LEHRSTUHL FÜR HOCHFREQUENZTECHNIK

Friedrich-Alexander-Universität
Technische Fakultät

Design of a point tracking system with depth camera for directional control of a mirror galvanometer for laser-based detection of muscle movements

Bearbeiter: Arda Buglagil

Betreuer: Prof. Dr. Björn Eskofier
Marius Schmidt, M. Sc.
Misha Sadeghi, M. Sc.

Ausgabedatum: 01.04.2023
Abgabedatum: 30.09.2023

Abstract

This work aims to design and implement a laser system for precise muscle targeting, accounting for patient movement. The project integrates a galvanometer, a laser, and a depth camera to monitor muscle activity in real-time. Coordination of the depth camera with the galvanometer enables the system to control the laser automatically. A calibration method is developed based on photodiodes to align the depth camera and galvanometer coordinate systems. Target detection algorithms are investigated and incorporated into the system. The system can adjust the laser's position with high accuracy, but it lacks real-time tracking capability in its current form due to the latency of the depth camera.

Contents

Symbols

P_0	incoming beam origin
n_0	incoming beam direction (unit vector)
M	mirror center
C	center of rotation
d	distance between M and C
n_m	mirror normal vector
P_1	reflected beam origin
n_1	reflected beam direction (unit vector)
D	distance between mirror center(C) and target plane
n_t	target plane normal vector
P_2	the point reflected beam hits the target plane
α	rotation degree of the target plane with respect to steering mirror
O	coordinate system origin
T	target plane origin
n_t	target plane normal vector
r_{OT}	vector from O to T
r_{TP_2}	vector from T to P_2
n_0	incoming laser direction
A_{IT}	transform matrix to convert frame of reference from I to T
A_{TI}	transform matrix to convert frame of reference from T to I
r_{OC}	vector from O to C
R	3×3 orthogonal rotation matrix
t	3×1 translation vector
H	cross covariance matrix
A	points recorded at camera coordinate system
B	points recorded at mirror coordinate system

1 Introduction

Galvanometers (or galvos) are rotary axes that use mirrors to deflect laser beams with precision, making them ideal for applications requiring accurate laser positioning. This work is part of a research project that focuses on a new method for detecting muscle movements without physical contact. In this report, tracking and laser directing parts of the project are covered. The main goal of this work is to develop a method to track a marker placed on a patient's body using a depth camera and to direct the laser beam to the tracked position as shown in Figure 1.1. To ensure precise measurements, the direction of the laser beam must be adjusted dynamically to account for any patient movement during data acquisition. This requires precise coordination between the depth camera and the galvanometer.

The objective of this study is to design, implement, and evaluate a system that can focus a laser on a specific muscle, compensate for patient movement, and ensure consistent measurement points.

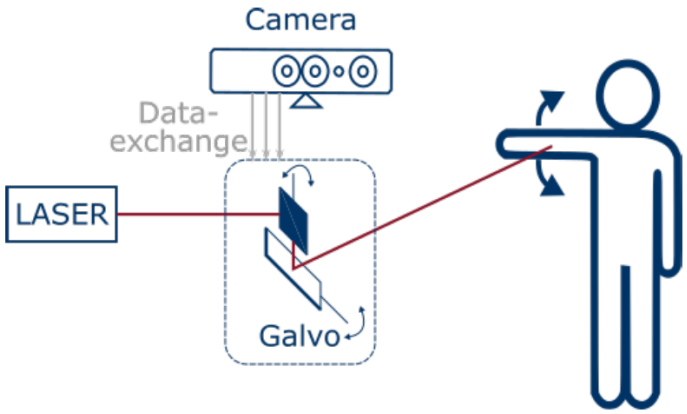


Figure 1.1: Tracking system schema

2 Mapping Steering Mirror Coordinates to Target Plane Coordinates

To direct the laser beam, MR-E-2 beam steering mirror (galvanometer) is used. The laser beam is produced by a stationary laser. The laser is positioned to hit the steering mirror in the center. After the beam hits the mirror, it is directed to the required position by adjusting the rotation of the mirror. The described experiment setup is shown in Figure 2.1.

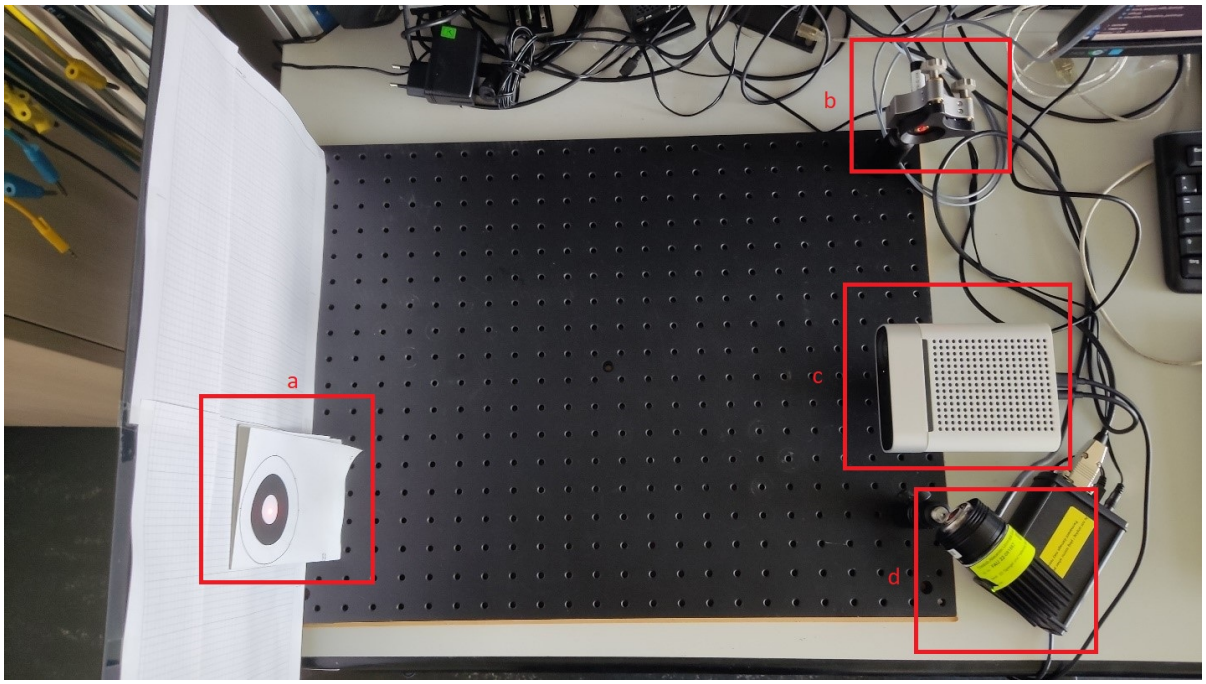


Figure 2.1: Target(a), laser(b), depth camera(c), and beam steering mirror(d)

Using the mirror to steer the laser beam requires a mapping from the 3D world coordinate system to the mirror coordinate system. The mapping calculates the required rotation

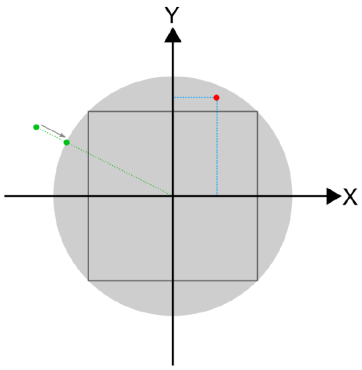


Figure 2.2: Internal mirror coordinate system [1]

of the mirror around its axes to direct the laser beam to the intended 3D position. The mirror coordinate system is shown in Figure 2.2.

The coordinate system is a Cartesian coordinate system with X and Y axes. The rotation of the mirror around its horizontal and vertical axes is expressed as x and y values. The range of both axes is $[-1, 1]$ interval. The relation between mirror rotation and mirror coordinate is shown in Figure 2.3. θ is the angle between the incoming and reflected beam. $\theta = +50°$ corresponds to +1 and $\theta = -50°$ corresponds to -1. x and y values are required to be inside a unit circle in order to be a valid point accessible by the mirror.

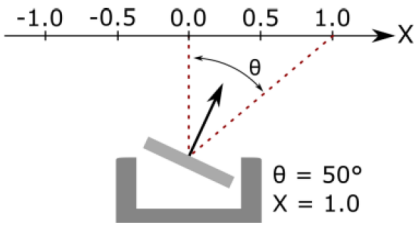


Figure 2.3: Relation between rotation of the mirror (θ) and the mirror coordinate (x) in X-axis [1]

In the experiments, the following configuration is used:

- Reflected beam origin P_1, mirror center M, and center of rotation C coincide with each other.
- Coordinate system origin O, reflected beam origin P_1, mirror center M, and center of rotation C coincide with each other.
- Distance between M and C, d is 0 mm.
- Incoming ray is in the y-z plane.

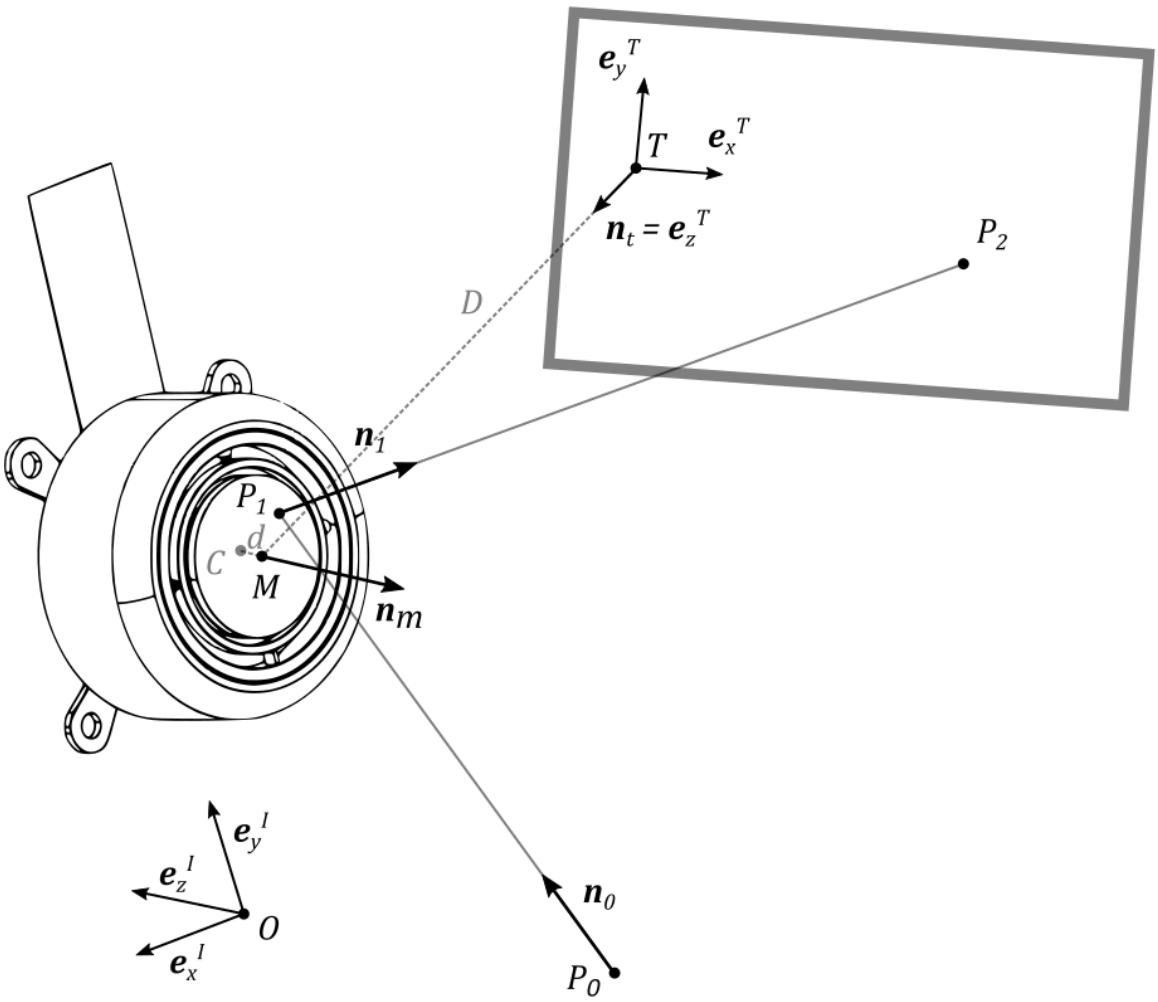


Figure 2.4: 3D mirror coordinate system [1]

- Rotation degree of the target plane with respect to steering mirror α is 45 °.

The conversion of coordinates on target plane (x_t, y_t) to mirror coordinates (x, y) is performed in two main steps. Firstly, the mirror normal (n_m) is calculated based on position of the laser, position of the target plane and (x_t, y_t). In the next step, corresponding mirror coordinates (x, y) is calculated using n_m.

2.1 Computation of mirror normal vector (n_m)

There are 2 different frames of reference: I and T. Superscripts I and T denote the reference frames of the vectors. $\mathbf{T}$ superscript denotes the transpose operation. Reference frame I is centered around O. Its z-axis is aligned with $-n_m$ direction and its y-axis is aligned so that the incoming laser beam propagates through the y-z plane. Reference

frame T is centered around target plane center T. Its z-axis is aligned with n_t direction and its y-axis is aligned so that the reflected laser beam propagates through the y-z plane. The transformation between the frames of reference is done with orthogonal transformation matrices A_{IT} and A_{TI}.

Vectors/matrices used in computations:

$$n_t^T = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad r_{OT}^T = \begin{bmatrix} 0 \\ 0 \\ -D \end{bmatrix} \tag{2.1}$$

$$r_{TP_2}^T = \begin{bmatrix} x_t \\ y_t \\ 0 \end{bmatrix} \qquad n_0^I = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.2}$$

$$A_{IT} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \qquad A_{TI} = (A_{IT})^{\mathbf{T}} \tag{2.3}$$

The mirror normal vector is calculated with the following steps [1]:

Target plane normal, n_t^T, is transformed into reference frame I.

$$n_t^I = A_{IT} \cdot n_t^T \tag{2.4}$$

Vector from O to T, r_{OT}^T, is transformed into reference frame I.

$$r_{OT}^I = A_{IT} \cdot r_{OT}^T \tag{2.5}$$

Vector from T to P_2, $r_{TP_2}^T$, is transformed into reference frame I.

$$r_{TP_2}^I = A_{IT} \cdot r_{TP_2}^T \tag{2.6}$$

$r^I_{OP_2}$ is obtained by vector summation.

$$r^I_{OP_2} = r^I_{OT} + r^I_{TP_2} \tag{2.7}$$

Reflected beam direction, n^I_1, is calculated by normalizing $r^I_{OP_2}$.

$$n^I_1 = \frac{r^I_{OP_2}}{||r^I_{OP_2}||} \tag{2.8}$$

Mirror normal direction n^I_m is calculated by subtracting n^I_0 from n^I_1 and normalizing the result.

$$n^I_m = \frac{n^I_1 - n^I_0}{||n^I_1 - n^I_0||} \tag{2.9}$$

2.2 Computation of mirror coordinates (x, y)

Vectors/matrices used in computations:

$$r^I_{OC} = \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix} \qquad n^I_0 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{2.10}$$

$$n^T_t = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad r^T_{OT} = \begin{bmatrix} 0 \\ 0 \\ -D \end{bmatrix} \tag{2.11}$$

$$A_{IT} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

The mirror coordinates (x, y) are calculated with the following steps [1]:

The intersection of incoming laser beam with mirror surface, $r^I_{OP_1}$, is calculated.

$$t_1 = \frac{(r^I_{OC} - r^I_{OP_0}) \cdot n^I_m + d}{n^I_0 \cdot n^I_m} \tag{2.13}$$

$$r^I_{OP_1} = r^I_{OP_0} + t_1 \cdot n^I_0 \tag{2.14}$$

Reflected beam direction, n^I_1, is obtained by applying the law of reflection.

$$n^I_1 = n^I_0 - 2 \cdot (n^I_0 \cdot n^I_m) \cdot n^I_m \tag{2.15}$$

Intersection point of reflected beam with target plane, $r^I_{OP_2}$, is calculated.

$$t_2 = \frac{(r^T_{OT} - r^I_{OP_1}) \cdot n^T_t}{n^I_1 \cdot n^T_t} \tag{2.16}$$

$$r^I_{OP_2} = r^I_{OP_1} + t_2 \cdot n^I_1 \tag{2.17}$$

x and y coordinates are extracted from $r^I_{OP_2}$ and are scaled by $D \cdot tan(50°)$.

$$x = \frac{r^I_{OP_2}[0]}{D \cdot tan(50°)} \tag{2.18}$$

$$y = \frac{r^I_{OP_2}[1]}{D \cdot tan(50°)} \tag{2.19}$$

3 Position Error Measurement System

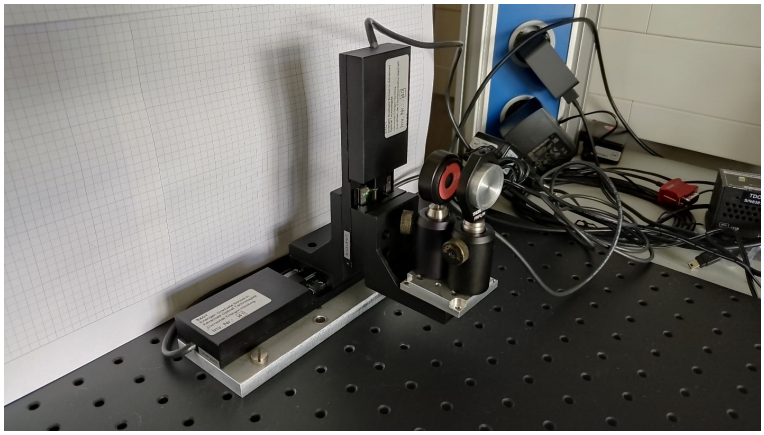


Figure 3.1: Position error test setup

Measurements are done with a 2-dimensional servo motor system (composed of a linear stage mounted perpendicularly on top of another linear stage) with a PM400 power meter attached as in Figure 3.1. Measurements are performed by pointing the laser to a specified target position and then sweeping the power meter in a linear trajectory. Power readings are recorded with recording times and plotted to find the instance with maximum power. The point with maximum power gives the center position of the laser. Power measurements are performed by a Python script which has a loop with a period of around 0.01 s. This sampling period is not constant throughout the measurements and might deviate from 0.01 s. For this reason, after the measurement, cubic interpolation is performed to get a uniformly sampled signal as shown in Figure 3.2.

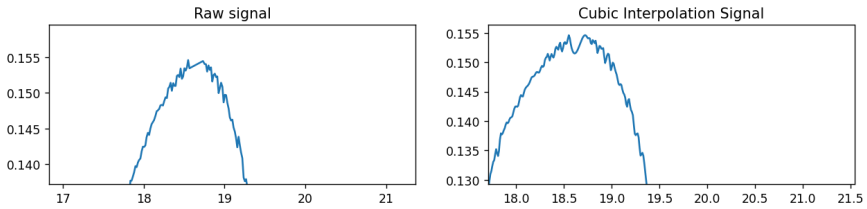


Figure 3.2: Power(mW) - Position(mm) graphs for raw signal and cubic interpolation signal

Interpolation solves the nonuniform sampling problem. Obtained signal still has noise in it which might alter the maximum position. To remove the noise a low pass filter in the

form of a moving average filter is applied. Figure 3.3 depicts the signal before and after the low pass filter. High-frequency noise components are removed from the signal while keeping the original signal mostly intact. This operation produced a smoother signal which is more suitable for peak finding operation.

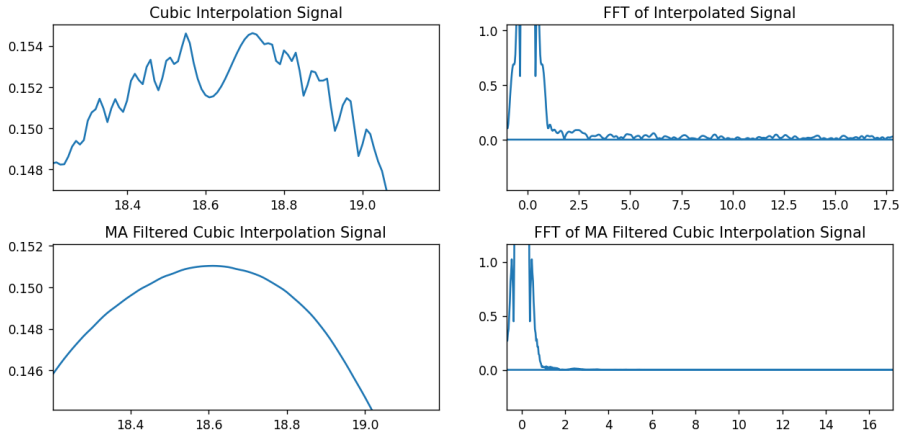


Figure 3.3: Power(mW) - Position(mm) graphs for raw signal and moving average filtered(MA) cubic interpolation signal

The recorded power signal is plotted with respect to distance in order to find the position of the laser's center. This method gives the position of the laser relative to the initial position of the servo motor. Figure 3.4 shows the power levels with respect to time and position.

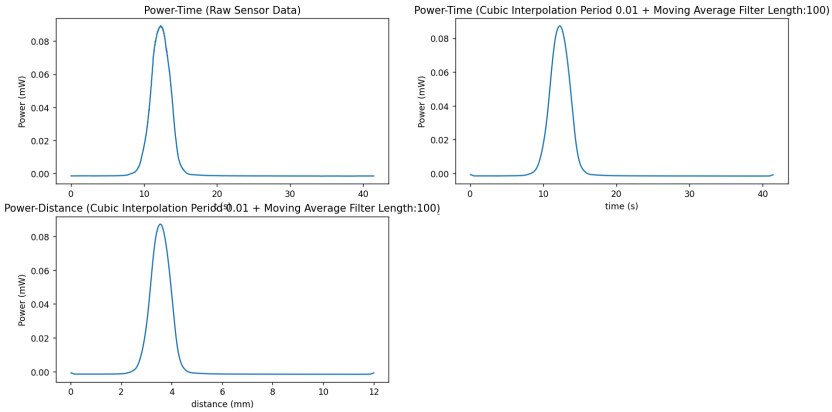


Figure 3.4: Power(mW) - Position(mm) graphs for raw signal and cubic interpolation signal

4 Mapping Depth Camera Coordinate System to Steering Mirror Coordinate System

The coordinate systems that are used in the steering mirror and the depth camera don't coincide as shown in Figure 4.1. They are rotated and translated versions of each other. Points recorded in camera coordinate system, A, can be transformed into points recorded in mirror coordinate system, B, with the following mapping:

$$B = RA + t \tag{4.1}$$

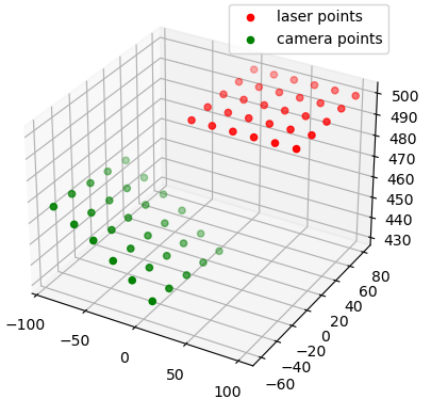


Figure 4.1: Recording of the same points in steering mirror(laser) and depth camera coordinate systems

R is 3×3 an orthogonal rotation matrix and t is a 3×1 translation vector. Optimal R and t are found by the following algorithm [2] [3] [4]:

To find the rotation matrix R cross covariance matrix H is calculated,

$$H = (A - m_A)(B - m_B)^T \tag{4.2}$$

Singular value decomposition of H is computed,

$$U, S, V^T = SVD(H) \tag{4.3}$$

Rotation matrix R is obtained by multiplying V and U^T,

$$R = VU^T \tag{4.4}$$

Translation vector t is found as,

$$t = m_B - Rm_A \tag{4.5}$$

Optimal transformation maps each point in camera coordinate system to a point in mirror coordinate system. After the mapping, mirror points and camera points are almost aligned as shown in Figure 4.2. With this mapping, any point recorded by depth camera can be transformed into a point which can be used by the mirror controller.

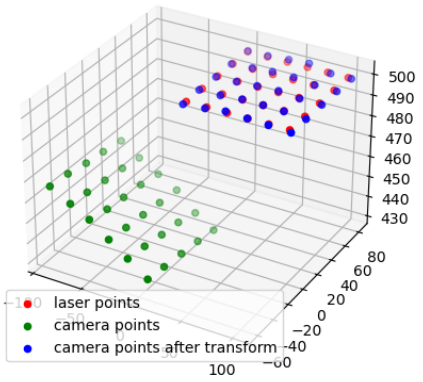


Figure 4.2: Camera points after the transformation

5 Calibration Procedure

Calibration of the system is done in order to coordinate the depth camera and mirror together. Both the steering mirror and depth camera have their own coordinate systems. The relationship between these coordinate systems is unknown which makes them unable to perform tasks together. The main objective of calibration is to find the rotation and translation of the coordinate systems relative to each other. For this purpose, the optimal R and t finding algorithm discussed in chapter 4 is used.

The calibration procedure is responsible for obtaining point pairs consisting of a point in depth camera coordinate system and its corresponding point in the steering mirror coordinate system. After the collection of the dataset, the mapping is computed and saved for later usage. Point pair collection operation depends on the type of laser used in the system. For easier testing, a visible laser is used. However, in the end system, an IR laser will be used. Using an IR laser restricts the usage of the camera since it is not visible by the camera sensor. IR laser requires IR light intensity sensors to detect and a different calibration procedure than visible laser which is discussed in chapter 5.2.

5.1 Calibration Based on Color Camera (Calibration Procedure for Visible Laser)

The laser is pointed at 30 different positions in a plane 560 mm away from the mirror center. These points are recorded as mirror points B with their 3D coordinates. Each laser position is extracted from the color images by using Hough Circle Detection algorithm. Once 2D pixel coordinates are found they are converted to 3D coordinates by Kinect Azure SDK's *k4a::calibration::convert_2d_to_3d* [5] function. Extracted 3D depth camera points are named A. Points A and B are formed into matrices with shape $3 \times N$. Finally, the algorithm in 5 is used to find R and t matrices.

5.2 Calibration Based on Photodiodes (Calibration Procedure for IR Laser)

IR laser requires a different calibration method from the visible laser. Point matching operation is done via laser intensity sensors. The sensor setup used to detect positions is in Figures 5.1 5.2. 3 sensors generate analog output signals based on the intensity of light hitting them. The analog signal is captured by ADCs on a Raspberry Pi Pico board and transmitted to the host computer via USB.

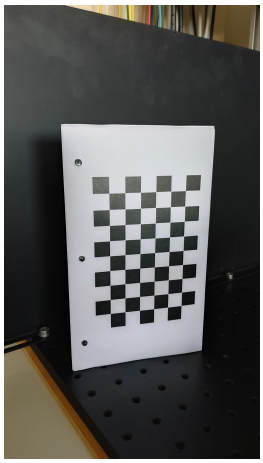


Figure 5.1: Sensor setup. 3 photodiodes are aligned vertically 75 mm apart from each other on the left part of the calibration plate, chessboard pattern is on the right side of the plate.

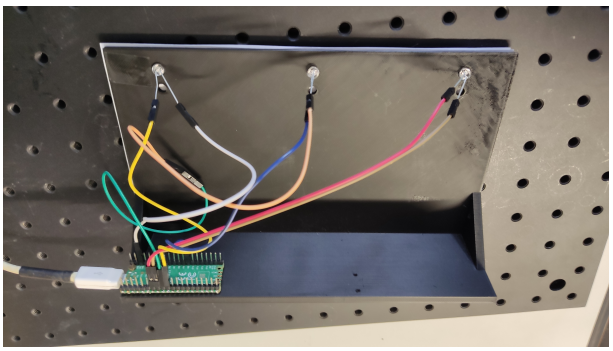


Figure 5.2: Sensor setup. 3 photodiodes are connected to 3 analog pins on Raspberry Pi Pico. Sensor 1, sensor 2, and sensor 3 (from left to right) are connected to respectively A0, A1, and A2 pins on Pico.

While the sensor is recording the signal intensity, the laser scans the area near the sensor position based on an initial 3D sensor position estimate. Each signal point is plotted with respect to the position of the laser at the time of recording. As a result, an intensity profile is generated as shown in Figures 5.3 5.4. The maximum intensity level of this

profile is chosen as the sensor position. This process is performed by all three sensors, and 3 different images are generated. In each image, a peak intensity is observed, and it is marked as that specific sensor's position.

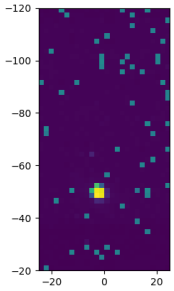


Figure 5.3: Sensor image created with 3mm scan resolution.

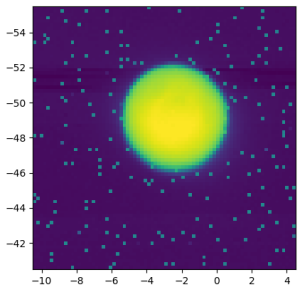


Figure 5.4: Sensor image created with 0.2mm scan resolution.

Three 3D coordinates generated are not exact coordinates due to the initial position estimate. The x and y positions determined by scanning the laser are valid if and only if the initial distance between the mirror center and target plane is correct. Since this won't be the case for most cases, found points should be used to calculate the real distance between the mirror center and the target plane.

To calculate the distance between the mirror center and the target plane, the geometry of the sensors is used. Sensors are placed on top of each other as in Figure 5.5 and the plate holding sensors is placed perpendicular to the ground. This configuration places all the sensors at the same distance from the mirror center in the z-direction as depicted in Figure 5.6. Using this constraint, the distance between sensors and the mirror center in the z-direction is calculated:

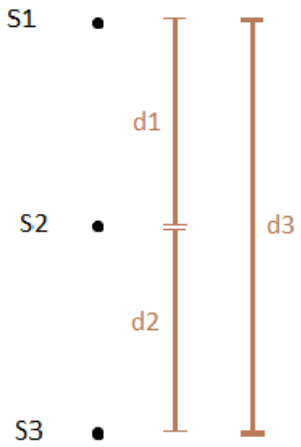


Figure 5.5: 3D sensor positions ($S1$, $S2$, $S3$), 3D detected sensor positions ($M1$, $M2$, $M3$), distances between sensors ($d1$, $d2$, $d3$) and steering mirror position (O) from camera view.

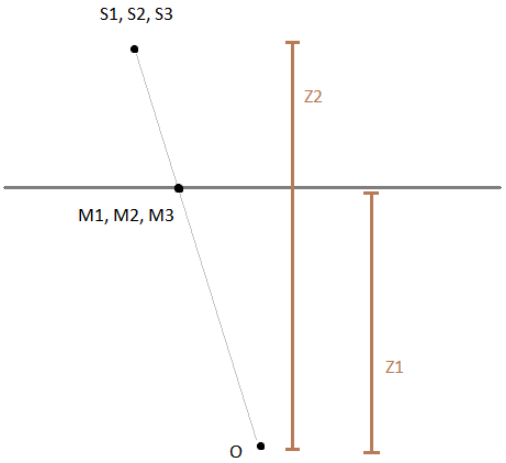


Figure 5.6: 3D sensor positions ($S1$, $S2$, $S3$), 3D detected sensor positions ($M1$, $M2$, $M3$), distance between steering mirror and detected sensor position plane $z1$, distance between steering mirror and actual sensor position plane $z2$ and steering mirror position (O) from top view.

$$d1 = |M1 \cdot \frac{z2}{z1} - M2 \cdot \frac{z2}{z1}| \quad d2 = |M2 \cdot \frac{z2}{z1} - M3 \cdot \frac{z2}{z1}| \quad d3 = |M1 \cdot \frac{z2}{z1} - M3 \cdot \frac{z2}{z1}| \tag{5.1}$$

$$z2 = \frac{d1 \cdot z1}{|M1 - M2|} \qquad z2 = \frac{d2 \cdot z1}{|M2 - M3|} \qquad z2 = \frac{d3 \cdot z1}{|M1 - M3|} \tag{5.2}$$

With the $z2$ distance obtained, previous 3D coordinates are updated by scaling them to be at the correct distance and they are saved as mirror points B. The next step is to find the sensor positions with the depth camera and to obtain depth camera points A. To localize the sensors, the marker pattern in Figure 5.7 is attached to the sensor plate. Pixel coordinates of each internal corner point in the chessboard pattern are detected. Then, 2D pixel coordinates are transformed into 3D coordiantes using Kinect Azure SDK's *k4a::calibration::convert_2d_to_3d* [5] function. Two unit vectors are extracted from the 3D coordinates of corners. These 3-dimensional unit vectors indicate the left and down directions on the surface of the calibration plate.

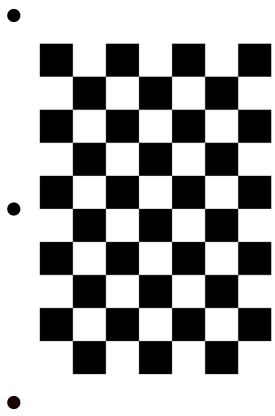


Figure 5.7: Localization marker with chessboard pattern. 3 circles on the left side show the position of the photodiodes.

Unit vectors u_l and u_d are used to locate the 3D position of the sensors. This is possible since markers and the marker pattern are on the same plane. Found points are recorded as depth camera points A. Points A and B are formed into matrices with shape $3 \times N$. Finally, algorithm in 5 is used to find R and t matrices.

6 Detection Algorithms

Position estimation is the first step to tracking the position of the body. One of the most common approaches to this problem is usage of visual markers. A visual marker can be distinguished from the environment without too much difficulty due to its characteristics. For the project 3 different algorithms are tested: ArUco, Hough Circle Detector, and WHYCon. Detection algorithms are evaluated based on their processing times in order to choose the best option for the system.

6.1 ArUco

ArUco is a popular binary square fiducial marker used for absolute pose estimation. ArUCo markers are black square shapes with an inner grid structure to encode a binary code. The binary code is used to identify the markers and distinguish multiple markers inside the same camera view. Due to the encoding, ArUco markers are not easily detected from far distances. [6]

Figure 6.1: ArUco marker with ids 0 up to 3 [6]

6.2 Hough Circle Detector

Hough Circle Transform is a feature extraction algorithm used for extracting circles from images. It is a general algorithm that can detect any circular shape unlike WHYCon and

ArUco markers which require a specific marker. During the implementation, the Hough Circle Detector can be used together with a color filter. The color filter eliminates all the colors except for the used marker's color. Then, the circle detector is applied to the filtered image to extract only the marker's circle. OpenCV's Hough Circle Detector implementation is used in the experiments [7].

6.3 WHYCon

WHYCon is an open-source marker-based localization system. It is highly computationally efficient [8]. WHYCon doesn't provide any identification information, its marker consists of black and white concentric circles. Due to its simplicity, it can be identified quickly and it requires low computational resources. During implementation, the only option that was able to run on a Windows machine was a C++ repository. To connect the algorithm to the rest of the system, Python bindings were implemented using Pybind11.

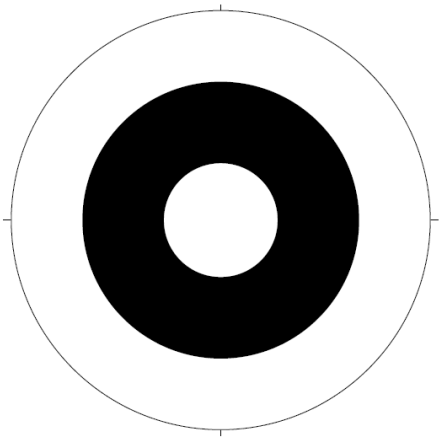


Figure 6.2: WHYCon marker [9]

7 Test Results

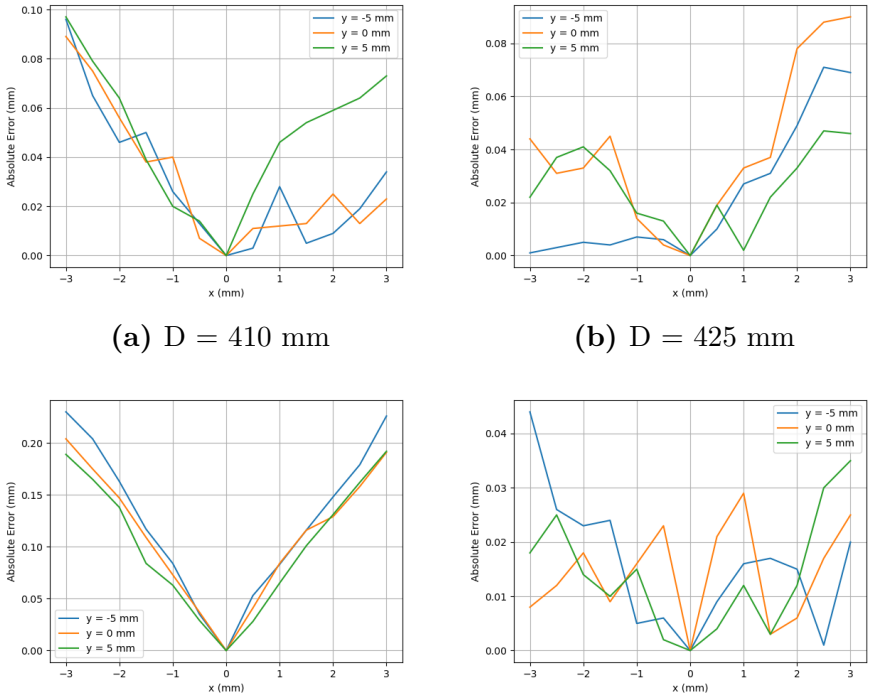
7.1 Position Error Test Results

During the tests effects of the distance between the mirror and target plane and the position of the laser on the error are observed. The results of these tests are recorded in Tables 9.1, 9.2, 9.3, 9.4. As seen in Figures 7.1a and 7.1c, when the real distance between the mirror and target plane doesn't match the distance entered into the system a systematic error occurs. The absolute value of the measured relative position is always larger than the expected position's absolute value. This is caused by the wrong scaling due to the wrong distance entered. When the actual distance and the distance entered into the system are the same this systematic error is not observed as in Figures 7.1d, 7.1b. The deflection mirror can point the laser to a specific target with less than 0.1 mm accuracy.

7.2 Calibration Error Test Results

Calibration accuracy is tested by the test setup in Figure 7.2. The target marker is detected by using WHYCon algorithm. The detected 3D point is shifted 40 mm in y-direction to hit the middle photodiode. The calibration error is measured by scanning around the detected point with the laser. At the end of the scanning process, an image as in Figure 5.4 is obtained. The position with the maximum intensity is selected as the center of the photodiode. The distance between the center of the photodiode and the center of the image is calculated to obtain the accuracy error.

Calibration accuracy depends on the number of points gathered and the position of these points. The calibration accuracy is tested with points gathered at different positions as displayed in Figure 7.3. Each blue point represents the position of the calibration plate from the top view. Increasing the number of points to calibrate the system and covering



(a) D = 410 mm **(b)** D = 425 mm

(c) D = 210 mm **(d)** D = 225 mm

Figure 7.1: Error(mm) - Position(mm) at various distances between mirror and target plane

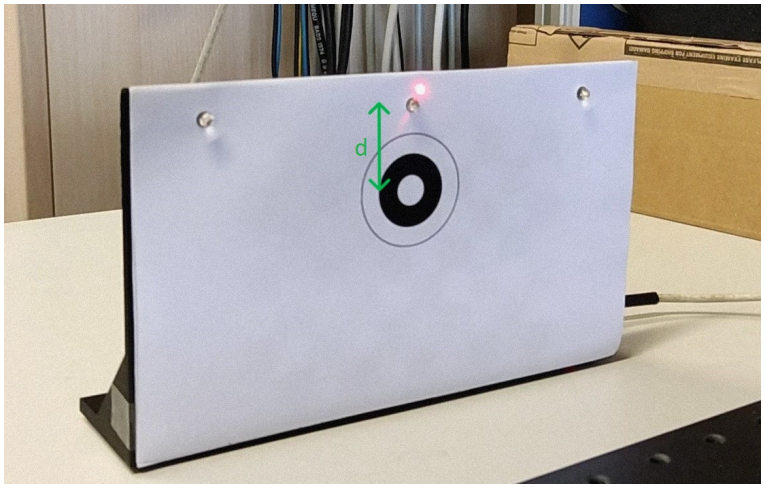
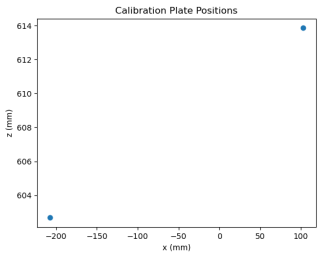
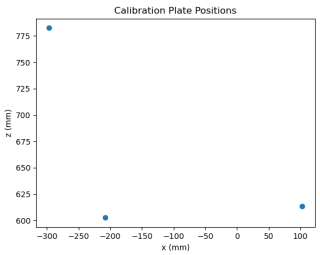


Figure 7.2: Calibration accuracy measurement test setup. $d = 40\,\text{mm}$.

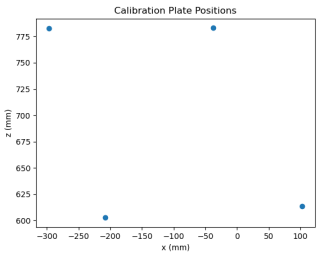
a larger area decreases the overall calibration error according to Figure 7.4. 1 mm error is the best result obtained during the tests and it is achieved by calibrating the system with 10 different positions.



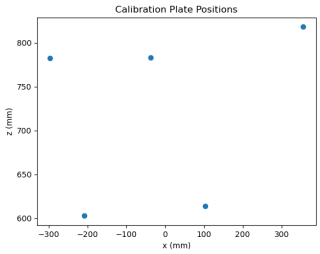
(a) Calibration with 2 positions



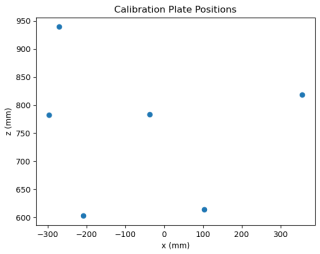
(b) Calibration with 3 positions



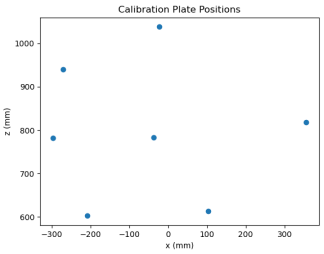
(c) Calibration with 4 positions



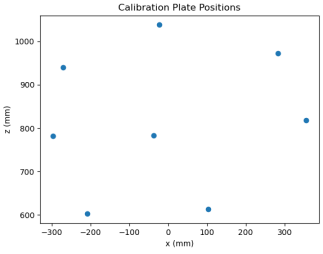
(d) Calibration with 5 positions



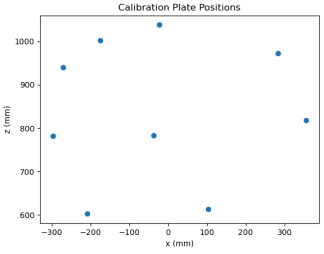
(e) Calibration with 6 positions



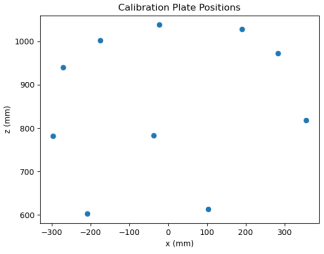
(f) Calibration with 7 positions



(g) Calibration with 8 positions



(h) Calibration with 9 positions



(i) Calibration with 10 positions

Figure 7.3: Calibration plate positions

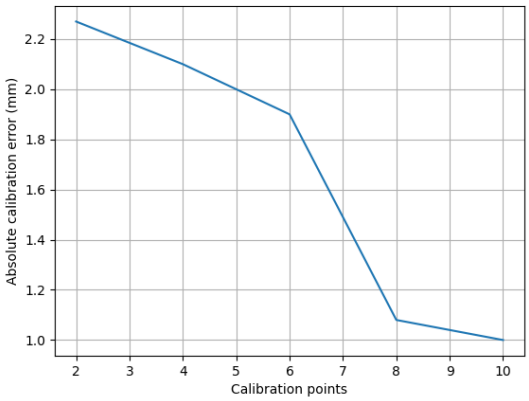


Figure 7.4: Absolute Calibration Error (mm) - Number of Calibration Positions

7.3 Detection Algorithm Performance Comparison Results

The algorithms used in the system are run on a computer with an Intel i7 processor. Based on their run times, it has been found that the WHYCon algorithm performs the best (as shown in Table 7.1). ArUco also performs well, with a runtime of 4ms, but its performance may decrease up to 30ms in case of movement or poor lighting conditions. This makes it unsuitable for real-time tracking. Although Hough Circle Detector has a good run time, it still falls short of WHYCon's performance. Therefore, WHYCon is used in the system.

The total system performance depends on multiple algorithms. Firstly, the Azure Kinect system captures color and depth images. Then, using the color image, the marker position is detected, and 2D pixel coordinates are extracted. The depth camera is used to transform these 2D pixel coordinates into 3D real-world coordinates. These 3D coordinates are then converted into mirror coordinates, as explained in section 2. Finally, the mirror API is used to rotate the mirror. Runtimes of each step are shown in Table 7.2. Total time is 23 ms, which means position of the laser can be smoothly updated with a frame rate higher than 30 fps.

Table 7.1: Detection Algorithm Performances at 720p resolution

Algorithm	**Runtime (ms)**
ArUco	4-30
WHYCon	2-3
Hough Circle Detector	8-10

Table 7.2: Runtime of different code sections

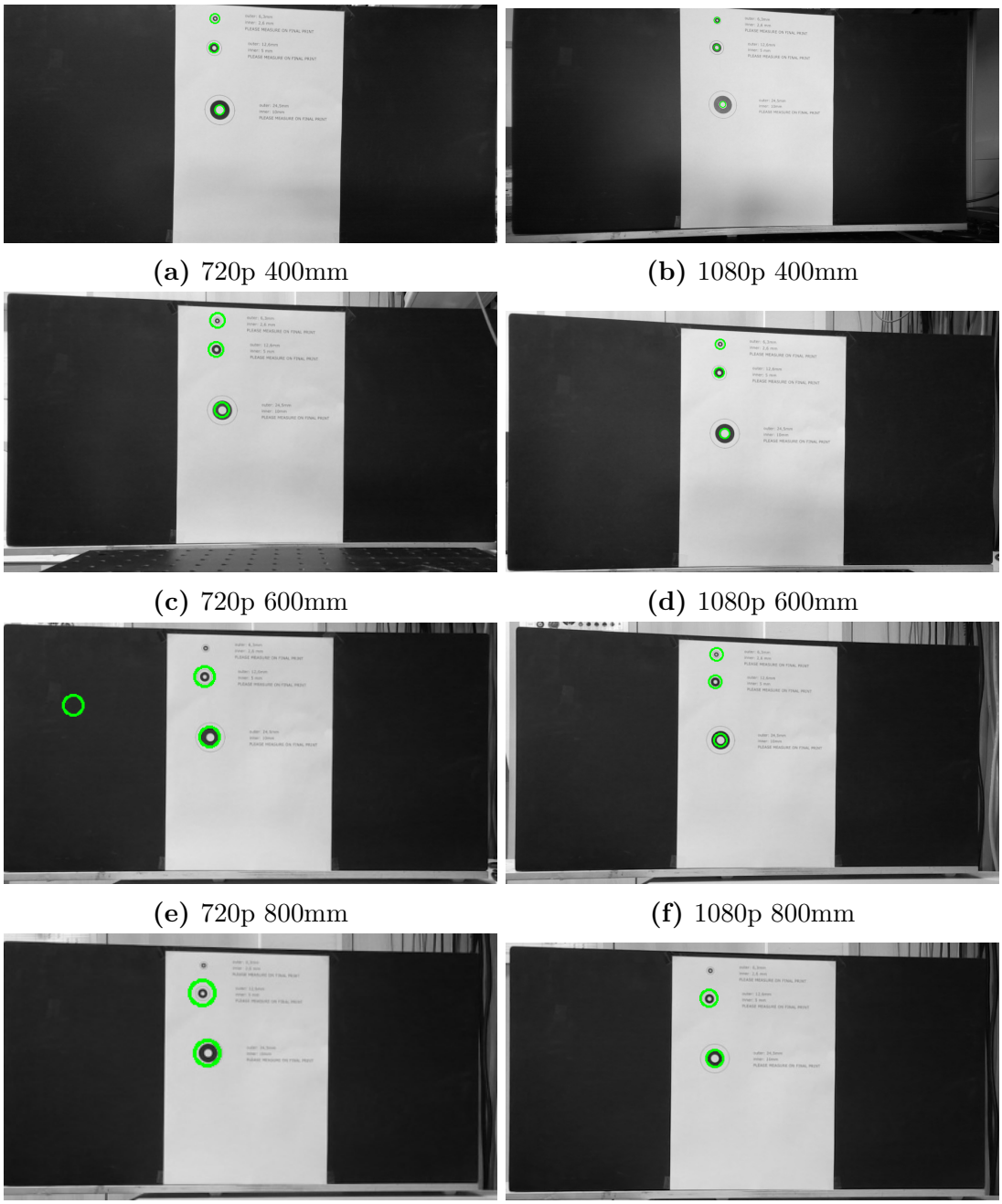
Code Sections	Elapsed Time (ms)
Capturing Color Image	3
Marker Detection	2
Transforming Depth Image to Color Camera's Coordinate System	15
3D to mirror coordinate conversion	1
Mirror Adjustment	2
Total Time	**23**

7.4 Maximum Distance of Detection Results

The maximum distance detection test is performed with the WHYCon detection algorithm. In each test, 3 different markers with diameters of 24.5 mm, 12.6 mm, and 6.3 mm are used. Tests are performed at distances 400 mm, 600 mm, 800 mm, and 1000 mm away from steering mirror center. Tests are repeated for 720p and 1080p image resolutions. Test results are shown in Figure 7.5. Increasing the resolution to 1080p increases the detection performance by making 6.3 mm diameter marker visible at 800 mm distance. At 1000 mm distance both 1080p and 720p resolutions fail to stably detect 6.3 mm diameter marker.

7.5 Marker Tracking Performance Results

The tracking performance of the system depends on the processing times of the algorithms used and the camera latency. As discussed in section 7.3, the average total processing time is 23ms. This means the software part of the system has a refresh rate of around 43.5 Hz. The Azure Kinect camera has a 30 fps refresh rate. This reduces the overall refresh rate to 30 fps. With this refresh rate objects can be tracked smoothly without abrupt changes in the laser position. Camera latency is the second most important factor which affects the tracking performance. Data supplied by the camera has to be fresh in order to achieve real-time tracking. The latency associated with the Azure Kinect camera is found to be 85ms. This value is estimated by moving laser position on a linear trajectory with a constant speed. The laser position is observed to be behind the intended position as shown in Figure 7.6a. Instead of supplying the current position, supplying the position of the laser 85 ms later produced the tracking image in Figure 7.6b. From this image it can be concluded that the latency of the system is approximately 85 ms. This latency figure causes lags and prevents real-time tracking when the tracked object is moving. For a typical target speed range of 0 – 500 mm/s, the tracking



(a) 720p 400mm **(b)** 1080p 400mm

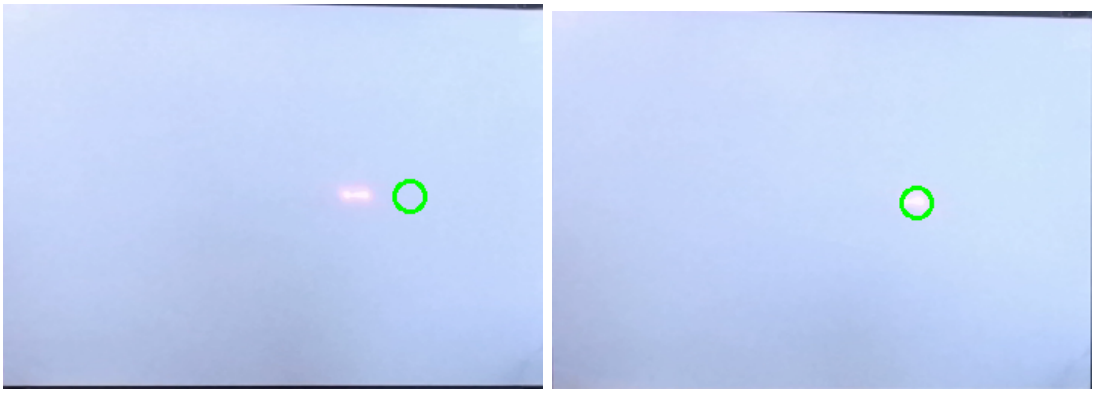
(c) 720p 600mm **(d)** 1080p 600mm

(e) 720p 800mm **(f)** 1080p 800mm

(g) 720p 1000mm **(h)** 1080p 1000mm

Figure 7.5: Detection results of WHYCon marker at different distances and different resolutions

lag is in the range of 0 – 42.5 mm. This means the laser will visibly lag behind the tracked target. Latency figures associated with Azure Kinect camera are as in Tables 7.3, 7.4. In the system setup, YUY2 format with 1280x720 resolution is used. Depth mode is selected as NFOW Binned due to depth image distortions at the edges of the captured image.



(a) Tracking with latency. **(b)** Tracking with latency compensation.

Figure 7.6: Effect of latency on tracking performance.

Table 7.3: Latencies of different color camera modes (The mode used in the system is marked by bold.) [10]

Resolution	MJPG	NV12	**YUY2**	BGRA32
4096 x 3072	111	-	-	168
3840 x 2160	72	-	-	161
2048 x 1536	59	-	-	72
2560 x 1440	67	-	-	85
1920 x 1080	63	-	-	75
1280 x 720	62	66	**59**	68

Table 7.4: Latencies of different depth camera modes (The mode used in the system is marked by bold.) [10]

Depth Mode	Resolution	Linux	Windows
NFOV Binned	320 x 288	42	**36**
NFOV Unbinned	640 x 576	42	35
WFOV Binned	512 x 512	33	27
WFOV Unbinned	1024 x 1024	73	62

8 Summary

This study aimed to design, implement, and assess a system capable of targeting fiducial markers with a laser using a galvanometer. The study encompassed a series of tasks that contributed to the realization of this objective.

The initial phase involved the establishment of the galvanometer and laser components. This step laid the groundwork for accurate laser positioning. The incorporation of depth camera introduced a new dimension, enabling real-time tracking of the targeted muscle. The development of a mechanism to align depth camera coordinates with the galvanometer's coordinate system was a critical milestone, facilitating the synchronization of visual data with laser positioning.

Progress was also made in the formulation of calibration methodologies, utilizing both color camera and photodiodes. These two different methods made matching depth camera and galvanometer possible. At the end of testing out different target detection algorithms WHYCon algorithm was chosen due to its fast runtime and reliability.

The latency associated with the depth camera introduced a limitation in real-time tracking, leading to delays in tracking the target. Consequently, the goal of seamless real-time tracking was partially unrealized.

As a possible next step, a depth camera with less latency can be used. This would reduce the lag during tracking. During calibration the sensor position is found by photodiodes with diameter of 5 mm. Reducing the diameter would provide a finer grained photodetector image and center location would be found more accurately. Another issue that should be incorporated is the usage of IR laser and IR photodetectors. In the operational system, the laser would operate in IR spectrum.

9 Appendix

In this section, experiment results are tabulated.

Table 9.1: Distance between mirror and target plane is set to 410mm. The actual distance between the mirror and the target plane is 425mm.

Mirror Input x	Mirror Input y	Peak Power Position	Relative Position
−3 mm	−5 mm	2.322 mm	−3.096 mm
−2.5 mm	−5 mm	2.853 mm	−2.565 mm
−2 mm	−5 mm	3.372 mm	−2.046 mm
−1.5 mm	−5 mm	3.868 mm	−1.550 mm
−1 mm	−5 mm	4.392 mm	−1.026 mm
−0.5 mm	−5 mm	4.905 mm	−0.513 mm
0 mm	−5 mm	5.418 mm	0 mm
0.5 mm	−5 mm	5.921 mm	0.503 mm
1 mm	−5 mm	6.446 mm	1.028 mm
1.5 mm	−5 mm	6.923 mm	1.505 mm
2 mm	−5 mm	7.427 mm	2.009 mm
2.5 mm	−5 mm	7.937 mm	2.519 mm
3 mm	−5 mm	8.452 mm	3.034 mm
−3 mm	0 mm	2.278 mm	−3.089 mm
−2.5 mm	0 mm	2.792 mm	−2.575 mm
−2 mm	0 mm	3.311 mm	−2.056 mm
−1.5 mm	0 mm	3.829 mm	−1.538 mm
−1 mm	0 mm	4.327 mm	−1.040 mm
−0.5 mm	0 mm	4.860 mm	−0.507 mm
0 mm	0 mm	5.367 mm	0 mm
0.5 mm	0 mm	5.878 mm	0.511 mm
1 mm	0 mm	6.379 mm	1.012 mm
1.5 mm	0 mm	6.880 mm	1.513 mm
2 mm	0 mm	7.392 mm	2.025 mm
2.5 mm	0 mm	7.880 mm	2.513 mm
3 mm	0 mm	8.390 mm	3.023 mm
−3 mm	5 mm	2.391 mm	−3.097 mm
−2.5 mm	5 mm	2.909 mm	−2.579 mm
−2 mm	5 mm	3.424 mm	−2.064 mm
−1.5 mm	5 mm	3.949 mm	−1.539 mm
−1 mm	5 mm	4.468 mm	−1.020 mm
−0.5 mm	5 mm	4.974 mm	−0.514 mm
0 mm	5 mm	5.488 mm	0 mm
0.5 mm	5 mm	6.013 mm	0.525 mm
1 mm	5 mm	6.534 mm	1.046 mm
1.5 mm	5 mm	7.042 mm	1.554 mm
2 mm	5 mm	7.547 mm	2.059 mm
2.5 mm	5 mm	8.052 mm	2.564 mm
3 mm	5 mm	8.561 mm	3.073 mm

Table 9.2: Distance between mirror and target plane is set to 425mm. Actual distance between mirror and target plane is 425mm.

Mirror Input x	Mirror Input y	Peak Power Position	Relative Position
−3 mm	−5 mm	6.723 mm	−2.999 mm
−2.5 mm	−5 mm	7.225 mm	−2.497 mm
−2 mm	−5 mm	7.727 mm	−1.995 mm
−1.5 mm	−5 mm	8.218 mm	−1.504 mm
−1 mm	−5 mm	8.716 mm	−1.007 mm
−0.5 mm	−5 mm	9.228 mm	−0.494 mm
0 mm	−5 mm	9.722 mm	0.0 mm
0.5 mm	−5 mm	10.212 mm	0.49 mm
1 mm	−5 mm	10.695 mm	0.973 mm
1.5 mm	−5 mm	11.191 mm	1.469 mm
2 mm	−5 mm	11.673 mm	1.951 mm
2.5 mm	−5 mm	12.151 mm	2.429 mm
3 mm	−5 mm	12.653 mm	2.931 mm
−3 mm	0 mm	6.788 mm	−2.956 mm
−2.5 mm	0 mm	7.275 mm	−2.469 mm
−2 mm	0 mm	7.777 mm	−1.967 mm
−1.5 mm	0 mm	8.289 mm	−1.455 mm
−1 mm	0 mm	8.758 mm	−0.986 mm
−0.5 mm	0 mm	9.248 mm	−0.496 mm
0 mm	0 mm	9.744 mm	0.0 mm
0.5 mm	0 mm	10.225 mm	0.481 mm
1 mm	0 mm	10.711 mm	0.967 mm
1.5 mm	0 mm	11.207 mm	1.463 mm
2 mm	0 mm	11.666 mm	1.922 mm
2.5 mm	0 mm	12.156 mm	2.412 mm
3 mm	0 mm	12.654 mm	2.91 mm
−3 mm	5 mm	6.849 mm	−2.978 mm
−2.5 mm	5 mm	7.364 mm	−2.463 mm
−2 mm	5 mm	7.867 mm	−1.959 mm
−1.5 mm	5 mm	8.359 mm	−1.468 mm
−1 mm	5 mm	8.843 mm	−0.984 mm
−0.5 mm	5 mm	9.34 mm	−0.487 mm
0 mm	5 mm	9.826 mm	0.0 mm
0.5 mm	5 mm	10.308 mm	0.481 mm
1 mm	5 mm	10.824 mm	0.998 mm
1.5 mm	5 mm	11.304 mm	1.478 mm
2 mm	5 mm	11.793 mm	1.967 mm
2.5 mm	5 mm	12.279 mm	2.453 mm
3 mm	5 mm	12.781 mm	2.954 mm

Table 9.3: Distance between mirror and target plane is set to 210mm. Actual distance between mirror and target plane is 225mm.

Mirror Input x	Mirror Input y	Peak Power Position	Relative Position
−3 mm	−5 mm	6.723 mm	−2.999 mm
−2.5 mm	−5 mm	7.225 mm	−2.497 mm
−2 mm	−5 mm	7.727 mm	−1.995 mm
−1.5 mm	−5 mm	8.218 mm	−1.504 mm
−1 mm	−5 mm	8.716 mm	−1.007 mm
−0.5 mm	−5 mm	9.228 mm	−0.494 mm
0 mm	−5 mm	9.722 mm	0.0 mm
0.5 mm	−5 mm	10.212 mm	0.49 mm
1 mm	−5 mm	10.695 mm	0.973 mm
1.5 mm	−5 mm	11.191 mm	1.469 mm
2 mm	−5 mm	11.673 mm	1.951 mm
2.5 mm	−5 mm	12.151 mm	2.429 mm
3 mm	−5 mm	12.653 mm	2.931 mm
−3 mm	0 mm	6.788 mm	−2.956 mm
−2.5 mm	0 mm	7.275 mm	−2.469 mm
−2 mm	0 mm	7.777 mm	−1.967 mm
−1.5 mm	0 mm	8.289 mm	−1.455 mm
−1 mm	0 mm	8.758 mm	−0.986 mm
−0.5 mm	0 mm	9.248 mm	−0.496 mm
0 mm	0 mm	9.744 mm	0.0 mm
0.5 mm	0 mm	10.225 mm	0.481 mm
1 mm	0 mm	10.711 mm	0.967 mm
1.5 mm	0 mm	11.207 mm	1.463 mm
2 mm	0 mm	11.666 mm	1.922 mm
2.5 mm	0 mm	12.156 mm	2.412 mm
3 mm	0 mm	12.654 mm	2.91 mm
−3 mm	5 mm	6.849 mm	−2.978 mm
−2.5 mm	5 mm	7.364 mm	−2.463 mm
−2 mm	5 mm	7.867 mm	−1.959 mm
−1.5 mm	5 mm	8.359 mm	−1.468 mm
−1 mm	5 mm	8.843 mm	−0.984 mm
−0.5 mm	5 mm	9.34 mm	−0.487 mm
0 mm	5 mm	9.826 mm	0.0 mm
0.5 mm	5 mm	10.308 mm	0.481 mm
1 mm	5 mm	10.824 mm	0.998 mm
1.5 mm	5 mm	11.304 mm	1.478 mm
2 mm	5 mm	11.793 mm	1.967 mm
2.5 mm	5 mm	12.279 mm	2.453 mm
3 mm	5 mm	12.781 mm	2.954 mm

Table 9.4: Distance between mirror and target plane is set to 225mm. Actual distance between mirror and target plane is 225mm.

Mirror Input x	Mirror Input y	Peak Power Position	Relative Position
−3 mm	−5 mm	4.985 mm	−3.044 mm
−2.5 mm	−5 mm	5.503 mm	−2.526 mm
−2 mm	−5 mm	6.007 mm	−2.023 mm
−1.5 mm	−5 mm	6.505 mm	−1.524 mm
−1 mm	−5 mm	7.034 mm	−0.995 mm
−0.5 mm	−5 mm	7.536 mm	−0.494 mm
0 mm	−5 mm	8.029 mm	0.0 mm
0.5 mm	−5 mm	8.539 mm	0.509 mm
1 mm	−5 mm	9.045 mm	1.016 mm
1.5 mm	−5 mm	9.546 mm	1.517 mm
2 mm	−5 mm	10.045 mm	2.015 mm
2.5 mm	−5 mm	10.529 mm	2.499 mm
3 mm	−5 mm	11.009 mm	2.98 mm
−3 mm	0 mm	4.989 mm	−2.992 mm
−2.5 mm	0 mm	5.493 mm	−2.488 mm
−2 mm	0 mm	5.999 mm	−1.982 mm
−1.5 mm	0 mm	6.49 mm	−1.491 mm
−1 mm	0 mm	6.998 mm	−0.984 mm
−0.5 mm	0 mm	7.459 mm	−0.523 mm
0 mm	0 mm	7.981 mm	0.0 mm
0.5 mm	0 mm	8.461 mm	0.479 mm
1 mm	0 mm	8.952 mm	0.971 mm
1.5 mm	0 mm	9.485 mm	1.503 mm
2 mm	0 mm	9.976 mm	1.994 mm
2.5 mm	0 mm	10.464 mm	2.483 mm
3 mm	0 mm	10.956 mm	2.975 mm
−3 mm	5 mm	5.156 mm	−2.982 mm
−2.5 mm	5 mm	5.664 mm	−2.475 mm
−2 mm	5 mm	6.153 mm	−1.986 mm
−1.5 mm	5 mm	6.649 mm	−1.49 mm
−1 mm	5 mm	7.154 mm	−0.985 mm
−0.5 mm	5 mm	7.64 mm	−0.498 mm
0 mm	5 mm	8.138 mm	0.0 mm
0.5 mm	5 mm	8.634 mm	0.496 mm
1 mm	5 mm	9.126 mm	0.988 mm
1.5 mm	5 mm	9.635 mm	1.497 mm
2 mm	5 mm	10.126 mm	1.988 mm
2.5 mm	5 mm	10.609 mm	2.47 mm
3 mm	5 mm	11.103 mm	2.965 mm

Table 9.5: Calibration error with respect to number of calibration points and distance between mirror and target plane

Distance (mm)	10 points	8 points	6 points	4 points	2 points
570	0.56mm	1.28mm	1.45	2.66	1.07
620	0.79mm	0.82mm	1.21	1.61	0.89
670	0.89mm	0.20mm	3.05	1.00	0.82
720	0.28mm	1.13mm	2.44	2.33	1.21
770	1.28mm	0.72mm	2.41	1.44	1.81
820	1.89mm	0.19mm	2.12	2.03	3.84
870	1.79mm	2.44mm	1.81	2.41	2.82
920	0.89mm	1.34	0.80	2.55	4.05
970	0.60mm	1.60	1.84	2.88	3.92
Average	**1.00**	**1.08**	**1.90**	**2.10**	**2.27**

List of Figures

List of Tables

Bibliography

[1] *MR-E-2 Development Kit Operation Manual.* Mar 2022

[2] https://simonensemble.github.io/posts/2018-10-27-orthogonal-procrustes/

[3] https://nghiaho.com/?page_id=671

[4] https://en.wikipedia.org/wiki/Kabsch_algorithm

[5] https://microsoft.github.io/Azure-Kinect-Sensor-SDK/master/structk4a_1_1calibration_a84577df64d47642d0b8f1fee11b21a96.html

[6] Ferrão, José ; Dias, Paulo ; Neves, António J.: Detection of ARUCO markers using the quadrilateral sum conjuncture. In: *Lecture Notes in Computer Science* (2018), S. 363–369. http://dx.doi.org/10.1007/978-3-319-93000-8_41. – DOI 10.1007/978–3–319–93000–8_41

[7] https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html

[8] Nitsche, Matias: *ROS Workshop on Aerial Open-source Robotics.* 2015

[9] Lrse: *LRSE/whycon: Vision-based external localization system. note: Not currently being developed.* https://github.com/lrse/whycon

[10] Microsoft: *Microsoft/Azure-Kinect-sensor-SDK: A cross platform (linux and windows) user mode SDK to read data from your Azure Kinect device.* https://github.com/microsoft/Azure-Kinect-Sensor-SDK