Tampere University of Applied Sciences

Robot Camera to Probe Offset Calibration Comparison

Joona Muje

BACHELOR'S THESIS
March 2020

ICT Engineering
Software Engineering

ABSTRACT

Tampere University of Applied Sciences
ICT Engineering
Software Engineering

JOONA MUJE:
Robot Camera to Probe Offset Calibration Comparison

Bachelor's thesis 29 pages
March 2020

The purpose of this thesis was to compare the accuracies of camera to probe offset calibrations and offer recommendations for different use cases based on accuracy and other characteristics, such as speed and cost.

Camera to probe offset is required to link the image produced by the camera to the robot tool. With the link in place, an object can be detected in the camera image and manipulated (e.g. icon can be tapped) with the robot tool. Using CAD drawings for the measurement is not accurate enough for high precision applications due to manufacturing tolerances, so a calibration is required.

Data collection was done by automating the calibration process completely. Scripts were used to run the calibration multiple times from pre-defined starting values, and the results of each calibration were saved for later analysis.

Results were analyzed using Gaussian Kernel Density Estimation plots with 99% confidence interval and color based on temperature measurements. Dual Camera and Ball offset calibrations were the most accurate at under ±10µm 99% confidence interval. Tap offset calibration was likely limited by the touch screen calibration, and only achieved approximately ±200µm accuracy.

Based on the results, some improvements were suggested to improve the calibration accuracy and/or repeatability. Recommendations for different needs were made, based on accuracy, cost and speed.

camera, computer vision, robot, calibration, offset

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

JOONA MUJE:
Robotin kamera-työkalu etäisyyskalibrointien vertailu

Opinnäytetyö 29 sivua
Maaliskuu 2020

Työn tarkoituksena oli vertailla kamera-työkalu etäisyyskalibrointien tarkkuutta ja antaa suosituksia eri käyttötarkoituksiin perustuen tarkkuuteen sekä muihin ominaisuuksiin, kuten nopeus ja hinta.

Kamera-työkalu etäisyyskalibrointi tarvitaan yhdistämään kameran kuva robotin työkaluun. Kun etäisyys tiedetään, voidaan kuvasta havaita objekti ja sitä voidaan käsitellä (esim. ikonia voidaan painaa) robotin työkalulla. CAD malleista saatu mittaus ei ole riittävä sovelluksissa, joissa vaaditaan hyvää tarkkuutta, johtuen muun muassa valmistustoleransseista, joten kalibrointi vaaditaan.

Tiedonkeruuta varten kalibrointiprosessi automatisoitiin täysin. Kalibroinnit ajettiin skriptien avulla useita kertoja ennalta määritellyistä alkuasennoista ja tulokset kerättiin myöhempää analyysia varten.

Analyysissä tulokset piirrettiin gaussisella ydinestimoinnilla, 99% luottamusvälillä ja lämpötilaan perustuvilla väreillä. Dual Camera ja Ball kalibrointien 99% luottamusväli oli alle ±10µm ja olivat täten tarkimpia. Tap kalibroinnin tarkkuutta rajoitti luultavasti kosketusnäytön resoluutio, ja tarkkuudeksi saatiin ainoastaan noin ±200µm.

Tulosten perusteella tehtiin parannusehdotuksia kalibrointien tarkkuuden tai toistettavuuden parantamiseksi. Erilaisiin käyttötarkoituksiin tehtiin suosituksia tarkkuuden, hinnan ja nopeuden perusteella.

kamera, konenäkö, robotti, kalibrointi, etäisyys

CONTENTS

GLOSSARY

AR	Augmented Reality
VR	Virtual Reality
TnT Station	Touch and Test Station software
DUT	Device Under Test
API	Application Programming Interface
UI	User Interface
JSON	JavaScript Object Notation
PPMM	Pixels Per Millimeter
CAD	Computer-Aided Design
HID	Human Interface Device

1 INTRODUCTION

1.1 PURPOSE

Calibrations are needed to maximize the accuracy of robots and other equipment, where manufacturing tolerances are not high enough for the system requirements. Single parts are often manufactured with high tolerances, but as these parts are put together to build a complete system, the inaccuracies add up and can become problematic. If these inaccuracies did not exist, measurements from technical drawings and CAD models could be used directly, and system could be used without calibrating it (Mooring, Benjamin W. & Roth, Zvi S. & Driels, Morris R.).

This thesis was done for OptoFidelity to compare the calibrations used to calibrate the distance between the robot tip (typically a touch tool) and the camera. Camera to Probe offset calibration is required to match the image produced by the camera to robot coordinates, in order to, for example, tap on an icon detected on a smartphone screen. After the research and development phase of the Ball calibration ended, a better understanding of strengths and accuracies of the available calibrations was needed, to deliver robots fit for different use cases. Robot built for functional testing can only require accuracy measured in millimeters and can use a less accurate calibration method to reduce costs or speed up the calibration process.

OptoFidelity Oy is a company offering test systems for touch screen devices (such as mobile phones and tablets) as well as VR/AR devices. OptoFidelity systems are often customized based on customer needs and can include purpose-built tools and other hardware. Most common use cases include production line testers, measuring touch screen performance (e.g. accuracy, sensitivity, latency) and R&D testers for measuring overall or software-only (i.e. does a web browser stutter while scrolling?) performance.

1.2 CALIBRATIONS

1.2.1 CALIBRATION TYPES

OptoFidelity robots require calibrations for axes, cameras and offsets. Axis calibrations are required to be able to drive the robot to a position in the workspace accurately. Non-calibrated axes may have large errors, especially as the robot size grows. Axis calibration is typically done with a laser tracker but is also possible with a high accuracy magnetic encoder (Mooring, Benjamin W. & Roth, Zvi S. & Driels, Morris R.).

Distortion and exposure calibrations are most common camera calibrations, but some cameras also require flat field, dark frame and other corrections and/or enchantments. Distortion calibration calculates a correction matrix for the lens distortion and other imperfections in the physical camera (Tsai, Roger Y.), and exposure calibration finds the optimal exposure time for taking images with the most detail possible, without under- or overexposing the image. Offset calibrations are done with specialized tools, depending on the instrument being calibrated. This thesis specializes in measuring the offset between the robot tool and the camera.

1.2.2 CAMERA TO PROBE OFFSET CALIBRATION

Knowing the camera to probe offset allows for detections from camera image to be translated to robot coordinates, for example detecting an icon on mobile phone screen and tapping it using the robot tool. While this number is easy enough to measure from the CAD drawings, there are inaccuracies in camera or robot tool mounts, requiring calibration.

Tool changing can require re-calibration depending on the accuracy requirements and the tool changing mechanism. Aligning two different tools perfectly in the tool holder is practically impossible, but in functional testing small inaccuracies are often acceptable after one of the tools is calibrated.

2 CALIBRATION API

2.1 TNT STATION

OptoFidelity Touch and Test Station, typically referred to as TnT Station is a robot control software, that allows the user to calibrate robots, position DUTs and run custom scripts using an API, for example via Python client library. TnT Station connects to robots and cameras mostly using Ethernet, but USB or serial connections are uncommonly used.

Robot configuration is stored on the robot, and downloaded on startup, if not already present on the machine. This configuration includes axis specifications (e.g. axis name and length), cameras, calibrations and any other components relating to the robot and its functionality.

2.2 API DESIGN

Calibrations in TnT Station are typically run manually using the user interface. An automated way of running the calibrations was however required for this thesis and it was decided to create an API endpoint for running calibrations. While most components in TnT Station were designed to be highly modular, calibrations were tied quite heavily to the UI, which we had to work around and in places change.

Figure 1: Calibration flow from UI shows the calibration flow when a calibration is run from the user interface. When the user interface is started, user can select the correct calibration and the user interface will display any arguments that user needs to give. Once user gives the arguments and starts the calibration, the server backend checks that the arguments are valid and starts the calibration sequence. If the sequence fails for any reason, an error is displayed to the user, and otherwise the calibration results are shown.

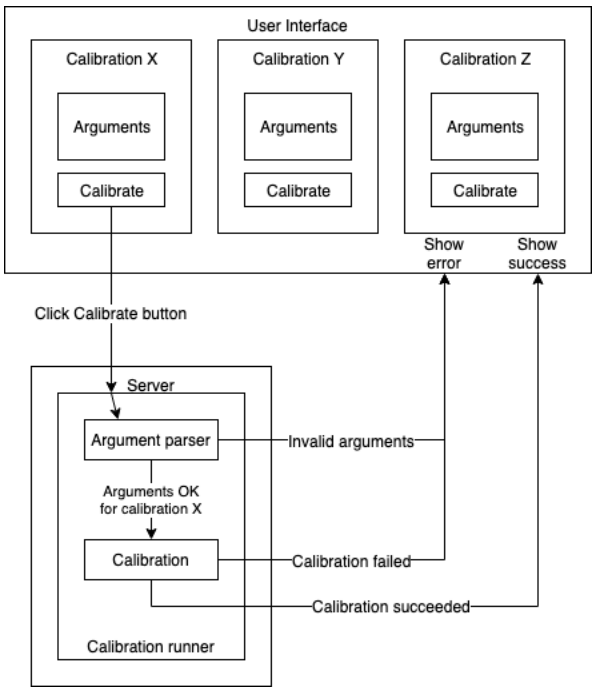

Figure 1: Calibration flow from UI

Code changes were required where the calibration results were returned to be shown on the UI, so that the code updating UI would only be triggered when the calibration is run from the UI.

Figure 2 shows the calibration flow when a calibration is run through the API. The calibration API has stricter error checking and handling than the UI, as the user can (not) provide any arguments they like. Server provides a PUT endpoint for running calibrations. User can run a calibration by creating a PUT request with the calibration name, and optionally the arguments required by the calibration. When server receives the request, the argument validity is checked, and calibration is started. Server will return an error or success message with the calibration results in the response.

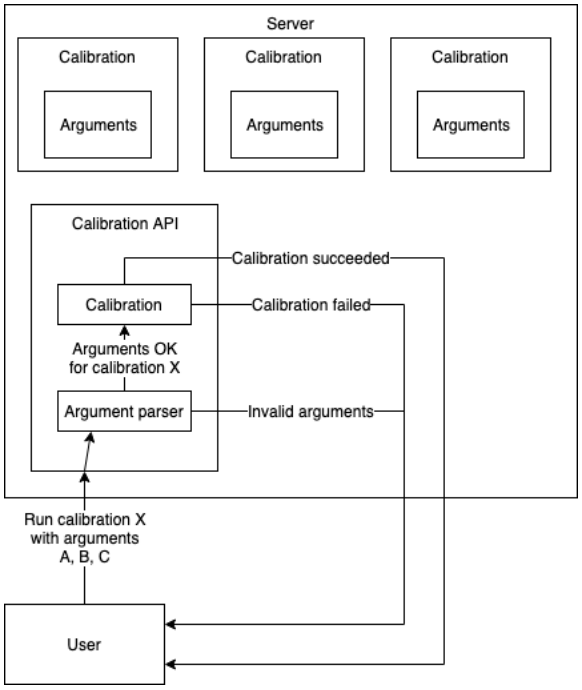

Figure 2: Calibration flow from API

Most of the code is shared between UI and API is shared, except for the code handling the execution of the required functions. Major difference lies in the fact that the calibrations run from the UI must be run on their own thread to avoid locking up the UI, while the API calls to the server automatically run in their own threads.

2.3 RUNNING CALIBRATIONS

Figure 3 shows the flow of the simple calibration script. When the script was started, robot and camera objects are initialized and connected to, and logging was started. A loop was then entered, where the robot was first moved to a pre-determined starting position, and the calibration started. Calibration result, which could be either success or error, was logged and the loop started again.

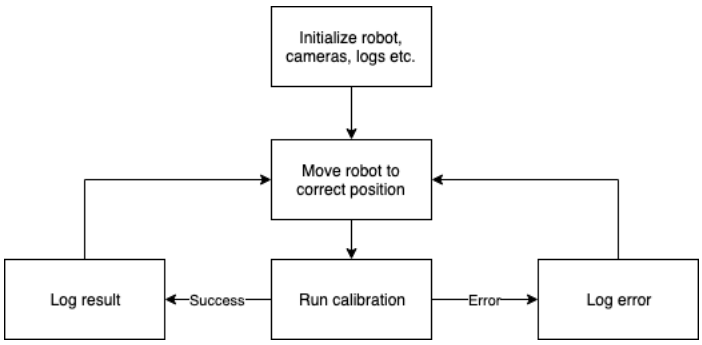

Figure 3: Calibration script flow

The scripts were written in Python and logged the results of the calibrations along with temperature and humidity values to a JSON-file. In case an error occurred, the calibration was restarted. Errors were typically caused by cameras losing connection or crashing.

3 TEST SETUP

3.1 ROBOT SETUP

OptoFidelity OptoStandard cartesian 3-axis portal robot (shown in Figure 4) was used in the measurements. Z-axis along with the touch tool and the camera were mounted on the X-axis. Unusually, the robot X-axis was mounted on the same robot base as Y-axis but moved independently. Most common cartesian robot axes are stacked on top of each other, instead of being separated as here.

Camera was *Basler acA2500-14gm* with *Edmund Optics 8mm/F1.8* lens. Camera requires Gigabit Ethernet connection to the computer with support for jumbo frames. Camera PPMM was approximately 36 (~0.028mm per pixel). *T&D TR-72nw* was used for temperature and humidity measurements, connected via Ethernet.

Figure 4: OptoFidelity OptoStandard cartesian 3-axis portal robot.

3.2 DUAL CAMERA OFFSET CALIBRATION

Dual Camera offset calibration used a special fixture, described in Figure 5, containing two *Basler acA1600-60gm* cameras (10) with *VS-TCH08-65* lenses at a 90-degree angle, with a *CCS TH-27X27SW* backlight (11) for both of them. An LED (12) was mounted at the crossing point of the camera images. The cameras were connected via Gigabit Ethernet with support for jumbo frames. Backlights were turned on by the cameras when capturing images, via output lines on the camera.

To calibrate, the camera is first centered on the LED to find the exact camera position, and then the robot tool is centered on top of the LED using the two cameras. Strong backlight causes the LED mounting and the robot tool to display as shadows in an otherwise bright white image.

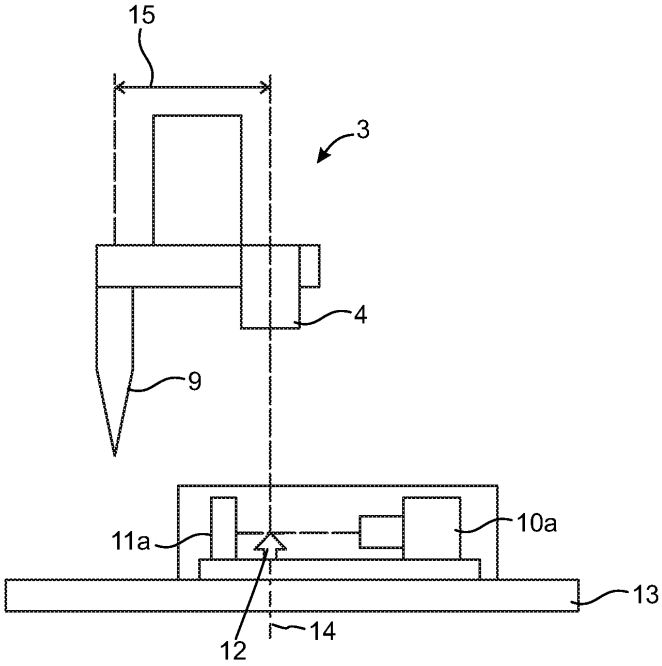

Fig. 3b

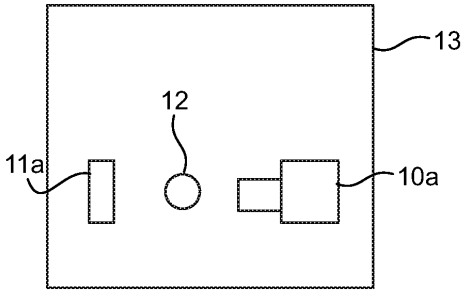

Fig. 3a

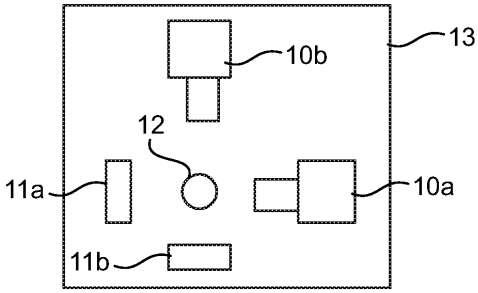

Fig. 3c

Figure 5: Dual Camera Offset Calibration setup (US20170339335A1)

3.3 BALL OFFSET CALIBRATION

Ball offset calibration used a standard 9mm brass finger electrically connected to the drives for X- and Y-axes. An electrically conductive sphere was placed on a tripod mount on the robot Y-axis, that was connected also connected to the drives

for X- and Y-axes. An electrical circuit was completed when the finger touched the sphere, like a switch.

To calibrate, the camera image is first centered on the ball, and then the ball is probed by the finger from multiple sides to complete an electrical circuit. When the circuit is completed, the exact robot position is stored to memory. With enough positions, the center point of the ball can be calculated, and used to calculate the camera to probe offset.

3.4 TAP OFFSET CALIBRATION

Tap Offset Calibration used a standard 9mm brass finger with an Apple iPhone 8+, mounted on the robot Y-axis with standard mounting brackets. The phone ran a custom application that displayed a circle in the last detected touch location.

To calibrate, the robot taps on a pre-determined location, and the phone displays a black circle on a white background at the position where the tap was detected. The camera is centered on the circle to find the camera position, from which the camera to probe offset can be calculated.

4 ANALYSIS

Python was used to analyze the results, because of familiarity with the language and the quality of the libraries.

Matplotlib, a 2D plotting library, was used for creating the graphical plots and histogram analysis. Matplotlib is used to visualize the results of many NumPy and SciPy examples. Figure 6 describes how to create a simple scatterplot like Figure 7 using Matplotlib. Example uses hardcoded X/Y-axis values, but they can be loaded from a file, or be generated at run-time.

```
import matplotlib.pyplot as plot

x_axis = [6, 9, 12, 8, 11, 18, 1, 4, 7, 13]
y_axis = [16, 20, 5, 6, 14, 12, 9, 4, 3, 18]
plot.xlabel("x-axis (mm)")
plot.ylabel("y-axis (mm)")
plot.scatter(x_axis, y_axis)
plot.show()
```

Figure 6: Matplotlib example to create a scatterplot of axis measurements.

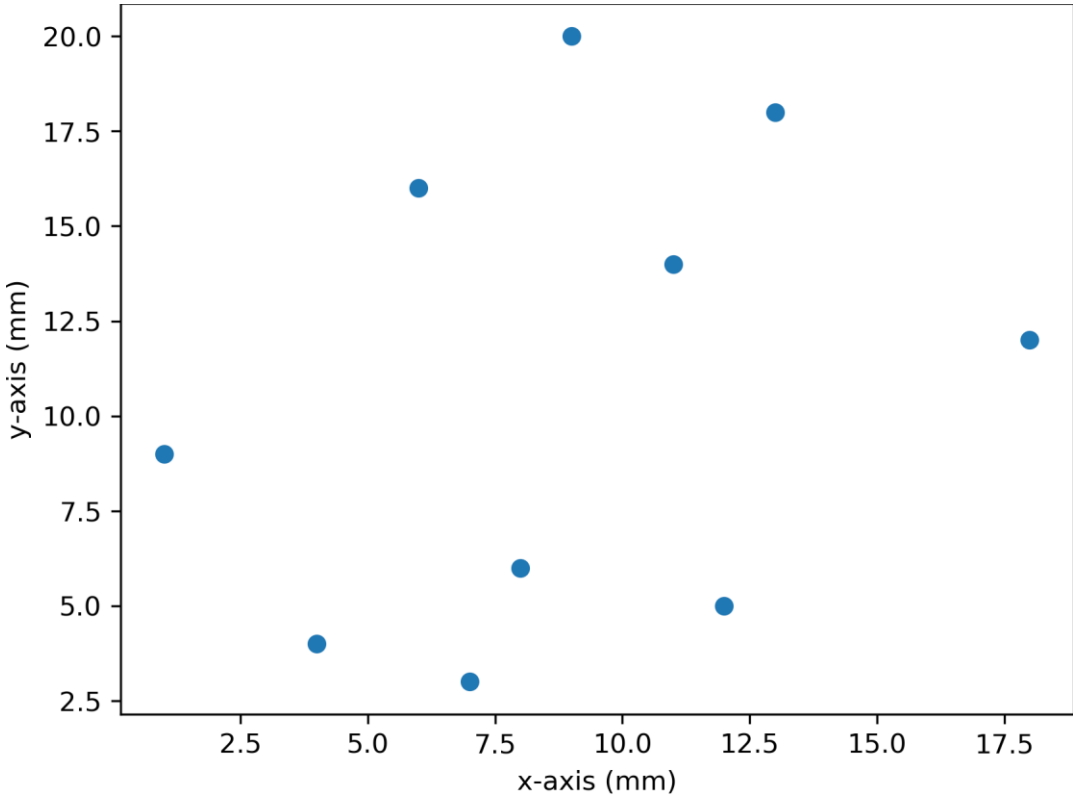

Figure 7: Matplotlib example scatterplot.

SciPy, a scientific analysis library that extends NumPy, a numerical analysis library, was used for Gaussian Kernel Density Estimation (Gaussian KDE) analysis. SciPy was chosen for Gaussian KDE due to its good documentation and easy usage.

```
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats

# Generate some random two-dimensional data
m1 = np.random.normal(size=2000)
m2 = np.random.normal(size=2000, scale=0.5)
m1, m2 = m1+m2, m1-m2
xmin, xmax = m1.min(), m1.max()
ymin, ymax = m2.min(), m2.max()

# Perform a kernel density estimate on the data
X, Y = np.mgrid[xmin:xmax:100j, ymin:ymax:100j]
positions = np.vstack([X.ravel(), Y.ravel()])
values = np.vstack([m1, m2])
kernel = stats.gaussian_kde(values)
Z = np.reshape(kernel(positions).T, X.shape)

# Plot the results
fig, ax = plt.subplots()
ax.imshow(np.rot90(Z), cmap=plt.cm.gist_earth_r, extent=[xmin, xmax, ymin,
ymax])
ax.plot(m1, m2, 'k.', markersize=2)
ax.set_xlim([xmin, xmax])
ax.set_ylim([ymin, ymax])
plt.show()
```

Figure 8: Example of SciPy Gaussian Kernel Estimation, adapted from official documentation (SciPy Reference Guide).

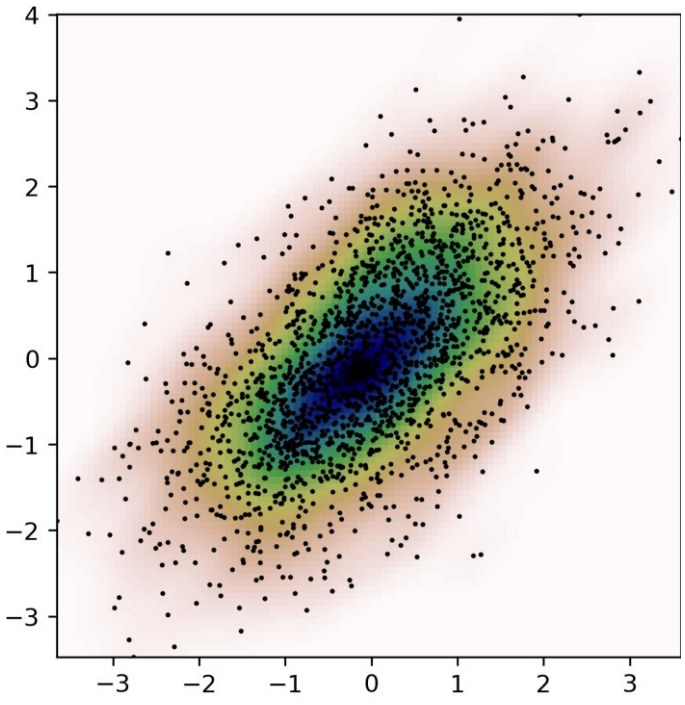

Figure 9: SciPy Gaussian KDE example plot.

The analysis for the gathered data consisted of calculating and plotting Gaussian KDE and a histogram. 99% confidence interval was calculated together with the Gaussian Kernel Density Estimation and included in the graphs. Gaussian KDE was used instead of normal distribution because normal distribution would not achieve a good fit in some cases. Colors were added to the scatterplots in order to analyze the effect of temperature on the measurements.

5 RESULTS

5.1 DUAL CAMERA OFFSET CALIBRATION

Figure 10 shows the X- and Y-axis offset measured in each measurement. The results are mostly contained within ±0.01mm box, which can be considered extremely good when taking in to account the PPMM of the camera. There is no visible relation between temperature and the results.

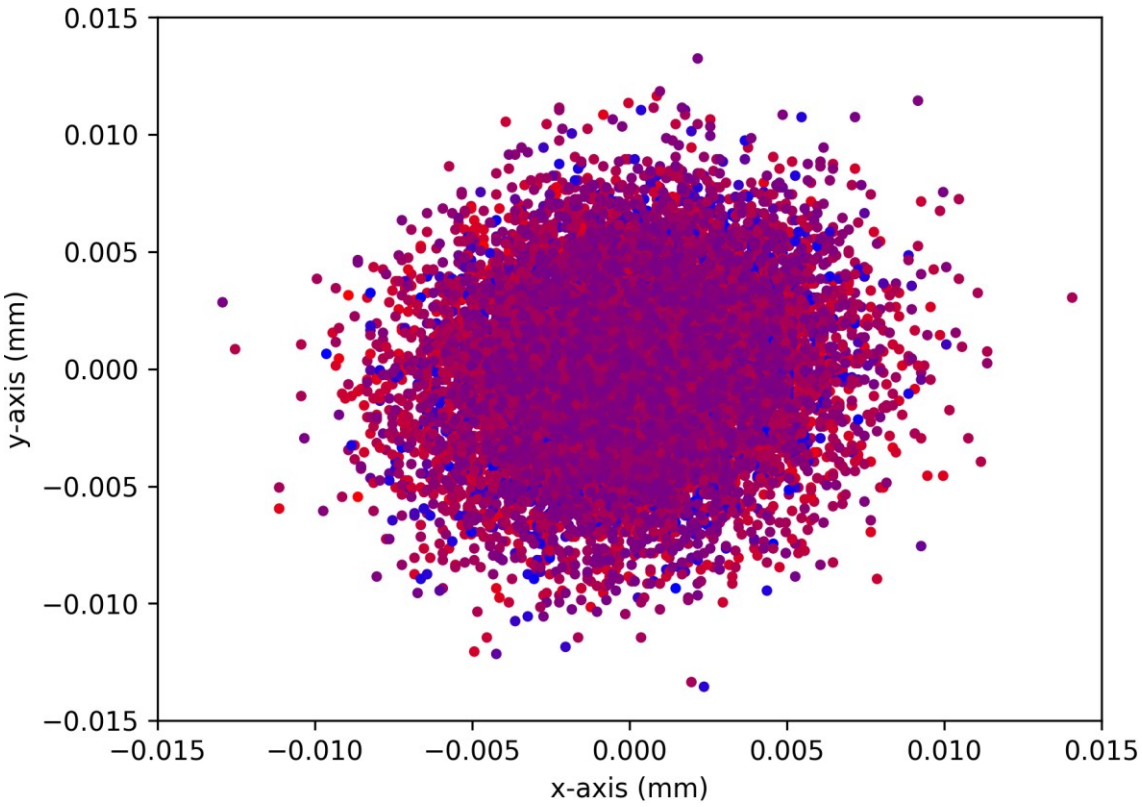

Figure 10: Dual Camera Offset Calibration X/Y scatterplot

Figure 11 and Figure 12 show the Gaussian KDE, histogram and 99% confidence interval for both axes. The plots seem to be normally distributed, however Gaussian KDE was used for consistency with other results.

X-axis 99% confidence interval was +7.23/-7.17 µm and Y-axis +7.77/-8.16 µm.

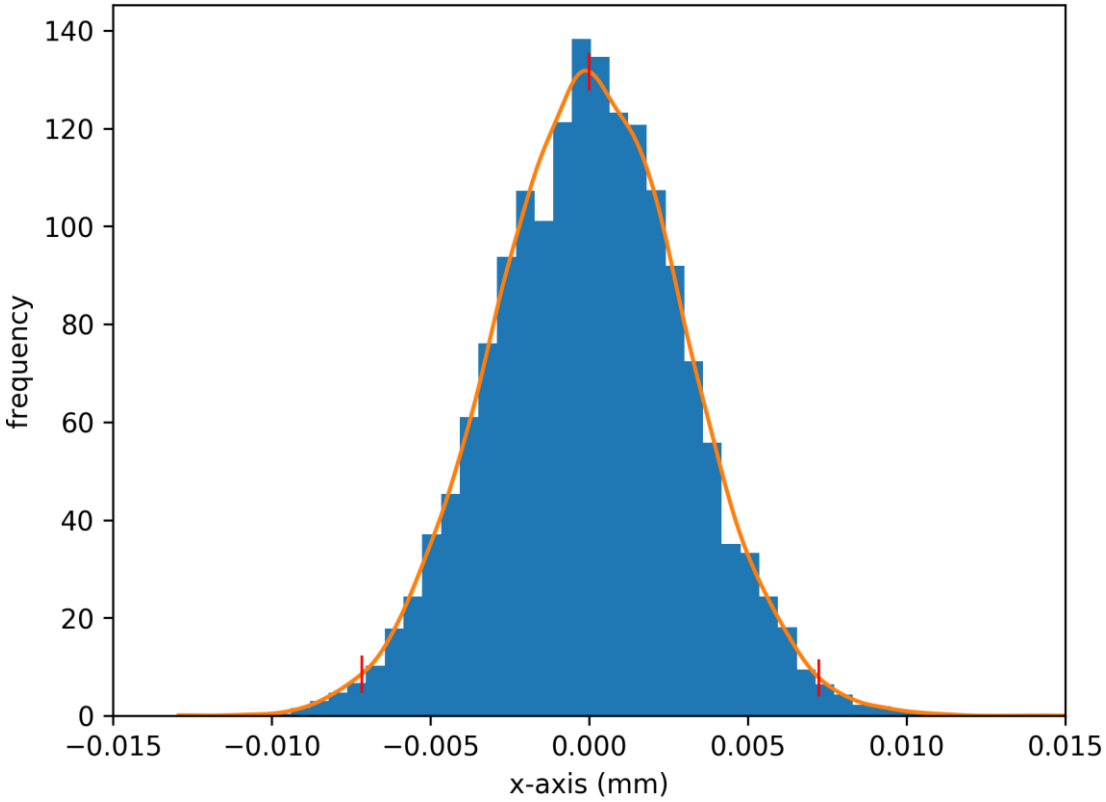

Figure 11: Dual Camera Offset Calibration X-axis Gaussian KDE and histogram

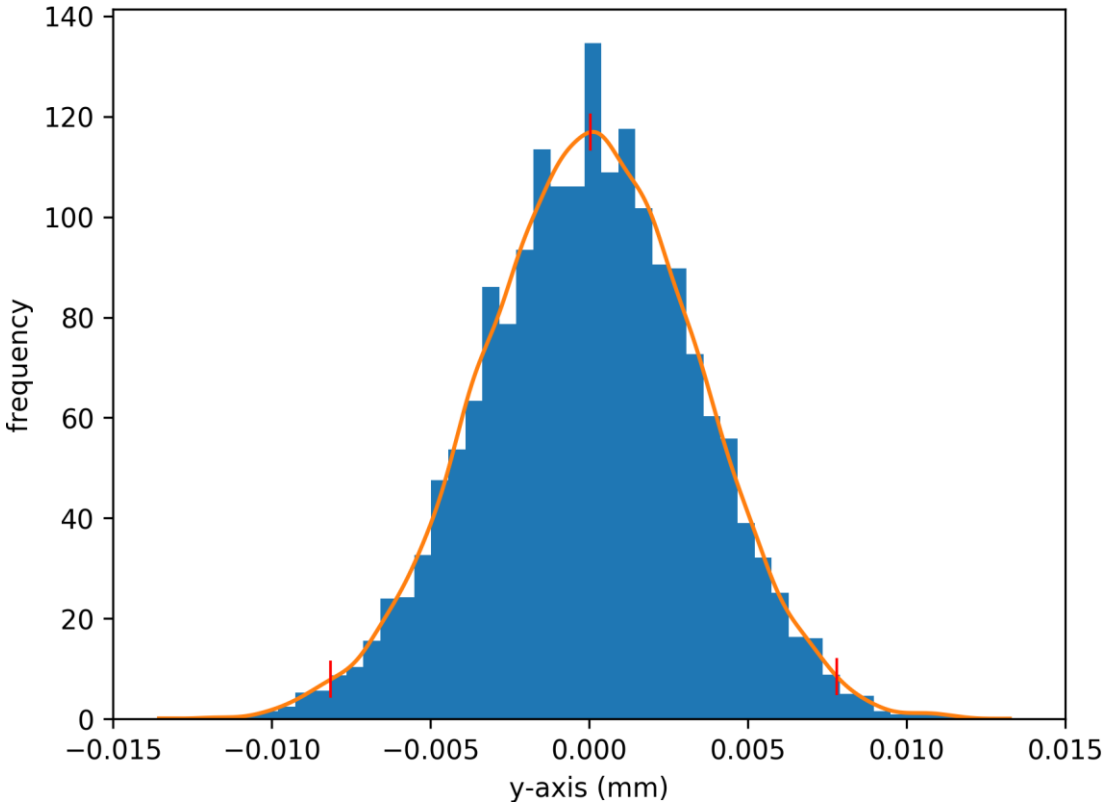

Figure 12: Dual Camera Offset Calibration Y-axis Gaussian KDE and histogram

5.2 BALL OFFSET CALIBRATION

Figure 13 shows the X- and Y-axis offset measured in each measurement. The results are mostly contained within ±0.01mm box, which can be considered extremely good when taking in to account the PPMM of the camera. There is noticeable relation between temperature and the results, with X-axis shifting in positive direction as temperature lowers, and Y-axis shifting in positive direction as temperature raises.

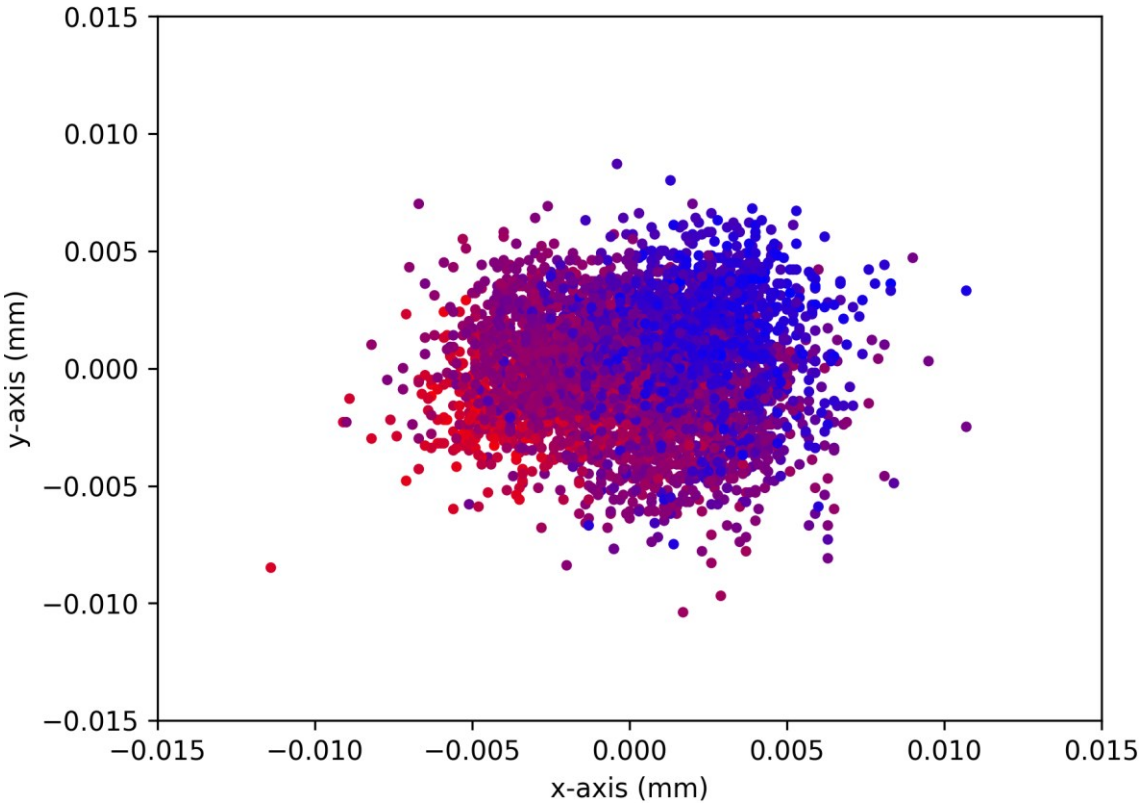

Figure 13: Ball Offset Calibration X/Y scatterplot

Figure 14 and Figure 15 show the Gaussian KDE, histogram and 99% confidence interval for both axes. The plots seem to be mostly normally distributed, however Gaussian KDE was used for consistency with other results.

X-axis 99% confidence interval was +6.39/-6.07 µm and Y-axis +5.45/-6.22 µm.

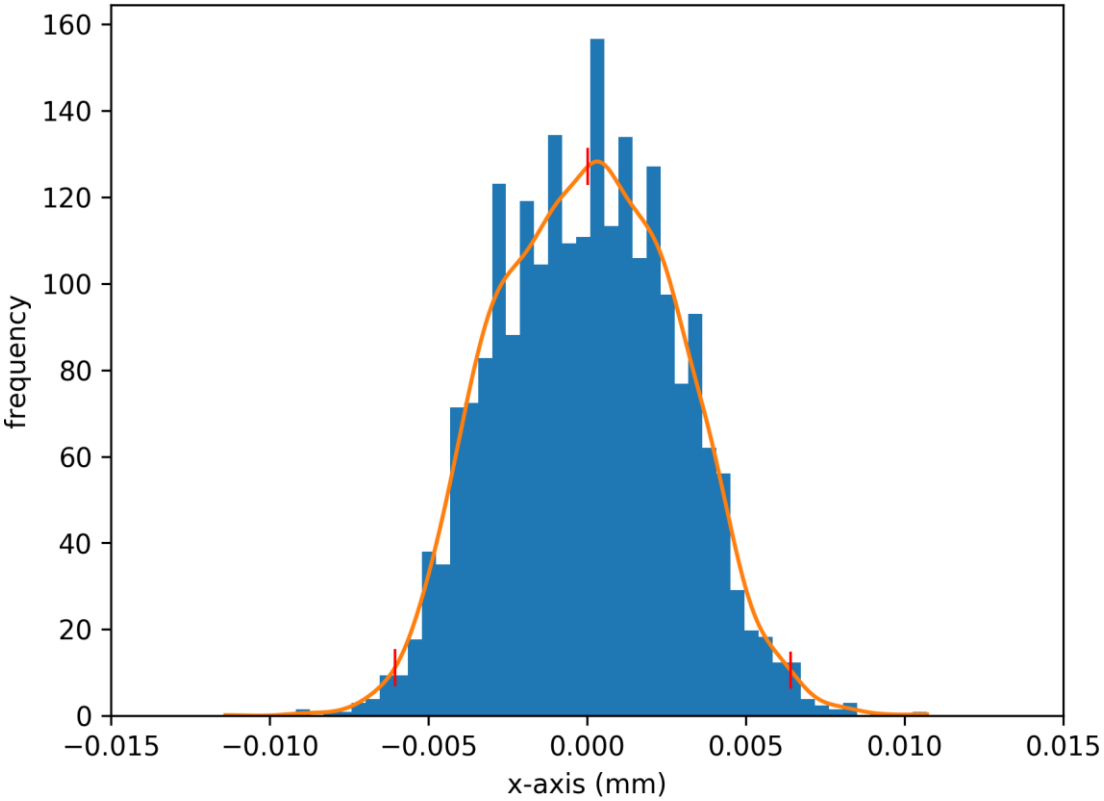

Figure 14: Ball Offset Calibration X-axis Gaussian KDE and histogram

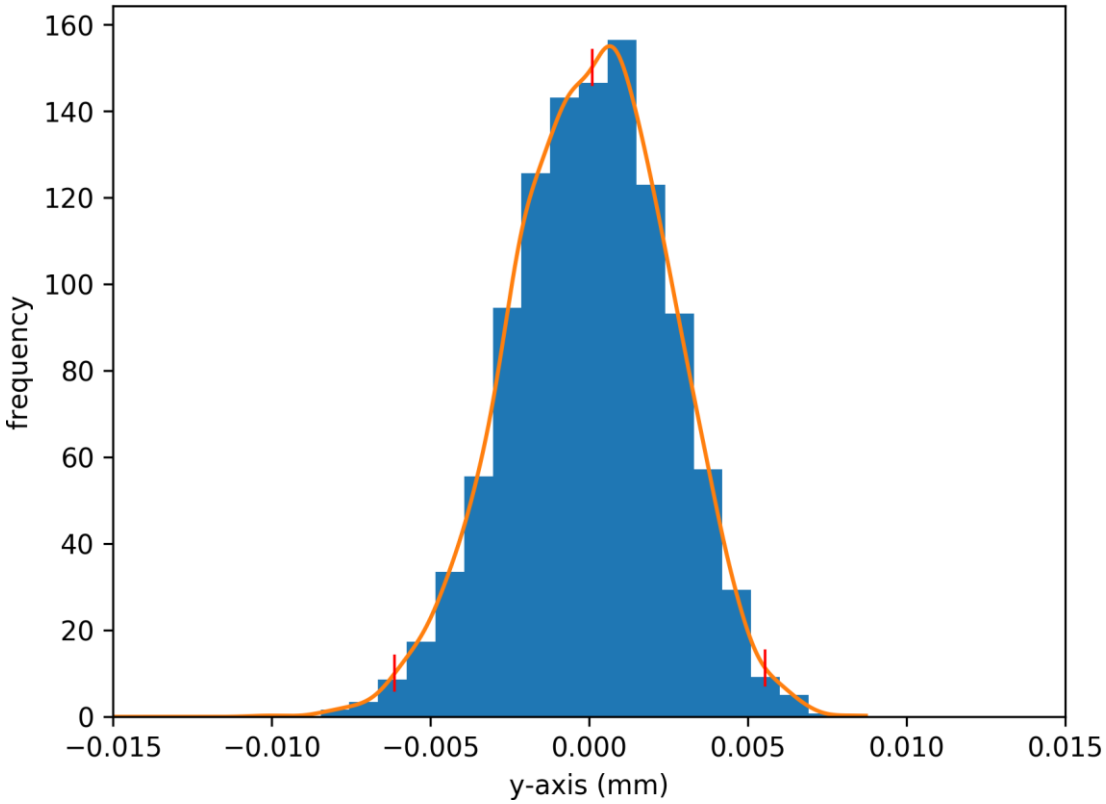

Figure 15: Ball Offset Calibration Y-axis Gaussian KDE and histogram

5.3 TAP OFFSET CALIBRATION

Figure 16 shows the X- and Y-axis offset measured in each measurement. Note the increased axis values. The results are grouped with approximately 0.17mm between each group on X/Y-axes. Grouping is caused by the device's touch screen resolution. Effect of temperature cannot be seen in the plot.

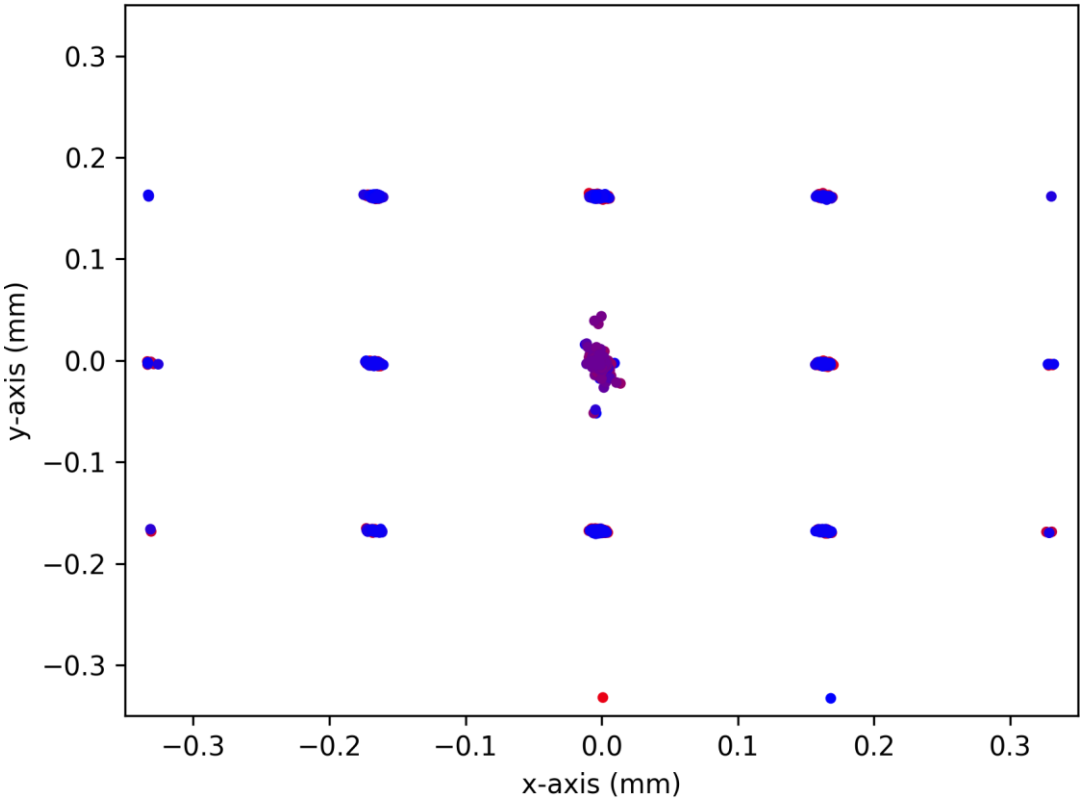

Figure 16: Tap Offset Calibration X/Y scatterplot

Figure 17 and Figure 18 show the Gaussian KDE, histogram and 99% confidence interval for both axes. Note the increased axis values. The plots are not normally distributed, because the touch screen resolution caused grouping.
X-axis 99% confidence interval was +185/-185 µm and Y-axis +166/-185 µm.

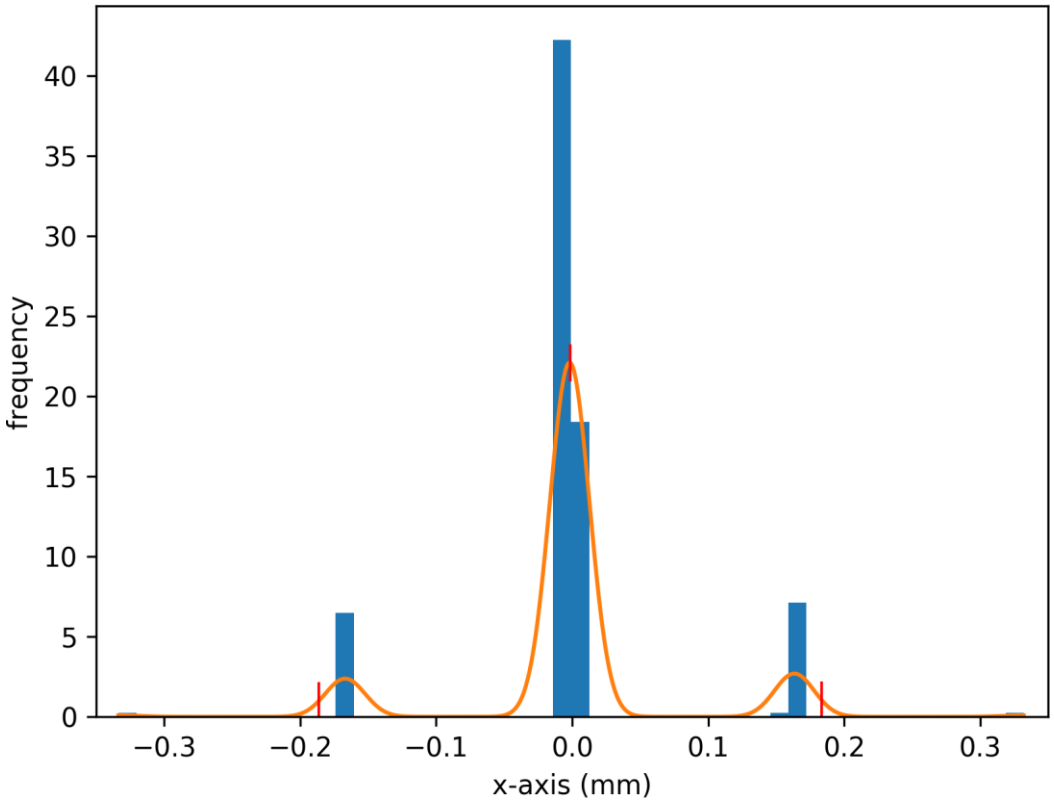

Figure 17: Tap Offset Calibration X-axis Gaussian KDE and histogram

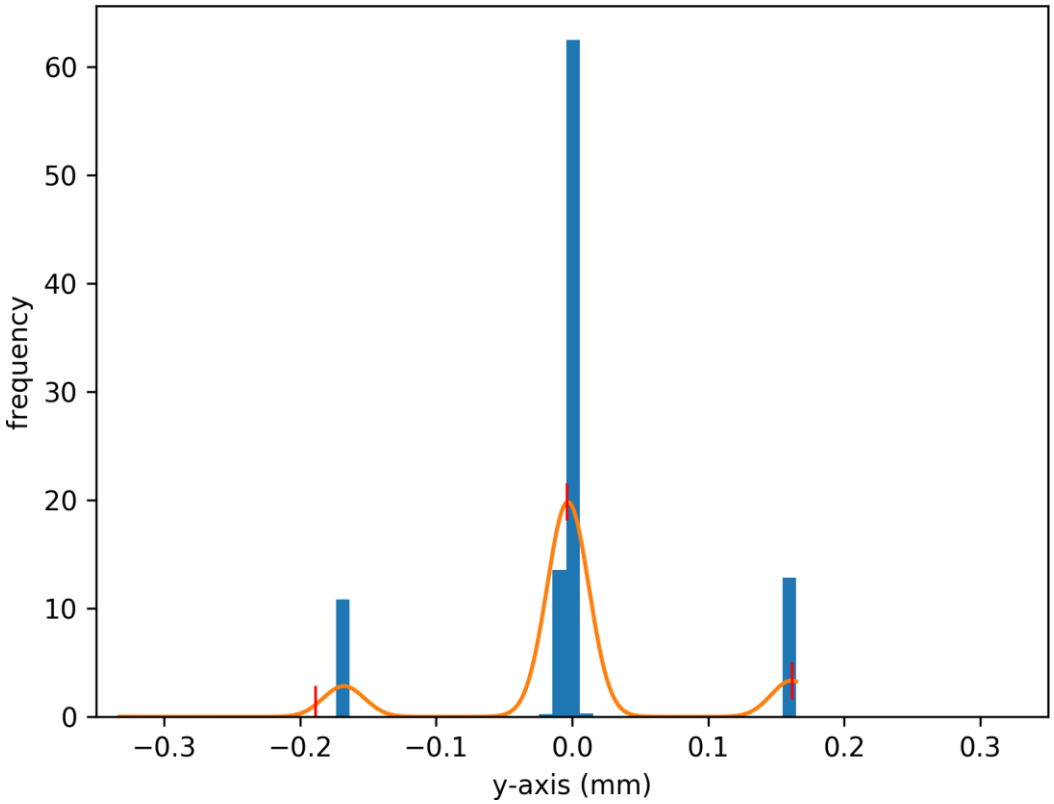

Figure 18: Tap Offset Calibration Y-axis Gaussian KDE and histogram

6 FURTHER IMPROVEMENTS AND RECOMMENDATIONS

Further improvements to the accuracy of all calibrations can be found by increasing the PPMM of the camera, for example with a higher resolution camera or a telecentric lens. This will allow detection algorithms to be more accurate in detecting patterns, such as the LED in Dual Camera calibration. This may not be possible in all applications, as features larger than the calibration pattern may need to be detected by the end user.

Tap offset calibration results could be improved by using a device with higher resolution touch screen. Such touch screen devices are however uncommon, as touch screens are used as a human interface devices (HIDs) where accuracies required are not as high as robots can typically reach (Microsoft Hardware Dev Center).

Ball offset calibration results could be improved by using a temperature-controlled environment or less temperature sensitive materials. It’s suspected that the ball or the ball mounting moved slightly due to thermal expansion.

Figure 19 is a recommendation matrix created based on the analysis results. Based on the matrix, recommendations can be made based on specific needs, for example if a fast and accurate calibration method is needed, Dual Camera is the best solution. However, if cost is an issue, and speed can be sacrificed, Ball can be a suitable alternative.

	Cost	Speed	Accuracy	Ease of Setup
Cost		Tap	Ball	Tap
Speed	Tap		Dual Camera	Dual Camera Tap
Accuracy	Ball	Dual Camera		Dual Camera Ball
Ease of Setup	Tap	Dual Camera Tap	Dual Camera Ball	

Figure 19: Recommendation matrix

7 CONCLUSION

The accuracy of the calibrations in this thesis were in the expected range, but the effect of temperature at the scale it was seen with Ball offset calibration was surprising. No issues were found in either Dual camera nor Tap offset calibrations, and it is recommended to use one of them depending on the system requirements.

Future calibration testing should be completed in a temperature-controlled environment to minimize the thermal expansion in the whole system, including robot axes and the calibration fixtures.

The work done to test the calibrations, especially the API for TnT Station, can be integrated in the software offered by OptoFidelity to improve the calibration process and analysis.

REFERENCES

Mooring, Benjamin W. & Roth, Zvi S. & Driels, Morris R. 1991. Fundamentals of Manipulator Calibration.

Tsai, Roger Y. 1987. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses.

Microsoft Hardware Dev Center. Touch Accuracy. Accessed 12.02.2020. https://docs.microsoft.com/en-us/windows-hardware/design/component-guidelines/touch-accuracy

SciPy Reference Guide. scipy.stats.gaussian_kde. Accessed 12.02.2020 https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html

US20170339335A1. Finger camera offset measurement. Accessed 09.01.2020. https://patents.google.com/patent/US20170339335A1