CHALMERS
UNIVERSITY OF TECHNOLOGY

Autonomous Air Hockey

SSYX02-16-82

Bachelor thesis at the Department of Signals and Systems

Christofer Berntsson, Lars Brown, Simon Fitz,
Albin Hallberg, David Sondell, Kim Svensson

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Abstract

The purpose of this bachelor project was to develop a robot that is capable of playing air hockey against either another robot or a human. First, by using a camera, a program controlling the robot tracks the puck and calculates a trajectory. The calculations are made on an external computer running a program written in C++. The program continuously sends information to a micro controller which controls a mechatronic system. The mechatronic system consists of two stepper motors and two stepper drivers together with a combination of linear rails, linear bearings and custom 3D printed parts. This system moves the mallet based on the information sent from the computer to emulate a human player and thereby defeat the opponent.

The system uses different strategy algorithms to decide how to move the mallet depending on the speed and direction of the puck and is capable of both defending against incoming pucks and attacking if the puck moves below a certain speed. The project has been completed with the implementation being successful although there are definitely some points of improvement.

Sammanfattning

Syftet med detta kandidatarbete var att utveckla en robot som kan spela air hockey mot antingen en annan robot eller en människa. Genom att använda en kamera följer programmet som styr roboten puckens rörelse och beräknar en förutsägelse av hur den kommer att röra sig. Beräkningarna sker på en extern dator som kör ett program skrivet i C++. Programmet skickar kontinuerligt information till en mikrokontroller som styr det mekatroniska systemet. Det mekatroniska systemet består av två stegmotorer och två stegmotordrivare som driver den mekaniska delen som är byggd av kombination av axlar, linjärlager och 3D utskrivna delar. Detta system förflyttar klubban för att efterlikna en människas spelsätt och förhoppningsvis vinna över motståndaren.

Systemet använder olika strategialgoritmer för at avgöra hur klubban ska flyttas beroende på puckens hastighet och riktning. Roboten är kapabel att både försvara mot inkommande puckar samt attackera om pucken rör sig långsammare än en specificerad hastighet. Slutsatsen är att projektet överlag är lyckat även om förbättringar kan göras.

Acknowledgements

The whole team would like to thank the other air hockey project for support and motivation.

Hakan Köroğlu for feedback and support throughout the project.

The department of Signals and Systems at Chalmers University of Technology.

Contents

1
Introduction

1.1 Background

Mechatronic systems are introduced to solve a growing number of every day tasks. If a system is to perform well, it is crucial that it is fast, accurate and reliable. Detecting and reacting to quickly moving objects are challenges that are found frequently in manufacturing and other applications. The developments in mechatronic systems technology have made it possible to introduce robotic units to various game platforms like table-tennis [1], football [2] and baseball [3]. This project is based on the development of such a system for an air hockey table.

Traditionally, air hockey is a game played by two opposing players on a rectangular field as represented in Figure 1.1. The winner of the game is decided based on the number of goals scored by each player. Each player has a slot on his or her side of the table and a goal is scored when the puck enters the opponent's slot. Around the whole field there is a rail which prevents the puck from falling off the side of the table. The puck bounces easily against the rail without losing a significant amount of its momentum. The puck floats on an airbed with negligible friction, which leads to a fast game play. Due to the fast game play, the reaction time of each player and the estimation of the path of the puck is vital for the outcome of the game. A video from the world championship final 2010 [4] displays the game play.

To imitate a human playing air hockey, the robot must be able to perform the same actions as a human during the game. These actions include tracking the puck, moving the mallet and having an understanding of the game mechanics and following the rules.

There are many different kinds of robots that are able to play air hockey: robot arms, combinations with linear rails and more complex solutions. To protect the goal, the robot only needs to move along the short side of the table, but if an attack mode is desired, the mallet will have to be able to move in two dimensions, thus making the system more complex.

1.2 Purpose

The purpose of our project is to develop and build an air hockey robot that operates semi-autonomously. The robot is supposed be fully autonomous when the game is

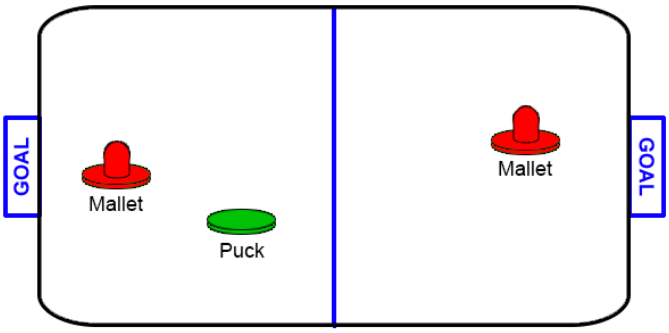


Figure 1.1: Illustration of an air hockey table and gaming accessories (two mallets and a puck). The vertical blue line is referred to as the middle line and seperates the two players' sides.

on, but will need human interaction to place the puck back on the table when either the robot or the opponent has scored a goal. The robot should be able to play against either a robot or a human. A second group is also assigned to develop a similar, separate robotic unit to be mounted to the other side of the same table. As a common project goal, the different robots are supposed to play against each other to compare their performances.

During the project, members expect to gain more knowledge and improve their skills in the analysis and design of vision-based control systems (with all the related mechanical and electronic units) as well as project management.

1.3 Problems and Tasks

With this project, multiple challenges were faced because of the diversity of the tasks to be performed. To cope with these challenges, the project work was divided into several separate tasks.

1.3.1 Tracking the Puck

Air hockey is a fast-paced game, which means that the puck must be located as early as possible. Since a green puck is used, a vision system that is able to track the puck on the field based on color is hence needed. Sampling rate is crucial since it sets the limitations for calculating the trajectory of the puck. Accuracy in acquiring the position of the puck is also very important, since the error from locating the puck can not be compensated for when controlling the motion of the mallet.

1.3.2 Predicting the Future Trajectory of the Puck

Once at least two measured positions of the puck and the time between them have been determined, both the speed and direction of the puck can be calculated. When these two parameters have been calculated, a trajectory can be estimated easily since the physics of the game is very simple with linear motion under very low friction. From this point a vector representation of the expected trajectory will give the system an indication of where to move the mallet before the puck reaches the foreseen position.

1.3.2.1 Multiple Measurement Points

With two measured locations of the puck, a very basic estimation of the puck's trajectory can be made. However, with only two measured points the contingency is high and the predicted trajectory might not be accurate enough. If there are more than two measured locations, the accuracy of the predicted trajectory will be improved, but the method to calculate a vector that fits the trajectory becomes more complex.

1.3.2.2 Bounce Against the Rail

The bouncing against the rails adds to the complexity of the mathematical model and makes the estimation of the trajectory more difficult. The law of reflection [5] will not perfectly apply to this system and thus a new mathematical model of reflection might have to be used. The bouncing itself might alter the velocity of the puck and thus the mathematical model of the trajectory will have to take this into account. If only two samples from the vision system are used and a bounce occurs in between the samples, the bounce might not be recognised and a false trajectory might be calculated as illustrated in Figure 1.2.

1.3.3 Controlling the Movement of the Mallet

After the puck has been located and a first trajectory has been calculated, the mallet has to be moved by a mechatronic system. The system has to be fast enough to defend the goal against the incoming puck and to shoot the puck towards the goal of the opponent. The system will have to be able to move the mallet simultaneously in two directions, along and across the table. This means that there will have to be a mechanical system that holds the mallet and that is able move it with high speed and accuracy. An essential goal will hence be to keep the weight of the moving parts low and design a smooth, robust, low-friction rail system without exceeding the available budget.

1.3.4 Programming a Game Strategy

The last step is to program a strategy which the robot will use to defend the goal and attack the opponent. It should be able to use different modes, which could be defense, attack or a combination of the two.

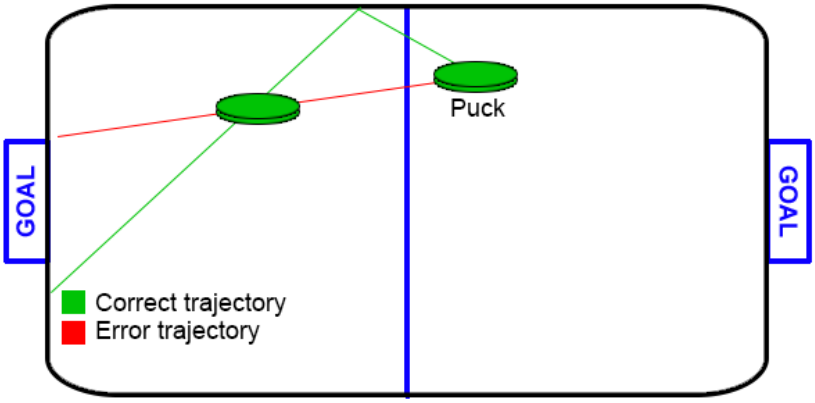


Figure 1.2: Representation of a potential error when a bounce occurs. When a bounce occurs between two samples, shown as the green pucks, the data suggest that the puck traveled straight as shown in red. However, the puck could also have traveled as the green line suggests.

1.4 System Requirements

Listed below are the overall requirements and specifications for developed the prototype system. A full list of requirements which have been evaluated can be found in Appendix A.

- The prototype should be able to block a moving puck from the middle line at 3 m/s towards the goal.
- The prototype must not protrude over the middle line into the opponent's half.
- The prototype should be able to strike the puck, sending it towards the opponent at a speed of 8 m/s.
- The camera system should be able to locate a still puck at an accuracy of ± 5 mm.
- When moving the mallet 400 mm with the actuators and without feedback control, the actual distance moved should not exceed a fault of ± 10 mm.
- The prototype should be constructed with materials which leads to a sustainable design.

1.5 Delimitations

The delimitations are as follows:

- The system will require human interaction when starting the game and/or if a player scores a goal. This is due to the limited time and budget of the project.
- The system might need help with software calibration when operating from a cold start, thus making the robot require human interaction.
- Only one prototype will be constructed as the robot opponent will be designed and constructed by another development group. This decision was made together with the other group to further promote competition and thus create motivation for better designs.
- The airspace above the middle line will be shared due to complications with interfering with the camera's field of view.
- The robot will not be required to be able to operate in an arbitrary environment. Limited lightening might cause the vision system to be unable to perform its task. Due to the limited project budget and duration, only one vision system will be implemented and the environment has to be adjusted to the requirements of the chosen system.
- The system will be designed to function on a specific air hockey table with dimensions 198 x 106.5 *cm* and will therefore not be operational on a table with other dimensions without reprogramming and changing the mechanical construction.

1.6 Thesis Outline

The outline of this bachelor thesis is as follows. Chapter 2 presents an overview of the system with its most essential subsystems described briefly. This chapter aims to facilitate an understanding of the overall system before getting into a more detailed description of each subsystem in the chapters that follows. Chapter 3 describes the vision system by explaining how the puck is detected and what methods are used. Chapter 4 presents the mechanical design that has been chosen and built in order to move the mallet. In Chapter 5 a description of the electrical design is presented which connects the vision system with the mechanical system. In order to calculate the trajectory of the puck and make an appropriate move according to these calculations, Chapter 6 explains the game strategies and algorithms behind the decision making of the robot. Chapter 7 then sets up different tests to check if the system requirements in the introduction have been met. The next chapter of the report discusses the results of the project work and possible improvements. Latsl follows a conclusion about the project as a whole.

2
Overall System Description

This chapter will give a brief description of the system as a whole, introduce the main concepts of the following chapters and explain the interaction of the subsystems. The main subsystems of this project are the vision, mechanical and electrical systems. After this follows a description of the strategy and prediction algorithms. Each subsystem will be explained in further detail in the following chapters.

An overall description of the whole system is provided in Figure 2.1. Basically, a camera will send information to a computer which will perform the image analysis. The image analysis is done in a self-developed program written in C++, which implements an open source library called OpenCV. The OpenCV library simplifies several aspects of the programing, for which there would not be enough time to do otherwise. The program tracks the puck using the camera and calculates a prediction of the trajectory and uses the strategy algorithms to decide what information to send. The program then sends information to the micro controller unit (MCU) about where the mallet should move. The communication with the MCU is performed through serial communication. The MCU will control the stepper motors through the motor drivers, which is power by a power supply unit (PSU), and thereby move the mechanical system. The mechanical system consists of steel shafts, linear bearings and custom 3D-printed parts. The belt system used is called CoreXY, and eliminates the need to move one of the motors, which significantly reduces the weight of the moving parts.

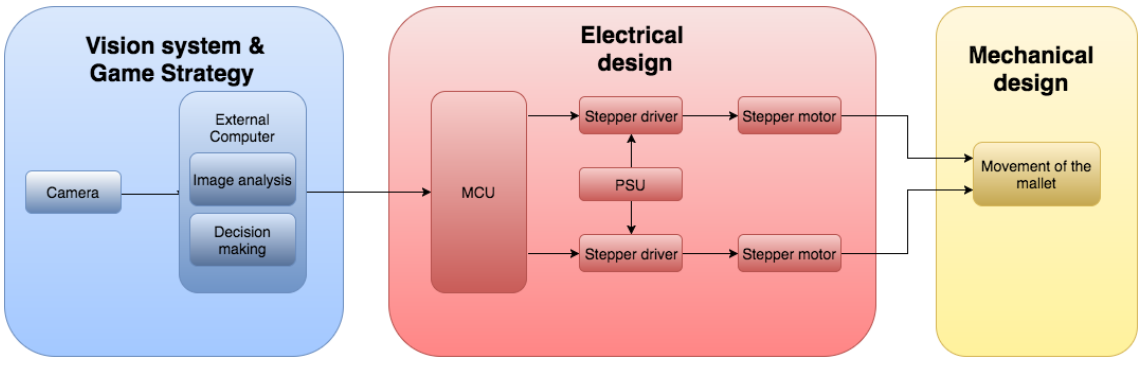


Figure 2.1: Flowchart of the system where the input signals flow through the system and end up as motions in of the mallet.

3

Vision System

This chapter addresses the main source of input data for the air hockey robot, which is the camera. The main point of the vision system is to filter out the position of the puck and the mallet by using their distinctive color. In this chapter more details about the camera hardware and image analysis software will be described.

Information about the hardware will be presented first in this chapter followed by a brief description of the image analysis libraries that were used in this project. After the library has been introduced, the filtering algorithms and calculation algorithms regarding image analysis are described with some mathematical background.

3.1 Camera Hardware

The problem of locating the puck was addressed with a vision system. A camera solution was best suited for this project and a couple of different camera solutions have been examined. There are a couple of features which were especially important for this project. The camera had to capture images at a high frame rate so that an accurate trajectory could be predicted at an early stage. The images also had to be clear enough to be analysed with suitable image processing algorithms [6] for accurate positioning of the puck and the mallet.

The project used a camera from ELP which captures images at a frame rate of 120 frames per second (fps) with a resolution of 640 x 480 pixels. This camera does not have integrated image analysis algorithms and thus these algorithms had to be developed and run on a separate computer.

The camera was placed above the table as low as possible, but still covering the whole table in its field of view. The game field is 200 *cm* long and the ELP camera came with a 70° angle of view lens. The required height was calculated to be 143 *cm* by using equation 3.1 where Θ is the camera's angle of view. The height was measured from the game field surface to the camera lens.

$$Height = \frac{Length/2}{tan(\Theta/2)} = \frac{100cm}{tan(35°)} \approx 143cm \tag{3.1}$$

3.2 Image Analysis and Puck Detection

OpenCV [7] is a library used for image processing with features which were particularly useful in this project. The OpenCV library was used in this project and the functions used will be described in this section. In Table 3.1 some of the features of OpenCV used in this project are listed and briefly explained.

Table 3.1: Selection of most important features from OpenCV used in this project.

Feature	Description
VideoCapture	Object that listens and reads the input from a camera or video file.
Mat	Class used to create an object that can store an image as a matrix representation.
cvtColor	Convert an image from one representation to another. Example RBG to HSV.
inRange	Function to analyse an image. Returns a mask where pixels with the right properties are given a high value and other pixels are given a low value.
Moments	Creates an object that gives each pixel a weight equal to its intensity. Borrowing formulas from physics the centroid or center of mass can be calculated.
erode	Morphological operation that erodes the source image based on its structures with a specified pixel neighborhood.
dilate	Morphological operation that dilates the source image based on its structures with a specified pixel neighborhood.

VideoCapture is an object that handles input from the camera. With a VideoCapture object a single frame from a video feed could be retrieved and converted to a Mat object which is an image container using matrix representation.

The Mat object used in this project is a three dimensional matrix where the first dimension represents the height of the frame in pixels. The second dimension represents the width of the frame and the third dimension represents the color of the pixel. The third dimension contains three integer values and these values set the color of the pixel.

3.2.1 Color Representation: RGB and HSV

When representing a color with three integers, the most common way is called RGB (Red Green Blue). In this representation, the first integer represents the amount of red in the pixel as a value between 0 and 255. The other two integers work in the same way but for green and blue. The concept of RGB is shown in Figure 3.1. In OpenCV the order of the colors are different and a BGR (Blue Green Red) representation is used, but it works in the same way as the RGB representation.

	0	1	2
0	R = 0 G = 255 B = 114	R = 255 G = 255 B = 0	R = 0 G = 186 B = 255
1	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 94 B = 94
2	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255

Figure 3.1: A 3 x 3 matrix where each cell represents a pixel and the background of the cell corresponds to the RGB numbers written inside it.

RGB or BGR representation is not ideal for filtering out an object based on color. The problem with using RGB when tracking a color is that a certain value of any one of the three integers does not correspond to a color. The color of a pixel is depending on all three parameters, which makes defining the desired color more difficult than with another representation. This is displayed in Figure 3.2. For color filtering in this project, the BGR frame was converted to an HSV (Hue Saturation Value) representation. The conversion was done using a function from OpenCV called cvtColor, which converts a Mat object to another representation.

Hue in HSV is a value between 0 and 180 which represents a color spectrum where values around 70 correspond to green and by testing different values, the interval of 60 to 75 was found to work best for detecting the green puck. Saturation determines how intense the color green has to be in order to be considered green. Without this parameter a gray pixel with slightly green tendencies would be considered green. Value corresponds to light intensity, for example, the sun is considered yellow but when looking directly at it, a human can not determine the color due to the high intensity of the light. By leaving this parameter untouched, the light environment

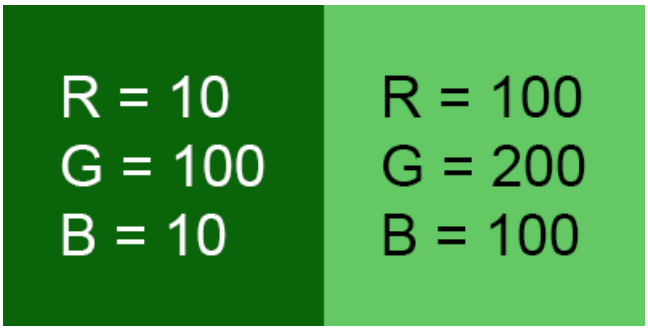


Figure 3.2: Shows two different types of green with RGB representation where the values differ significantly. This illustrates the difficulties of using RGB representation in color tracking. The numbers in the image represent the color of the background.

has very little effect to the system's ability to track the puck.

The HSV image and the parameters for hue, saturation and value were then used to create a binary image mask with the OpenCV function inRange, which gave every pixel that was considered as green the value of 1 (white) and every pixel that was not considered green the value of 0 (black).

3.2.2 Mathematical Morphology

In the binary image created by the inRange function, shown in Figure 3.3, the center of the puck could clearly be identified by a human. However, some noise bleed through the previous filter which decreases the accuracy of the algorithm that calculates the center of the puck (center of mass algorithm is described in Section 3.3). This is because it uses every pixel in the image with the value 1 (white pixels in Figure 3.3) and weigh them together. Note that this also means that the shape of the puck affects the algorithm's accuracy.

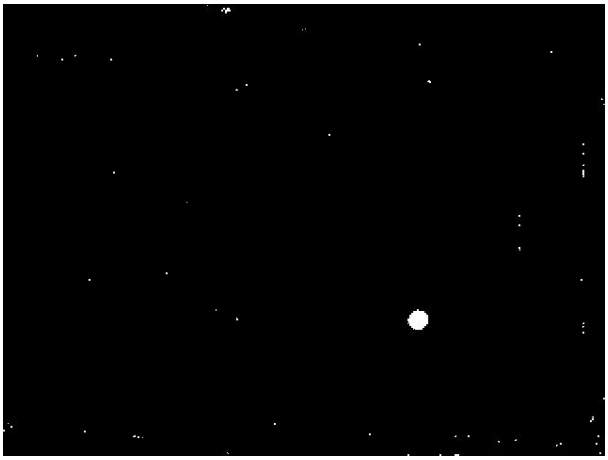


Figure 3.3: Binary image of air hockey table after filtering by threshold levels over HSV values. The large white circle in the lower right corner is the puck.

To get around this problem mathematical morphology was used to eradicate the noise. Mathematical morphology literally means *"using mathematical principals to do things to shapes"* [8]. Two different but similar operations was used to spread the black pixels, making them cover the white noise. The erosion also covers some parts of the puck. In order to restore it a white spreading operation is used to restore the circular puck. The darkness spreading operation is called erosion, and the white restoring operation is called dilation.

A more detailed description of the operations is found in the following sections: 3.2.2.1, 3.2.2.2 and 3.2.2.3. These sections are not essential for an overview of the system so a jump to section 3.3 from here can be made.

The binary image can be viewed as a set and the notations shown in Table 3.2 will be used to describe two binary operations.

Table 3.2: Mathematical notations used to describe dilation and erosion.

A	the image
B	the structuring element
$A \cup B$	the union of the sets A and B
$A \cap B$	the intersection of the sets A and B
$\oplus$	dilation sign
$\ominus$	erosion sign
$\circ$	opening sign
$\bullet$	closing sign

An introduction to the notion of an translated image set will also be needed. A movement vector t describes a translated image A in equation 3.2.

$$A_t = \{c | c = a + t \text{ for some a} \in A\} \tag{3.2}$$

3.2.2.1 Morphological Dilation

Dilation is a translation-invariant operation where an image, represented as the set A is manipulated using a structuring element B [8]. The mathematical definition of dilation can be seem below in equation 3.3.

$$A \oplus B = \{c | c = a + b \text{ for some a} \in A \text{ and b} \in B\} \tag{3.3}$$

By taking copies of A and translating them with help of the movement vectors defined by all the pixels in B and then by a union of these superimposed sets together, it can be written as in equation 3.4.

$$A \oplus B = \bigcup_{t \in B} A_t \tag{3.4}$$

Just as A can translate over B, B can translate over A. This can be interpreted as using the structuring element B as a stamp and making copies of B over every pixel of A. Similarly the union $B \oplus A$ is created as in equation 3.5.

$$A \oplus B = \bigcup_{t \in A} B_t \tag{3.5}$$

To further illustrate dilation, consider that A is the following 11 x 10 matrix and the structuring element B is represented by a 3 x 3 matrix given in Figure 3.4.

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	1	1	0
0	0	1	1	0	0	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0

Matrix A

0	1	0
1	1	1
0	1	0

Matrix B

Figure 3.4: Matrix A is to be dilated by using the structuring element B.

The dilation of A by B would then be executed by first moving the center of B over each pixel in A. With a copy of the structuring element over a specific pixel in A, then if the field created by the structuring element contains a 1, then the specific pixel in A also becomes a 1 in the output image. The result can be seen in Figure 3.5 where the ones has spread out.

3.2.2.2 Morphological Erosion

Erosion is a translation-invariant operation where an image, represented as the set A is manipulated using an structuring element B [8]. The mathematical definition of erosion can be seen below in equation 3.6.

$$A \ominus B = \{x | x = x + b \in \text{ A for every b} \in B\} \tag{3.6}$$

0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	0

Figure 3.5: Result after dilation using matrices found in figure 3.4.

Just like with dilation, by taking copies of A and translating them with help of the by movement vectors defined by all the pixels in B. However, this time the copies are intersected together and in the opposite direction. It can be written as in equation 3.7.

$$A \ominus B = \bigcap_{t \in B} A_{-t} \tag{3.7}$$

By using the same image to illustrate erosion, consider that A is the following 11 x 10 matrix and the structuring element B represented by a 3 x 3 matrix, seen in Figure 3.6.

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	1	1	0
0	0	1	1	0	0	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0

Matrix A

0	1	0
1	1	1
0	1	0

Matrix B

Figure 3.6: Matrix A is to be erosion by using the structuring element B.

The erosion of A by B would then be executed by first moving the center of B over each pixel in A. With a copy of the structuring element over a specific pixel in A, then if the field created by the structuring element only contains 1's then the

specific pixel in A also becomes a 1 in the output image. The result would look like in Figure 3.7 where the zeroes have spread over the ones.

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	1	0	0
0	0	0	0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Figure 3.7: Result after erosion using matrices found in Figure 3.6.

3.2.2.3 Morphological Opening and Closing

When translating ones and zeroes to black and white, it becomes clear that the effect of dilation expands white parts while erosion expands black. Because of this, one can think that they are the opposite to each other and can cancel out the other, just as in simple arithmetic with addition and subtraction. But in fact, this is not the case, see equation 3.8.

$$(A \ominus B) \oplus B \neq (A \oplus B) \ominus B \tag{3.8}$$

One needs to keep in mind that erosion requires the structuring element to fit inside the image, thus eliminating objects smaller than the structuring element. When an erosion is followed up by a dilation, the objects that remain will be somewhat restored, but the smaller objects would have been completely removed. The restoration from the dilation will also have trouble with filling in details. This combination of operations is known as an opening [9] and is illustrated in Figure 3.8. Openings can be very useful when trying to remove small objects, smoothing out edges or removing thin protrusions from objects.
An opening is written as in equation 3.9, which is nothing but the first part of 3.8:

$$A \circ B = (A \ominus B) \oplus B \tag{3.9}$$

The same tale is told about the second half of equation 3.8. But now it is a dilation followed up by an erosion. This is known as a closing [9] and works in the opposite manner. Whereas opening removes all the pixels that the structuring element will not fit, closing on the other hand will fill in all places where the structuring element will not fit in the image background.
An closing is written as the second half of equation 3.8 or as in 3.10.

$$A \bullet B = (A \oplus B) \ominus B \tag{3.10}$$

Again, one needs to keep in mind that this duality of the functions does not mean that an inverse operation is possible (i.e. an opening followed by an closing will not restore the image). An example of how an opening followed by a closing can be seen in Figure 3.8.

3.3 Position and Distortion

The mask image of ones and zeroes described in the previous section from Figure 3.8 was used to calculate the center of the puck. Each pixel was handled as a physical part with a mass, either 1 or 0. Using the laws of physics about center of mass, the location of the puck could be found. The coordinates for the center of mass in the mask are the same as the coordinates of the center of the puck [10]. The puck's location was handled as a point in its center.

The position could not be used directly due to the barrel distortion caused by the wide angle lens. This effect leads to that the position point from the camera does not correspond to the actual position on the air hockey table. OpenCV has a method for correcting the distortion based on the parameters of the lens. Figure 3.9 shows the distorted image to the left and the calculated correction image to the right. However, this method requires a lot processing power, thus taking a long time to perform. Testing this method showed that only 5 fps could be reached which is below what was required. Therefore another method had to be used.

For this project, a method that decreases the distortion of a single point was needed so that only the necessary calculations were made. This in order to transform only the relevant information from the curved (i.e. distorted) image to the flat (i.e. undistorted) game field, thus minimizing the required processing time. The method found to work best was based on the formulas provided by Imatest [11] that was proved to work well in the project from the previous year[12]. The following steps were taken to transform a curved position to a flat. The first step was to find the distance (marked R_{curv} in Figure 3.10) from the curved position to the center of the image, which is where the camera is located (marked Camera in figure 3.10), using equation 3.11.

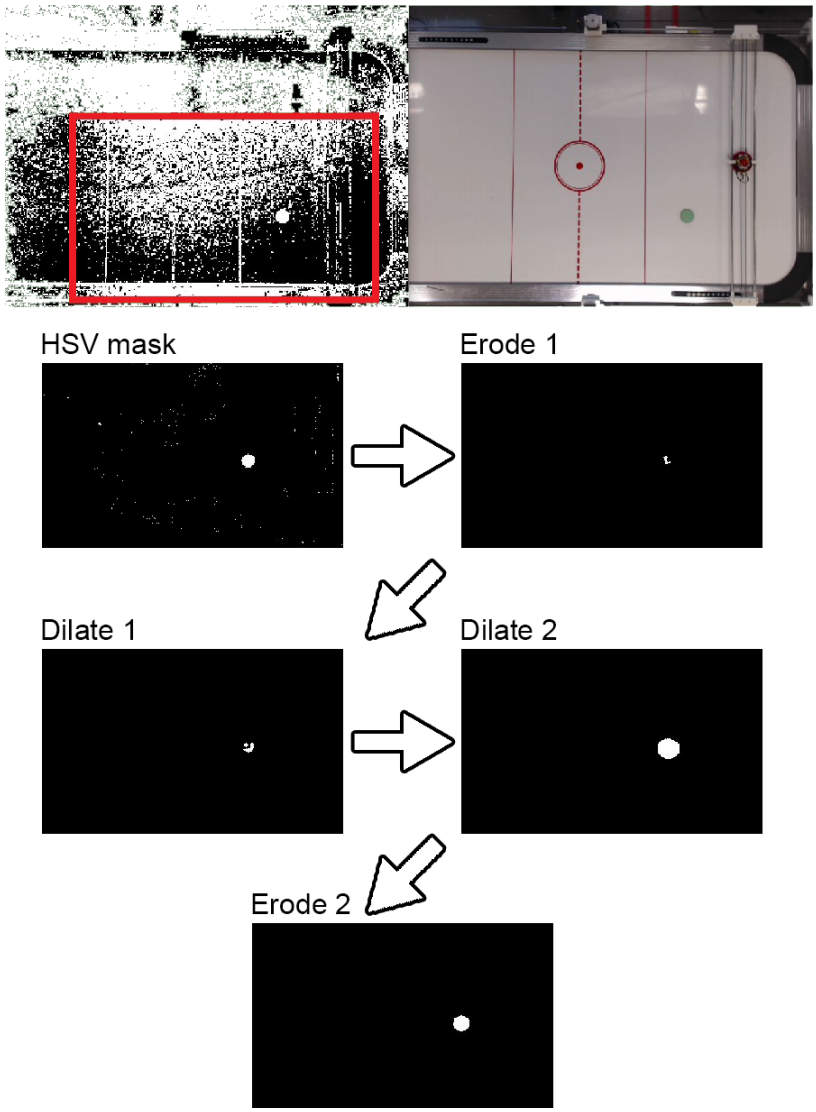


Figure 3.8: Illustrating of each step through a morphological filtering process. By eroding, dilating, dilating and eroding in that specific order; shortly known as opening and closing.

$$R_{curv} = \sqrt{x_{curv}^2 + y_{curv}^2} \tag{3.11}$$

x_{curv} and y_{curv} are the distances from the camera to the position of the puck in X and Y directions. R_{curv} was then transformed to R_{flat} using equation 3.12 where a, b, c, and d are parameters that were changed in order to counteract the distortion caused by the camera lens.

$$R_{flat} = a \cdot R_{curv}^4 + b \cdot R_{curv}^3 + c \cdot R_{curv}^2 + d \cdot R_{curv} \tag{3.12}$$

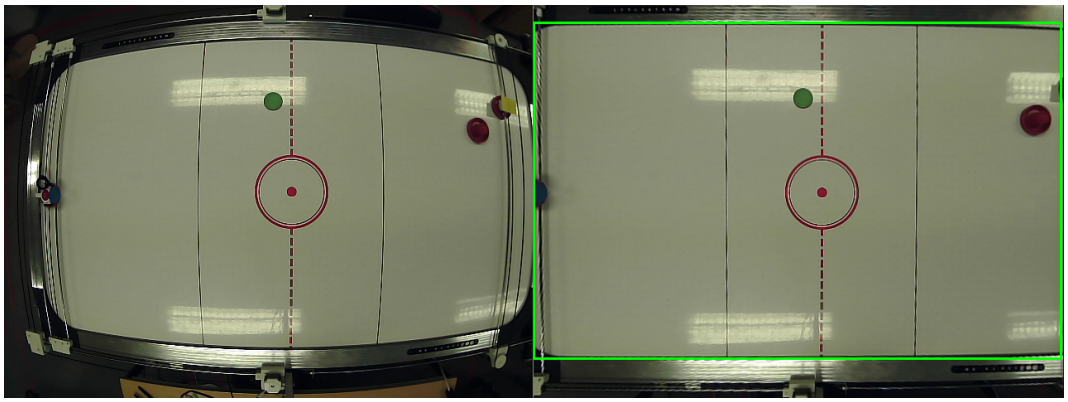


Figure 3.9: Original image (left) from the 70° angle of view lens compared to the undistorted image (right) generated by the built in OpenCV method.

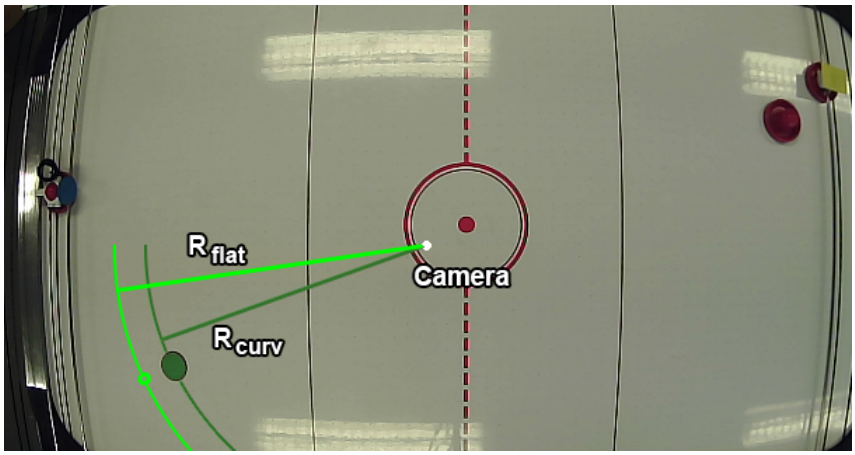


Figure 3.10: Figure shows the difference in position from the image taken by the camera compared to the flat position used in strategy algorithms. The camera is represented by a white dot, the light green represent the flat position and the dark green represent the actual puck.

The flat position (x_{flat}, y_{flat}) corresponding to x_{curv} and y_{curv} was then calculated using the ratio between R_{curv} and R_{flat} as given in equations 3.13 and 3.14,.

$$x_{flat} = x_{curv} \cdot \frac{R_{curv}}{R_{flat}} \tag{3.13}$$

$$y_{flat} = y_{curv} \cdot \frac{R_{curv}}{R_{flat}} \tag{3.14}$$

The flat position x_{flat} and y_{flat} was used in game strategy algorithms. To develop a strategy the vision system process had to be done for both the puck and mallet positions. Further information about the strategy can be found in Chapter 6 of this report.

4
Mechanical Design

The system needs some mechanical construction to move the mallet to the desired position. The chosen mechanical design was based on a system called CoreXY. CoreXY is a principle that has been proven to work well for various CNC (Computer Numerical Control) projects [13]. The group did not develop the CoreXY principle; it was merely implemented for the purpose of this project. In this section, CoreXY is further explained and the construction of the system is presented.

Most of the design was made in CATIA [14] CAD-program. The robot was made of various components, such as steel shafts, aluminum brackets and distance MDF-blocks (Medium Density Fiberboard), but most parts were made by 3D-printing in PLA plastic. This was a fast, easy and precise manufacturing method.

4.1 CoreXY

The mechanical design CoreXY, shown in Figure 4.1, consists of two motors and two belts which control the mallet. The belts were structured in a way that one motor with its corresponding belt moves the mallet diagonally. Therefore, in order to move the mallet along the table, both motors have to run simultaneously. The CoreXY system has the advantage that both motors are stationary, meaning that there is less mass to be moved, compared to last year's design[12] and it is possible to use larger, heavier motors without affecting the mass of the moving parts.

Since the CoreXY system uses a different coordinate system than the vision system, equations for converting coordinates between the two systems had to be developed, and the result can be seen in equation 4.1 and 4.2.

$$\Delta X = \frac{1}{2}(\Delta A_{belt} - \Delta B_{belt}) \tag{4.1}$$

$$\Delta Y = \frac{1}{2}(-\Delta A_{belt} - \Delta B_{belt}) \tag{4.2}$$

These equations describe the motion in X and Y direction in terms of the displacement of the belts moved with the two motors. The desired belt displacements for a certain amount of motion in X and Y directions are obtained with equations 4.3

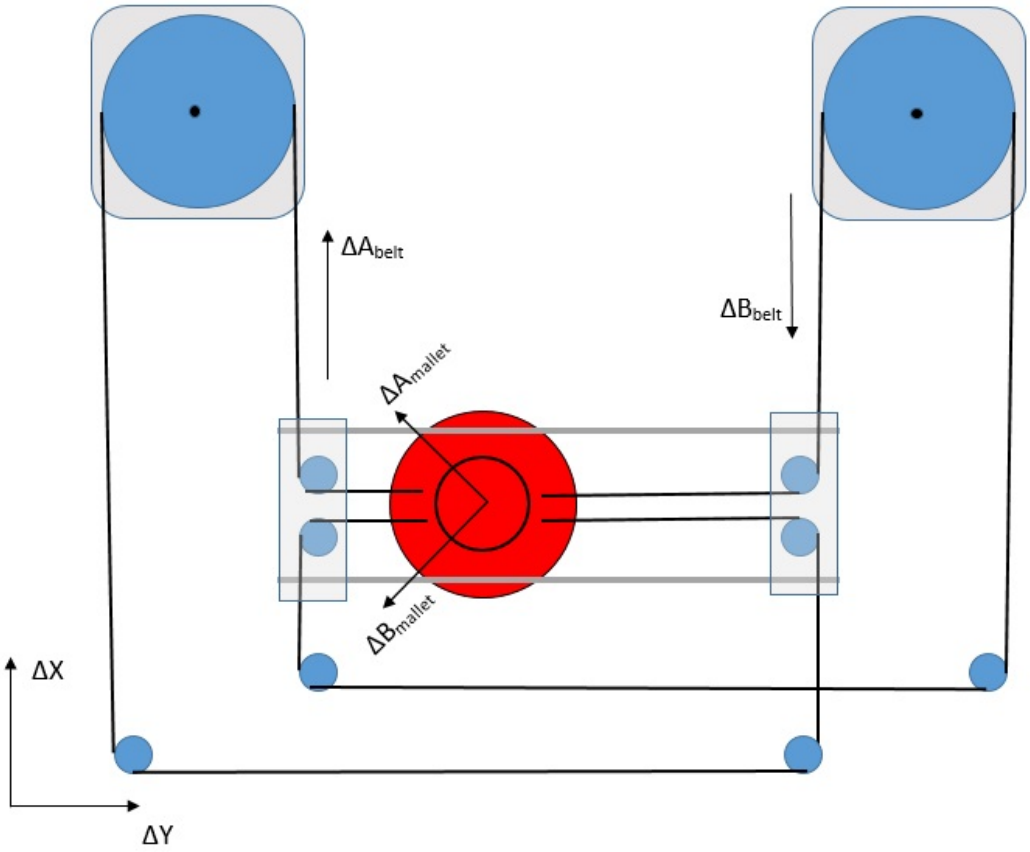


Figure 4.1: Illustration of the CoreXY belt system. The motors are in the top left and right corners.

and 4.4.

$$\Delta A_{mallet} = \Delta X - \Delta Y \tag{4.3}$$

$$\Delta B_{mallet} = -\Delta X - \Delta Y \tag{4.4}$$

Equations 4.3 and 4.4 show that the mallet will move in both X and Y directions by running one motor. Therefore, to make movements in only X or Y direction, both stepper motors need to operate at the same time in different combinations of rotational directions and speeds. ΔA_{belt} and ΔB_{belt} depend on the diameter of the belt pulleys and the number of steps per revolution of the stepper motors. The equations 4.5 and 4.6 relate the motion of the motors to the movement of the mallet where D is the diameter of the belt pulleys and n is the number of steps per revolution.

$$\Delta A_{belt} = \frac{D\pi}{n} \tag{4.5}$$

$$\Delta B_{belt} = \frac{D\pi}{n} \tag{4.6}$$

The prototype built in this project used belt pulleys with a diameter of 50 *mm* and stepper motors with 400 steps per revolution which resulted in the mallet moving approximately 0.4 *mm/step*. This made it possible to convert between mm and steps which was important for the program in the MCU since it only handles steps. A more detailed description of the program in the MCU is found in Chapter 5.

4.2 Choice of Design

The CoreXY system can be designed in a variety of different ways, but for this project, only two different design concepts were created: a one layer belt design and a two layer belt design.

4.2.1 One Layer Belt Design

The one layer belt design (OLBD) has the two belts at the same height, therefore the design is lower and slimmer than the two level belt design. One downside of the OLBD is that the belts need to cross each other at one point, which makes this design more of a challenge. The corner pieces (seen in Appendix C) will not be symmetrical and the belts and the linear shafts are on the same level. Another downside is that the OLBD leaves less space and freedom to change and modify the finished design and the mallet will not be able to move as far along the table as with the two layer belt design. This design was eventually chosen and constructed because of its slimmer and lower profile design and lighter moving parts.

4.2.2 Two Layer Belt Design

The two layer belt design (TLBD) differs from the OLBD mainly in the fact that the belts are stacked in two layers. The construction of the corner pieces would be easier, with the left and right mountings being almost identical, only mirrored. The belts would be mounted at different heights, which would eliminate the problem with crossing belts. However, a problem with this design might have been that the corner mountings would be higher and possibly more unstable. This design was considered inferior compared to the OLBD because of the risk that the moving parts would be heavier and that the system might have been less rigid than the OLBD.

4.3 Choice of Mechanical Components

4.3.1 Linear Shafts

The choice of material for the shafts was between steel, aluminum and carbon fibre. Steel was chosen due to low price, availability and robustness. Carbon fibre and aluminium are lighter but the surfaces might not have been strong enough for the linear bearings.

Dimensioning of the steel shafts was made with respect to the deflection. The maximum deflection of the shafts will occur when the mallet is in the middle of the robot's area of operation. The shafts are simply supported beams with the load positioned in the centre and a spread load from the weight of the shafts. The deflection was calculated by using equation 4.7 [15].

$$p = \frac{PL^3}{48EI} + \frac{5WL^4}{384EI} \tag{4.7}$$

The variables used in this equation are identified as follows:

- p is the deflection.
- P is the weight of the mallet holder (and the Y-shafts' weights when calculating X-shaft deflection).
- E is the elastic modulus of steel.
- L is the length of the shaft.
- W is the load per length unit (weight of shaft).
- I is the second moment of area for a circular cross-section.

The weight of the mallet holder and linear bearings were approximated to 0.1 kg and the shafts for the Y-axis has a diameter of 8 mm. The deflection for the shafts along the Y-axis was 2.15 mm. The shafts for the X-axis were at first chosen to be 8 mm in diameter but due to the weight of the Y-axis shafts as well as the length of the X-axis, the deflection was 9.5 mm. This was considered too high and thus, the X-axis shafts were changed to be 10 mm in diameter, which made the total maximum deflection 3.7 mm. Details about the calculation can be found in a MatLab script in Appendix B.

4.3.2 Linear Bearings

The bearings which were bought from the same distributor [16] as the belts had a low price and was advertised as high quality bearings. The model type of the bearings is LM10UU and they were manufactured by Fushi. The bearings have an inner diameter of 10 mm, an outer diameter of 19 mm and a length of 29 mm. The spacing between the lock track for the retaining rings are 23 mm. The sledge houses (see Appendix C for explanation) had to be designed so that the bearings would be

able to fit inside and lock in place with the retaining rings. The linear bearings for the 8 *mm* X-axis shafts were reused from last year's project [12]. The linear bearings were tested and compared to a set of 3D-printed plastic bearings. The ball bearings were chosen since their friction was significantly lower compared to the plastic ones. Specific data and dimensions of these linear bearings are found in the data sheet [17].

4.3.3 Belts

There were two different types of belts commonly available: GT2 and T2.5 (see Figure 4.2 for a comparison of the profiles). These belts differ in the tooth spacing and profile. The GT2 belt has a rounder profile to minimize skidding and was thus preferred but due to it being out of stock at the reseller at the time, the T2.5 belt was chosen [18]. The distributor [16] was only able to deliver a width of 6 or 9 *mm* of T2.5 belt. The belt chosen was the 6 *mm* belt to keep the design slimmer. The strength of the belt was specified at 36 N/*mm*, for a 6 *mm* belt that results in 216 N. The 6 *mm* belt was able to carry the weight the entire construction and was therefore considered sufficient.

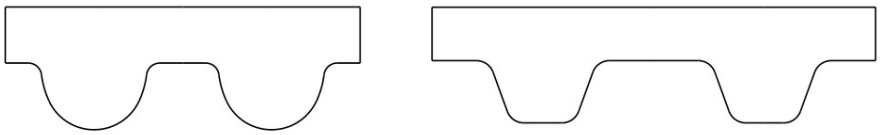


Figure 4.2: Difference between the GT2 (left) and T2.5 (right) belt profiles.

4.3.4 Dimensioning of the Belt Pulleys

Designing and dimensioning of the belt pulleys were made with inspiration from last year's project. Since the motors and load forces were mostly the same on this system, last year's construction was used as a rule of thumb. The equations presented later in this chapter were used to verify that the chosen dimensions were correct.
When dimensioning the belt pulleys, things to consider were the rated torque that the stepper motors could deliver, the stepper motor's maximum angular velocity and the maximum final speed of the mallet. The belt pulleys had to be large enough diameter so that the speed of the mallet would reach the requirements yet not too large so that the torque required to accelerate the mallet would exceed the torque that the motors were capable of delivering.

There are three different load cases for the stepper motors. One case where only one motor is active and the mallet is moving diagonally, and two cases when both of the motors are active and the mallet is moving forward (X-direction) or across

the table (Y-direction). When the robot is moving in the Y-direction, the two steel shafts across the table do not move and will not need to be accelerated, unlike the case when moving in X-direction. The load case when moving in the X-direction was the most demanding because of the need to accelerate the steel shafts across the table. Due to this, the X-direction movement with both stepper motors active was the movement calculated for when dimensioning the belt pulleys.

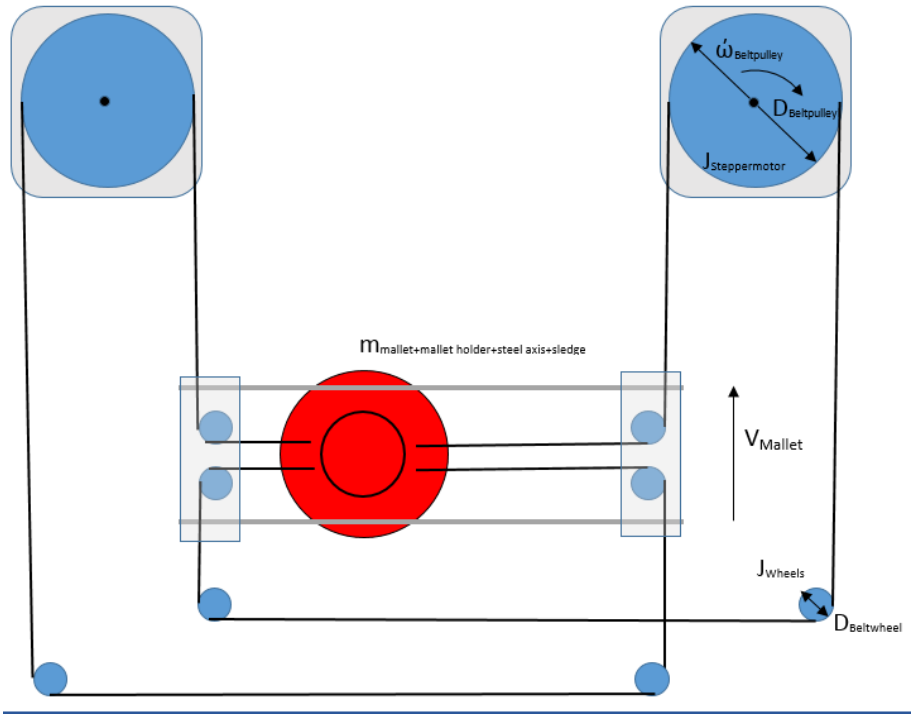


Figure 4.3: The masses and inertia of the system that affects the equation of motion and calculations for the belt pulley dimensioning.

The following notation are used in equation 4.8 based on which the calculation was performed. Also see Figure 4.3 for a visual representation.

- $J_{steppermotor}$ is the inertia of the stepper motor, from the stepper motor data sheet [19].
- $J_{beltpulley}$ is the inertia of the belt pulleys.
- J_{wheels} is the total inertia of the small belt wheels.
- m_{load} is the total mass of the mallet, mallet holder, bearings, steel shafts for the y axis and belt wheels.
- $T_{steppermotor}$ is the stepper motors' rated torque, and this is multiplied with two in the equation, since both motors are active.
- n_{wheels} is the number of rotating belt wheels (4 in this case).
- $D_{beltpulley}$ is the diameter of the belt pulley.
- D_{wheel} is the diameter of the belt wheels. See Figure 4.3.

The relation between the torque and acceleration is expressed in equation 4.8. No friction or losses were considered in the calculations since they were very uncertain

and hard to calculate. The friction is still a prominent factor of loss in the system.

$$\begin{aligned} T_{steppermotor} \cdot 2 =& (J_{steppermotor} + J_{beltpulley}) \cdot \dot{\omega} + m_{load} \cdot \dot{\omega} \cdot \frac{D_{beltpulley}}{2} \\ &+ n_{wheels} \cdot J_{wheels} \cdot \dot{\omega} \cdot \frac{D_{beltpulley}}{D_{wheels}} \end{aligned} \tag{4.8}$$

$\dot{\omega}$ is the angular acceleration of the stepper motor axis, which was determined from the desired acceleration time and acceleration distance and requested maximum velocity of the mallet in equation 4.9.

$$V_{max,mallet} = V_{0,mallet} + a_{mallet} \cdot t = \{V_{0,mallet} = 0\} + a_{mallet} \cdot t = \cdot\dot{\omega}_{beltpulley} \cdot \frac{D_{beltpulley}}{2} \cdot t \tag{4.9}$$

The belt pulley is approximated as a thin disc $J_{beltpulleys}$ and calculated in equation 4.10.

$$J_{beltpulleys} = \frac{1}{8} \cdot n_{wheels} \cdot m_{beltpulley} \cdot D_{beltpulleys} \tag{4.10}$$

The stepper motors were the same as the ones used in last year's project. The rated torque was found in the data sheet for the stepper motors [19]. The calculation validated the rule of thumb from the last year's project and the decided diameter of the belt pulley was 50 *mm*. The height of the belt pulleys was designed to fit the 6 *mm* T2.5 belt. The CAD-drawing of the belt pulleys was made in Makerbot Customizer [20] and then 3D-printed in PLA plastic.

4.3.5 Camera Stand

The camera needed to be mounted at a height of 143 *cm* over the playing surface according to the calculations for the camera lens in equation 3.1. The exact distance was uncertain before the camera was mounted as it was a decision made together with the other air hockey project group, since the stand was to be used by both groups. The cameras of both groups needed to be at the same height to avoid the other team's camera to interfere and hide parts of the game field. Since the groups used different cameras with different lenses, the groups had different preferred heights for the cameras. Therefore was an agreement made on a height that worked for both groups. The camera mounting was designed and built in square steel tubing, and then adjusted to fit both groups' demands as well as possible, which turned out

to be at 122 *cm* over the game field surface. It was of important that the mounting would be stable and not connected to the table to avoid vibrations from the robots which would increase the risk of bad image capturing.

5
Electrical Design

This chapter provides more information about the choice of electrical components and the programming of the MCU in order to control the mechanical design. When deciding which electrical components to use, there were several different aspects to consider. It was very important that the motors were strong enough to control the mechanical construction and that the MCU was capable of controlling the stepper drivers. The stepper drivers needed to be able to provide a sufficient amount of current to the motors so that there was less chance that the motors would skip steps. In Figure 5.1 the electronic components are shown as mounted on the side of the table.

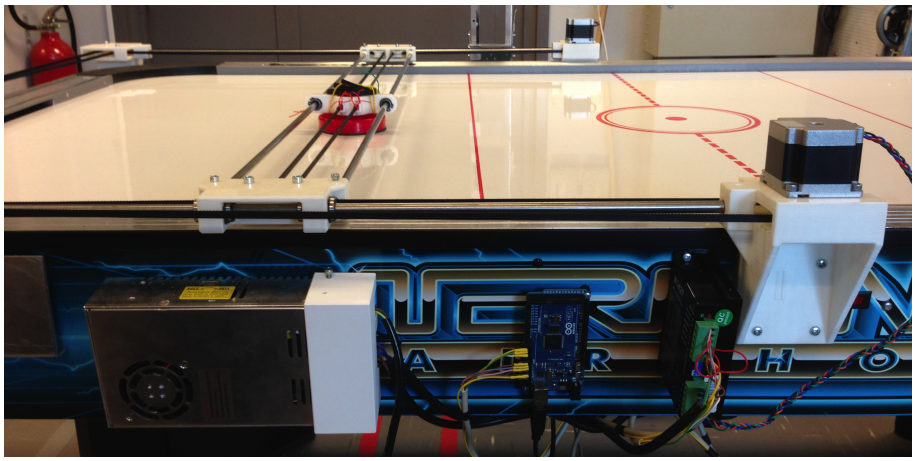


Figure 5.1: The electronic components are mounted on the side of the air hockey table. Furthest to the left is the power supply. There are two drivers, one on each side of the table. In the middle is the Arduino board (MCU).

5.1 Choice of Components

The first choice concerning the electrical components was deciding about which motors to use. Since there was a group with the same project the year before, the group chose to reuse the same stepper motors to save money. The main reason for using stepper motors was that they are very exact in their positioning, but they also have a disadvantage since they are not supposed to run fast. Normally, stepper motors are used to control CNC machines with high precision at low speeds and high torque,

but in this application there was a need for higher speeds. This has been taken into consideration when designing the mechanical system, especially concerning the diameter of the belt pulleys.

The motors required a maximum current of 2.8A which led us to choose drivers capable of delivering this current. The drivers that were chosen and bought were micro-stepping drivers called M542T [21]. They are capable of delivering a current between 1.5-4.5A which covered the requirements of the motors. When driving the motors in full step mode, the motors started to resonate at a certain speed, which would have become a problem. Instead the drivers were configured to run in half step mode, so that the resolution is 400 steps per revolution instead of 200. This removed the problem with resonance but the disadvantage was that the torque was reduced by approximately 15% [22] compared to full step mode. Controlling the stepper drivers was simple: 3 pins were used for each motor, one for stepping and one to choose which direction it should move. The third one was optional but gave the possibility to turn off the current to the motors so that they did not draw unnecessary power and overheat.

To supply the motor drivers with current, a Power Supply Unit (PSU) was used which could supply enough current at a suitable voltage. The PSU that was chosen [23] is capable of delivering 7.3A at a voltage of 48V. The reason this voltage was chosen is that the higher the voltage is, the faster the motor will be able to run [24].

The MCU which was used was an Arduino Mega [25], which was handed over from previous year's project [12]. The Arduino MCUs are easy to program and the Mega version had enough processing power and memory for the application in this project. Since there was an external computer running software for object tracking and strategy algorithms, which is described in Chapters 3 and 6, information needs to be sent from the computer to the MCU. This was done through serial communication.

5.2 Programming of the MCU

The main assignment for the MCU was to control the stepper motors based on the target position given from the external computer. Hence the MCU only set the direction and speed of the motors. The direction was set by either giving high or low on the driver direction pin and the speed of the motors was set by sending pulses with desired frequency to the step pin. These pulses were created in the program by the usage of the built-in hardware timer functions on the MCU. These timers provided the basic function of being able to pulse different output ports without interrupting the rest of the program. This solved many problems and removed the need to manually pulse the outputs, which was hard to complete without losing performance.

The target position given from the external computer was based on the coordinate system used by the vision system which had the X-coordinate in direction of the long side of the table and the Y-coordinate perpendicular to X. The MCU used the

CoreXY coordinate system which meant that the coordinates had to be converted and that was done with equations 5.1 and 5.2. The origin of the CoreXY coordinate system was set to be in $X = 0$ and $Y = \frac{width}{2}$. The coordinates were given in millimeters which were converted into steps with equation 4.5 and 4.6 described in Chapter 4. This is represented with the constant Ψ in the equations (5.1) and 5.2 shown below, which is in $steps/mm$ in order for the result to be in steps.

$$m1_{target} = (X - Y + \frac{width}{2}) \cdot \Psi \tag{5.1}$$

$$m2_{target} = (-X - Y + \frac{width}{2}) \cdot \Psi \tag{5.2}$$

When the target positions had been calculated, it was sent to *move_steps*, which basically was the main function of the program and other functions like acceleration was called from there. The *move_steps* function first calculated what speed the two motors should run at. The maximum speed was given as a parameter to the function and the motor which had to travel furthest was given that speed. The speed of the other motor was calculated from the quotient of both distances, and this was to make sure that motors stopped at the same time. The program also kept track of the current speed and direction of the motors, which the function used to determine how the acceleration or deceleration should be performed. If a motor was running in the opposite direction, it would first have to decelerate before the change of direction.

In order for the stepper motors to perform well, they had to start and stop smoothly. To achieve this, a linear change of speed was programmed. This was done with acceleration and deceleration functions which took the difference between the current speed and the target speed and divided it into a certain amount of steps. Since the timers have time period as input and frequency had to be used for the calculations, the frequency was calculated by inverting the time period. If the period just decreases linearly, the acceleration would not be constant. An array with evenly spaced increasing or decreasing frequencies was calculated and then the frequency values are traversed through and inverted then sent to the timers, starting from current frequency and making its way up or down to the target frequency, which translates to the target speed.

When the target speed had been reached after the acceleration, the *move_steps* function would have entered a while-loop waiting for new instructions, since the timer functions ran in the background constantly running the motors. When the required steps for the deceleration function to start was reached, the program exited the while-loop and started to decelerate so the mallet would end up at the correct location. A flowchart of the program is presented in Figure 5.2 and the basic procedure described so far is basically shown in the top loop. There is also a condition in the while-loop waiting for new information from the external computer.

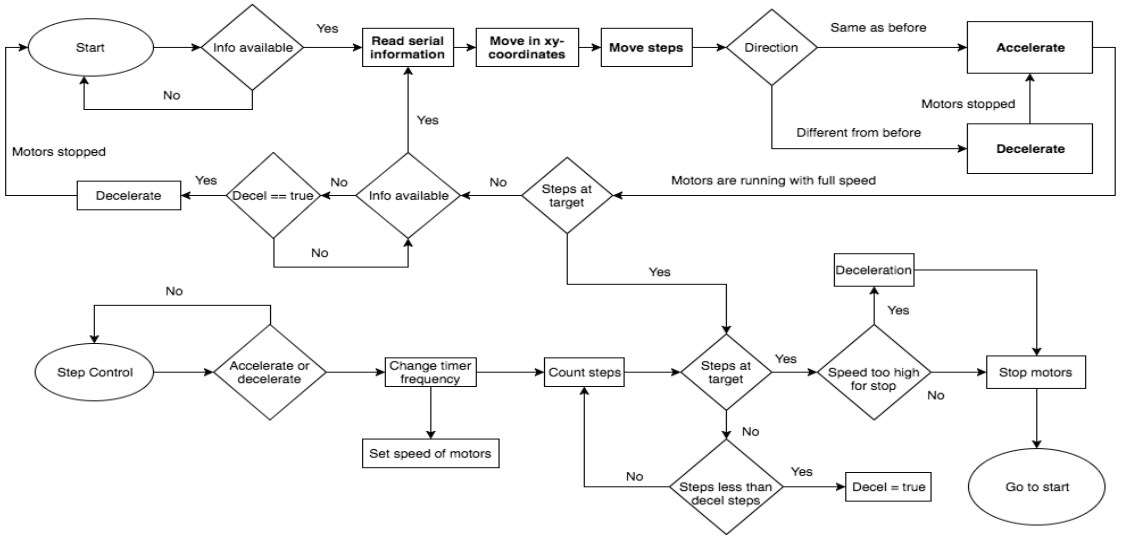


Figure 5.2: Flowchart over the stepperprogram in the Arduino. The program reads the serial information and converts it into steps. Then runs the stepper motors to the target position with the step control as reference.

The program always had to know how many steps it had sent to the motors, in order to start decelerating at the right time. The bottom half of the flowchart in Figure 5.2 shows how step control, which basically is position control, was managed by counting steps. A function in the timers called attachInterrupt made it possible to run a short function each time a pulse was given to the stepper drivers. This function was used for two tasks: to count steps and trigger a deceleration when the motors had run far enough.

If new information from the external computer made the counted steps equal to the target position without a deceleration having been made, the program would check the current speed and stop the motors if the speed was low enough for a smooth stop. Otherwise it would start to decelerate at that position, making the mallet overshoot a little. This was not an issue since the program still counts steps and goes to the right position when new information is acquired next time.

However, since it was possible for the stepper motors to skip steps, the system could not rely only on the position calculated from the timers. The plan was to solve this by using feedback control from the external computer which would tell the MCU where the mallet was located. If the difference between the calculated position and the position given from the camera was too large, the step count would calibrate using the position given by the camera. However, it was not implemented and therefore a calibration done manually had to be made regularly by placing the mallet at the centre of the coordinate system.

6

Game Strategy and Algorithms

This chapter is about how the robot acts based on the information about the position of the puck and the mallet. The first part is predicting the trajectory of the puck based on the position information and later deciding what to do.

6.1 Predicting the Trajectory of the Puck

The algorithm which calculates the trajectory of the puck does this without taking the speed into consideration. A function takes an X coordinate as input and calculates the Y position of the puck at the given X coordinate. The function does this by using the equation 6.1.

$$Y_C = \frac{\Delta X_m}{\Delta X} \Delta Y + Y \tag{6.1}$$

The variables used in equation 6.1 are illustrated in Figure 6.1.

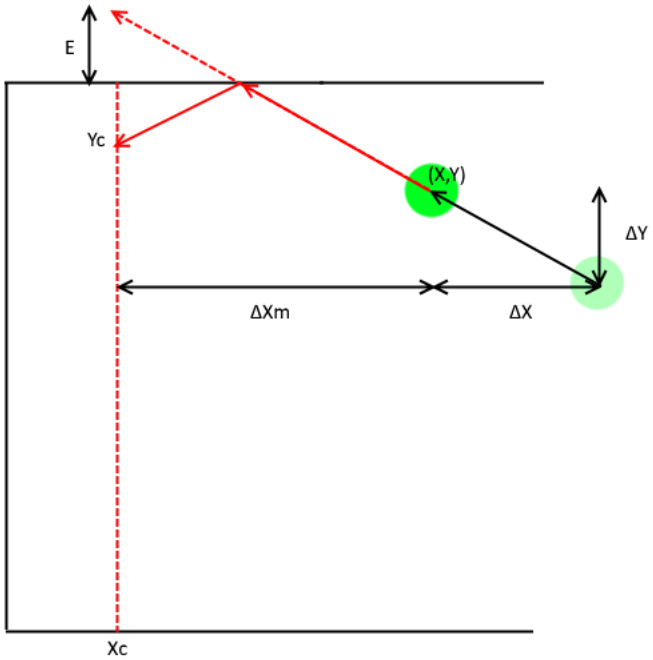


Figure 6.1: Illustration of the algorithm which calculates the trajectory

If the calculated Y position lies outside the field of the game as seen above, the distance E from the edge is multiplied by -1. If the Y position would still be outside the game field, this is repeated until the calculated position is within the edges. The algorithm results in a Y-position regardless of where along table it needs to be known.

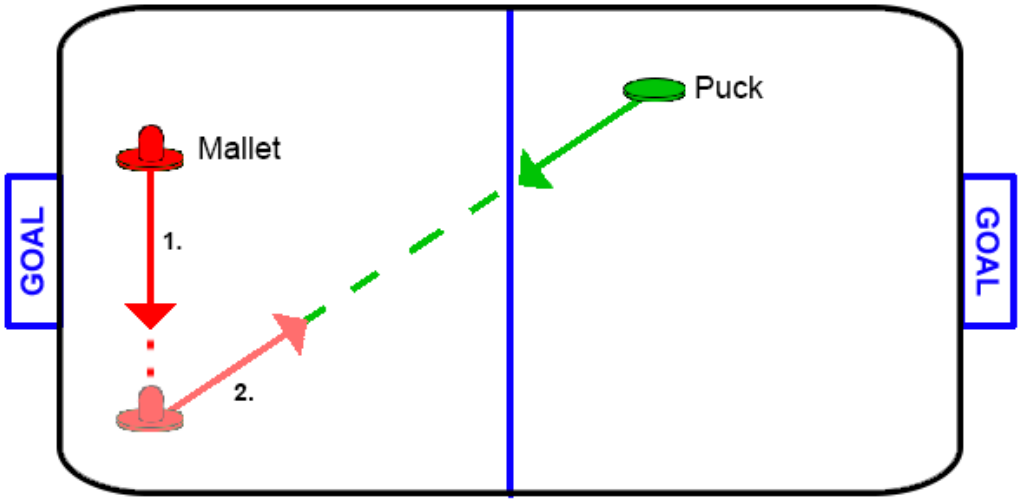


Figure 6.2: Illustration of attacking algorithm. The first move is to place the mallet at an intersection between the trajectory and the default defense axis (1). The second and last move is to the mallet along the trajectory of the puck (2).

6.2 Strategy

There are four basic modes used to decide what the robot should do. The decision about which mode to use is based on the speed, position and direction of the puck. The speed basically translates to the difference in position between the last and the current frame, assuming that the frame rate is constant.

- If the puck is moving towards us faster than a certain threshold speed set by the user, the robot enters a defensive mode and only moves the mallet within the edges of the goal with the only intention to defend. This motion in the Y-axis is done at a certain distance in the X-axis from the goal.

- If the puck is moving slower than the threshold speed towards us, the robot enters an attack mode. In this mode, the vision system calculates the intersection between the puck's trajectory and the default X-axis of the defense mode. The mallet is first moved to the resulting intersection and when the mallet is

in place, it moves along the trajectory of the puck to make an attack. See Figure 6.2. In this way, the timing of the attack is less important since the robot will hit the puck regardless of when it starts the attack if the trajectory is correct.

- If the puck is laying still on our side of the game field the robot just hits the puck to prevent the need of human interaction.

- If the the puck does not move towards our side, the robot returns to the default position, the middle in front of the goal.

When using the prediction algorithm, there is always a small error in the calculation because the vision system's algorithm returns data which is fluctuating. To decrease this error, a mean value is calculated. The mean value calculation puts more weight on the latest value which makes the algorithm work better since the prediction gets better when the puck is closer to the final position.

7
Validation

Testing and evaluation have been done to verify whether the requirements and specifications in Appendix A have been reached. The validation of the system requirements will all be tested separately from each other and listed in separate sections.

R1.1 Block puck at low speed

When the defensive capabilities of the robot was tested, it was concluded that it was capable of protecting the goal from pucks moving at 3 m/s towards the robot. The requirement was **fulfilled**.

R1.2 & R1.3 Pinpoint a still puck and mallet

The puck was placed on our half of the game field and the correct X and Y coordinates were measured with a measuring rod. This point was compared with the value given by the detection system, see Figure 7.1. The mallet was tested in the same way as the puck.

The result for the puck gave a maximum deviation from the real position by 29 mm which **does not fulfill** the requirement of ± 5 mm.
The result for the mallet gave a maximum deviation from the real position by 31 mm which **does not fulfill** the requirement of ± 5 mm.

R1.4 Actuator motion accuracy

When the mallet was programmed to move 400 mm at maximum speed, the traveled distance was 399 mm which was in the error range of ± 10 mm. This meant that this requirement was **fulfilled**.

R1.5 Construction and table space

A simple visual conformation gave the result for this requirement. See Figure 7.2. The construction requirement was **fulfilled**.

R1.6 Striking the puck

A special program for the Arduino was created to be able to hit the puck repeatedly at an optimal angle to give the puck maximum speed. The same acceleration

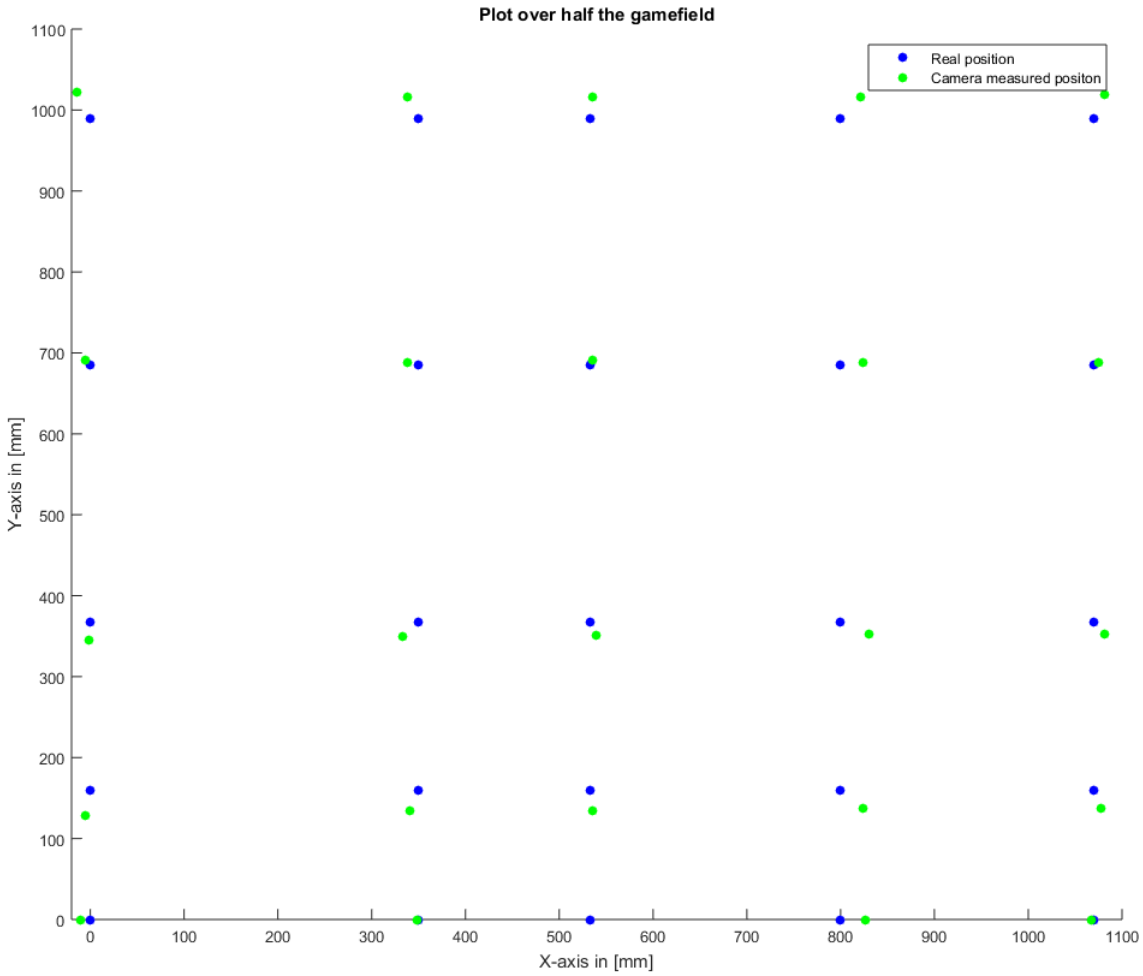


Figure 7.1: Plot over our half of the game field with points showing where the puck actually are and where the camera system think it is.

and speed of the stepper motors as in the game mode were used and the puck was completely still. The speed of the puck was measured by recording a video with a camera that uses a known frame rate. By counting how many frames it takes for the puck to travel a certain distance, the speed is acquired.

The mean value from the results was : 1.44 m/s which **does not fulfill** the requirement of 8 m/s.

R1.7 Materials used in the process

The prototype is constructed with materials that leads to a sustainable design. The requirement on the materials was **fulfilled**.

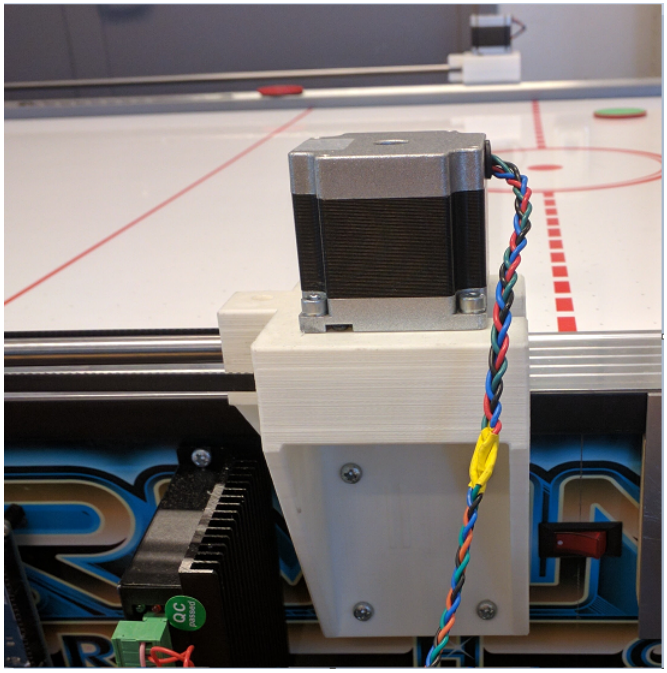


Figure 7.2: Picture of the motormounts and the puck on the opponents side of the table.

R1.8 Dismantling the prototype

When dismantling the prototype, only handheld tools is required. 24 Pozidriv screws hold the prototype in place.
The construction requirement was **fulfilled**.

D1.1 Calculation power

The calculations made on the computer does not become a bottle neck in the system which makes the desire **fulfilled**.

D1.2 Block puck at high speed

When the defensive capabilities of the robot was tested, it was concluded that it wasn't able to protect the goal from pucks moving at 6 m/s towards the robot.
The desire **was not fulfilled**.

D1.3 Actuator motion accuracy

When the mallet was programmed to move 400 mm at maximum speed, the traveled distance was 399 mm which was in the error range of ± 5 mm. This meant that this desire was **fulfilled**.

8

Discussion

This chapter discusses the results of the project. Discussion will focus on whether the prototype fulfilled the requirements or not and the reason behind the result. It will go into further detail about the performance of each subsystem and discuss possible improvements of the systems. Lastly a short section about what environmental aspects that have been taken into consideration will be described.

8.1 Prototype Performance

The construction of all the subsystems has been finished in time and the subsystems are working together as intended. However there are some minor issues within each subsystem. All the functionality of the robot is working but the overall performance does not reach all the requirements and desires set up in the system requirements, as discussed in Chapter 7.

Requirement R1.6, where the system should be able to strike a puck towards the opponent with a speed of 8 *m/s*, is not met. A strike from the system on a puck laying still sends the puck moving towards the opponent at about 1.4 *m/s* which is below the desired speed. Calculating the speed of the mallet based on the data sent to the motor drivers results in a speed of about 1.57 *m/s*. The problem with the low speed of the puck could be addressed by moving the mallet faster. To move the mallet faster there are a couple of solutions. One is to increase the radius of the belt pulleys so that one revolution of the motors correspond to a larger movement of the mallet. Another solution is to decrease the time between each step command being sent to the motor drivers, but this would cause the stepper motors to miss some of the commands, thus leading to imprecise positioning. In hindsight, the requirement R1.6 was set too high as it was based on the speed of the previous year's system [12], which was able to defend against an incoming puck moving at 8 *m/s*. Striking is a more complex maneuver compared to defending and outperforming the last year's defending maneuver with just raw speed is not possible with the prototype built in this project.

Pinpointing a puck laying still and a mallet as described in requirement R1.2 and R1.3 with an accepted margin for error at ± 5 *mm* is not achieved with the prototype in this project. The maximum error when testing the puck location was 29 *mm* as described in the system requirements chapter. The requirement is not met but the maximum errors found are located far out to the sides of the game field (shown in

Figure 7.1) where the robot would rarely (or never) have to hit the puck and the problem with low accuracy in pinpointing is not a problem in practice. The low accuracy can be fixed by further tuning the parameters in the method for removing the distortion for point described in the vision system chapter. Since this error in pinpointing does not affect the performance of the system there should be little to no advantage gained by further fine tuning the parameters.

There is however a problem with the vision system that affects the performance of the prototype. That is the error in hardware settings for the camera used. The camera specification states that is should be able to generate 120 fps and it is able to reach 100 fps using a special webcam software. When accessing the camera trough OpenCV, there is no option to set the frame rate and the camera uses its default frame rate of 30 fps. Capturing frames at 30 fps is a considerable decrease in frame rate compared to the stated frame rate of 120 fps. This causes the system to identify puck movement slower and it generates fewer points measured. Identifying the puck slower is a problem as it decreases the time left for the prototype to take an appropriate counter action to whatever is happening on the game field. A couple of actions to fix the problem with the low frame rate has been taken. The manufacturer was contacted and a development kit was provided but it was written in Mandarin Chinese. Due to lack of knowledge in Mandarin within the group as well as the manufacturer's lack of knowledge in English, communication was hard and no solution was found. Some other ways of accessing the camera settings were also attempted including Matlab and Python scripts, but these methods did not grant us access to the required settings and no solution to the frame rate problem could be found.

The frame rate problem is the most significant problem but there are some other performance issues with the vision system. The processing time for the vision system is a factor that is affecting the overall system. This project uses the OpenCV library which is free and open source with a large community which is very helpful but some of the methods in the library are not properly optimized. This causes the processing time to increase, which adds to the already slow data gathering caused by the low frame rate.

Directly depending on the data gathering is the game strategy algorithms. The algorithms described in the chapter called Game Strategy and Algorithms which makes the prediction about where the puck will be located in the future is flickering. The methods described to stabilize the prediction works well enough for the system to play a game of air hockey. Adding a Kalman filter to the prediction would lead to a more stable prediction, but the calculations would take additional computing time, which caused this option to be left unexplored. It was decided not to take the possible loss of momentum from the bounces in consideration, as it was observed to be negligible.

8.2 Possible Improvements and Future Project

If the problem with the camera is not resolved, there are a few different options of what to do. One possibility is to use it at 30 fps with high resolution or another camera could be purchased. If another camera is purchased, it could be a PS3 Eye camera, which can be used at 60 fps at a resolution of 640x480 pixels [26].

The Pixy CMUcam5 [27] is a promising alternative since it has integrated image analysis algorithms. With the use of such a camera, which means that an external computer would not be needed. However, this integrated analysis can also become a problem as it is not flexible. Another drawback of the Pixy CMUcam5 is that it only takes pictures at 50 fps, which might cause some problems since some interaction can happen between the two pictures, especially when the puck is moving at a high speed.

Another possibility might be to use a mobile phone with a high frame rate camera as web cam.

The corner mountings were designed with some extra space for larger belt pulleys if it would show that the stepper motors had more torque than anticipated. If this would be the case, then larger belt pulleys could be manufactured for increased speed and acceleration.

When the problems discussed above have been solved, a lot of time could be spent on developing well-performing game strategy algorithms.

8.3 Environmental Aspects

The following considerations have been made concerning the environmental aspects:

- The parts are printed in a material called Polylactide (PLA) which is biodegradable and made from corn starch.
- The robot has electrical motors which have high efficiency and no emissions, and since the electrical production in Sweden has low emissions, the total emission will be low.
- All parts in the construction are easily removable.

9
Conclusion

The purpose of this project was to develop a robot which is capable of replacing a human playing air hockey. This goal is considered to be achieved to some degree in our project, although not at the same level as a human player. The finished robot consists of two stepper motors moving the mallet using a CoreXY system, a camera made by ELP, a computer processing the camera data and an MCU which receives data from the computer and controls the stepper drivers which drive the motors.

Since the prototype fulfills most of the specified requirements and desires, the project is considered as successfully completed. The prototype is capable of playing against either a human or a robot even though human interaction will be needed during the game. Even though it works as intended for the most part, there is definitely room for improvement. The biggest area of improvement is within the software that was developed. The software on the computer could be optimised for faster filtration, better game strategies and the communication between the computer and the MCU could definitely be improved.

Bibliography

[1] K. Mülling, J. Kober, and J. Peters, *A Biomimetic Approach to Robot Table Tennis*. Technische Universität Darmstadt Germany, Max Planck Institute Tübingen Germany, 2011.

[2] L. Jodensvi, V. Johansson, C. Lanfelt, and S. Löfström, *ONE ROBOT TO ROLL THEM ALL. Construction of a self-balancing, two-wheeled, football playing robot*. Chalmers University of Technology, Gothenburg, 2015.

[3] T. Senoo, A. Namiki, and M. Ishikawa, *High-Speed Batting Using a Multi-Jointed Manipulator*. New Orleans, USA, 4.28: 2004 IEEE International Conference on Robotics and Automation, 2004.

[4] Video of air hockey world championship finals 2010. [Online]. Available: https://www.youtube.com/watch?v=iArc6FTLISA [Accessed: 17.05.2016].

[5] R. Fitzpatrick. Professor of physics at the university of texas describes the law of reflection in a lecture note. [Online]. Available: http://farside.ph.utexas.edu/teaching/302l/lectures/node127.html [Accessed: 17.05.2016].

[6] D. Eugster, *High-Speed Motion Tracking for Robot Control*. Eidgenössische Technische Hochschule Zürich, 2012.

[7] D. Rubino. Documentation for open source library opencv. [Online]. Available: http://docs.opencv.org/master [Accessed: 17.05.2016].

[8] B. Morse. Lecture notes about binary morphology. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/morph1.pdf [Accessed: 17.05.2016].

[9] N. Efford, *Digital Image Processing: A Practical Introduction Using Java*, P. Education, Ed., 2000.

[10] J. Flusser and T. Suk. Rotation moment invariants for recognition of symmetric objects. [Online]. Available: http://library.utia.cas.cz/separaty/historie/flusser-rotation%20moment%20invariants%20for%20recognition%20of%20symmetric%20objects.pdf [Accessed: 17.05.2016].

[11] Imatest llc, distortion. [Online]. Available: http://www.imatest.com/docs/distortion/ [Accessed: 17.05.2016].

[12] E. Bergendal, M. Ganelius, J. Gustafsson, N. Hellberg, J. Hasselgren, and J. Karlsson, "Självspelande air-hocketspel," [Accessed: 17.05.2016].

[13] I. Moyer. Corexy, principle of operation. [Online]. Available: http://corexy.com/theory.html [Accessed: 17.05.2016].

[14] Dassault systèmes, catia. [Online]. Available: http://www.3ds.com/se/produkter-och-tjaenster/catia/ [Accessed: 17.05.2016].

[15] B. Sundström, *Handbok och formelsamling i hållfasthetslära.* Institutionen för hållfasthetslära KTH, 1998.
[16] 3D-Grottan. Parts distrubitor. [Online]. Available: http://3d-grottan.se/ [Accessed: 17.05.2016].
[17] F. Bearings. Lm10uu datasheet. [Online]. Available: http://www.euro-bearings.com/eurocatonline.pdf [Accessed: 17.05.2016].
[18] D. Grottan. Corexy, principle of operation. [Online]. Available: http://3d-grottan.se/T2.5-Timing-Belt?search=T2.5 [Accessed: 17.05.2016].
[19] R. Components. 57mm 1.8' high torque stepper. [Online]. Available: http://uk.rs-online.com/web/p/stepper-motors/5350423/ [Accessed: 17.05.2016].
[20] droftarts. Parametric pulley - lots of tooth profiles. [Online]. Available: http://www.thingiverse.com/thing:16627 [Accessed: 17.05.2016].
[21] StepperOnline. High performance microstepping driver. [Online]. Available: http://eu.stepperonline.com/download/pdf/M542T.pdf [Accessed: 17.05.2016].
[22] Minebea. Microstepping, full step & half step. [Online]. Available: http://www.nmbtc.com/step-motors/engineering/full-half-and-microstepping/ [Accessed: 17.05.2016].
[23] StepperOnline. 350w power supply s-350-48. [Online]. Available: http://eu.stepperonline.com/download/pdf/S-350-48.pdf [Accessed: 17.05.2016].
[24] Ericsson. Drive circuit basics. [Online]. Available: http://users.ece.utexas.edu/~valvano/Datasheets/StepperDriveBasic.pdf [Accessed: 17.05.2016].
[25] Arduino. Arduino mega. [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardMega [Accessed: 17.05.2016].
[26] Sony computer entertainment europe, playstation eye. [Online]. Available: http://uk.playstation.com/media/247868/7010571%20PS3%20Eye%20Web_GB.pdf [Accessed: 17.05.2016].
[27] CMUcam, "Cmucam5 pixy." [Online]. Available: http://cmucam.org/projects/cmucam5 [Accessed: 17.05.2016].

A

System Requirements

System requirements

Criteria	Requirements and desires	Target	Weight 1-5	Validation method
Requirements				
Block puck, low speed	R.1.1	Puck moving at 3 m/s, movin towards 300 mm horizontally form the mallets starting position	5	Physical testing
Locating a still puck	R.1.2	Pinpoint a still puck to ± 5mm	4	Physical testing
Locating a still mallet	R.1.3	Pinpoint a still mallet to ± 5mm	4	Physical testing
Actuator motion accuracy	R.1.4	When moving the mallet 400 mm without feedback control can not exeed a fault of ± 10mm	5	Physical testing
Construction and table space	R.1.5	The prototype may not protrude over the middle line towards the opponents half.	5	Physical testing
Striking the puck	R.1.6	Strike the puck, sending it towards the opponent at 8 m/s	4	Physical testing
Environmentally friendly	R.1.7	The prototype should be constructed with materials that make a sustainable design.	4	Analysis of chosen materials and methods.
Smart construction	R.1.8	Only hand tools shall be required when dismantling the prototype.	3	Physical testing
Desires				
Calculation power	D.1.1	Calculations made to track the puck does not become a bottle neck in the system	3	Time complexity analysis/physical testing
Block puck, high speed	D.1.2	Puck moving at 6 m/s, movin towards 400 mm horizontally form the mallets starting position	4	Physical testing
Actuator motion accuracy	D.1.3	When moving the mallet 400 mm without feedback control can not exeed a fault of ± 5mm	3	Physical testing

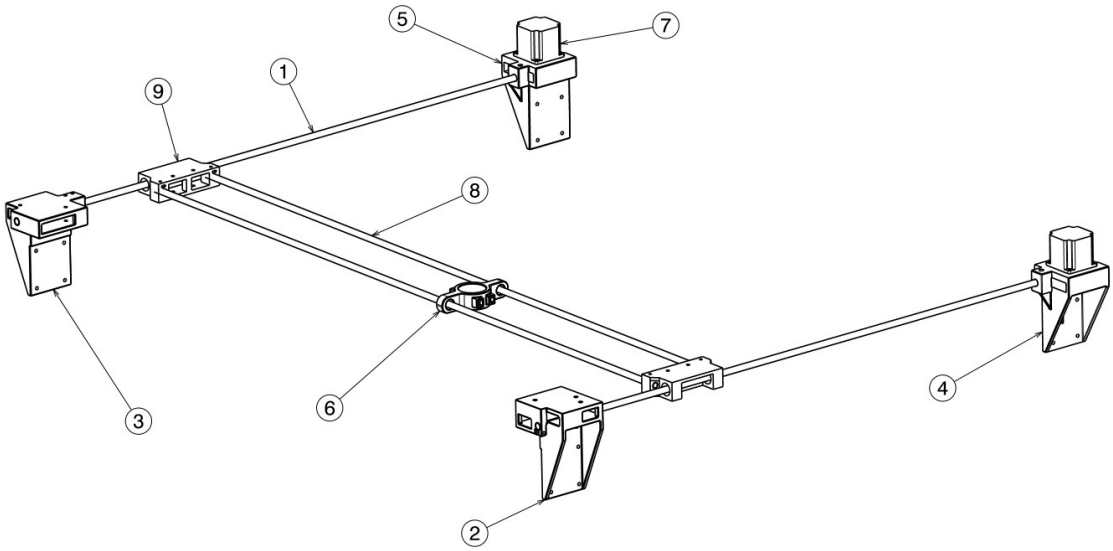
B

MatLab Script - Deflection

```
%aprox calc
clc
%input
E=210e9; %pa
g=9.82; %gravitation.
Diam=[8 10 12 15]*10^-3;
Lenght_axis_x=1.35;
I=pi*Diam.^4/64;
mass_2m_steel=[0.6 0.93 1.35 2.1];
Weight_our_axis=mass_2m_steel./Lenght_axis_x^2;
malletholder=0.1; %200g mallet holder.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%övre axlar egenutbojn: dvs utb pga axelns egenvikt utan hänsyn till last.
for i=1:4
        %deflection of the axis only due to the axis own weight
        d_weight(i)=5*Weight_our_axis(i).*g.*Lenght_axis_x^4/(E*I(i)*384);
end
'ö axl nedböjning pga egenvikt 8 10 12 15 mm:'
d_axisw_mm=d_weight*1000     %i mm

%övre axlar: lastutböjning (klubban)
P=malletholder*g;
for i=1:4
    d(i)=P*Lenght_axis_x^3/(E*I(i)*48);

end
'nedböjning ö axl pga last(klubbhållaren) 8 10 12 15 mm:'
d_last_mm=d*1000

'nedböjning ö axl tot 8 10 12 15 mm:'
d_tot_mm=d_axisw_mm+d_last_mm

%undre balken utböjn total. olika dim på axlar

for j=1:4
    Pu=(2*Weight_our_axis(j)+malletholder);
    for i=1:4
    du(i,j)=Pu*Lenght_axis_x^3/(E*I(i)*48);
    end
end
'utböjning på u axl (last=klubbahållare+öaxel): ( |undre axel 8 10 12 15, - överaxel 8 10 12 15)'
du_mm=du*1000
```

C

Drawing of the Prototype



1	X-Shaft
2	Back right corner pieces
3	Back left corner pieces
4	Front right motor mount
5	Front left motor mount
6	Mallet holder
7	Stepper motor
8	Y-Shaft
9	Sledge house