A Monocular Vision-based System Using Markers for a Real-Time 6D Pose Estimation of a Trailer

ROB 10

Natalie Mark

Robotics
Aalborg University

AALBORG UNIVERSITY
STUDENT REPORT

Robotics
Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg
http://www.robotics.aau.dk

Title:
A Monocular Vision-based System Using Markers for a Real-Time 6D Pose Estimation of a Trailer

Theme:
Computer Vision for Robotics

Project Period:
Spring 2022

Project Group:
1068d

Participant(s):
Natalie Mark

Supervisor(s):
Andreas Møgelmose
Galadrielle Humblot-Renaux

Copies: 1

Page Count: 56

Date of Completion:
June 1, 2022

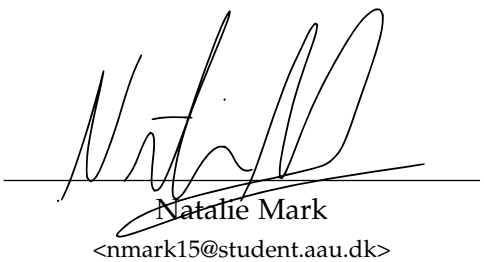
Abstract:

This project investigates the possibilities in object detection and pose estimation with a single monocular camera without the use of neural networks. The object in need of localization is a trailer. This report dives into current marker design along with current methods for detecting these designs. Furthermore, this report describes different sensor systems, along with the advantages and disadvantages of them to compare. Rather than detecting the object itself, a marker has been created to attach onto the object. The marker consists of two spheres of the color blue and orange. Detecting the marker, robotic perception method has been implemented, such as Hough circle detection and color segmentation. The pose estimation system works within a distance of maximum 560mm, when both of the spheres are in the camera frame. Last, the unexpected and expected test results have been discussed, along with thoughts upon future work.

Preface

Empty

Aalborg University, June 01, 2022



Natalie Mark
<nmark15@student.aau.dk>

Contents

1 - Introduction

Capra robotics is a company that develops mobile robots for outdoor environments[1], see figure 1.1. The robot is a platform that can be used for various applications. One of the applications that Capra wants to explore further is the ability to drive autonomously with a trailer attached to the robotic platform. Not only do they wish for the robot to be able to drive with a trailer but also to pick up the trailer by command. In order to do so the robot would need to be able to locate and attach the trailer to itself autonomously. The trailer can bee seen on figure 1.2 and is designed such that the end will be attached to a knob. The knob is placed on the top center of the base of the robot.

Figure 1.1: The base of the mobile robot platform by Capra Robotics

Figure 1.2: The trailer that can be attached to the robot. A close-up of the end of the rod.

We will focus on developing a solution for Capra Robotics such that the mobile robot can attach a trailer to itself.

1.0.1 Trailer Detection

Detection of a trailer can be done with state-of-the-art processes, such as using deep-learning for feature extraction[2]. The reason to use methods such as deep-learning feature extraction for classification and trailer detection would be because the outcome could be trained to be highly accurate and that deep learning architecture is flexible to be adapted to new problems in the future. The disadvantage to this method is the need for a very large amount of training data and the complex and expensive in relation to computation power, requiring expensive GPUs. It can be very time consuming to adapt in order to obtain a high accuracy[3].

For this project, we want to explore other possibilities for trailer detection that does not require the use of high computational power, such as deep-learning. A more simple possibility for detecting objects in space is to create point of reference on the trailer, such as a marker. Using markers as a reference point have earlier been used in the application of vision systems to navigate mobile robots[4]. There are both active markers and passive markers. Some examples of active markers being LEDs, RFID tags or using infrared. These all requires specific hardware attached to the trailer, in order to function fully[5][6]. Examples of the passive markers are retro reflective spheres, planar dots, QR code or fiducial markers[7]. The reason to use methods such as markers is to simplify the task of detecting a complex shaped object. The disadvantage of using passive markers is that the markers often are small, resulting in possible occlusion. Often, to avoid this problem, multiple markers are used[7].

1.1 Project Scope

The overall process of driving with a trailer can be divided into sub-tasks and can be described step-by-step. A visual representation of the tasks or steps can be seen in figure 1.3.

Figure 1.3: Process of driving with a trailer autonomously divided into steps

The goal of this project is to develop a camera based solution for **Step03 - Location and Orientation**. Step 03 starts when the trailer is within 1m to the camera of the robot and is located in the robot's field of view (FOV). Here it is needed to determine both the location and the orientation of the trailer with high accuracy and a precision of $\pm$10mm. We are specifically looking into detecting a marker placed on the trailer to simplify the recognition process. The marker must be passive sue to the limitation of hardware attached to the trailer, that would be needed in case of active markers.

To guide the research in the desired direction, the following research questions have been developed:

- In order to locate the trailer, an identifier should be created and attached to the trailer. What object is applicable to use identifying the location and orientation of the trailer and where should it be located?
- What camera is suitable for identifying and localizing the trailer in an outdoor environment?

1.2 Report Structure

Chapter 2 introduce solutions related to the scope of this project, exploring different marker designs and existing frameworks. **Chapter 3** describes various sensor systems, reviewing their general characteristics. **Chapter 4** provides the implementation of the system in detail, presenting the design of the marker along with the software implementation. **Chapter 5** describes the various tests, that is in **Chapter 6** then evaluated and concluded upon. **Chapter 7** will discuss the tests along with future improvements.

2 - Related Work

To create a camera based solution for locating a trailer in the Cartesian space along with its orientation, it is needed to study related projects and state-of-the-art methods. This chapter will describe different solutions and algorithms for solving similar problems. First a description and analysis of different solutions regarding the design of markers for pose estimation has been done, followed by a summary of different algorithms and specific methods that is considered a possible solution for this project.

2.1 Design of Marker for Pose Estimation

First it is wanted to analyse the different designs of calibration tools in the field of robotics in order to grasp the possibilities and limitations of current research methods. This section focuses on three different possible marker-types that can be used in the development of a solution for this project.

2.1.1 2D Shapes

QR codes have been used for many years and consists of high data capacity, reducing space printing with a high speed reading[8]. QR codes have been used in various aspects, such as for estimating the position and pose of a camera[9] and for mobile robot localization and navigation[10] where QR codes are used as landmarks to provide a global pose reference. A similar but different approach is the circular marker that has been proposed in [11]. The marker is a circle that consists of three rings, the outer rind is always black. The middle and inner ring is divided into bins that is either black or white. There exist a large amount of different 2D fiducial marker systems besides QR codes and the circular marker[12], these can be seen on figure 2.2.

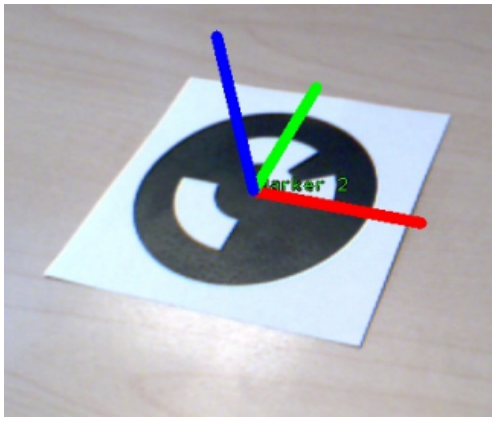


Figure 2.1: The circular marker design[11]

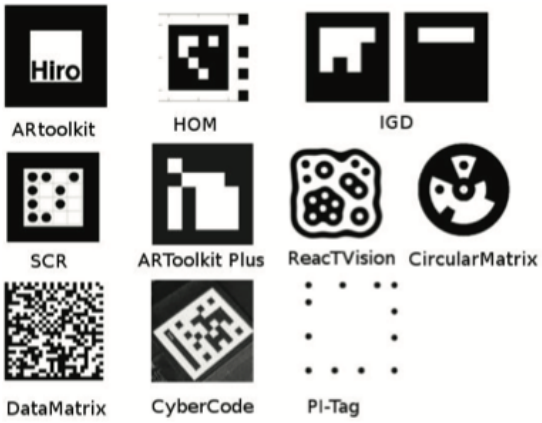


Figure 2.2: Variety of different 2D markers[12]

2.1.2 3D Shapes

Automation of a robotic surgical assistant requires precise movement. In order to calibrate the positions of the robots, [13] propose a solution being 3D-printed spherical fiducials being mounted on the end-effector and robot arm, see tool on figure 2.3. Specifically two red spheres have been placed on the shaft near the end effector and four different colored spheres in a cross-shaped reference frame have been placed on the end-effector. RGB-D images are used to estimate the robot's ground truth joint configurations relative to the commanded joint configurations. The proposed measurement system has a root mean square error of 0.32mm with a standard deviation of 0.18mm for single sphere detection.

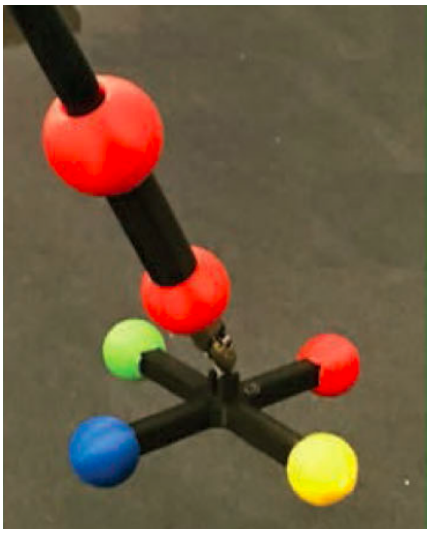


Figure 2.3: Tool for estimating position of the end-effector, using multiple colorful spheres[13]

The paper [14] is researching a solution for an augmented reality system using a color coded cube for calibration. Each side of the cube has a unique luscious, matte color. The matte color eliminate some of the highlights reflected from light. In this research first the geometrical form is identified by finding the corners of the cube. When a cube has been detected, each visible side of the cube are segmented as one region and used for determining the position of the cube. To do color segmentation, a color classifier was trained. The part most prone to errors is the color segmentation due to different lighting conditions. The proposed framework works offline and has a recognition rate of the cube that ranges between 74% and 93% depending on the illumination of the scene.

2.1.3 2D Patterns on 3D Shapes

The research paper [15] proposes a real-time pose estimation tool for an endoscopic instrument. The marker consists of a white paper with squares printed onto, that is designed to be wrapped around the tip of a cylindrical surgical device, as seen on figure 2.4. The squares are placed such that at least one will be visible to the camera at any instant and orientation. The tool has a green band at one end, and is used to calculate the distance from the detected square to the green band in order to estimate which square is detected. The green band is used as the global reference and the center of the square module is defined as a local reference of the marker. The paper shows that the pose estimation has an accuracy of 1.286mm with a standard deviation of 0.673mm and a mean rotational error of 1.497° with a standard deviation of 0.873°. The proposed framework copes with different lighting conditions along with corner, edge and whole marker occlusion under static and dynamic conditions.

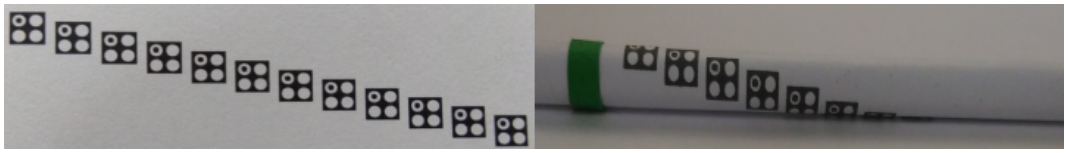


Figure 2.4: Left: Planar view of the paper with pattern, before wrapping onto the tool. Right: Tool after wrapped with the printed pattern[15]

A research on surgical tool tracking and pose estimation have resulted in the design of the *hybrid cylindrical marker*[16]. The hybrid marker consists of circular dots and chessboard vertices all places in a specific pattern on a cylinder, as seen on figure 2.5 and figure 2.6. The placements and rotation of the chessboard vertices are thought out such that the pattern is unique when detecting them on the 3D printed body. Using an RGB camera, they capture an image of the pattern on the cylinder, running it through a marker detection algorithm, identifying the patterns

and uses it for pose estimation. The proposed tracking framework allows large motions of both rotations and translations with a detection rate of 99.7% - 100% with a pose estimation accuracy, for the diameter of the cylinder being 6mm, of 1.43mm with a standard deviation of 1.09mm and a mean rotational error of 0.55° with a standard deviation of 0.38°. The proposed framework copes with self- and partial occlusion.

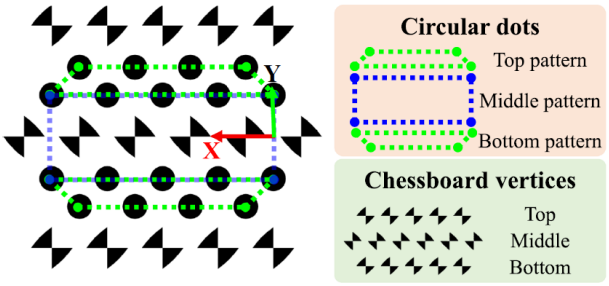


Figure 2.5: Top view of the pattern along with the specifications of the circular dots and chessboard vertices[16]

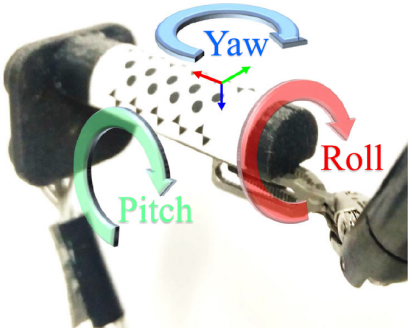


Figure 2.6: Coordinate system of the pattern on a 3D printed body[16]

2.2 Computer Vision Methods

The methods described in this section are possible approaches in computer vision, used to detect a marker for pose estimation. Method described here will be based on the previous section and the possible solutions for detecting the different markers proposed.

2.2.1 Detecting 2D Shapes

The need for detecting QR codes and similar 2D shapes have resulted in multiple existing solutions in respect to algorithms. Each different marker design follow a specific rule of design and with it an algorithm. It is fairly simple to find a working, open-source algorithm for detecting 2D markers, such as the QR code.

The structure of the QR code is a pattern of black and white squares, separator symbols, timing patterns, alignment patterns, information and much more, as seen on figure 2.7. The structure of a QR code can be split into two groups: Function Region and Encoding Region. The function region consists of detecting the marker as a QR code. The encoding region consists of the specific information that the QR code contains.[17]

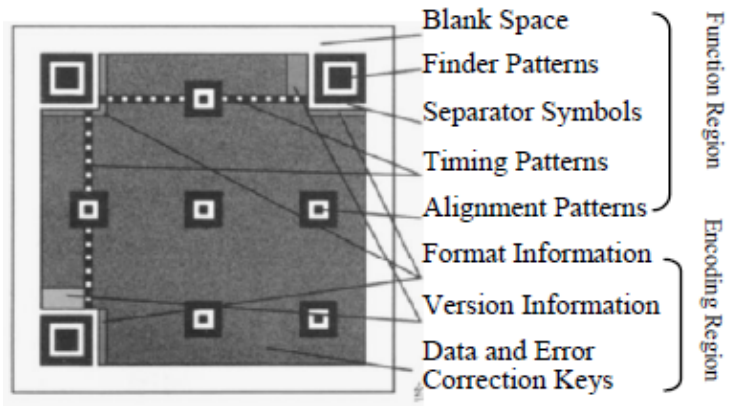


Figure 2.7: The structure of a QR code[17]

The paper [17] proposed a decoding algorithm that uses binarization, localization of the QR code, geometric rectification, localization of alignment patterns and image sampling.
Binarization is the act of changing the value of a pixel in a gray scaled image to either 1 or 0, using a threshold value k. The the next step is to **Localize the Finder Patterns**, being the three big squares in the corners of the cube. After these are found, **Geometric Rectification** is used to correct the QR code to a square. The image of the QR code may have been shot by an angle and it is wanted to correct this. The paper [17] solves this by using affine transformation. Next is to **Localize the Alignment Patterns**. Its purpose is to check if the distortions are detected and corrected. Last step is **Image Sampling**. The QR Code has been detected and now the version information is read and used to determine the size of the QR code. Then the bits represented by each grid, either 1 or 0, is obtained. The steps for decoding a QR code is shown in figure 2.8.

Figure 2.8: From left to right, the images represent: (a) Raw Image. (b) Binarization with k=128. (c) Localization of Finder Patterns. (d) Affine Transformation. (e) Localization of Alignment Pattern(s)

A method, called FastQR[18] has created an algorithm for pose estimation using multiple QR codes with monocular system. The paper concludes the system to be of low cost, small volume, low computing power and fast speed. But due to the size of QR code, camera resolution and illumination, the error will increase with

the increase in distance from the object to the camera. The system has a relative high X- and Y directional error when the camera has a distance of 400mm and more.

2.2.2 Detecting 3D Shapes

In order to discuss detection and classification of 3D shapes it is relevant to narrow the specific shapes which we will be looking into. We choose to research methods for simple 3D shapes, such as spheres and cubes.

A solution for detecting 3D spheres for calibration purposes has been proposed by [13]. The solution requires an RGB-D camera to track the fiducials of colored spheres. The paper states that the result is more robust in relation to detection of the spheres regardless of image background by masking both depth and color ranges. The end-effector has 4 spheres attached to ensure detection of at least 3 in case of occlusion. To detect the spheres in the image, they implement image segmentation using OpenCV. To then calculate and localize the center of each sphere, 3D points from the RGB-D camera is used.

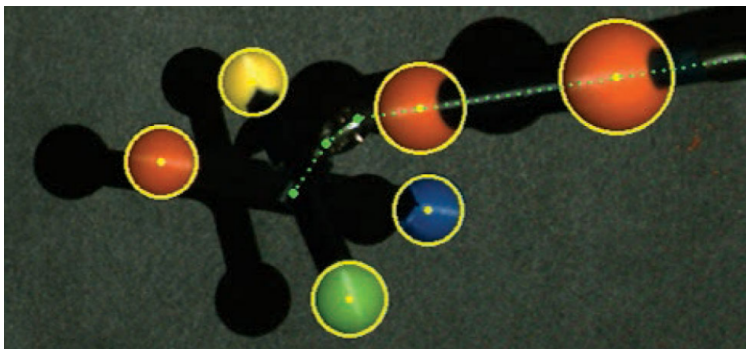


Figure 2.9: Yellow circles and dots indicating the detection of spheres and their center2.3

Another paper proposes a method for detecting multiple 3D objects in a scene, using and RGB-D sensor and linear spatial pyramid matching (LSPM)[19]. The sensor used for the proposed method is the Kinect sensor, which is using a combination of an infrared (IR) light projector and a simple camera. The system proposed by [19] can be divided into four sections: Acquisition and pre-processing, depth map analysis, object detection and filtering, image cropping.

Acquisition and pre-processing consists of capturing an RGB image and its corresponding depth image. The depth image may contain regions with no information. This can be the result of the IR light does not reflect well on all surfaces or because the object is located out of the sonsor's FOV. Different depth sensors

have different measure distances. The Kinect sensor can measure from 80cm up to 400cm[20]. The pixels within regions with no information will be assigned the depth value zero. To correct the zero values, these are replaced by their nearby non-zero values, see figure 2.10.

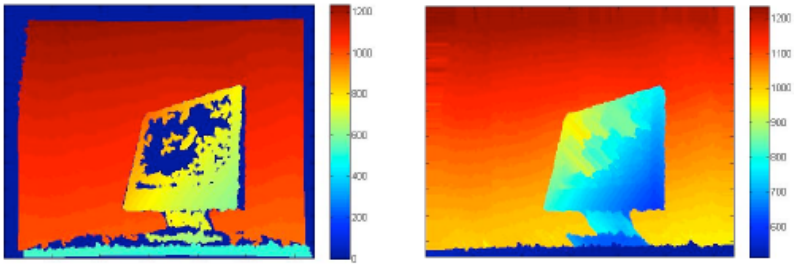


Figure 2.10: Depth image before and after depth normalization[19]

Depth map analysis consists of edge detection. The normalized depth image is processed with convolution by a 2D Gaussian filter to produce a gradient image to detect edges in the image. Edge detection is done on the depth image and not on the RGB image due to edges being detected where luminosity changes sharply. In an RGB image an object may have a sharp change in color, resulting in detection of edges that may not represent the real object boundaries. After the edges have been detected, small fragments of the actual edges might be missing. To ensure that the detected edges are connected a morphological closing algorithm is used, see figure 2.11.

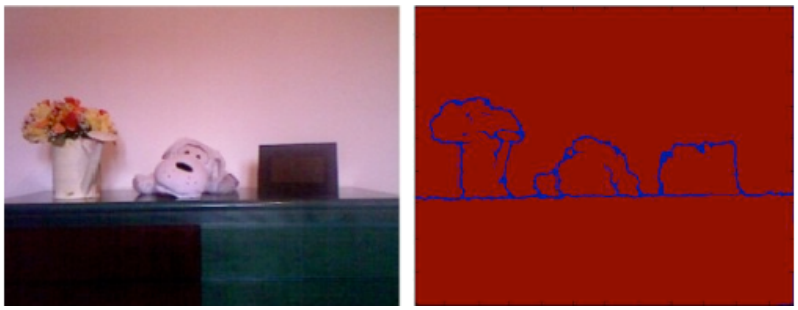


Figure 2.11: From left to right, the images represent: (a) RGB input image. (b) Depth image after edge detection and edge closing algorithm[19]

Object detection and filtering consists of differentiating between the detected objects. The connected components algorithm is then used on the image to gain useful information of each object such as the area of each object in pixels, the extrema points, the bounding box for each component etc. Figure 2.12 visualize the result of the use of connected components algorithm, here each object is given

a different color for us to differentiate between detected objects. After these objects are detected, next step is to filter out what is needed and what is not and is done so in three phases. First phase is to filter the object by their relative size. Second phase is to filter based on their width and height in pixels. Third phase is to filter out which objects are considered background. These filters are adjusted depending on user need.

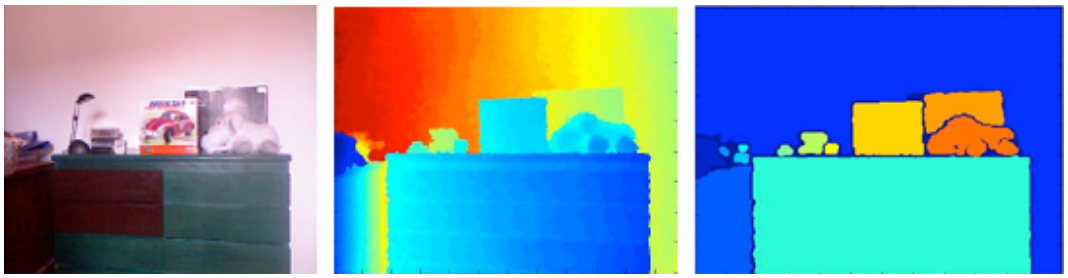


Figure 2.12: From left to right, the images represent: (a) RGB image. (b) Depth image after zero value correction. (c) Connected components filled with different colors[19]

Image cropping is used to crop each component that made it through the filters from the original RGB image, using the corresponding bounding box of that component. All pixels, in the new cropped image, that does not belong to the component get a pixel value of zero.

Last the LSPM algorithm is used to extract descriptors of each object and then to classify them using a linear classifier.

2.2.3 Detection of 3D Objects from 2D Features

Both of the described markers in section 2.1.3 have been proposed as a solution for surgical purposes. The objective in such setting is to create a small marker that could potentially be attached onto the surgical tool, while having a high accuracy in pose estimation.

The method proposed by [15] uses a printable marker to wrap around the surgical tool. The printable marker consists of identical squares placed with an offset both horizontally and vertically, creating an oblique line of small square markers. This is done such that at least one marker is always visible to the camera when the surgical tool is rotating. To know which square is detected by the camera, a green band is placed at one end of the tool as a reference point. The distance from the green band is then calculated, and it is then known which square has been identified. Each square consists of four white circles and within one of them, a black smaller circle, as seen on figure 2.13. The small black circle of the square module is defined as the local reference while the green band is defined as the

global reference.

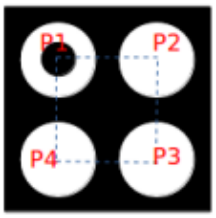


Figure 2.13: Numbering of each circle within the square module[15]

The method proposed in the paper uses an RGB camera to capture the input image and uses OpenCV libraries for their algorithm in order to detect the marker. The algorithm can be divided into three sections: Pre-processing, feature detection and marker identification.

Pre-processing consists of creating a binary image, see figure 2.14. First the RGB input image is converted into a grayscale image. A shadow removal algorithm is applied on the grayscale image and lastly using segmentation to obtain a binary image. The input image is also converted into an HSV image in order to detect the green band, using proper threshold values for the HSV channels. Morphological filters are applied to remove noise.

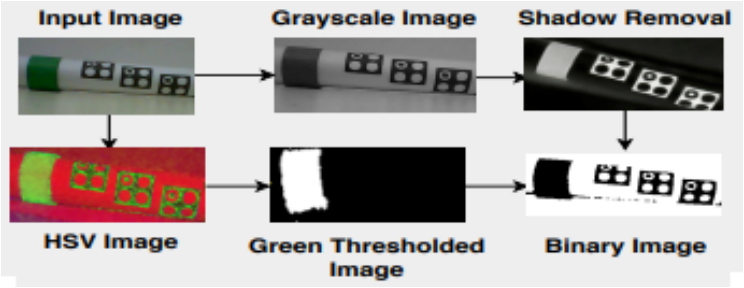


Figure 2.14: Preprocessing Block[15]

Feature detection consists of detecting both the square itself and the circles inside, see figure 2.15. To do so, contour detection is applied to the binary image, followed by quadrangle detection. Quadrangle detection is done by checking: (a) If the contour has four corners. (b) If the distance between the corners are greater than the set threshold. (c) If the area of the contour is between both the upper and lower thresholds. (D) If the contour is convex. Furthermore, the detected quadrangles are transformed into a square with the size of its bounding rectangle. Its ratio of length and height is tested to detect which rectangles are square. If more than one square is detected the one closest to the green band is selected. Then

contour detection is done again, this time inside of the selected square, to detect the four white circles. Centroids of the circles are stored as image points for pose estimation.

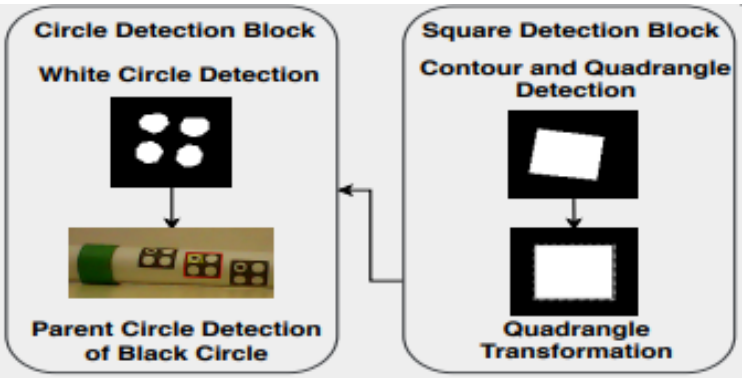


Figure 2.15: Feature Detection Block[15]

Marker identification consists of knowing the orientation . Knowing the centroids of the circles, it is checked which circle contains a black circle and which do not, to get a proper orientation of the image points. To estimate the position of the tool tip, the detected quadrangle is transformed such that it will look like a square and the same transformation is applied to the green band. The distance between the centroid of the detected square and the green band is estimated in pixels. The real-world distance is then calculated using a simple pinhole camera model.

The tool tracking and pose estimation method proposed by [16] uses a dot- and chessboard pattern as a marker, placed onto a cylinder. The method proposed in this paper uses an RGB camera to capture the input image and uses OpenCV libraries for their algorithm in order to detect the marker. The algorithm can be divided into four sections: Pre-processing, circular-dot detection, chessboard vertices detection and marker identification, as seen on figure 2.16

Pre-processing consists of getting the RGB input image and converting it into a grayscale image. A Gaussian filter is then applied to the grayscale image to remove speckles and noises.

Circular-dot detection uses the grayscale image to generate multiple binary images, based on thresholding. When using multiple binary images, it is possible to discard blobs that are false positives. The threshold values used in this paper is 70, 80, 90 and 100. Then contour tracking is used on the binary images to locate the circular dots. To filter out the dots the following criteria have been set: A blob is removing if it does not belong to the marker. A blob is removed if its area is not between the minimum and maximum set thresholds. A blob is removed if it is not

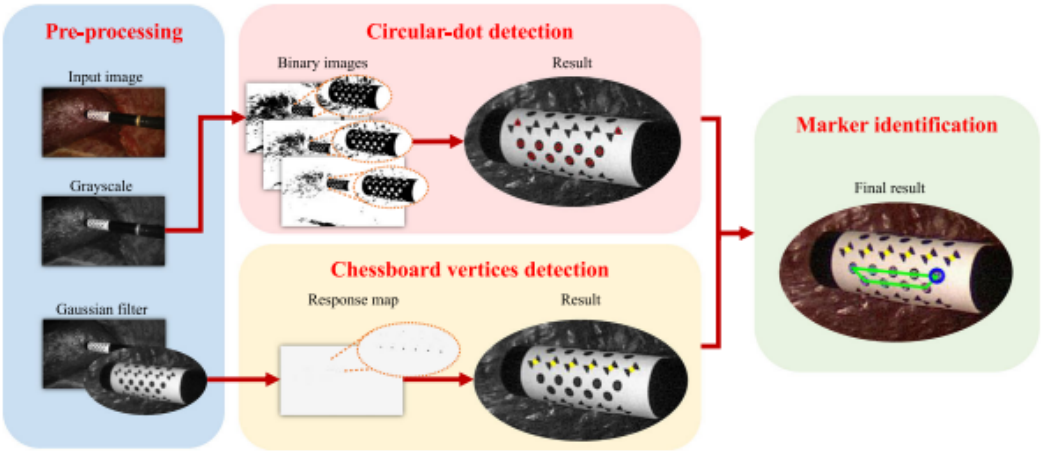


Figure 2.16: The algorithm and its four sections[16]

convex. A blob is removed if its circularity is not high enough. A blob is removed if it is not on at least N of the binary images.

Chessboard vertices detection uses the image with the Gaussian filter applied to in order to detect the chessboard vertices. Each pixel in the filtered image is sampled with a constant radius with equal angular spacing. A response map is calculated using these sampled pixels. If the value in the response map is greater than a threshold value, it is classified as a chessboard feature. The triangles forming the chessboard vertices may be detected as a dot. When classified as a chessboard feature, it is used to reject the false detection of a circular dot.

Marker identification uses the detected dots and chessboard vertices for pose estimation. Dots that are close to each other are assigned to the same cluster. The cluster with highest amount of dots is identified as a marker. The four sharp endpoints of the marker is identified. The pattern is matched with the model pattern to decide where the detected marker is located. The orientation of the pattern can be determined by looking at the relative position of the dots.

2.3 Advantages and Disadvantages

To summarize, the advantages and limitations of the different methods has been listed in this section. A table has been created for the different methods and has been categorized by the different shapes to detect. The purpose for this section is to create an overview of the pros and cons to determine what we, in this project, will use for our solution.

The advantages and limitations for 2D shapes, 3D shaped objects and 2D patterns

on 3D shaped objects and its detection methods have been listed in table 2.1.

2D Shapes	
Advantages	**Limitations**
High speed	Limited to size - detecting the label becomes less precise with the increase of distance
Low cost	Limited to surface
Low computing power	

3D Shapes	
Advantages	**Limitations**
Can be detected by a 3D sensor	Limited to illumination - reflected light can cause poor detection
Robust to distance of detection	

2D Patterns on 3D Shapes	
Advantages	**Limitations**
Can be detected by a 3D sensor	Limited to size - detecting the label becomes less precise with the increase of distance
Great for pose estimation of cylinder-formed objects	

Table 2.1: Table of the advantages and limitations of 2D shaped markers, 3D shaped markers and 2D patterns on 3D shaped markers

3 - Sensor Systems

There are various of sensors, each with different advantages and limitations. This chapter will describe which sensors that can be used for detecting the markers introduced and described in the previous chapter. Some sensors are better for different applications. Some systems are active, meaning that the sensor will measure on an emission of some kind. Other systems are passive, meaning that they are designed to receive natural data to measure upon[21]. We look into the various types such as monocular and stereo cameras, along with other depth sensors.

3.1 Monocular Camera

A monocular camera, is a passive sensor, meaning that nothing is emitted from the camera. It is a normal RGB camera, capturing reflected light. Before the camera can be used, it needs to be calibrated. Due to the lens of the camera, the image might be distorted, just like an image captured with a fisheye camera[22].

3.1.1 Pinhole Model and Camera Calibration

The pinhole model is a mathematical relationship between world coordinates and its projection onto an image plane. When light passes through a pinhole, an inverted image will appear, see figure 3.1. The larger the hole, the brighter the image, but it would also result in more blurry pictures. Shrinking the hole will produce sharper images, but can cause diffraction, see figure 3.2 Most real cameras are equipped with lenses. This is to gather more light while keeping a sharp image. The lens has two spherical surfaces, see figure 3.3, where F is the focal point and f is the focal length. The rays parallel to the optical axis are focused on the focal point F'.[23]

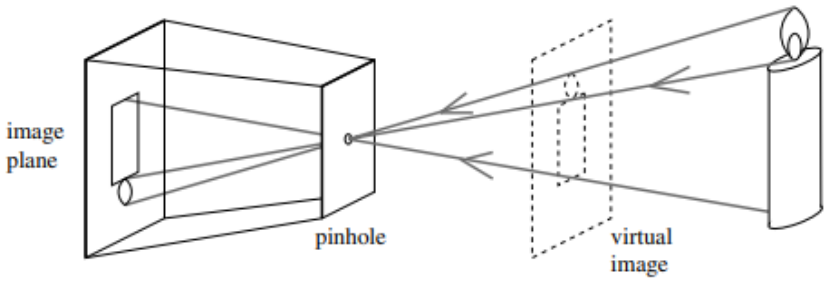


Figure 3.1: Pinhole Model[23]

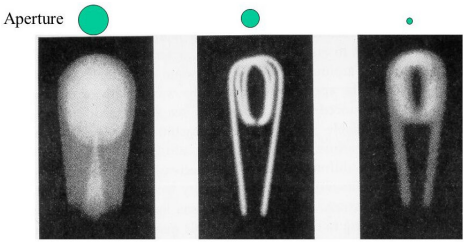


Figure 3.2: Images with different sizes of pinhole. Left: Very bright and blurry picture. Right: Diffraction caused by small pinhole

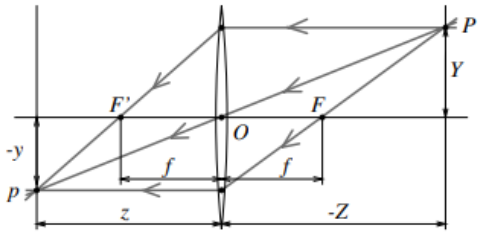


Figure 3.3: The pinhole model with a thin lens. Rays passing through O are not refracted[23]

When capturing an image by a camera with any lens, the image might be distorted, see figure 3.4. In order to use these images for position estimation purposes, we want calibrate the camera by finding the intrinsic and extrinsic parameters of the camera and calculating new parameters to compensate for that distortion, using objects of known dimensions. A very common calibration method for monocular cameras, called Zhang's Method[24], is to use a planar pattern, such as a chessboard pattern with known number of black and white squares, along with the dimensions of the squares. Taking multiple images of the chessboard in different positions and rotations, we can estimate new camera parameters. The extrinsic parameters of the camera represents a rigid transformation from 3D world coordinate system to the 3D camera's coordinate system. These parameters describes the position and orientation of the camera. The intrinsic parameters of the camera represents a projective transformation form the 3D camera's coordinates into the 2D image coordinates. These parameters includes the focal length, the optical center and the skew coefficient.[25]

The pinhole model in homogeneous form is represented by equation 3.1, that can also be represented by equation 3.2, where the two matrices in the middle represent the conversion from 3D to 2D and the scaling. Is the focal length f small,

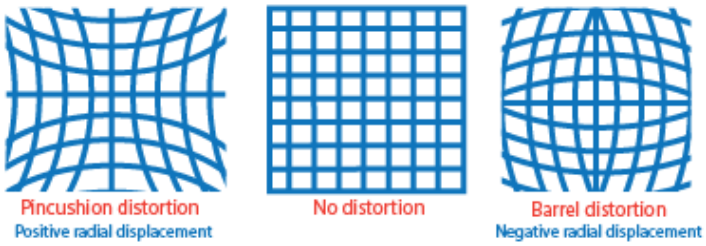


Figure 3.4: Radial distortion[25]

the image is going to be more wide angle compared to a bigger focal length.

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.2}$$

This can also be represented by the central projection model, as seen in figure 3.5

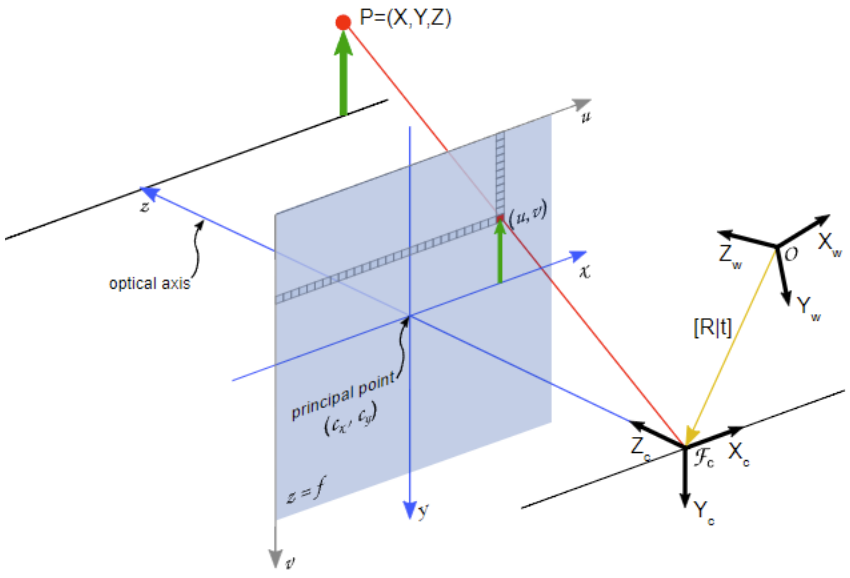


Figure 3.5: Central projection model

Converting the distance from metric to pixel values, we use equation 3.3 to scale from metres to pixels and to shift the origin (0,0) from the center to the top left corner.

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \frac{1}{p_u} & 0 & u_0 \\ 0 & \frac{1}{p_v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \tag{3.3}$$

The extrinsic parameters for the camera is given by equation 3.4, where **R** is the rotational matrix, t being the translation, describing the position and orientation of the camera.

$$\begin{pmatrix} \mathbf{R} & t \\ \mathbf{0}_{1x3} & 1 \end{pmatrix}^{-1} \tag{3.4}$$

The full camera parameters can then be described by equation 3.5.

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \frac{1}{p_u} & 0 & u_0 \\ 0 & \frac{1}{p_v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & t \\ \mathbf{0}_{1x3} & 1 \end{pmatrix}^{-1} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.5}$$

3.2 Stereo Vision

A stereo camera is two cameras that simultaneously captures images of the same scene with slightly shifted view points. It is a passive system, meaning that nothing is emitted from the camera system. It works like a normal RGB camera, capturing reflected light. The reason to use a stereo camera is to get the depth information of the object in the image. To obtain a depth map from the images is to first rectify the two images, such that corresponding image points can be computed, which is then used to calculate the depth. The general pipeline for obtaining depth with a stereo camera can be seen in figure 3.6.[26]

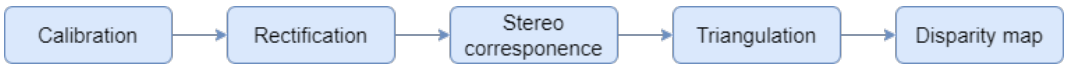


Figure 3.6: General stereo pipeline

3.2.1 Rectification

When calculating the depth in an image, a pixel in one image is compared to the placement of the pixel in the other image. For each pixel, the corresponding

epipolar line is found in the right image. Then all pixels on the epipolar line from the right image is examined and the pixel with the smallest descriptor distance would be matched with the point in the left image, see figure 3.7[27].

If the cameras are placed parallel to each other, the camera centers are at the same height, the focal lengths are the same and the epipolar lines fall along the horizontal scan lines of the images, see figure 3.8[28].

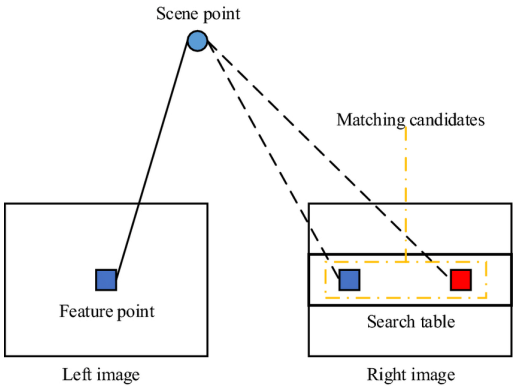


Figure 3.7: Stereo matching process[27]

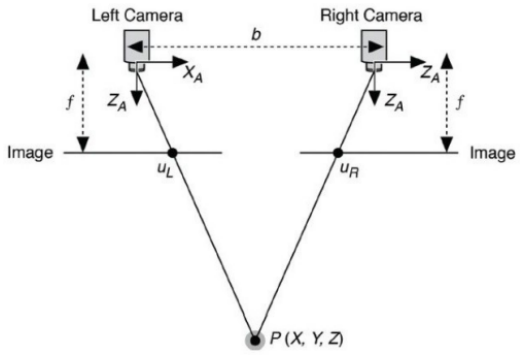


Figure 3.8: Image planes of the two cameras are parallel to each other and to the baseline[29]

When the cameras are not placed in parallel to each other we do image rectification. This is to reproject the image planes onto a common plane parallel to the line between the optical centers, as seen on figure 3.9[30][28]. It is now possible to find corresponding image points with a stereo matching algorithm, see figure 3.10.

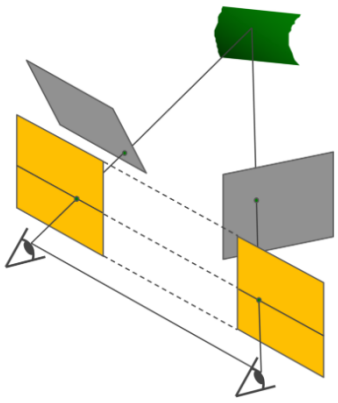


Figure 3.9: Reprojection of the two image planes. The reprojected images are parallel to each[30]

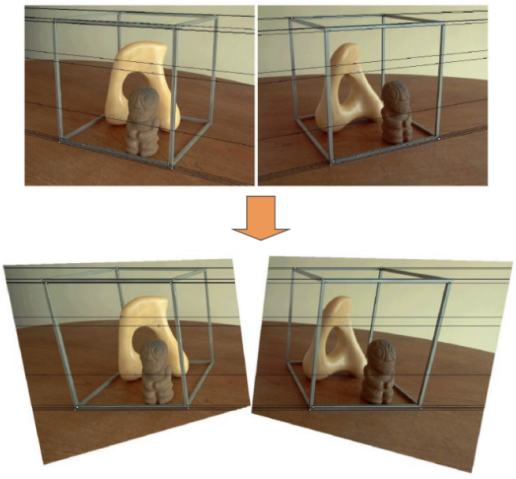


Figure 3.10: Image before and after rectification[30]

3.2.2 Disparity map

The disparity map represent the shift in pixels from left to right[30]. Due to rectification, the epipolar lines have been transformed into scanlines. For each pixel x in the first image, finding the corresponding epipolar scanline in the other image , examining all pixels on the scanline and picking the best match x'. The depth can be calculated with the matching points[27]:

$$z = \frac{f * b}{x_r - x_l} \tag{3.6}$$

Where z is the distance, f is the focal length, b is the baseline and the matching points x_r and x_l is from the right and left image plane respectively. The depth measurements is discretized into parallel planes, and can be visualized with grayscale, black being furthest and white being the closest to the camera, as seen on figure 3.11.

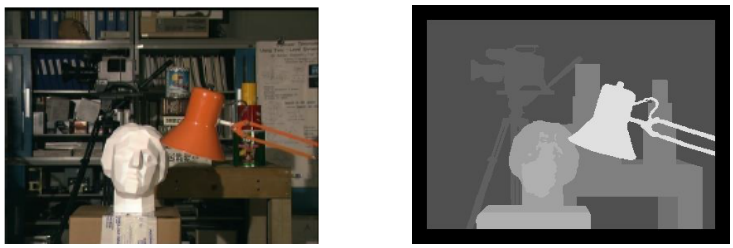


Figure 3.11: Depth map[31]

Some stereo vision cameras are also equipped with infrared light to project onto a scene to improve the accuracy of depth information. The stereo vision camera works in both indoor and outdoor environments. In bad lighting conditions, the infrared projector helps perceiving depth information. There is no limits to how many of these sensors that can be used in a particular space, the camera does not interfere with one another in a way that other depth sensors might[32]

3.3 Depth Sensors

There are variety of different type of extracting the depth information in an image, the stereo vision camera included. This section will focus on other types of depth sensors, along with the advantages and limitations of the different types.

3.3.1 RGB-D Sensor

An RGB-D sensor has, in recent years, been used in computer vision tasks. If we think of an image captured by a normal RGB camera, each pixel in the image has

3 channels, containing information about the amount of red, green and blue. A pixel from an image captured with an RGB-D sensor has four channels being red, green, blue and depth. This means that each pixel contains information about distance. The depth information can be used for object detection, pose estimation, shape analysis, image-based rendering, 3D reconstruction and visual tracking.[33]

3.3.2 Structured Light

Structured light rely on projected light from some kind of emitter onto the scene. The light is usually infrared light. The projected light is some sort of known pattern, such as a grid of lines or dots, as seen on figure 3.12. The sensor captures the distorted pattern from the scene and can calculate the depth from the distortion of the objects. These active sensors are vulnerable to other noise from the environment such as other devices emitting infrared [32].

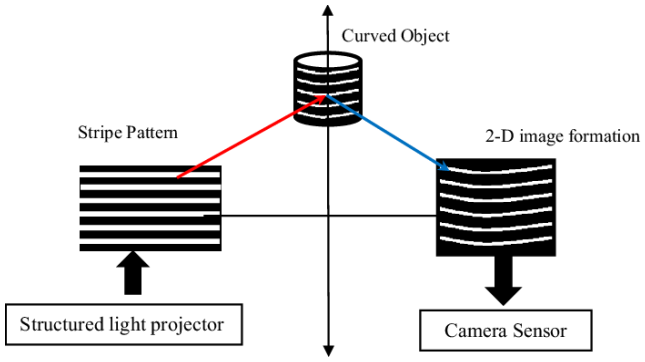


Figure 3.12: Structured light projected onto a curved object[34]

3.3.3 Time of Flight

Another active sensor is the time of flight (TOF). A TOF sensor will emit light and by knowing the speed of light, it is possible to calculate the distance by timing the return of that light. The light source is typically LED, IR or laser. With pulsed light systems, the camera shutter is synchronized with the illuminator, the delay in returning light pulse is measured to calculate the distance to the reflection point. In continuous light systems, the illuminator will emit modulated amplitudes of light to then measure the returning light to estimate the distance. Depending on the power and the wavelength of the light, TOF sensors can measure significant depth and distances. It can be used to map an environment from a robotic vehicle driving on the ground, or mapping a terrain form a helicopter. Depending on the specific sensor, the disadvantage of TOF sensors is that they can be sensitive to other cameras in the same space and it can also be vulnerable to sunlight, functioning less in outdoor environments.[32][35]

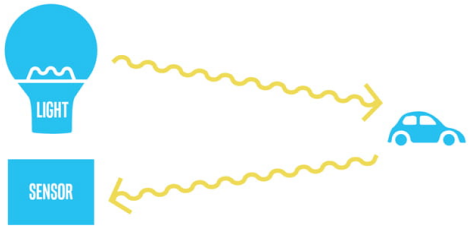


Figure 3.13: Time of flight sensor measuring the distance to an object by emitting light and capturing it with a sensor[32]

3.4 Advantages and Disadvantages

To summarize, the advantages and limitations of the different sensor systems has been listen in this section. The purpose for this, is to create an overview to determine which sensor system we chose for our solution. Table 3.1 has been created for comparing the different sensors.

	Monocular Camera	**Stereo Vision**	**Structured Light**	**Time of Flight**
Effective range (typical)	Camera focus lens dependent	Baseline dependent	< 5m*	< 7m*
Depth accuracy	cm-accuracy Camera focus lens dependent	cm-accuracy Baseline dependent Gradually diminishing with distance	mm-accuracy Rapid fall-off beyond projection range	cm-accuracy Rapid fall-off beyond light range
Compute load	Framework dependent	High	Medium	Low
Outdoor performance	Excellent	Excellent	Poor	Poor to good*
Low-light performance	Poor	Poor/Good (IR illuminated)	Excellent	Excellent
Sensor cost	Low	Low	High	Medium
Non-compute power	Low	Low	High	Medium
* For active systems, range and performance is highly dependent on the power of the light projector/emitter				

Table 3.1: Generalizing the characteristics of the different sensor systems[36][37]

4 - Implementation

Capra Robotics has expressed their wish for the marker to be implemented onto the trailer as a part of the design, possibly to be reshaped into a useful element of the trailer, such as a handle on the end of the rod, in the future. We have chosen to focus on designing a marker of 3D shapes, with the idea of further redesigning it into a handle on the rod of the trailer. We have been given a Logitech C930e webcam as a sensor for the implementation. This chapter will describe the design of a passive 3D marker to be detected by a monocular camera in order to gain orientation and position of the trailer. Implementing the system, OpenCV library are used for image manipulation due to its large number of computer vision algorithms. The coding language used for this project is python.

4.1 Design of marker

We have designed a marker consisting of two 3D printed spheres in different vivid colors and different sizes. One of the spheres is orange with a radius of 20mm while the second sphere is blue with a radius of 12.5mm. As a proof of concept a 3D printed stand has been designed such that the spheres can be moved around with a static distance of 100mm, see figure 4.1. The colors of the spheres has been chosen such that they do not lie next to each other in the HSV (Hue, Saturation, Value) space, see figure 4.2. On figure 4.3, the spheres have been added to the trailer, to visualize the idea behind the marker and its future placement.

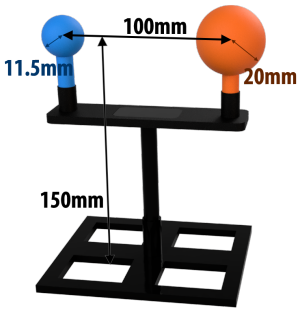
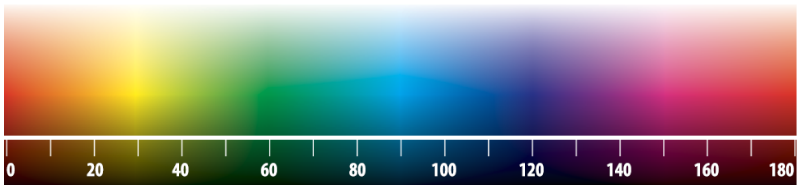


Figure 4.1: Dimensions of the marker including the proof-of-concept stand



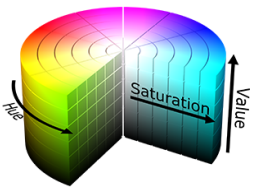


Figure 4.2: HSV colormap

Figure 4.3: Concept of how the marker would be placed onto the trailer

4.2 Camera calibration

When we look at the image captured by the Logitech C930e camera, we can see that straight lines near the edges bends slightly. This is called a barrel distortion. We want to remove this distortion, such that we can calculate the correct position of the marker. To calibrate the camera we use a chessboard pattern printed onto a A4 paper with known dimensions. The chessboard pattern has 11x8 squares, each the size of 25x25mm. The paper is taped onto a flat rigid object, to ensure that the paper does not warp, which could cause wrong calibration parameters. The chessboard is moved around with different positions, rotations and are also tilted. The calibration parameters that we get is the camera matrix, distortion parameters, rotation and translation vectors. These can be applied into a function in OpenCV to remove distortion.

Applying the Zhang's method, using 5 input images of the chessboard, we get the camera matrix:

$$C = \begin{bmatrix} 746.09 & 0 & 611.06 \\ 0 & 746.32 & 357.96 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

With the distortion coefficients:

$$dist = (0.1253, -0.252, -0.001, -0.009, 0.111) \tag{4.2}$$

The camera matrix and distortion coefficients are then used to calculate the new camera matrix:

$$C_{new} = \begin{bmatrix} 730.62 & 0 & 590.99 \\ 0 & 718.50 & 355.76 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

With a reprojection error of:

$$error = 0.015 \tag{4.4}$$

The new matrix is applied to remove the distortion caused by the camera lens, see figure 4.4.

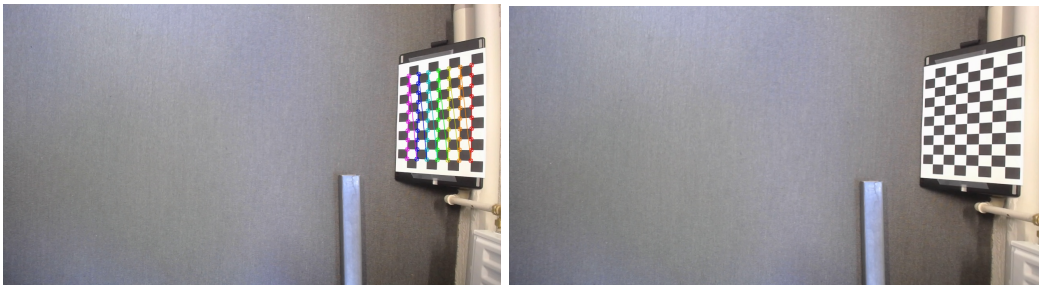


Figure 4.4: Calibration using a chessboard pattern. Left: Detecting 10x7 vertices. Right: After undistorting using new camera parameters. Notice the line of the wall and floor meeting is more straight on the right image compared to the left image.

4.3 Software Implementation

We have created a system in Python with the use of OpenCV functions that can detect and estimate the position and orientation of spherical markers. The system takes in a stream of images through a camera and calculates the position of each sphere, the position in between the spheres and the rotation matrix of the marker, in relation to the camera frame. The flowchart of the system can be seen on figure 4.5. Running the program, the first thing we check is if we have connection to the camera, if not, the program will stop. While we have connection to the camera, the program will run. Each frame is then undistorted with the distortion coefficients and new camera matrix found when calibrating the camera. We use a OpenCV function to find possible circles in the image. To do so we first convert the RGB image to a grayscale image and adding blur to eliminate possible noise. The OpenCV function `cv2.HoughCircles` returns a list of circles, if any. We check if any circles are found, to then check if they contain either orange or blue color, as seen in the flowchart on figure 4.6. We use OpenCV to convert the image to the HSV space in order to work with ranges of color values. We can then use the ranges of colors to binarize the image with a lower and higher threshold.

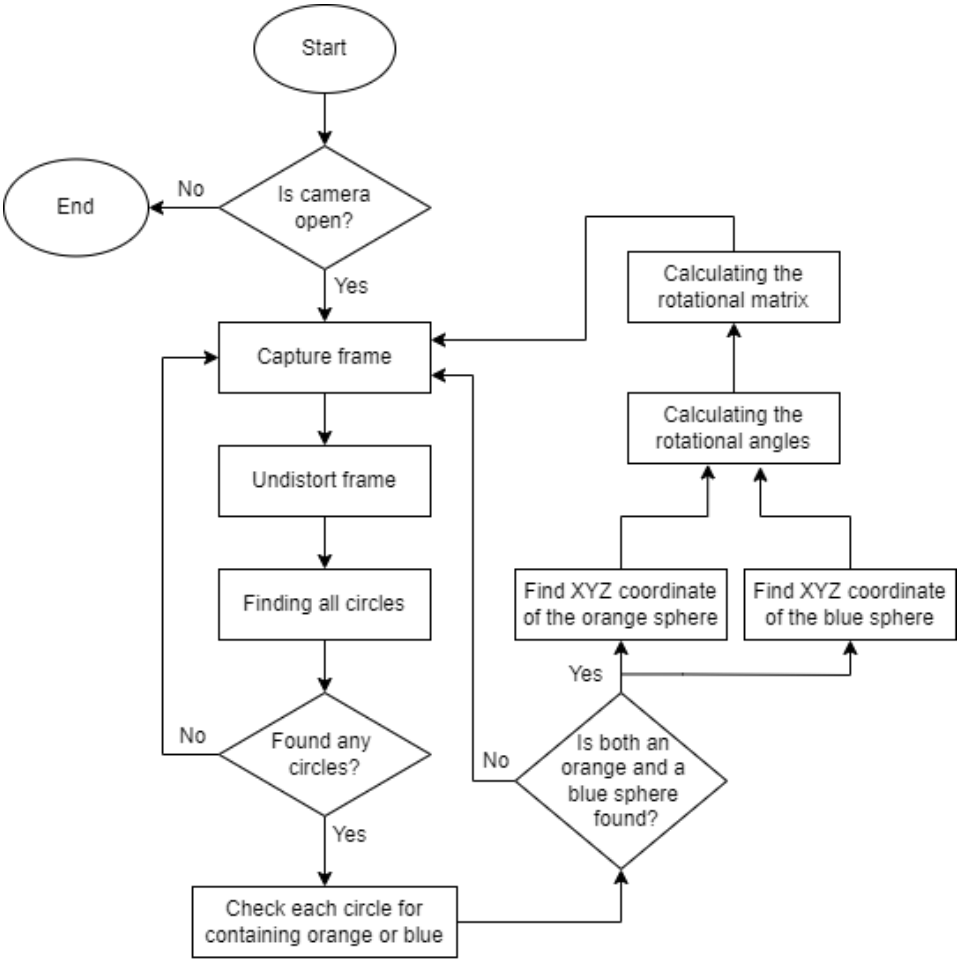


Figure 4.5: Flowchart of system

To find these thresholds, we use a python program[38] with sliders to find the HSV values that fits to differentiate between the blue and the oranges sphere, see figure 4.7 and figure 4.8. We derive the following thresholds:

- Orange lower threshold: (0, 125, 136)
- Orange higher threshold: (22, 255, 255)
- Blue lower threshold: (92, 62, 41)
- Blue higher threshold: (115, 255, 255)

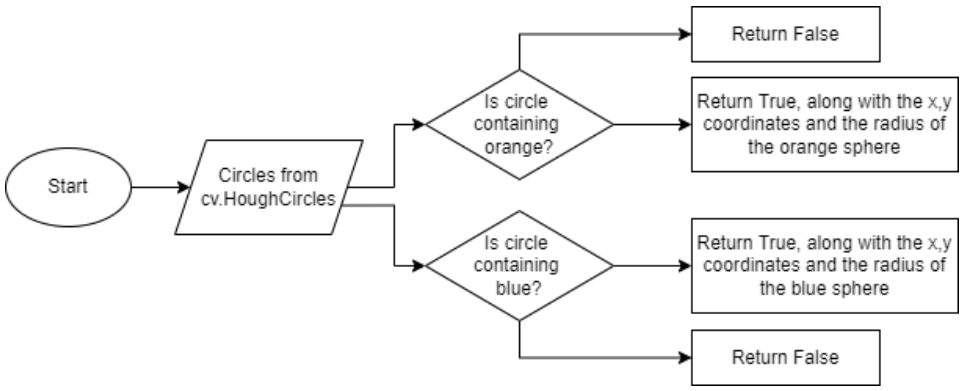


Figure 4.6: Flowchart of orange and blue sphere detection

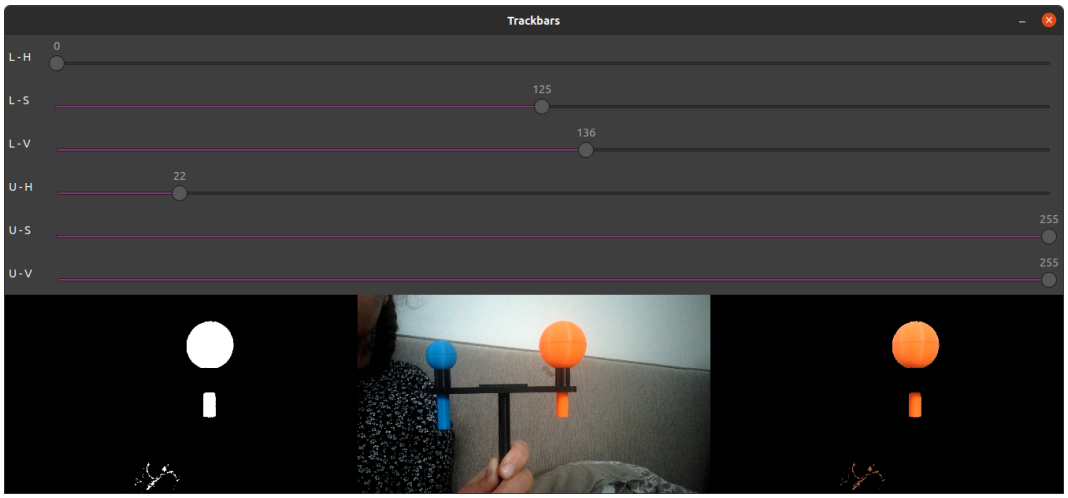


Figure 4.7: Lower HSV threshold (0, 125, 136). Upper HSV threshold (22, 255, 255)

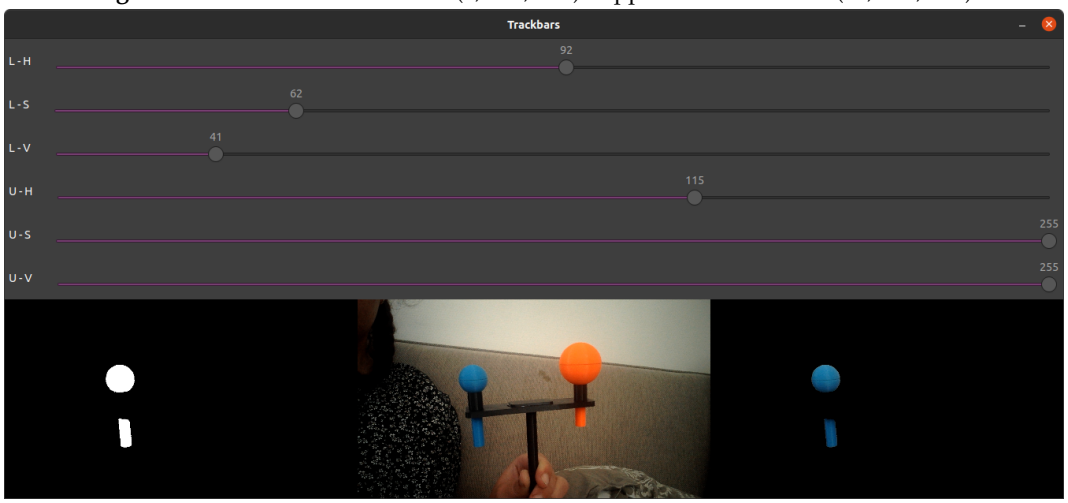


Figure 4.8: Lower HSV threshold (92, 62, 41). Upper HSV threshold (115, 255, 255)

We check each circle, if any lies between either the orange or the blue thresholds, we return `True`, along with the (x,y) coordinates of the center and the radius. If we detect both a blue and an orange sphere in the image, we calculate the distance to each sphere. The distance is found with the following equation:

$$distance = \frac{radius_{mm} * focallength}{radius_{pixel}} \tag{4.5}$$

The XYZ coordinates are then used to calculate the directional vector, which is used to calculate the rotational angles. The rotations about X, Y and Z is calculated using the directional vector with the following equations:

$$\begin{aligned} \phi &= math.atan2(z, y) \\ \theta &= math.atan2(z, x) \\ \psi &= math.atan2(y, x) \end{aligned} \tag{4.6}$$

Where ϕ is the rotational angle around X, θ is the rotational angle around Y and ψ is the rotational angle around Z. We know that the rotation about X can not be detected, due to the design of the marker, we have therefore changed ϕ, such that:

$$\begin{aligned} \phi &= 0 \\ \theta &= math.atan2(z, x) \\ \psi &= math.atan2(y, x) \end{aligned} \tag{4.7}$$

The found angles are used to calculate the rotation matrix:

$$\begin{aligned} R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \\ R_y(\theta) &= \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \\ R_z(\psi) &= \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.8}$$

The Rotational matrix is calculated by the ZYX-sequence:

$$
\begin{aligned}
R &= \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \\
&= \begin{bmatrix} C(\psi)C(\theta) & C(\psi)S(\theta)S(\phi) - S(\psi)C(\phi) & C(\psi)S(\theta)C(\phi) + S(\psi)S(\phi) \\ S(\psi)C(\theta) & S(\psi)S(\theta)S(\phi) + C(\psi)C(\phi) & S(\psi)S(\theta)C(\phi) - C(\psi)S(\phi) \\ -S(\theta) & C(\theta)S(\phi) & C(\theta)C(\phi) \end{bmatrix}
\end{aligned} \tag{4.9}
$$

The matrix can be converted into world coordinates, but this has yet to be implemented into the system.

5 - Test

We will test the system by acquiring the accuracy and precision of pose detection of the marker. The tests have divided into different categories each focusing on either distance, orientation, illumination or background noise to identify where the system fails and where it succeeds. Tests for distance and orientation will be performed with the same illumination and no background noise. The orientation about the X-axis can not be detected with the current marker design and implementation. The orientation has therefore been set to 0° at all times and is therefore not evaluated upon any test.

Due to the change in illumination at the test setup environment, the HSV values has been changed to the following:

- Orange lower threshold: (0, 150, 110)
- Orange higher threshold: (25, 255, 255)
- Blue lower threshold: (88, 100, 80)
- Blue higher threshold: (138, 255, 255)

5.1 Setup 1

The setup for tests on distance and orientation can be seen on figure 5.1. The marker (m) has been placed in front of a white wall. Red tape has been placed on the floor to mark the different positions to place the camera for the different tests. A small piece of red tape has been placed on each red line of tape for every 100mm. The back of the furthest piece measures 1060mm from the center of the foot of the marker. The closest piece measures 160mm. The camera module will be placed behind the red tape, such that the front of the camera foot aligns with the back of the red tape. The angles between the lines can be seen on figure 5.2.

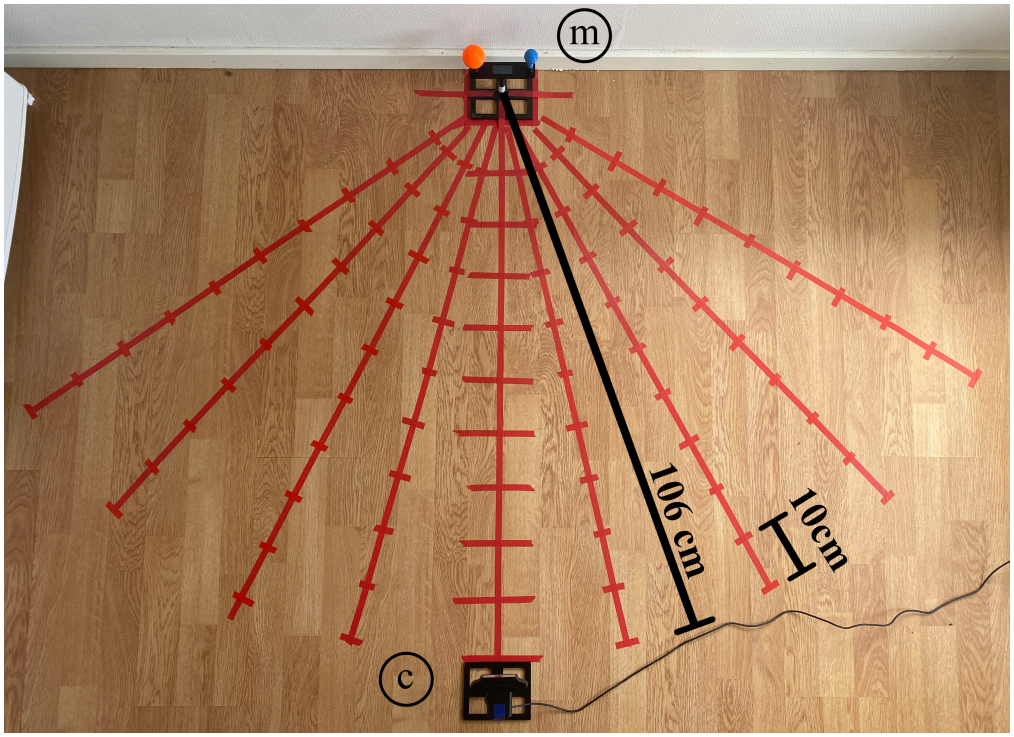


Figure 5.1: Top-down view of the test setup for acquiring distance and orientation between camera and marker. Red tape placed on floor for measuring actual pose of the marker (m) in relation to the camera (c)

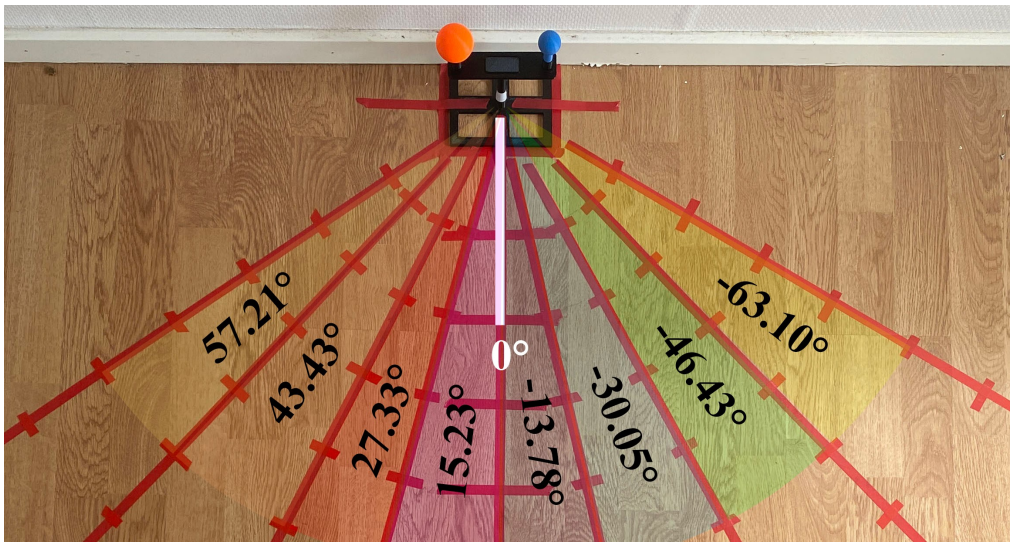


Figure 5.2: Top-down view of the test setup to visualize the angles of each red line in respect to the mid line, being being 0°.

5.2 Setup 2

A setup with a fixed length between the camera and marker has been created for the test of detecting the marker in different illumination and detection of the marker with background noise. To ensure a fixed length between camera and marker, a rod of 500mm has been placed between the bodies of both the camera and the marker, creating a distance of 470mm between the camera lens and the mid point of the two spheres. The setup can be seen on figure 5.3.

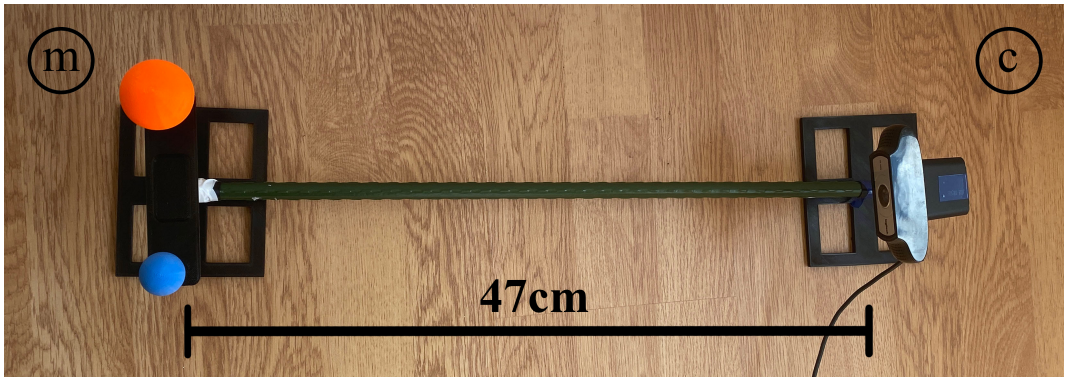


Figure 5.3: Top-down view of the test setup for testing sphere detection in different illumination and with background noise. A rigid rod is placed between the marker (m) and the camera (c)

5.3 Different Distances

This test will evaluate upon which distances the sphere detection works. The marker is placed on a fixed location with a fixed orientation with the camera placed facing the marker. The distance from the point in between both spheres to the camera will be measured physically and compared with the result of the system. The distance can only be calculated if both of the spheres are detected. The camera will be placed at 9 different angles each in 10 different distances to the marker. In each iteration of this test the camera will capture and save an image, use Hough circle detection to find all of the possible circles in the image. If there exists an orange circle and a blue circle in the image, the distance from the mid point of the spheres to the camera is measured. After each iteration, the camera will be moved 100mm closer to the marker. At the start of this test, the distance between camera and marker is 1060mm. When the camera has measured the closest distance, of 160mm, the camera is placed back with a new angle towards the marker, on the furthest point, resulting in a total of 90 iterations.

5.3.1 Result

The system could only detect both spheres when the camera was 160mm - 560mm to the marker. The measured distances from the camera to marker has been listed in table 5.1. It is noticeable how the error decreases as the camera gets closer to the marker.

Plotting the error data into a graph, see figure 5.4, it is noticeable how the error at distance 460mm is distinct higher than expected. Looking at the image taken from the camera at a distance of 460mm from the marker, as seen on figure 5.6, it can be seen that the position of the center of the orange sphere is wrongly detected.

Real Distance (mm)	Measured Distance (mm)	Error (mm)
560	575.50	15.58
460	482.14	22.14
360	368.67	8.67
260	264.81	4.81
160	161.1	1.1

Table 5.1: Data of the measured distance

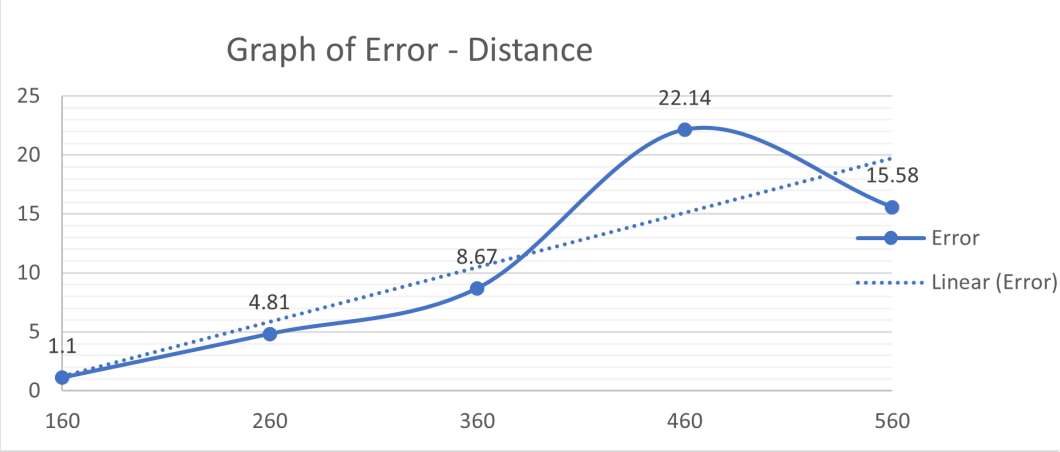


Figure 5.4: Graph of error in distance detection

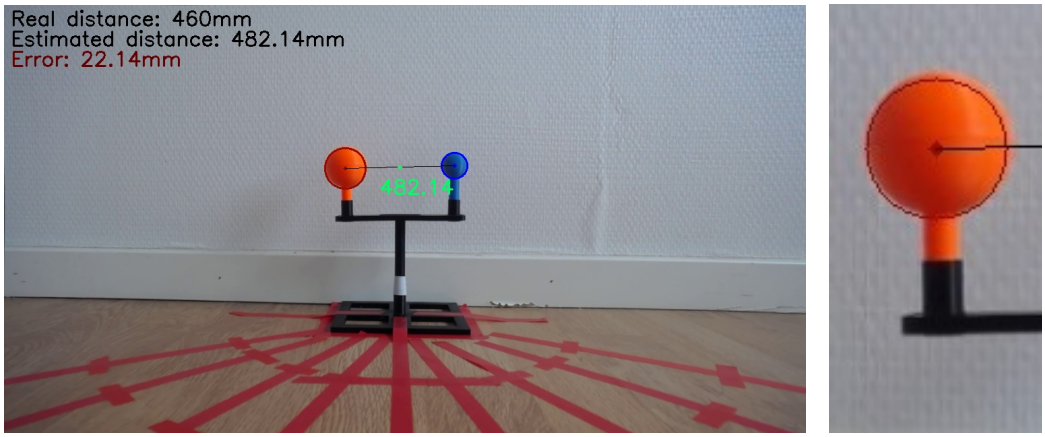


Figure 5.5: From left to right, the images represent: (a) Image captured by the camera at a physical distance of 460mm from the marker. (b) Close up of the orange sphere

Figure 5.6: Comparison of real distance and measured distance

5.4 Different Camera Orientations

This test will evaluate how well the orientation estimation of the system works when the marker is placed on a fixed location with a fixed orientation with the camera facing the marker at all times. The true orientation will be calculated with the known position and orientation of the camera in relation to the marker and compared with the result of the system. The orientation is calculated with the XYZ coordinates of the marker, that can only be derived when both of the spheres are detected. The camera will be placed at 9 different angles each in 10 different distances to the marker. In each iteration of this test the camera will capture and save an image, obtain the XYZ coordinates of the two spheres, calculating the direction vector in order to derive pitch, roll and yaw angles that defines the rotation of the marker in relation to the camera frame. After each iteration, the camera will be moved 100mm closer to the marker. At the start of this test, the distance between camera and marker is 1060mm. When the camera has measured the closest distance, of 160mm, the camera is placed back with a new angle towards the marker, on the furthest point, resulting in a total of 90 iterations.

5.4.1 Result

The system detected both spheres in the same image 29 times out of 90 images. The measured orientations of the marker in relation to the camera frame has been listed in table 5.2. The system could detect both spheres at all 9 angles, but limited to a detection distance of 160-560mm. The only rotation that has been changed is the rotation about the Y-axis, but the system detected a rotation about both

the Y- and the Z-axis. In figure 5.7 the difference between actual and measured orientation has been displayed.

Rotation Y (degrees) Y	**Distance (mm)**	**Measured Rotation YZ (degrees)** Y	Z	**Error (degrees)** Y	Z
57.21	360	47.87	5.43	9.34	-5.43
43.43	460	37.28	3.7	6.15	-3.7
43.43	360	38.32	3.92	5.11	-3.92
27.33	460	5.33	0.6	22.0	-0.6
27.33	360	18.58	2.47	8.75	-2.47
27.33	260	19.2	2.72	8.13	-2.72
27.33	160	16.57	1.99	10.76	-1.99
15.23	460	5.1	1.3	10.13	-1.3
15.23	360	12.98	2.3	2.25	-2.3
15.23	260	9.26	1.87	5.97	-1.87
15.23	160	12.26	0.99	2.97	-0.99
0	560	7.43	-0.98	-7.43	0.98
0	460	3.4	-1.05	-3.4	1.05
0	360	5.1	-0.47	-5.1	0.47
0	260	5.28	-0.4	-5.28	0.4
0	160	16.13	0.97	-16.13	-0.97
-13.78	560	-34.33	-2.34	20.55	2.34
-13.78	460	-8.26	0.51	-5.52	-0.51
-13.78	360	-24.62	-1.57	10.84	1.57
-13.78	160	-27.91	-3.89	14.13	3.89
-30.05	460	-39.83	-2.94	9.78	2.94
-30.05	360	-37.73	-1.98	7.68	1.98
-30.05	260	-35.49	-1.51	5.44	1.51
-30.05	160	-44.88	-6.06	14.83	6.06
-46.43	360	-48.02	-3.7	1.59	3.7
-46.43	260	-47.83	-3.08	1.4	3.08
-46.43	160	-52.23	-5.6	5.8	5.6
-63.1	260	-64.13	-7.66	1.03	7.66
-63.1	160	-60.87	-4.88	-2.23	4.88

Table 5.2: Data of the measured rotation about YZ-axis

We observe that the most successful detections, meaning that the error calculations of rotations that we obtain about Y-axis is lowest when the camera is placed at -46.43° in relation to the marker. The error calculations of rotations about Z-axis is

lowest when the camera is placed in front of the marker, at 0°.

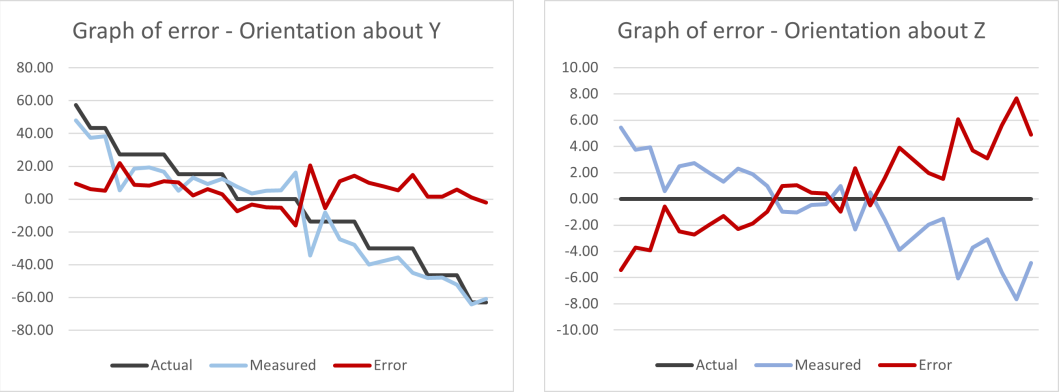


Figure 5.7: From left to right, the graphs represent: (a) Graph of error on the orientation about Y-axis. (b) Graph of error on the orientation about Z-axis.

We want to investigate upon the data even further. To do so we isolate the data for each distance and examine the different mean and standard deviation values. Both the mean and the standard deviation values of the gathered data can be seen on figure 5.3.

Distance	Error Y	Error Z
160	4.30±10.09	2.35±3.25
260	2.78±4.39	1.34±3.43
360	5.06±4.94	-0.80±2.99
460	6.52±9.19	-0.35±2.04
560	6.56±13.99	1.66±0.68

Table 5.3: Mean and standard deviation values of the calculated error for each distance

We also isolate each measurement for the individual angles, to investigate upon the mean and standard deviation values, to further compare them. Table 5.4 provides this information.

Angle	Error Y	Error Z
57.21*	9.34±0	-5.43±0
43.43	5.63±0.52	-3.81±0.11
27.33	12.41±5.62	-1.95±0.82
15.23	5.33±3.10	-1.62±0.51
0.00	-7.47±4.52	0.39±0.73
-13.78	10±9.62	1.82±1.58
-30.05	9.43±3.47	3.12±1.77
-46.43	2.93±2.03	4.13±1.07
-63.10	-0.6±1.63	6.27±1.39
*Only one measurement of this angle		

Table 5.4: Mean and standard deviation value of the calculated error for each angle

We can observe that both the mean value and the standard deviation of the error in the rotation about Y-axis is the smallest at a distance of 260mm.

5.5 Different Sphere Orientations

This test will evaluate how well the orientation estimation of the system works when the marker is at a fixed location and rotated with different angles while the camera is at fixed position. The true orientation will be calculated with the known position and orientation of the camera in relation to the marker and compared with the result of the system. The orientation is calculated with the XYZ coordinates of the marker, that can only be derived when both of the spheres are detected. The camera will be placed at 10 different positions, all placed on the mid red line at 0°, only changing the distance between marker and camera. In each iteration of this test the camera will capture and save an image, obtain the XYZ coordinate of the two spheres, calculating the direction vector in order to derive pitch, roll and yaw angles that defines the rotation of the marker in relation to the camera frame. After each iteration, the marker will be rotated. When the marker has reached a full 360° rotation, the camera will be moved 100mm closer to the marker. At the start of this test, the distance between camera and marker is 1050mm. When the camera has measured the closest distance of 150mm, resulting in a total of 80 iterations. The angles that the marker has rotated is the following: 180°, 149.451°, 77.831°, 34.859°, 1.437°, -51.084°, -108.3803° and -137.028°.

5.5.1 Result

The system detected both spheres in the same image frame 15 times out of a total of 72 images. The measured orientations of the marker in relation to the camera

frame has been listed in table 5.5. The only rotation that has been changed if the rotation about the Y-axis, but the system detected a rotation about both Y- and Z-axis. The system could detect both spheres at all given rotations but 77.831° and -108.3803°. One of the spheres overlap the other at these rotations, see figure 5.8. Both spheres are detected within a distance of 250mm and 550mm. Both spheres did not get detected at a distance of 150mm as in the other tests. At some of the rotations, one of the spheres does not fit into the image frame, see figure 5.9. The colors of the spheres are also brighter in the set of images collected from 150mm. We can observe that the lowest error measured about the Y-axis is at a positive rotation being 149.45°-180°, while the error measured around the Z-axis is lowest at a low rotation of 1.44°.

Rotation Y (degrees) Y	**Distance (mm)**	**Measured Rotation YZ (degrees)** Y	Z	**Error (degrees)** Y	Z
180.0	550	175.17	177.64	4.83	-177.64
180.0	450	180.0	178.62	0.0	-178.62
180.0	350	-170.06	-179.95	350.06	179.95
180.0	250	178.13	176.98	1.87	-176.98
149.45	450	150.61	175.33	-1.16	-175.33
149.45	350	148.21	174.95	1.24	-174.95
149.45	250	138.28	170.22	11.17	-170.22
34.86	350	33.59	3.84	1.27	-3.84
1.44	550	-13.65	-1.04	15.09	1.04
1.44	450	12.44	3.18	-11.01	-3.18
1.44	350	-2.83	1.15	4.27	-1.15
-51.08	450	-59.92	-10.32	8.83	10.32
-51.08	350	-48.97	-4.7	-2.12	4.7
-51.08	250	-44.32	-3.39	-6.77	3.39
-137.03	350	-138.79	-179.86	1.76	179.86

Table 5.5: Data of measured rotation about YZ-axis

Figure 5.8: From left to right, the images represent: (a) Image taken at a distance of 250mm, marker rotated 34.859° around the Y-axis. (b) Image taken at a distance of 250mm, marker rotated -108.3803° around the Y-axis

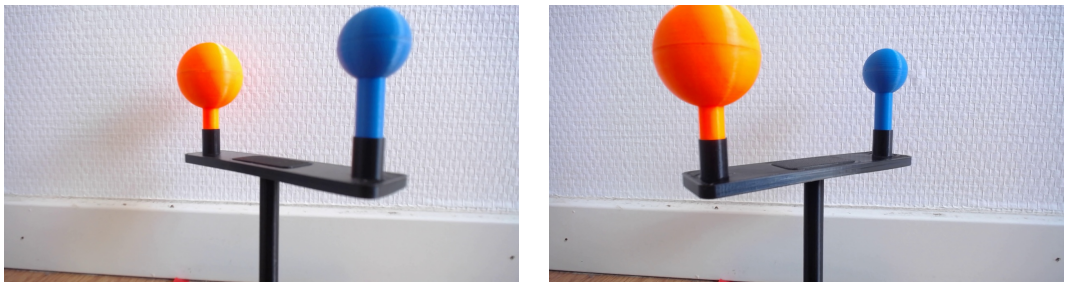


Figure 5.9: From left to right, the images represent: (a) Image taken at a distance of 150mm, marker rotated 77.831° around the Y-axis. (b) Image taken at a distance of 150mm, marker rotated -108.3803° around the Y-axis

In figure 5.10 the difference between the actual and measured orientation has been displayed. We can, in figure 5.10(a), identify an outlier at 180°. The same can be observed in figure 5.10(b). At around 180° the graph spikes.

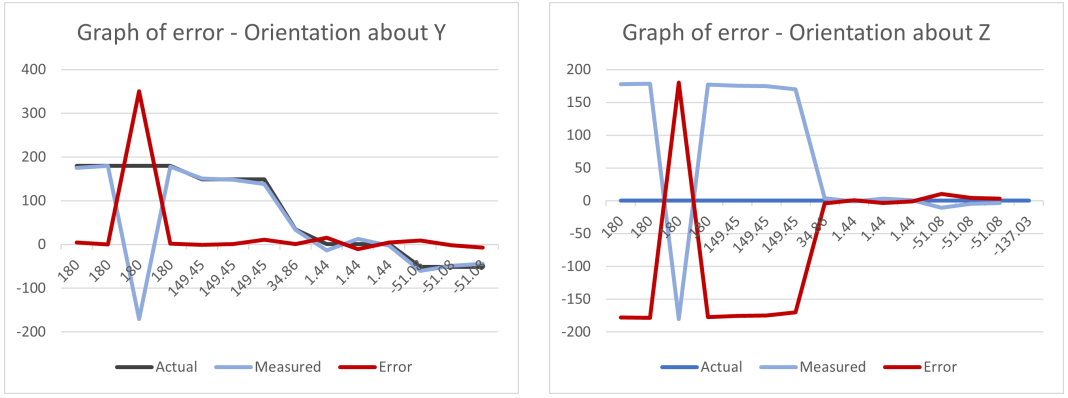


Figure 5.10: From left to right, the graphs represent: (a) Graph of data on sphere orientation about the Y-axis from test on illumination. (b) Graph of data on the orientation about the Z-axis from test on sphere orientation.

To analyse the data even further, we split up the data such that each group involves the data measured at a specific angle, and then compare the mean and standard deviation calculated between the different angles. Table 5.6 presents the mean and standard deviation of the calculated error around both the Y- and the Z-axis.

We also split the data into groups involving the different distances, to analyse the mean and standard deviation of the measured angles around the Y- and Z-axis, at different distances. This can be seen in table 5.7.

Angle	Error Y	Error Z
180	89.19±150.62	-88.32±154.89
149.45	3.75±5.34	-173.5±2.32
1.44	2.78±10.71	-1.1±1.72
-51.08	-0.02±6.54	6.14±3.01
34.86*	1.27±0	-3.84±0
-137.03*	1.76±0	179.86±0
*Only one measurement from this angle		

Table 5.6: Mean and standard deviation value of the calculated error for each angle

Distance	Error Y	Error Z
350	59.41±129.99	30.76±122.27
450	-0.84±7.03	-86.70±90.41
550	9.96±5.13	-88.3±89.34

Table 5.7: Mean and standard deviation values of the calculated error for each distance

5.6 Accuracy and Precision

This test will evaluate the accuracy and precision of the system. To perform this test, the marker is placed on a fixed location with a fixed orientation with the camera placed facing the marker. The orientation of the marker in relation to the camera is calculated with the system and compared to the true orientation that is calculated with the known position and orientation of the camera in relation to the marker. The camera will be placed at three different positions. In each iteration of this test the camera will capture and save an image, obtaining the XYZ coordinates of the two spheres, calculating the direction vector in order to derive pitch, roll and yaw angles that defines the rotation of the marker in relation to the camera frame. In each iteration, the camera will capture 10 images without moving either the marker or the camera. After each iteration, the camera will be moved to a new position.

5.6.1 Result

The system detected both spheres 30 times out of a total of 30 images. Each of the three test consists of 10 images. The measured orientation of the marker in relation to the camera frame of each iteration of the test has been listed in table 5.8, table 5.9 and table 5.10. The placement for the camera and the orientation of the sphere has been set at random, with two constraints: The camera should be placed within a distance of 560mm to the marker and the orientation of the sphere should never be turned such that one sphere would cover the other in the camera frame. The mean and standard deviation for the three iterations of 154.23°, 197.66° and 124.18° around the Y-axis is 153.21°±2.89°, -161.73°±6.23° and 122.77°±1.34° respectively. The mean and standard deviation for the three iterations of 154.23°, 197.66° and 124.18° around the Z-axis is 175.48° ±0.63°, 107.77°±143.15° and 168.36°±0.93° respectively.

Rotation Y (degrees)	**Measured Rotation YZ (degrees)**		**Error (degrees)**	
Y	Y	Z	Y	Z
154.23	152.74	175.96	1.48	-175.96
154.23	152.74	175.96	1.48	-175.96
154.23	161.2	176.91	-6.97	-176.91
154.23	152.74	175.33	1.48	-175.33
154.23	152.74	175.33	1.48	-175.33
154.23	152.74	174.78	1.48	-174.78
154.23	148.95	174.54	5.27	-174.54
154.23	152.74	175.33	1.48	-175.33
154.23	152.74	175.33	1.48	-175.33
154.23	152.74	175.33	1.48	-175.33

Table 5.8: Data measured of 10 images from same pose: 154° and a distance of 360mm

Rotation Y (degrees)	**Measured Rotation YZ (degrees)**		**Error (degrees)**	
Y	Y	Z	Y	Z
197.6554	-166.97	179.12	364.62	-179.12
197.6554	-167.09	179.67	364.75	-179.67
197.6554	-162.62	179.26	360.28	-179.26
197.6554	-150.03	-178.53	347.69	178.53
197.6554	-167.09	179.13	364.75	-179.13
197.6554	-167.09	179.13	364.75	-179.13
197.6554	-150.03	-178.53	347.69	178.53
197.6554	-162.12	179.31	359.77	-179.31
197.6554	-162.12	179.31	359.77	-179.31
197.6554	-162.12	179.85	359.77	-179.85

Table 5.9: Data measured of 10 images from same pose: 197.66° and a distance of 360mm

Rotation Y (degrees)	**Measured Rotation YZ (degrees)**		**Error (degrees)**	
Y	Y	Z	Y	Z
124.18	124.03	170.11	0.15	-170.11
124.18	123.82	168.88	0.35	-168.88
124.18	121.86	167.14	2.31	-167.14
124.18	124.08	168.22	0.09	-168.22
124.18	123.45	168.72	0.72	-168.72
124.18	123.82	168.88	0.35	-168.88
124.18	120.86	168.07	3.32	-168.07
124.18	123.82	168.88	0.35	-168.88
124.18	121.10	166.62	3.07	-166.62
124.18	120.86	168.07	3.32	-168.07

Table 5.10: Data measured of 10 images from same pose: 124.18° and a distance of 260mm

Looking at figure 5.11(a) and 5.13(a) the precision and accuracy is high. Figure 5.11(b), figure 5.12(a) and figure 5.13(b) has a high precision but low accuracy. Looking at figure 5.12(b) has a low precision and low accuracy. We can observe that the measurements that revolves around 180° often has outlier, and that the calculation of the orientation about Z-axis is off by $\pm 180°$. in all of the tests. Figure 5.14 represents the three different poses of the marker in the image frame.

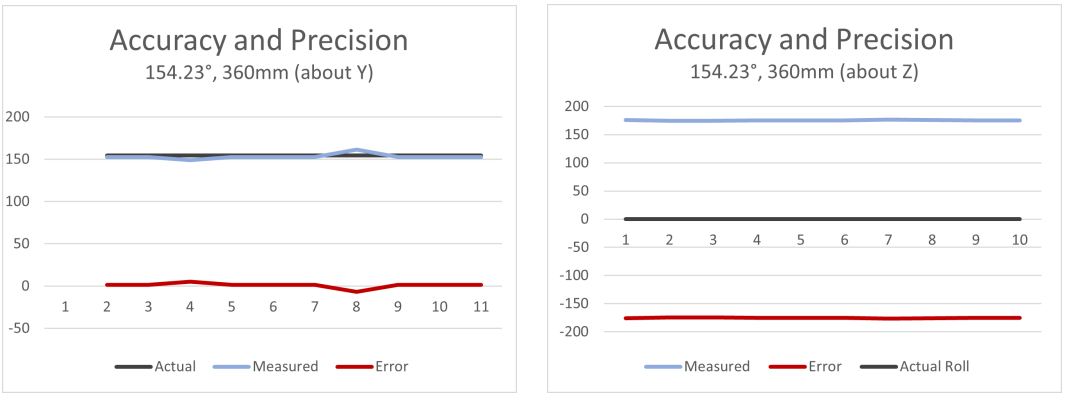


Figure 5.11: From left to right, the graphs represent: (a) Accuracy and precision about Y-axis. (b) Graph of error on the orientation about Z-axis. Both graphs represent data measured of 10 images from the following pose of the marker in relation to the camera: 154.23°, 360mm distance

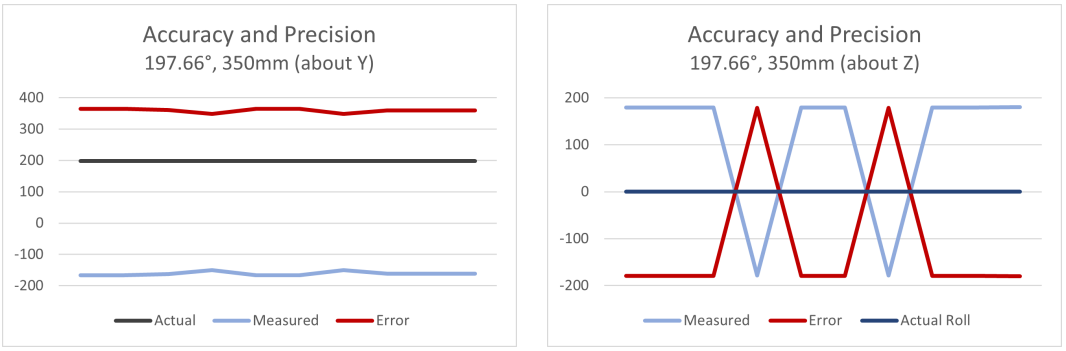


Figure 5.12: From left to right, the graphs represent: (a) Accuracy and precision about Y-axis. (b) Graph of error on the orientation about Z-axis. Both graphs represent data measured of 10 images from the following pose of the marker in relation to the camera: 197.66°, 360mm distance

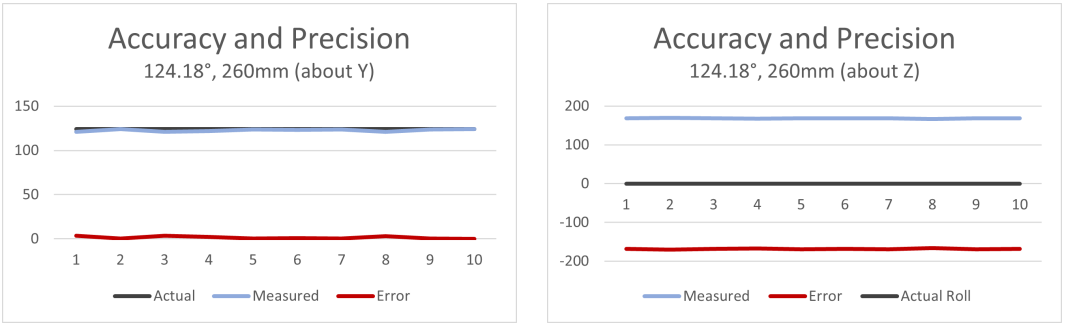


Figure 5.13: From left to right, the graphs represent: (a) Accuracy and precision about Y-axis. (b) Graph of error on the orientation about Z-axis. Both graphs represent data measured of 10 images from the following pose of the marker in relation to the camera: 124.18°, 260mm distance

Angle	Error Y	Error Z
197.66	359.38±6.23	-107.77±143.15
154.23	1.01±2.89	-175.48±0.63
124.18	1.40±1.34	-168.36±0.93

Table 5.11: Mean and standard deviation value of the calculated error for each angle

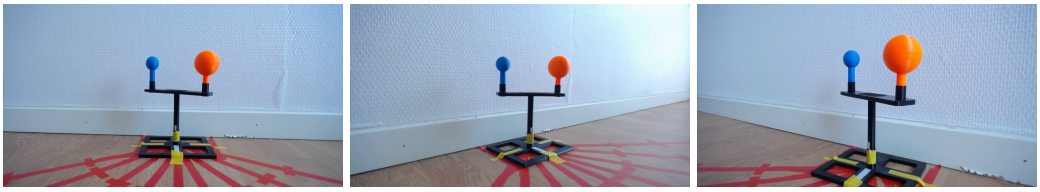


Figure 5.14: From left to right, the images represent the marker detected from: (a) A distance of 360mm and an expected rotation of 154.23°. (b) A distance of 360mm and an expected rotation of 197.66°. (b) A distance of 260mm and an expected rotation of 124.18°.

To investigate further, we calculate the mean and standard deviation values of each of the 10 measurements. We can observe, from the measurements of the error of both 154.23° and 124.18° around the Y-axis, that the system prooves very precise. We can also observe that the measurements when the angle is rotated 197.66° is almost 360° off.

5.7 Different Illuminations

This test will evaluate the robustness of the system, when placed in different illuminations. To isolate possible interference, the background will be plain in all test frames. The camera will be attached to the marker with a rigid rod, in order to move the setup around to different places with different illuminations, expecting the same result each time. The distance from the camera to the marker is 470mm and the marker is rotated with 0°. For each iteration of this test, the camera will capture an image, the system will detect both spheres, calculate the distance to the mid of both spheres and calculate the orientation of the marker. This is done four times.

5.7.1 Result

The system detected both spheres on all images, making a total of 4 images. The measured data has been listed on table 5.12. The mean and standard deviation for measuring the orientation about the Y-axis is 87.34°±149.97° and 88.46°±154.76° when measuring the orientation about the Z-axis. Looking at figure 5.15, one of the estimations is off. Calculating the mean and deviation without the result from

figure 5.15(a), the result is 173.81°±8.76° about the Y-axis and 177.8°±1.64°. The data has been plotted on a graph to visualize the result and can be seen on figure 5.16.

Rotation Y (degrees)	**Measured Rotation YZ (degrees)**		**Error (degrees)**	
Y	Y	Z	Y	Z
180.0	-172.08	-179.57	352.08	179.57
180.0	161.42	175.52	18.58	-175.52
180.0	180.0	178.6	0.0	-178.6
180.0	180.0	179.3	0.0	-179.3

Table 5.12: Data of the measured rotation about YZ-axis when exposed to different illuminations

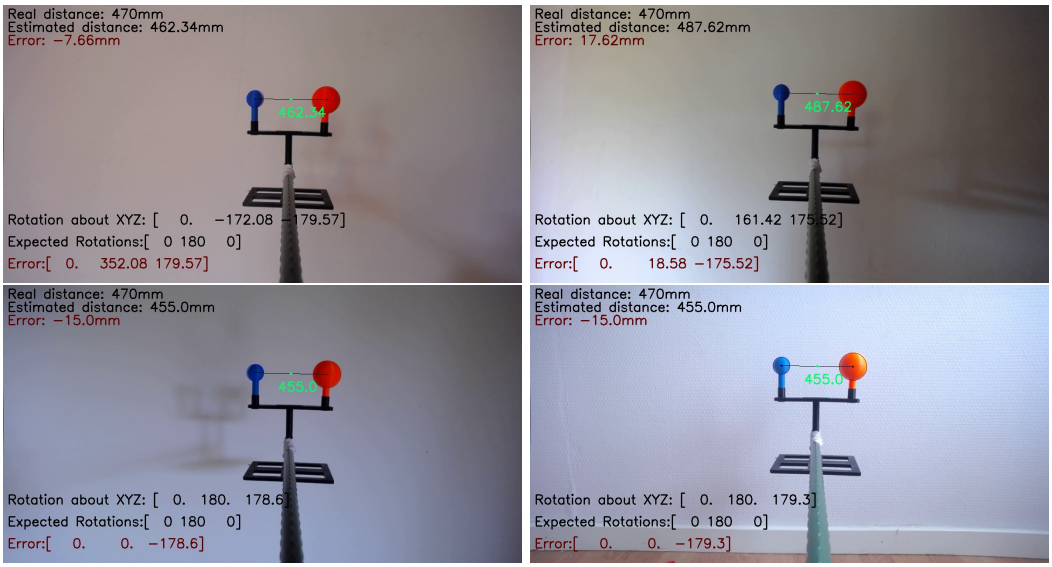


Figure 5.15: From left upper corner, the images represent: (a) Marker in a slightly darker setting than the previous tests. (b) Marker in an even darker setting. (c) Marker in a different lighting. (d) Marker placed in a slightly lighter setting than the previous tests.

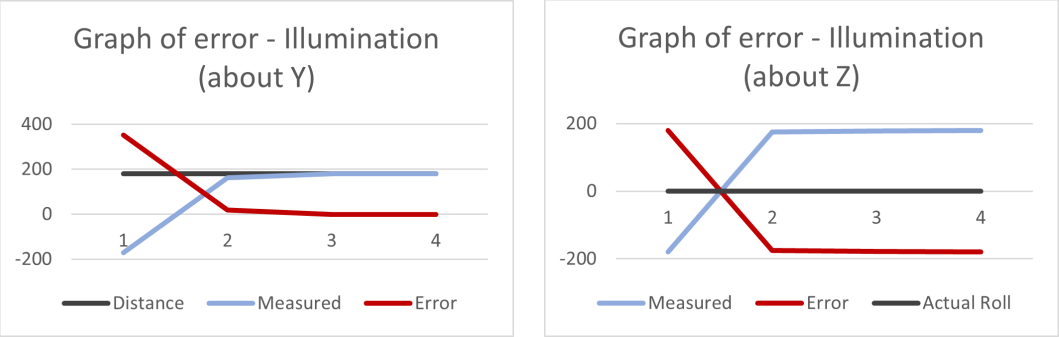


Figure 5.16: From left to right, the graphs represent: (a) Graph of data on the orientation about the Y-axis from test on illumination. (b) Graph of data on the orientation about the Z-axis from test on illumination

6 - Conclusion

In this chapter, we elaborate on the test results. We can conclude that the system only works within a distance of maximum 560mm. In addition, we can from tests in section 5.3, observe that the greater the distance between the marker and the camera, the higher an error we get. Looking at the data from test section 5.4, we observe that the measurements of the rotation about Y-axis is not very precise, not accurate. The best result are found at a rotation, where the camera is facing the marker such that the blue sphere is closer to the camera than the orange. We can also conclude from this test, that the distance has a say in both accuracy and precision of the measurements. A smaller distance gives a higher accuracy and precision. The various angles does not appear to have a great impact on the error. We do get more precise measurements at the angles that are at further distance to 0°, but we do also not get as much data from these angles, due to the lack of sphere detection. We can, with data derived from test section 5.5 conclude that the system does not detect the marker when one sphere is occluded. The smallest error is found at quite a high rotation of the marker. The highest error measured the rotation as being almost a full rotation from the expected result, meaning that the detection was not far from the truth when visualizing the difference with geometry, but the calculations produce the high value. The detection is great with a smaller distance, but due to the outlier in detection, this can not be seen in the tables describing the mean and standard deviation for this test included in this report. With the result from test section 5.6, we can conclude that the system in these almost static scenarios are fairly accurate with only a small deviation. We can also conclude that the error of the measured rotation about Y-axis at a high angular rotation is appears to be from accurate nor precise. This is caused by miscalculations in the system. The system calculates the rotation from -180° to 180°. If the error is close to 360°, and we are working with a circle, we know for a fact that the actual error is far less. The results from test section **??** are hard to conclude upon, due to the lack of repeated measurements in different illumination. The four different images used for tests could all provide information about the position and orientation of the marker. We can observe that the accuracy of the marker detection are fairly better in lighter settings. The HSV threshold values set limitations for detecting the spheres in different illuminations.

Not the camera nor the marker is rotated about the Z-axis purposely, yet the results from the tests shows that there is always some rotation applied in this dimension. We detect that the minimum error of the rotation about the Z-axis is when both the camera and the marker has little to no rotation. The greater the rotation about the Y-axis, the greater of an error when calculating the orientation about Z-axis. We can therefore conclude that the distance does not have a great influence on the error of the measurements of orientation about Z-axis. That is affected by the rotations about the Y-axis.

To sum up, we can conclude that the detection of the marker is successful when the distance between the camera and the marker is < 560mm and both spheres are in the camera frame, with a fairly illuminated environment with no background noise. The calculation of the orientation of the marker needs to be adjusted such that a successful calculation does not appear to be unsuccessful.

7 - Discussion

In this chapter all of the thoughts and reflections done doing the end of this project have been divided into different focus areas. Section 7.1 describes the reflections upon the marker design. Section 7.2 discuss the completed tests, their results and the need for further tests. Section 7.3 describes the large error calculations in orientation estimation, why the calculations was far off and how it might not even be such a big error. Section 7.4 and Section 7.5 reflect upon further improvements, both in marker detection and the step that comes after.

7.1 Design of the Marker

We designed each sphere on the marker to be of different colors and different sizes. As of right now, the system does not use the difference in sizes of the markers. Both spheres could have been of same size and the system would supposedly have been working the same, with little to no difference in test results. With two spheres located with a static distance, the rotation about X-axis can not be calculated. This is an intentional design choice, the detection of that rotation is redundant, due to the proposed placement of the marker on the trailer. We do not expect the trailer to be rotated in that direction when it is parked and in need for being picked up by the robotic vehicle.

7.2 Reflecting Upon the Errors from Tests

The test section provides test done on the distance between the camera and the marker, along with the rotation about the Y-axis, but does the results reflect upon the actual error? We want to test knowing ground truth, but the accuracy for the ground truth is limited. The manual "ground truth" measurements have been done with tape on the floor and a ruler. The measurements are not as precise as we ideally would want. With more time we could use an active tracking technology. It is possible to track and measure the position of moving objects with high accuracy. It is not easy to tweak the software gaining accurate measurements, if the test is compared to non-accurate manual measurements. We do observe a large error around the Z-axis. The tests where this happens, the spheres are rotated 180° in comparison to the first tests. The directional vector is calculated from the orange

sphere to the blue. The system could potentially calculate this as 180° around the Z-axis instead of the Y-axis, in which it was actually performed. To the system this would be the same. The biggest flaw in this case, is that the system outputs a rotation matrix in which the marker is rotated 180° around both the Y-axis and the Z-axis, which is not correct.

7.2.1 More Tests

We have tests that we would love to perform to understand how to improve the system. Due to the lack of time of executing and analysing more tests, we did not manage to perform the following tests: More tests in different illuminations, test with background noise, change of rotation about Z-axis.
We only captured four images for tests on the system's performance in different illuminated environments. We would ideally have multiple (at least 10) images in the same illuminated environment. We would ideally have used darker and brighter images, as well as warmer(orange) and cooler(blue) light. We predict the system to work poorly with background noise, but did not run any test. It would have been interesting to see in which cases the system would perform great and which it would not, to understand the limitations better, and with that - the possibility to optimize the design of the marker and/or the software. We only performed rotations about the Y-axis doing tests, it would be interesting to do similar tests, but changing the rotation about the Z-axis. This would require a new base for the spheres. The base only had the possibility of turning around itself. Creating a new base and running tests would be far from impossible to achieve, but due to time limitations, this was not prioritized.

7.3 Flaws in the Orientation Calculation

The system is created to detect two colored circles in the same image frame and to calculate the orientation between them. The orientation of the marker can only be calculated when two circles of the specific color are detected, limiting the system to not being able to detect the marker at all 360°, since the spheres will occlude one another at certain orientations. The software algorithm calculating the angular degrees has a range from -180° to 180°. When the sphere is turned 181° around the Y-axis, the system will calculate that orientation as being -179°. Right now the system calculated the possible orientation about all of the axis, meaning that when the marker is turned 180° around the Y-axis, it could just as well have been rotated 180° around the Z-axis. The system does not compensate for this error, which results in a rotation matrix that believes the marker to be rotated 180° about both axis.

7.4 Possible Improvements

The system would be greatly improved if the orientation calculation is improved upon, such that the rotation matrix will not be calculated as if a rotation has happened about two axes, when it has only happened about one axis. The system does not allow one sphere to be occluded, which should be improved upon such that the marker can be detected from all orientations. The desired distance in which to detect the markers has been mentioned in the introduction as being 1m. Right now the system can only detect the spheres at a distance of half a meter. The system should improve on the distance of detection in order to satisfy the requirement.

7.5 Future Work

When the system has been optimized and is able to detect the marker from all orientations, the next step would be to attach the spheres onto the trailer and a camera to the robot. Then a transformation matrix should be calculated from the robot to the camera, such that the robot knows where the marker is positioned in relation to itself and not in relation to the camera. The robot would then receive a go-to command, containing pose estimation of the marker in relation to the robot. The next step will be to test out the system in real life scenarios, first inside, then outside, tweaking the system such that the robotic vehicle will be robust enough to reach the trailer within a precision low enough to being able to attach the trailer to itself.

Bibliography

[1] Capra Robotics. *About Capra Robotics*. (Accessed on 1/03/2022). URL: https://capra.ooo/about-capra-robotics/.

[2] Ashok Dahal, Jakir Hossen, Chennupati Sumanth, Ganesh Sistu, Kazumi Malhan, Muhammad Amasha, and Senthil Yogamani. "DeepTrailerAssist: Deep Learning Based Trailer Detection, Tracking and Articulation Angle Estimation on Automotive Rear-View Camera". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 2339–2346. DOI: 10.1109/ICCVW.2019.00287.

[3] RFWirelessWorld. "Advantages of Deep Learning | disadvantages of Deep Learning". In: (Accessed on 18/04/2022). URL: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Deep-Learning.html.

[4] Małgorzata Łaganowska. "Application of vision systems to the navigation of mobile robots using markers". In: *Transportation Research Procedia* 40 (2019). TRANSCOM 2019 13th International Scientific Conference on Sustainable, Modern and Safe Transport, pp. 1449–1452. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2019.07.200. URL: https://www.sciencedirect.com/science/article/pii/S2352146519303709.

[5] Filippo Costa, Simone Genovesi, Michele Borgese, Andrea Michel, Francesco Alessio Dicandia, and Giuliano Manara. "A Review of RFID Sensors, the New Frontier of Internet of Things". In: *Sensors* 21.9 (2021). ISSN: 1424-8220. DOI: 10.3390/s21093138. URL: https://www.mdpi.com/1424-8220/21/9/3138.

[6] Advanced Realtime Tracking. *Markers*. (Accessed on 31/05/2022). URL: https://ar-tracking.com/en/product-program/markers.

[7] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[8] Manu Mathuria Khushboo Mantri Neeraj Bhargava Ritu Bhargava. "The Effective QR Code Development using VB.NET". In: *IJCART* (2013). (Accessed on 18/02/2022). URL: https://www.academia.edu/3675096/The_Effective_QR_Code_Development_using_VB_NET?from=cover_page.

[9] Chin-Hung Teng and Bing-Shiun Wu. "Developing QR Code Based Augmented Reality Using SIFT Features". In: *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*. 2012, pp. 985–990. DOI: 10.1109/UIC-ATC.2012.56.

[10] Huijuan Zhang, Chengning Zhang, Wei Yang, and Chin-Yin Chen. "Localization and navigation using QR code for mobile robot in indoor environment". In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2015, pp. 2501–2506. DOI: 10.1109/ROBIO.2015.7419715.

[11] Alain Pagani, Johannes. Koehler, and Didier Stricker. "Circular markers for camera pose estimation". In: *WIAMIS 2011*. 2011.

[12] Artur Sagitov, Ksenia Shabalina, Roman Lavrenov, and Evgeni Magid. "Comparing fiducial marker systems in the presence of occlusion". In: *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*. 2017, pp. 377–382. DOI: 10.1109/ICMSC.2017.7959505.

[13] Minho Hwang, Brijen Thananjeyan, Samuel Paradis, Daniel Seita, Jeffrey Ichnowski, Danyal Fer, Thomas Low, and Ken Goldberg. "Efficiently Calibrating Cable-Driven Surgical Robots With RGBD Fiducial Sensing and Recurrent Neural Networks". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5937–5944. DOI: 10.1109/LRA.2020.3010746.

[14] Jochen Schmidt, Ingo Scholz, and Heinrich Niemann. "Placing Arbitrary Objects in a Real Scene Using a Color Cube for Pose Estimation". In: vol. 2191. Dec. 2001. ISBN: 978-3-540-42596-0. DOI: 10.1007/3-540-45404-7_56.

[15] Aniket Gadwe and Hongliang Ren. "Real-Time 6DOF Pose Estimation of Endoscopic Instruments Using Printable Markers". In: *IEEE Sensors Journal* 19.6 (2019), pp. 2338–2346. DOI: 10.1109/JSEN.2018.2886418.

[16] Ye M. Chan PL. et al. Zhang L. "Real-time surgical tool tracking and pose estimation using a hybrid cylindrical marker". In: (2017). (Accessed on 18/02/2022), 921–930. URL: https://link.springer.com/article/10.1007/s11548-017-1558-9.

[17] Weibing Chen, Gaobo Yang, and Ganglin Zhang. "A Simple and Efficient Image Pre-processing for QR Decoder". In: (Sept. 2012). DOI: 10.2991/emeit.2012.46.

[18] Yuheng Yan, Yiqiu Liang, Zihan Zhou, Bin Jiang, and Jian Xiao. "FastQR: Fast Pose Estimation of Objects Based on Multiple QR Codes and Monocular Vision in Mobile Embedded Devices". In: *Wireless Communications and Mobile Computing* 2021 (Sept. 2021), pp. 1–9. DOI: 10.1155/2021/9481190.

[19] Georgios Triantafyllidis, Michalis Dimitriou, Tsampikos Kounalakis, and Nikolaos Vidakis. "Detection and Classification of Multiple Objects using an RGB-D Sensor and Linear Spatial Pyramid Matching". In: *Electronic Letters on Computer Vision and Image Analysis* 12.2 (Apr. 2013), p. 78. DOI: 10.5565/rev/elcvia.523. URL: https://doi.org/10.5565/rev/elcvia.523.

[20] Anuj Shivaji Sabale and Yogita M. Vaidya. "Accuracy measurement of depth using Kinect sensor". In: *2016 Conference on Advances in Signal Processing (CASP)*. 2016, pp. 155–159. DOI: 10.1109/CASP.2016.7746156.

[21] Thuy Mai. *What are passive and active sensors?* (Accessed on 31/05/2022). 2017. URL: https://www.nasa.gov/directorates/heo/scan/communications/outreach/funfacts/txt_passive_active.html.

[22] MathWorks. *Camera calibrator*. URL: https://www.mathworks.com/help/driving/ug/calibrate-a-monocular-camera.html#:~:text=A%20monocular%20camera%20is%20a,camera%2C%20you%20must%20calibrate%20it..

[23] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. An Alan R. Apt book. Prentice Hall, 2003. ISBN: 9780130851987. URL: https://books.google.dk/books?id=VAd5QgAACAAJ.

[24] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[25] Mathworks. *What Is Camera Calibration?* (Accessed on 30/05/2022). URL: https://www.mathworks.com/help/vision/ug/camera-calibration.html.

[26] vision team. *What is a stereo vision camera?* (Accessed on 31/05/2022). 2018. URL: https://www.e-consystems.com/blog/camera/technology/what-is-a-stereo-vision-camera-2.

[27] Chuanye Tang, Xinwen Zhao, Jianfeng Chen, Long Chen, and Yazhou Zhou. "Fast stereo visual odometry based on LK optical flow and ORB-SLAM2". In: *Multimedia Systems* (June 2020). DOI: 10.1007/s00530-020-00662-9.

[28] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[29] Robin C. Colclough. *Viva3D Real-time Stereo Vision*. (Accessed on 31/05/2022). 2016. URL: https://www.viewpoint-3d.com/media/Viva3D\%20Stereo\%20Vision\%20user\%20manual\%20en\%202016-06.pdf.

[30] Joseph Robinson. *How Computers See Depth: Recent Advances in Deep Learning-Based Methods*. (Accessed on 31/05/2022). 2021. URL: https://towardsdatascience.com/how-computers-see-depth-deep-learning-based-methods-368581b244ed.

[31] Andreas Klaus, Mario Sormann, and Konrad Karner. "Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure". In: vol. 3. Jan. 2006, pp. 15–18. DOI: 10.1109/ICPR.2006.1033.

[32] intel RealSense. *Beginner's guide to depth (Updated)*. (Accessed on 31/05/2022). 2019. URL: https://www.intelrealsense.com/beginners-guide-to-depth/.

[33] Zhihan Lv, Jaime Lloret Mauri, and Houbing Song. "Editorial RGB-D Sensors and 3D Reconstruction". In: *IEEE Sensors Journal* 20.20 (2020), pp. 11751–11752. DOI: 10.1109/JSEN.2020.3015417.

[34] Hemprasad Patil, Ashwin Kothari, and K. Bhurchandi. "3-D face recognition: Features, databases, algorithms and challenges". In: *Artificial Intelligence Review* 44 (Oct. 2015), pp. 393–441. DOI: 10.1007/s10462-015-9431-0.

[35] Kudan. *Depth cameras and RGB-D camera SLAM*. (Accessed on 31/05/2022). 2020. URL: https://www.kudan.io/archives/517.

[36] Alphonse Pja and venkata sriharsha Kuncham. "Depth perception in single rgb camera system using lens aperture and object size: a geometrical approach for depth estimation". In: *SN Applied Sciences* 3 (June 2021). DOI: 10.1007/s42452-021-04212-4.

[37] Ricardo Oliva-García, Sabato Ceruso, José G. Marichal-Hernández, and José M. Rodriguez-Ramos. "Monocular Real Time Full Resolution Depth Estimation Arrangement with a Tunable Lens". In: *Applied Sciences* 12.6 (2022). ISSN: 2076-3417. DOI: 10.3390/app12063141. URL: https://www.mdpi.com/2076-3417/12/6/3141.

[38] Praveen. *How to find HSV range of an Object for Computer Vision applications?* (Accessed on 30/05/2022). July 2020. URL: https://medium.com/programming-fever/how-to-find-hsv-range-of-an-object-for-computer-vision-applications-254a8eb039fc.