pixels.

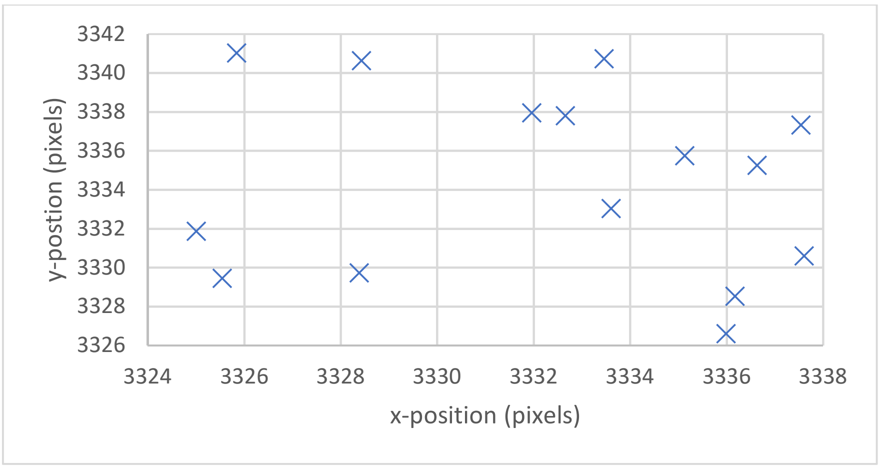


Figure 34: Focal point comparison of multiple calibrations

For the same 15 measurements, the mean reprojection error is shown which has an average of 0.86 pixels with a deviation of 0.10 pixels.

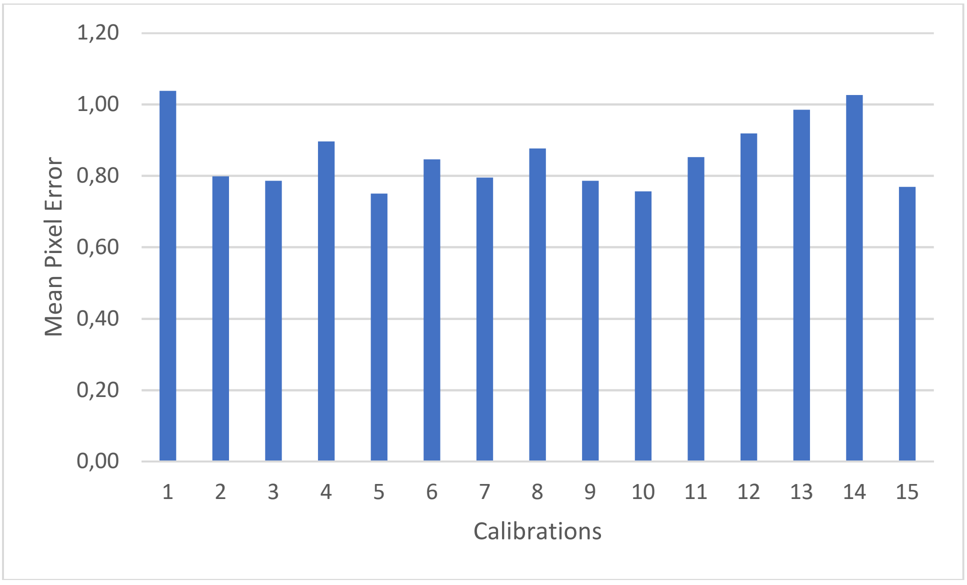


Figure 35: Mean reprojection errors of 15 calibrations

Translation

The distance measurements are the distances measured from the camera center to the calibration pattern when the begin position is known. This distance is derived from the translation vector, calculated for each calibrated image. Each iteration, an advancement of 10 full steps or 160 micro steps is made, which corresponds to 6.67 mm and the movement is ended at a full step size. The results are showed in Figure 36. The error between the real and measured distance is shown in Figure 37. The average error is 0.77 mm with a standard deviation of 0.58 mm for 30 measurements.

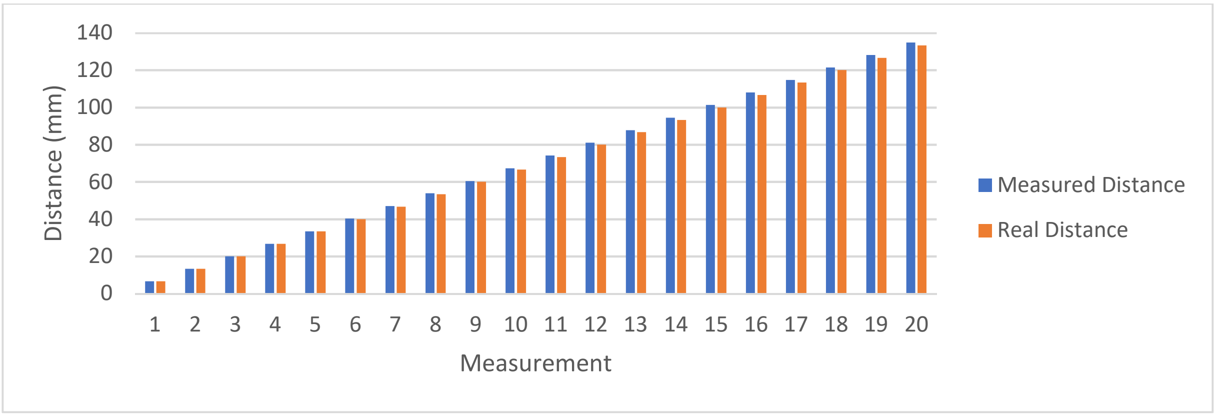


Figure 36: Measured distance versus the real distance in microstepping mode

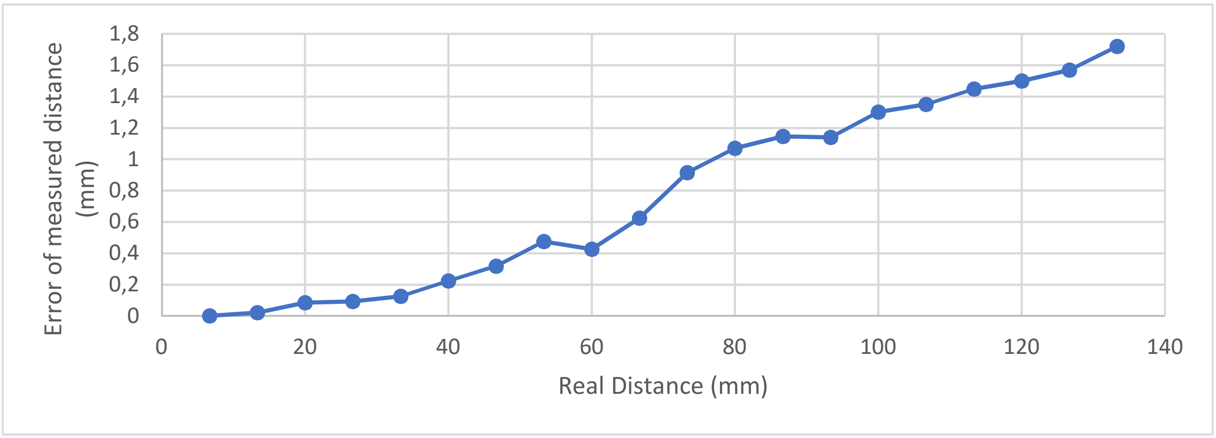


Figure 37: Error of the measured distance in microstepping mode

The distance measurement is repeated for full step size motion only. The full steps are not divided into micro steps to ensure that each movement ends on a full step. Each iteration, an advancement of 10 full steps is made, which corresponds to 6.67 mm and the movement is ended at a full step size. The results are showed in Figure 38. The error between the real and measured distance is shown in Figure 39. The average error is 0.67 mm with a standard deviation of 0.70 mm for 30 measurements.

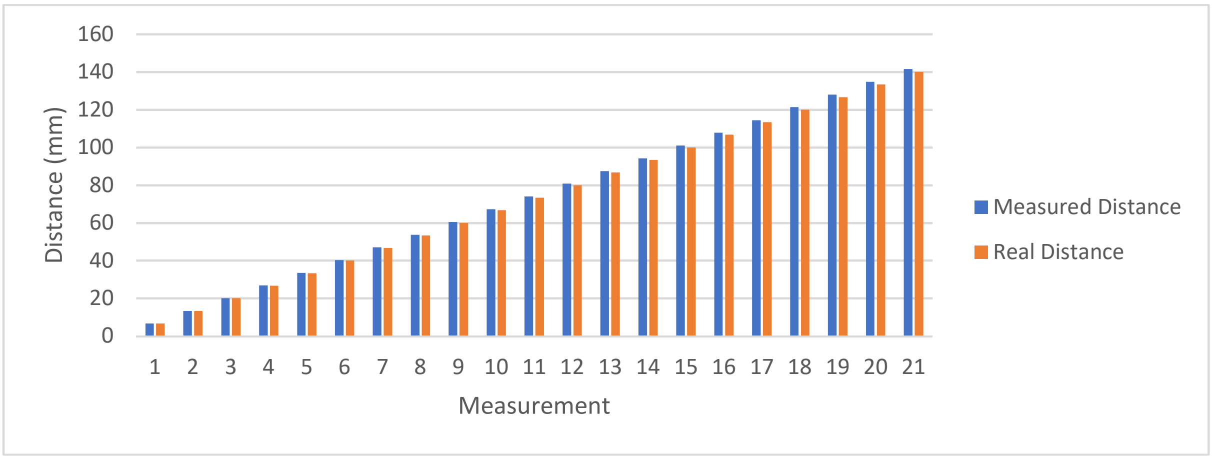


Figure 38: Real distance versus measured distance in full stepping mode

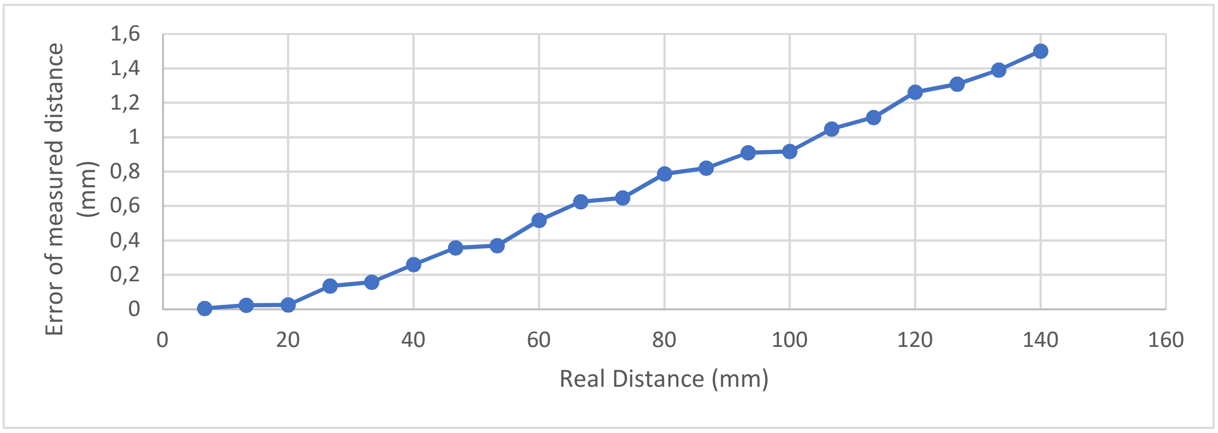


Figure 39: Error of the distance measurement in full stepping mode

Cross translation

The setup is turned 90° from the camera lens so the movement is now measured from the camera center to the origin point of the grid pattern. The camera now captures the movement in the Y-axis, which is placed parallel with the fixed rod that sets the direction of movement of the stepper motor. With this configuration, the X-and Z translations are negligible as they are almost constant. The distance is again derived from the translation vector. Each advancement is 10 full steps, which corresponds to 6.67 mm or 0.667 mm per step. The results are shown in Figure 40. The error between the real distance and measured distance is shown in Figure 41. The average error is 0.69 mm with a standard deviation of 0.58 mm for 25 measurements.

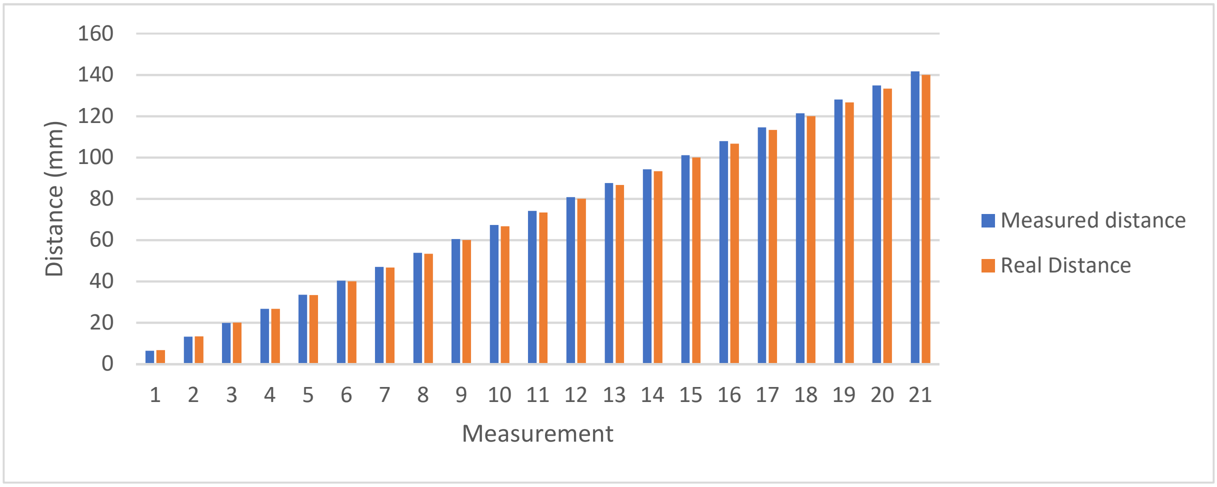


Figure 40: Measured distance versus real distance of transverse direction

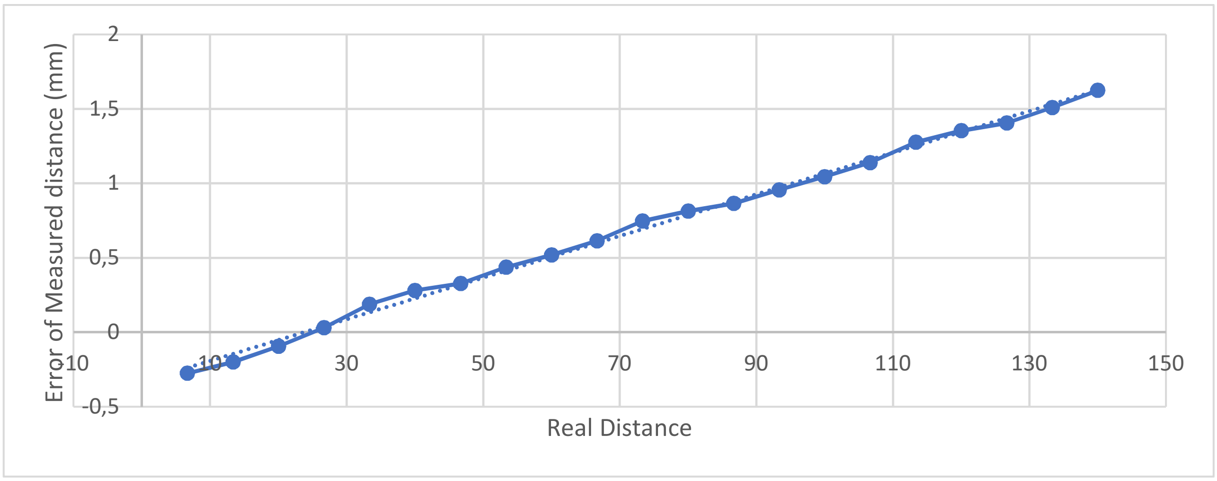


Figure 41: Error of the distance measurement in transverse direction

Rotation

The rotation of the calibration pattern is determined by the Rodrigues transform [4] of the rotation matrix. The transform decomposes the matrix into a single rotation vector around each of the axis. Only the rotation around the y-axis is plotted, since the design of the pattern applicator only allowed for rotations around its y-axis, but note that the camera can be repositioned to allow for other angles. The measurements are done on a fixed angle pattern of 0°, 20° and 45° over a 100mm. The results are shown in Figure 42. The measured angles are an average of 0.21° with a standard deviation of 0.03°, an average of 19.84° with a standard deviation of 0.06° and an average of 44,81° with a standard deviation of 0.07°.

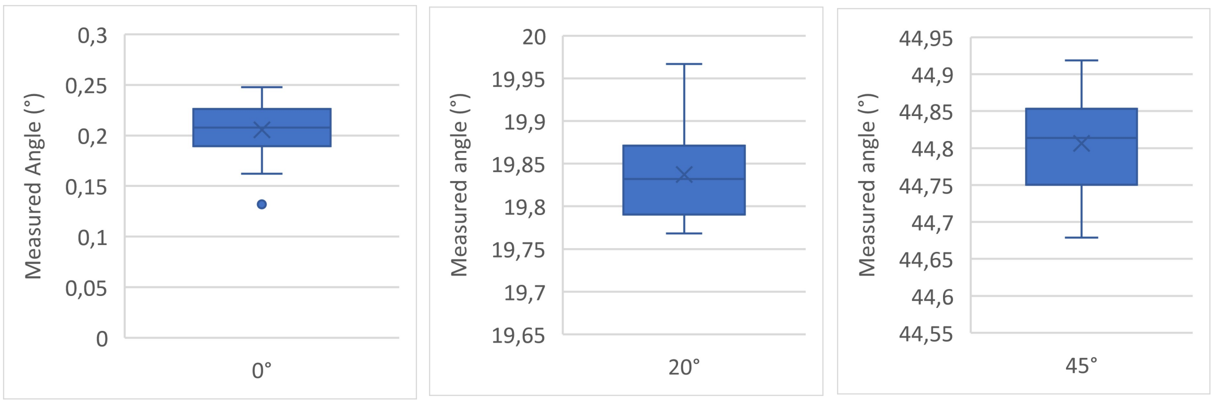


Figure 42: Rotation of pattern under 0°, 20° and 45° angles

The average rotation of 30 measurements is calculated and shown in Figure 43. The average error is 0.20° with a standard deviation of 0.03°.

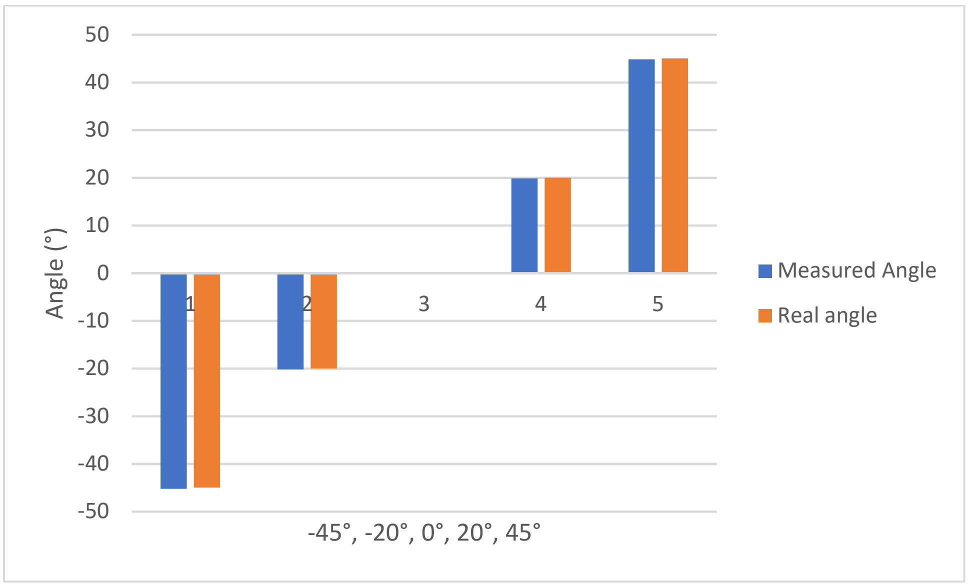


Figure 43: Real angle compared to the measured angle

Fixed distance

The origin point of the calculated grid at a fixed distance is determined. This point will be equal to the upper left saddle-point of the chessboard, as shown in Figure 44. The camera is placed at a distance of 100 mm and the camera position is estimated for 15 origin points. The results are shown in Figure 45.

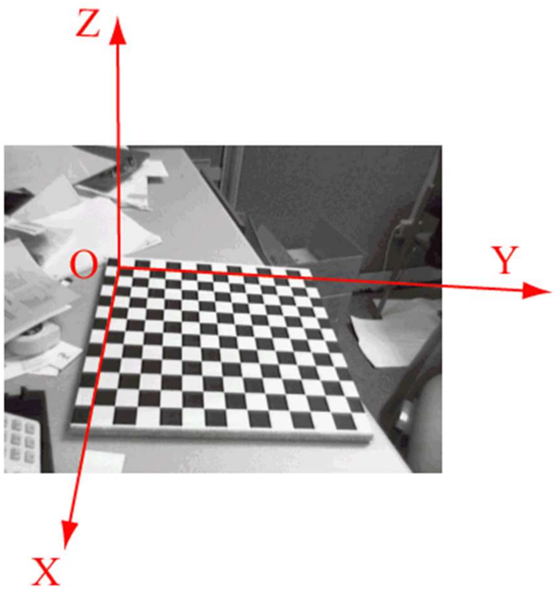


Figure 44: Origin point of the grid pattern.

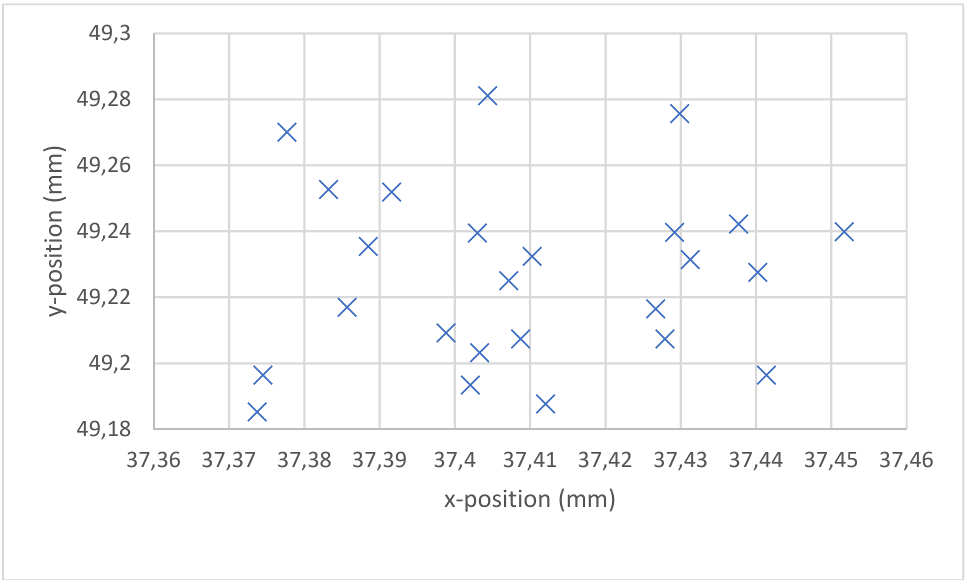


Figure 45: Origin points for 15 measurements.

5.3.4 Inaccuracies during measurements

The first inaccuracy to mention is the error that is present in the distortion matrix. Since we can only move the calibration pattern along one axis perpendicular to the camera, there is no way to capture images in the camera corner except by manually moving the setup. The pattern is placed as close as possible to the camera lens so that most of the camera view is covered to compensate for this. A bad calibration can lead to a less qualitative estimation of the camera parameters. The measurements used a large MP camera to capture as much pixels as possible, but because of the wide-angle lens the occurring distortions are large.

The second inaccuracy arises during the manufacturing of the pattern applicator and the stubs that are placed in the holes. The laser cutter that was used to process the MDF has an error of 10 µm. This error could result in a small inaccuracy of the measured angle.

The measurements to determine the accuracy of the movement of the stepper motor are done with a digital caliper which has a repeatable accuracy of ±50 µm. The measured errors that are smaller in absolute value than 50 µm precision could therefore deviate from the actual value.

The calibration chessboard pattern determines the pixel-to-mm ratio from which the real-world distances are calculated. In theory, the pattern should be placed as flat as possible on the pattern applicator. A transparent surface, like a plexiglass substrate, can be used to press the pattern flat. This would affect the quality of the images as a transparent surface would bend the incoming light because it has a different refractive index than the underlying material. The pattern can also be improved by increasing the dots per inch or pixel. This setting is determined by the quality of the printer. A higher DPI will result in a better pattern.

Chapter 6

Conclusion

6.1 Measurements

The measurements of the hardware indicate that the Adafruit Motorshield was the best choice to drive the stepper motor. Both tested drivers should have been able to achieve the same accuracy. For the Motorshield, there were preexisting C libraries to control the function of the stepper motor. For the ULN2003A, all the code had to be written manually to be able to perform microstepping. It is possible that ULN2003A performed worse because there was a miscalculation in the Arduino code. There were no technological specifications available for the stepper motor so its specifications had to be estimated through measurements.

In Table 3, the overall results of the calibration measurements are shown. Note that the represented errors are the standard error, not the standard deviation.

Table 3: Result of measurements with standard error

	Manual Calibration (pixels)	Calibration With Setup (pixels)
Focal length		
Fx	3330,537 ± 4,904	3332,266 ± 7,160
Fy	3335,727 ± 4,891	3334,416 ± 6,916
	Reprojection Error (pixels)	Reprojection Error (pixels)
Mean Pixel Error	0,53	0,86
	Measured Distance (mm)	Calculated Distance (mm)
100 mm	100,21 ± 0,05	100,84 ± 0,19
	Real Angle (°)	Calculated Angle (°)
	0°	0,20 ± 0,09
	20°	19,84 ± 0,18
	45°	44,81 ± 0,21

The average estimation of the focal length shows to be consistent with the focal length of the manual calibration. The reprojection error is significantly larger for the measurements but compared to the 16-megapixel resolution of the images this is considered a small error.

The accuracy of the translation with microstepping mode is also significantly lower for the measurements, being ± 190 µm compared to the ± 50 µm accuracy of the caliper. A possible explanation is that the measurements were not perfectly perpendicular to the camera lens. The distance measurements transverse to the camera showed approximately the same error. The distance measurements with full stepping mode have approximately the same error but a larger deviation of the average.

The measured angles have an average error of 0.20°. Table 3 shows that all calculated angles deviate from the real angles by about the same value. This could indicate that the measurements were not fully perpendicular but at an angle of the average error.

References

[1] Zhengyou Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, November 2000.

[2] Roger Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323-344, August 1987.

[3] Jef Stals, "Pose estimation with a camera for orthognathic surgery planning". Thesis, University of Hasselt – Department of industrial sciences Electronics-ICT, 2015.

[4] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed. Sebastopol: O'Reilly Media, Inc, 2008.

[5] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion model and accuracy evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10. IEEE Transactions, pp. 965–980, 1992.

[6] "pinhole camara science.jpg (JPEG-afbeelding, 729 × 262 pixels)." [Online]. Available: http://paksc.org/pk/images/stories/pinhole camara science.jpg. [Accessed: Dec-2018].

[7] P. Monasse, U. P. Marne-la-vallée, P. Monasse, J. Morel, and Z. Tang, "Epipolar rectification Epipolar rectification," no. October, 2014.

[8] S. Kapoor, "An Introduction to Epipolar Geometry | Sanyam Kapoor," 2017. [Online]. Available: https://www.sanyamkapoor.com/machine-learning/an-introduction-to-epipolar-geometry/. [Accessed: Dec-2018].

[9] G. Vass and T. Perlaki, "Applying and removing lens distortion in post production," *Proc. 2nd Hungarian Conf. Comput. Graph. Geom.*, pp. 9–16, 2009.

[10] "cameracalibrator_tangentialdistortion.png (PNG-afbeelding, 700 × 374 pixels)." [Online]. Available: https://nl.mathworks.com/help/vision/ref/cameracalibrator_tangentialdistortion.png. [Accessed: 14-Jan-2018].

[11] D. Brown, "Decentering Distortion of Lenses," *Photom. Eng.*, vol. 32, no. 3, pp. 444–462, 1966.

[12] I. E. Sutherland, "Three-Dimensional Data Input by Tablet," Proc. IEEE, vol. 62, no. 4, pp. 453–461, 1974.

[13] "computation_3d_point2.png (PNG-afbeelding, 1058 × 467 pixels)." [Online]. Available: https://support.pix4d.com/hc/article_attachments/115018247263/computation_3d_point2.png. [Accessed: Jan-2018].

[14] T. E. Kissell, *Industrial electronics: applications for programmable controllers, instrumentation and process control, and electrical machines and motor controls*. Prentice Hall, 2000.

[15] "Unipolar-Stepper-Motor.gif (GIF-afbeelding, 650 × 275 pixels)." [Online]. Available: https://www.circuitspecialists.com/blog/wp-content/uploads/2012/01/Unipolar-Stepper-Motor.gif. [Accessed: Jan-2018].

[16] G. Leger, “Unipolar Stepper Motor vs Bipolar Stepper Motors,” 2012. [Online]. Available: https://www.circuitspecialists.com/blog/unipolar-stepper-motor-vs-bipolar-stepper-motors/. [Accessed: Jan-2018].

[17] “Timing Belt Dimensions.jpg (JPEG-afbeelding, 480 × 241 pixels).” [Online]. Available: http://www.wychbearings.co.uk/user/Timing Belt Dimensions.jpg. [Accessed: Jan-2018].

[18] “Tutorial: Calibrating Stepper Motor Machines with Belts and Pulleys.” [Online]. Available: https://www.norwegiancreations.com/2015/07/tutorial-calibrating-stepper-motor-machines-with-belts-and-pulleys/. [Accessed: Jan-2018].

[19] J. Valvano, “Microstepping,” p. 11, 2015.

[20] “Jangeox’ blog: Stepper motor 28BYJ-48.” [Online]. Available: http://www.jangeox.be/2013/10/stepper-motor-28byj-48_25.html. [Accessed: Jan-2018].

[21] Ardino Store, “Arduino UNO Rev3,” *Online*, 2014. [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3%0Ahttps://store.arduino.cc/usa/arduino-uno-rev3%0Ahttps://store.arduino.cc/usa/arduino-uno-rev3%0Ahttps://store.arduino.cc/arduino-uno-rev3%5Cnhttp://store.arduino.cc/product/A000066. [Accessed: Jan-2018].

[22] “ULN2003A High-Voltage, High-Current Darlington Transistor Arrays | TI.com.” [Online]. Available: http://www.ti.com/product/ULN2003A. [Accessed: Jan-2018].

[23] “Adafruit Assembled Data Logging shield for Arduino ID: Adafruit Industries, Unique & fun DIY electronics and kits.” [Online]. Available: http://www.adafruit.com/products/1141. [Accessed: Jan-2018].

[24] “DENVER ACT-8030W - Denver.dk.” [Online]. Available: http://www.denver-electronics.com/denver-act-8030w/. [Accessed: Dec-2018].

[25] Itseez, “About - OpenCV library,” *OpenCV*, 2000. [Online]. Available: https://opencv.org/about.html%0Ahttp://opencv.org/about.html. [Accessed: Dec-2018].

[26] J.-Y. Bouguet, “Camera Calibration Toolbox for Matlab,” 2009. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Accessed: 15-Jan-2018].

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Calibration and evaluation system for 3D camera systems

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2018**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Aerts, Thomas

Datum: **15/01/2018**