School of Electronic and Electrical Engineering
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

ELEC3875

Project Report

An Autonomous Chess Playing Robot

Assaad Pierre Tabet

SID: 201145961	Project No. 93
Supervisor: Roger Berry	**Assessor:** Dr Chris Wood

Declaration of Academic Integrity

School of Electronic and Electrical Engineering
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

ELEC3875 Individual Engineering Project

Declaration of Academic Integrity

Plagiarism in University Assessments and the Presentation of Fraudulent or Fabricated Coursework

Plagiarism is defined as presenting someone else's work as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance.

Fraudulent or fabricated coursework is defined as work, particularly reports of laboratory or practical work that is untrue and/or made up, submitted to satisfy the requirements of a University assessment, in whole or in part.

Declaration:

- I have read the University Regulations on Plagiarism [1] and state that the work covered by this declaration is my own and does not contain any unacknowledged work from other sources.
- I confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes.
- I confirm that details of any mitigating circumstances or other matters which might have affected my performance and which I wish to bring to the attention of the examiners, have been submitted to the Student Support Office.

[1] Available on the School Student Intranet

Student Name: Assaad Pierre Tabet Project No. 93

Signed: Pierre Tabet Date: 21/05/2020

Abstract

This report describes the design process for a 3-DOF autonomous chess playing robot. This requires real-time detection of chess moves from a camera using a convolutional layer neural network and a chess engine to determine a response. This is used to calculate the inverse kinematics required to move the chess pieces. The hardware and software were designed from first principles, with the aim of maximising the amount of self-written code and self-designed parts.

Acknowledgements

I would like to thank my supervisor Mr. Roger Berry, for his advice on how to approach developing a mechatronic system. The weekly meetings were vital, and a starting point for much of the technical depth required for this project. Mr. Berry's expertise derives from years in industry, which is invaluable. Additionally, he was very generous with his time, resources and patience. The Robotics and Machine Intelligence course developed my understanding in the conceptual design process for the computer vision and hardware aspects of the project. Leeds University Library's Ebook system was valuable and I made use of relevant chapters in a selection of textbooks. The School of Electronic and Electrical Engineering lab technician was helpful, allowing me the use of the Ultimaker S5 3D printer.

Table of Contents

Introduction

Chess is a widely played strategic game where until recently all official matches were required to be played across a physical board in accordance to The Fédération Internationale des Échecs (FIDE) regulations [1]. Chess is played online by millions of users [2], however many players would still prefer to play using a physical board [3]. This project aims to bring the convenience of digital chess to a physical chess board. For this project to be deemed a success, the experience of playing against an autonomous robot must not be inferior to playing online. Therefore, the system must be able to read, calculate and move pieces as effectively as human players.

An autonomous robot is a mechatronic system. Therefore, it was decided to divide the report into mechanical, electronical, and computing chapters.

Mechanical Systems

Manipulator Designs

4 degrees of freedom design

This design was based off a simplified Programmable Universal Manipulation Arm (PUMA). The robot has 4 revolute joints allowing for 4 degrees of freedom.

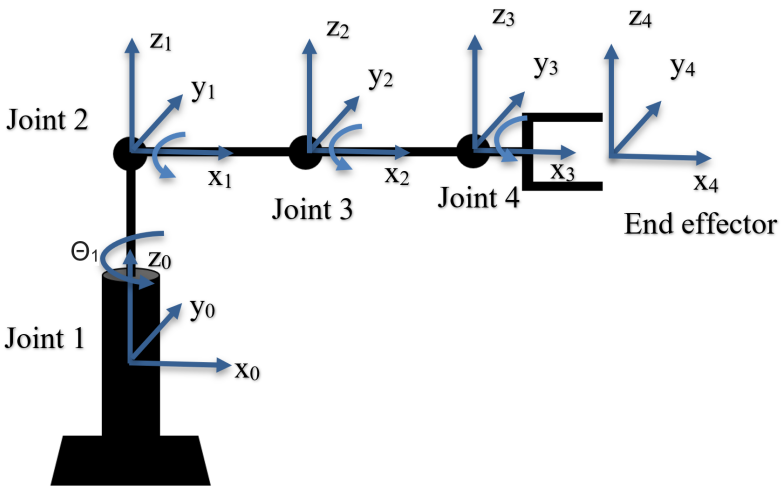


Figure 1 - 4 DOF design

The inverse kinematics problem was solved in order to control the manipulator. Inverse kinematics can be solved algebraically or geometrically. The geometric method was chosen because it is a more intuitive approach when the geometry of the joints have revolute pairs that intersect at a point. The frames were assigned using the Denavit-Hartenberg convention rules [4]:

1. The Z axis must be the axis of revolution or direction of motion
2. The X axis must be perpendicular to the Z axis of the previous frame
3. The X axis must intersect the Z axis of the previous frame
4. The Y axis must be drawn to follow the right-hand rule.

Assigning the frames with these rules allows a matrix table to be made with only 4 parameters instead of a the typical 6. These are the Denavit-Hartenberg parameters, where:

1. θ- is the rotation needed to get the X_{n-1} to match the X_n frame by rotating around Z_{n-1} axes.
2. α is the rotation needed around X_n that is required to get the Z_{n-1} axes lined up with Z_n
3. A is the distance between the centre of the two frames in the
4. X_n direction d is the distance between the two frames in the Zn-1 direction

Link	θ	α	A	d
1	Θ_1	90	L_1	0
2	Θ_2	0	L_2	0
3	Θ_2	0	L_3	0
4	Θ_2	0	L_4	0

$$^{i-1}T_i = \begin{bmatrix} Cos(\theta_i) & -Cos(\alpha_i) \times Sin(\theta_i) & Sin(\alpha_i) \times Sin(\theta_i) & A_i \times Cos(\theta_i) \\ Sin(\theta_i) & Cos(\alpha_i) \times Cos(\theta_i) & -Sin(\alpha_i) \times Cos(\theta_i) & A_i \times Sin(\theta_i) \\ 0 & Sin(\alpha_i) & Cos(\alpha_i) & D_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the first link:

$$^0T_1 = \begin{bmatrix} Cos(\theta_1) & 0 & Sin(\theta_1) & L_1 \times Cos(\theta_1) \\ Sin(\theta_1) & 0 & -Cos(\theta_1) & L_1 \times Sin(\theta_1) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the second link:

$$^1T_2 = \begin{bmatrix} Cos(\theta_2) & -Sin(\theta_2) & 0 & L_2 \times Cos(\theta_2) \\ Sin(\theta_2) & Cos(\theta_2) & 0 & L_2 \times Sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the third link:

$$^2T_3 = \begin{bmatrix} Cos(\theta_3) & -Sin(\theta_3) & 0 & L_3 \times Cos(\theta_3) \\ Sin(\theta_3) & Cos(\theta_3) & 0 & L_3 \times Sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the fourth link:

$$^3T_4 = \begin{bmatrix} Cos(\theta_4) & -Sin(\theta_4) & 0 & L_4 \times Cos(\theta_4) \\ Sin(\theta_4) & Cos(\theta_4) & 0 & L_4 \times Sin(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Overall transformation matrix:

$$^0T_4 = {}^0T_1 \times {}^1T_2 \times {}^2T_3 \times {}^3T_4$$

$$ {}^{0}T_4 = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix} $$

A = $Cos(\theta_1) \times (Cos(\theta_4) \times Cos(\theta_2 + \theta_3) - Sin(\theta_4) \times Sin(\theta_2 + \theta_3))$
B = $Cos(\theta_1) \times (-Sin(\theta_4) \times Cos(\theta_2 + \theta_3) - Cos(\theta_4) \times Sin(\theta_2 + \theta_3))$
C = $Sin(\theta_1)$
D = $Cos(\theta_1) \times (L_4 \times Cos(\theta_4) \times Cos(\theta_2 + \theta_3) - L_4 \times Sin(\theta_4) \times Sin(\theta_2 + \theta_3) + L_3 \times Cos(\theta_2 + \theta_3) + L_1 + L_2 \times Cos(\theta_2))$
E = $Sin(\theta_1) \times (Cos(\theta_4) \times Cos(\theta_2 + \theta_3) - Sin(\theta_4) \times Sin(\theta_2 + \theta_3))$
F = $Sin(\theta_1) \times (-Sin(\theta_4) \times Cos(\theta_2 + \theta_3) - Cos(\theta_4) \times Sin(\theta_2 + \theta_3))$
G = $-Cos(\theta_1)$
H = $Sin(\theta_1) \times (L_4 \times Cos(\theta_4) \times Cos(\theta_2 + \theta_3) - L_4 \times Sin(\theta_4) \times Sin(\theta_2 + \theta_3) + L_3 \times Cos(\theta_2 + \theta_3) + L_1 + L_2 \times Sin(\theta_2))$
I = $Cos(\theta_4) \times Sin(\theta_2 + \theta_3) + Sin(\theta_4) \times Cos(\theta_2 + \theta_3)$

Using the transformation matrix, it was calculated that joints 1 and 2 would need to be controlled to within one tenth of a degree assuming a board size of 220 millimetres and accuracy requirement of 3 millimetres. Therefore, it was decided the requirements were beyond the scope of what could be manufactured given the restraints on time and resources.

3 degrees of freedom design

There are several different designs that allow 3 degrees of freedom. The main variable in design is the number of revolute joints with the easiest designs having 0 or 1 revolute joints.

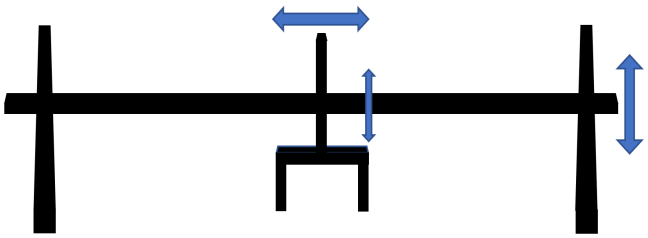


Figure 2 - 3 DOF, 0 Revolute Joint Design

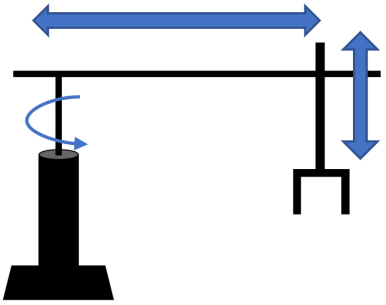


Figure 3 - 3 DOF, 1 revolute joint design

These designs were seen to be too far removed from the aim of making a humanoid manipulator. It was decided that the project should focus on designs with 2 revolute joints and 1 prismatic joint. The simplest version of this design can be seen in figure 4.

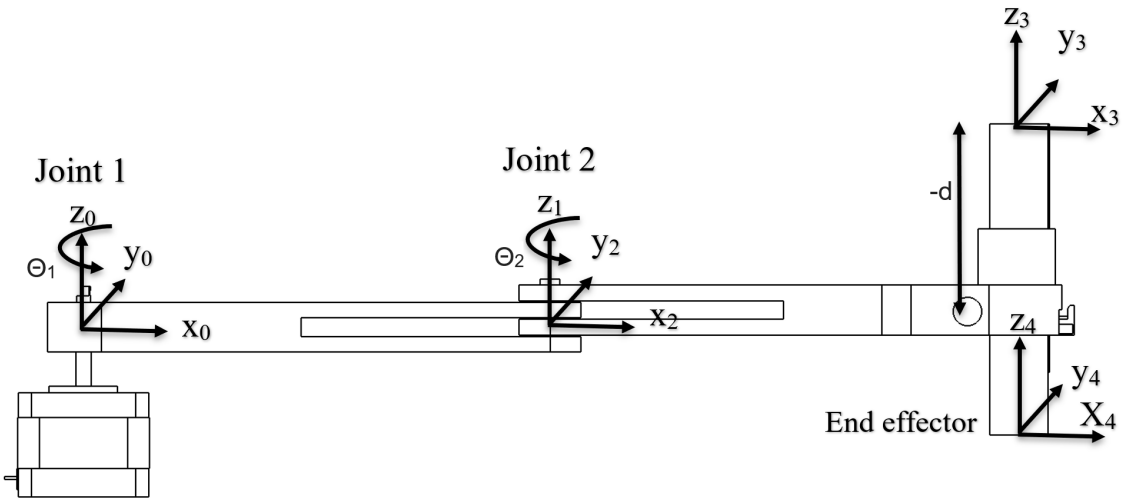


Figure 4 - 3 DOF, 2 revolute joint design

Link	θ	α	A	d
1	Θ_1	0	L_1	0
2	Θ_2	0	L_2	0
3	0	0	0	d

For the first link:

$$ {}^{0}T_{1} = \begin{bmatrix} Cos(\theta_1) & -Sin(\theta_1) & 0 & L_1 \times Cos(\theta_1) \\ Sin(\theta_1) & Cos(\theta_1) & 0 & L_1 \times Sin(\theta_1) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For the second link:

$$ {}^{1}T_{2} = \begin{bmatrix} Cos(\theta_2) & -Sin(\theta_2) & 0 & L_2 \times Cos(\theta_2) \\ Sin(\theta_2) & Cos(\theta_2) & 0 & L_2 \times Sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For the third link:

$$ T_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Overall transformation matrix:

$$T_3^0 = \begin{bmatrix} Cos(\theta_1 + \theta_2) & -Sin(\theta_1 + \theta_2) & 0 & (L_1 \times Cos(\theta_1)) + (L_2 \times Cos(\theta_1 + \theta_2)) \\ Sin(\theta_1 + \theta_2) & Cos(\theta_1 + \theta_2) & 0 & (L_1 \times Sin(\theta_1)) + (L_2 \times Sin(\theta_1 + \theta_2)) \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The generic transformation matrix, where *R* represents the rotation aspect and *d* represents the translational aspect:

$$T = \begin{bmatrix} R_{x1} & R_{x2} & R_{x3} & d_x \\ R_{y1} & R_{y2} & R_{y3} & d_y \\ R_{z1} & R_{z2} & R_{z3} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore:

$$d_x = (L_1 \times Cos(\theta_1)) + (L_2 \times Cos(\theta_1 + \theta_2))$$
$$d_y = (L_1 \times Sin(\theta_1)) + (L_2 \times Sin(\theta_1 + \theta_2))$$
$$d_z = d$$
$$Cos(\theta_2) = \frac{1}{2 \times L_1 \times L_2} \times ({d_x}^2 + {d_y}^2 - {L_1}^2 - {L_2}^2)$$

$$Sin(\theta_2) = \pm\sqrt{1 - Cos(\theta_2)^2}$$

$$\theta_2 = -\alpha \times Tan\left(\frac{Sin(\theta_2)}{Cos(\theta_2)}\right)$$

$$\theta_1 = \alpha \times Tan\left(\frac{+d_x \times L_2 \times Sin(\theta_2) + d_y \times (L_1 + L_2 \times Cos(\theta_2))}{d_x\big(L_1 + L_2 \times Cos(\theta_2)\big) - (d_y \times L_2 \times Sin(\theta_2))}\right)$$

An issue with this design is that if the motors for joint 2 and 3 are placed at their joints, there is the possibility of beam deflection affecting the inverse kinematics. Calculations for deflection and slope of a cantilever beam with a point load for the first link were done to determine what the minimum thickness of the beam should be.

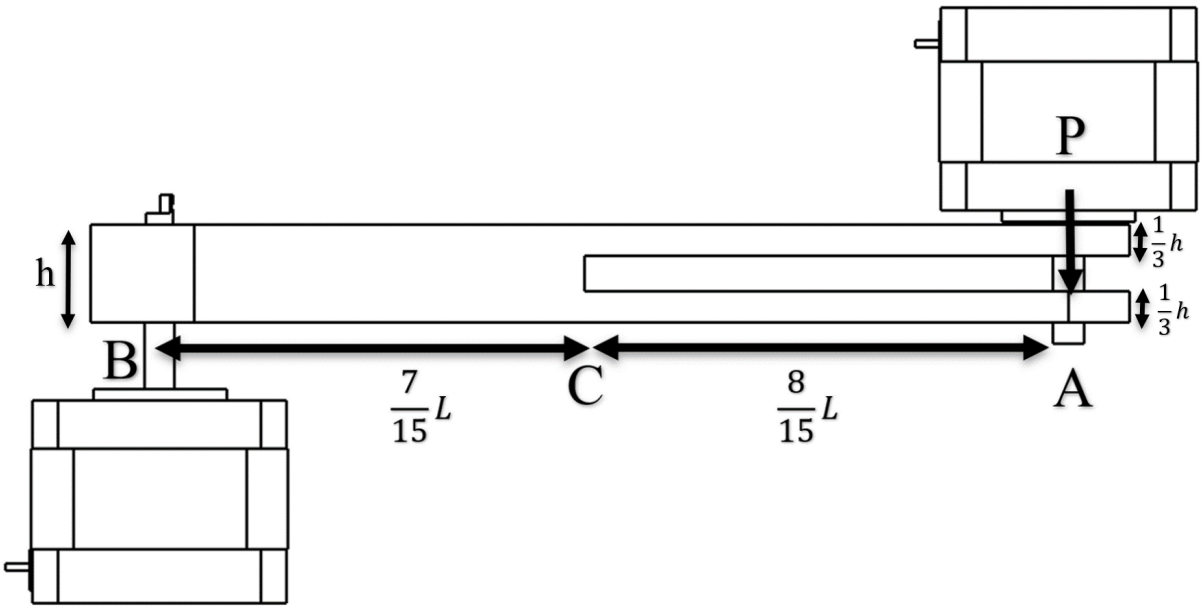


Figure 5 - Link 1 cantilever beam

This is equivalent to:

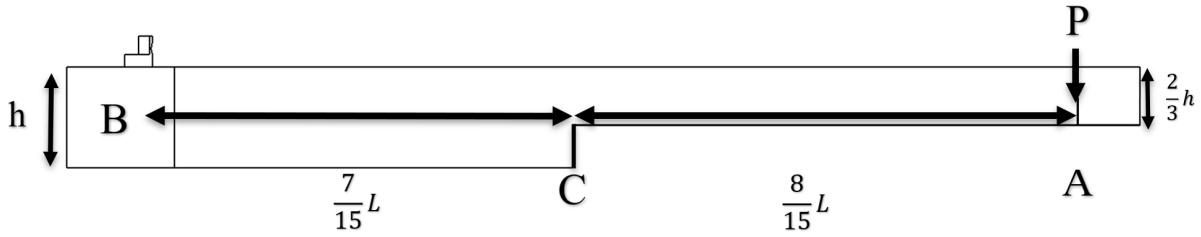


Figure 6 - Link 1 cantilever beam

In this design the moment of inertia (I) is not constant throughout the beam.

$$I = \frac{bh^3}{12}$$

$$I_{AC} = \frac{b(\frac{2}{3}h)^3}{12}$$

$$I_{BC} = \frac{27}{8}I_{AB}$$

$$\delta_1 = \frac{P\left(\frac{8L}{15}\right)^3}{3EI} \approx \frac{5PL^3}{101EI}$$

$$\delta_c = \frac{P\left(\frac{7L}{15}\right)^3}{3E(\frac{27}{8}I)} + \frac{(\frac{8PL}{15})\left(\frac{7L}{15}\right)^2}{2E(\frac{27}{8}I)} \approx \frac{17PL^3}{625EI}$$

$$\theta_c = \frac{P\left(\frac{7L}{15}\right)^2}{2E(\frac{27I}{8})} + \frac{(\frac{8PL}{15})(\frac{7L}{15})}{E(\frac{27I}{8})} \approx \frac{43PL^2}{405EI}$$

$$\delta_2 = \delta_c + \frac{8L\theta_c}{15} \approx \frac{21PL^3}{250EI}$$

$$\delta_A = \delta_1 + \delta_2 \approx \frac{2PL^3}{15EI}$$

Rearranging for the I:

$$I \approx \frac{2PL^3}{15E\delta_A}$$

Substituting:

$$\frac{bh^3}{12} \approx \frac{2PL^3}{15E\delta_A}$$

$$h \approx \sqrt[3]{\frac{8PL^3}{5bE\delta_A}}$$

For a deflection aim (δ_A) of $0.1 * 10^{-3}m$, assuming a modulus of elasticity (E) of polylactic acid polymer (PLA) is $3.5 * 10^9$Pa as demonstrated in [5],a beam with (b) of $20 * 10^{-3}m$, a length (L) of $150 * 10^{-3}m$ and a force (P) of 2.5N .

$$h = \sqrt[3]{\frac{(8)(2.5)(150 * 10^{-3})^3}{(5)(20 * 10^{-3})(3.5 * 10^9)(0.1 * 10^{-3})}}$$

$$h = 12.4 * 10^{-3}m$$

A height (h) of $16 * 10^{-3}m$ was chosen for the arm, this allowed enough height for bearings to be inserted into the arms.

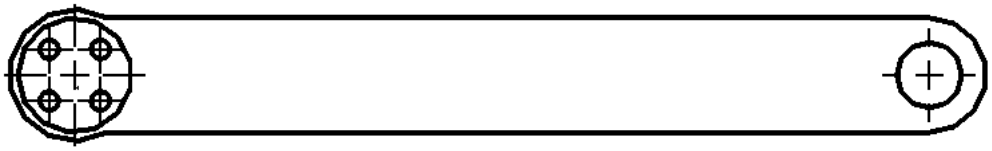


Figure 7 - Link 1 Below view

Figure 8 - Link 1 Isometric view

This design can be improved further by reducing the rotational inertia. Assuming the centre of mass of the link acts halfway along the link, the rotational inertia is given by:

$$I = \frac{m_1 L^2}{4} + m_2 L^2$$

m_1- Mass of link 1

m_2- Mass of motor at joint 1

L- Length of link 1

By removing the stepper motor from joint 2, the rotational inertia is decreased significantly. In order to rotate link 2 a belt pulley could be used. However, this has a few disadvantages such as movements of link 2 relative to link 1 whilst motor 2 is stationary. An alternate design is to increase the amount of links as shown in figure 9. This design remains a 3 degrees of freedom design as shown by Grübler's formula:

Links=6 Full Joints=6 Half Joints=0

M= 3L – 2J -3G

M=3(L-1)-2J-3(0)

M=15-12

M=3 D.o.F

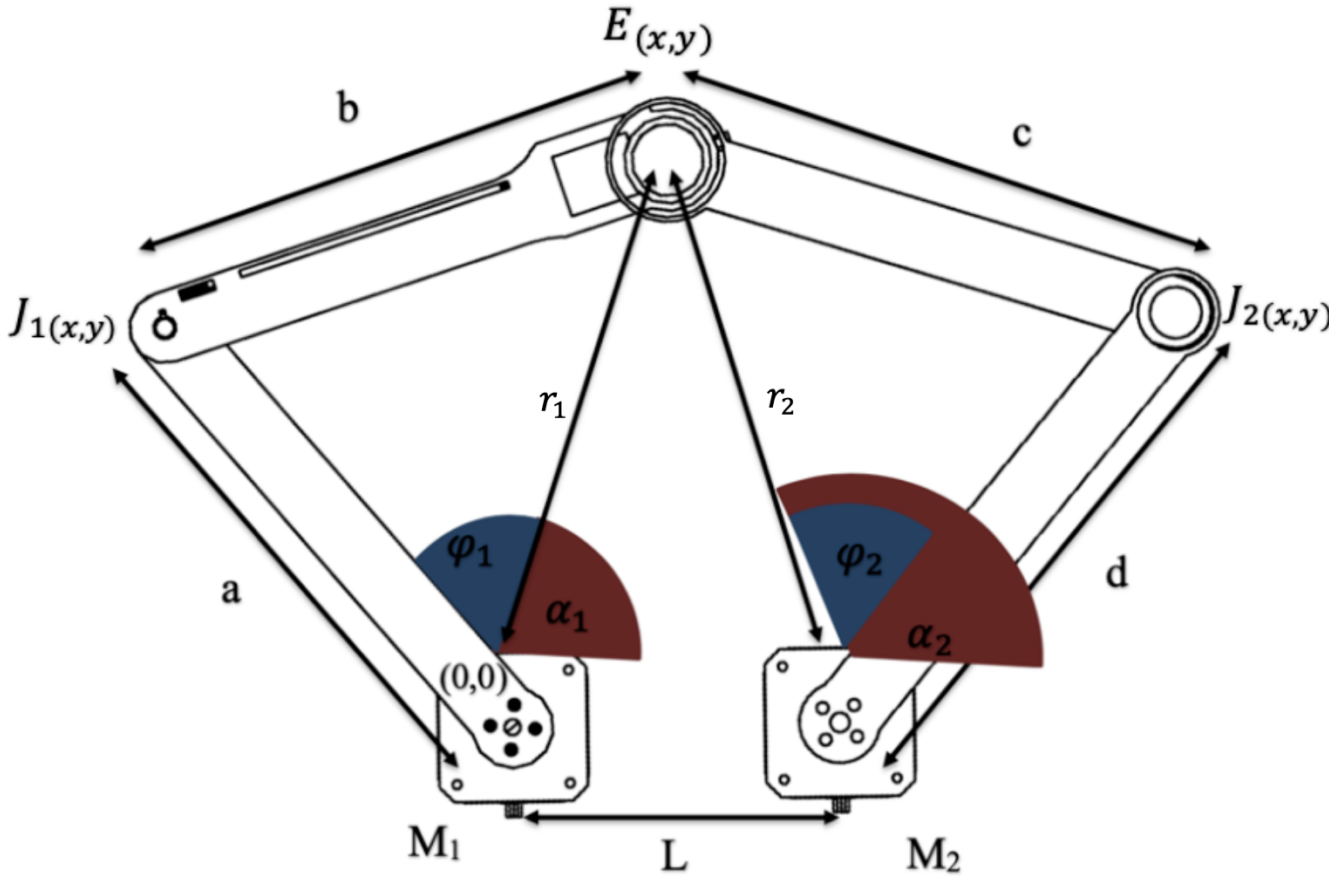


Figure 9 - 3 DOF, Alternate Design with 6 Link and 6 Joint design

Where:

$E_{x,y}$ is the position of the end effector.

$J_{1(x,y)}$ and $J_{2(x,y)}$ are the positions of the joints.

L is the horizontal offset of the second motor

a, b, c and d be the lengths of the arms.

r_1 and r_2 be straight lines drawn between motors M_1 and M_2, and $E_{x,y}$, respectively.

φ and α be the angles between r and a, and r and the zero-position of M_1, respectively.

γ_1 and γ_2 be the angles that motors M_1 and M_2 must move to reach $E_{x,y}$

The inverse kinematics for this design were simulated in MATLAB. Many variations of length links a, b, c, and d were tested against distance L with the aim of finding the smallest possible link lengths that still allows the end effector to reach all points within the work surface. The optimal solution was to make L a length of zero, and the links a, b, c, and d a length of 150 millimetres. The MATLAB script can be found in the appendix.

For the manipulator to pick up the its own king or queen piece, $E_{x,y}$ will be approximately (0,40) millimetres. Therefore, the length of the groove cut within the mass supporting links were calculated:

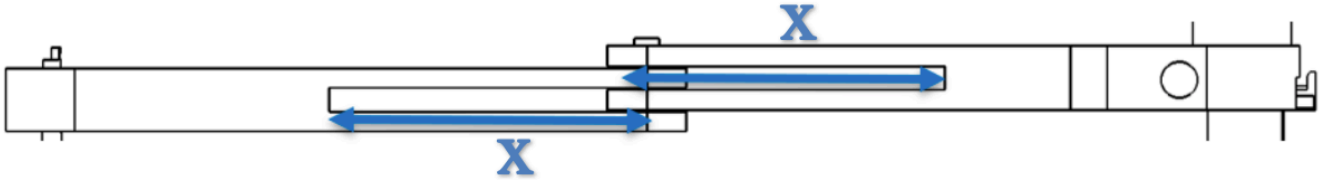
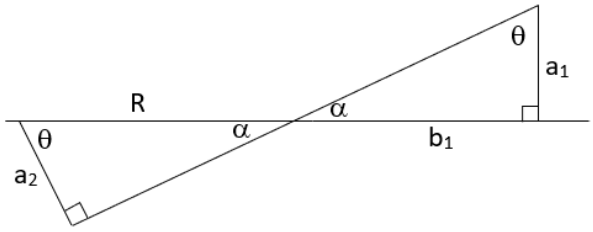


Figure 10 - Determining the Groove cut Length

Cosine rule:

$$cosa = \frac{150^2 + 150^2 - 40^2}{2 * 150 * 150}$$
$$a = 15.32$$

Geometry:



$$x = b_1 + R$$

$$x = a_1 \times Tan(\theta) + \frac{a_2}{Cos(\theta)}$$

X – Length of groove cut
a_1 – Half the width of link a

The length of groove cut must be 75 millimetres.

Having calculated all the essential parameters for the manipulator, SolidWorks was used to simulate the movement of the links before manufacturing.

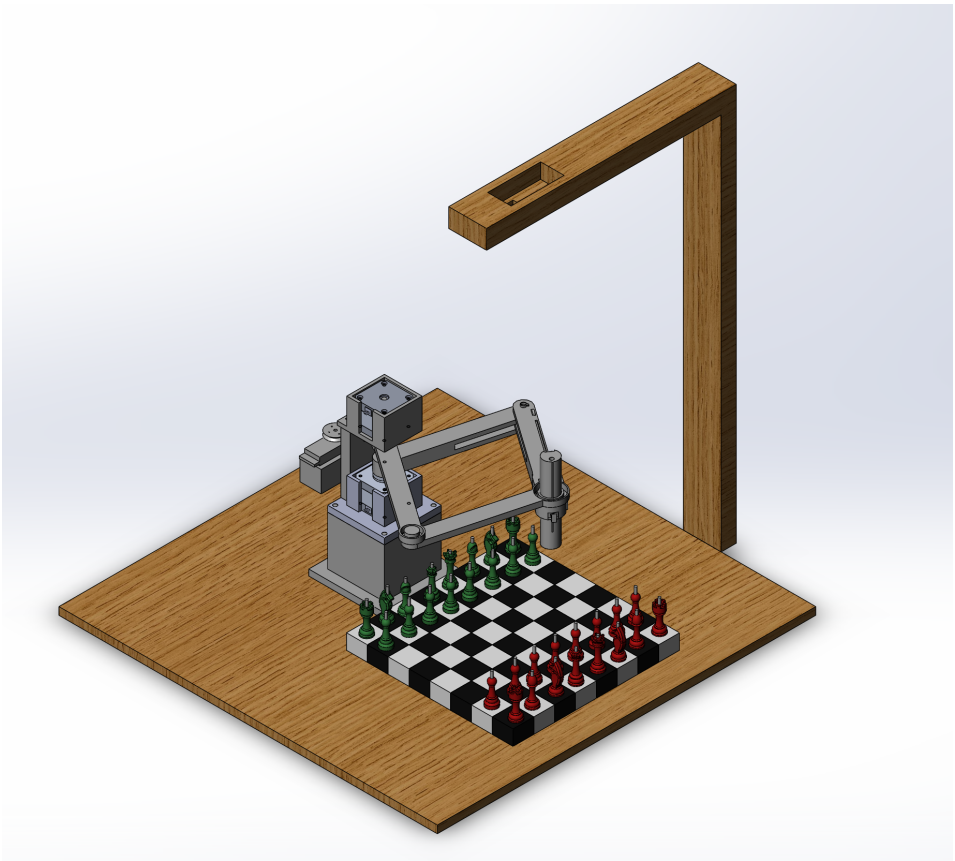


Figure 11 - SolidWorks Model

Rack and Pinion

It was decided to separate the motor from the prismatic joint in order to keep the suspended mass and rotational inertia at a minimum. In order to move the prismatic joint a rack and the pinion was chosen. The gear rack is a cylindrical gear with an infinite pitch cylinder radius. A rack consists of teeth cut at equal distance and of the same shape and size. By combining the two, one can convert rotational motion into linear motion. The university's license of SolidWorks does not include use of the gear toolbox, therefore both rack and pinion were made from blank sketches.

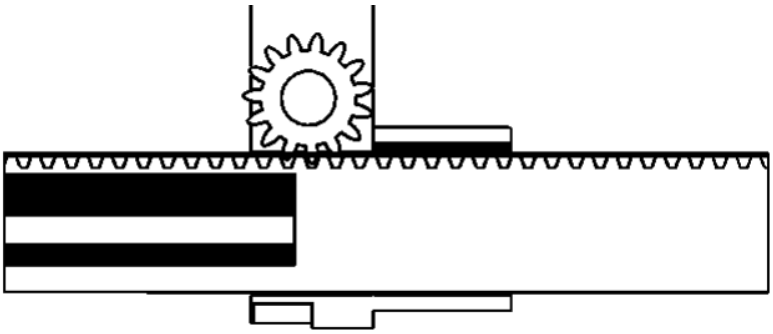


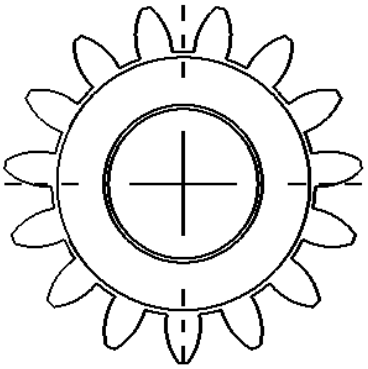
Figure 12 - Rack and Pinion

The gear parameters were chosen to allow 180-degree servo to move the rack by its entire length. The pinion rotates twice upon one half a rotation of the servo pulley, because the pinion has a smaller pulley attached to its side as shown in figure 14. The diameters of the pulleys were calculated with:

$$\frac{d_1}{d_2} = \frac{n_2}{n_1}$$

d_1- servo pulley diameter
d_2- pinion pulley diameter
n_1- revolutions of servo pulley
n_2 - revolutions of pinion pulley

The pinion was given 15 teeth, as this is was good compromise between transmitted torque and teeth being resistant to breaking. The physical diameter of the gear was limited by the width of arm *b*, therefore the pinion pitch circle was calculated by dividing the diameter by the number of teeth. The linear pitch or linear distance between teeth of the rack was then calculated by multiplying the pinion pitch circle by pi. The pressure angle, which is the slope of the tooth at the pitch circle of the pinion was set to 25 degrees. When the rack switches directions, there will be a small amount of travel lost known as backlash, for smooth rotation and to minimise wear a backlash of 0.3 mm was chosen. The servo rotates anticlockwise to lower the rack containing the electromagnet towards a chess piece. A spring between arm *b* and the rack ensures the fishing line remains taught whilst the servo rotates.



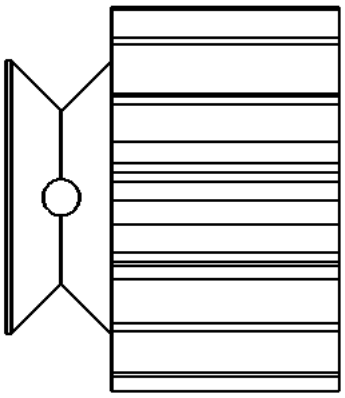
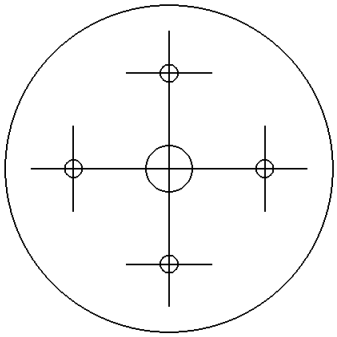


Figure 13 - Pinion side view Figure 14 – Pinion and Pulley View



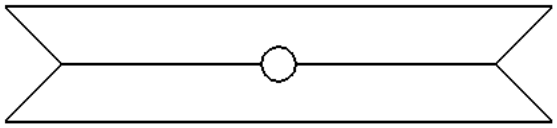


Figure 15 - Servo Pulley

End Effector Designs

Servo Actuated Gripper

This design requires a servo to actuate the opening and closing of the gripper. The RB-Dfr-358 robot gripper allowed for extension fingers to be attached, and the intention was to 3D print fingers that could lift the underside of a chess piece.

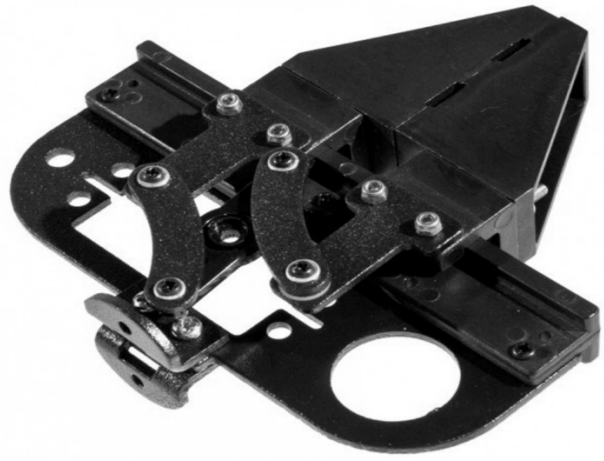


Figure 16 - Robot Gripper

Source: [6]

The issue with this design is that the servo adds additional weight and requires additional design changes to accommodate the wiring along the arm. The placement of this weight also raises the moment of inertia of the arm, an undesirable characteristic.

Parker Pen Inspired Gripper

In order to reduce the weight, the design was improved by removing the need for a servo motor. It was decided to attempt to actuate the gripper by using the distance travelled by the rack in z axis. In order to do this the gripper needed to follow the sequence as shown in figure 17.

Figure 17 - Cycle Flow Chart

In order to achieve this sequence, the cam body of a projecting and retracting ballpoint pen could be modified to convert the linear motion into a rotational one. A piece of string would be used to cycle through the pen's projected and retracted state, representing the opening and closing of the gripper. When the arm is fully lowered the mechanism will pick up or release a chess piece whilst being able to keep hold of a chess piece whilst in the raised state. The mechanism used would be similar to the cam body and plunger used in Parker pen detailed in figure 18. The mechanism works because the cam body is set free when it is lowered below the stop members, allowing it to rotate when the spring pushes it up. When the pen has been extended and retracted once the cam body has been rotated by 45 degrees 4 times, requiring two cycles for a full rotation.

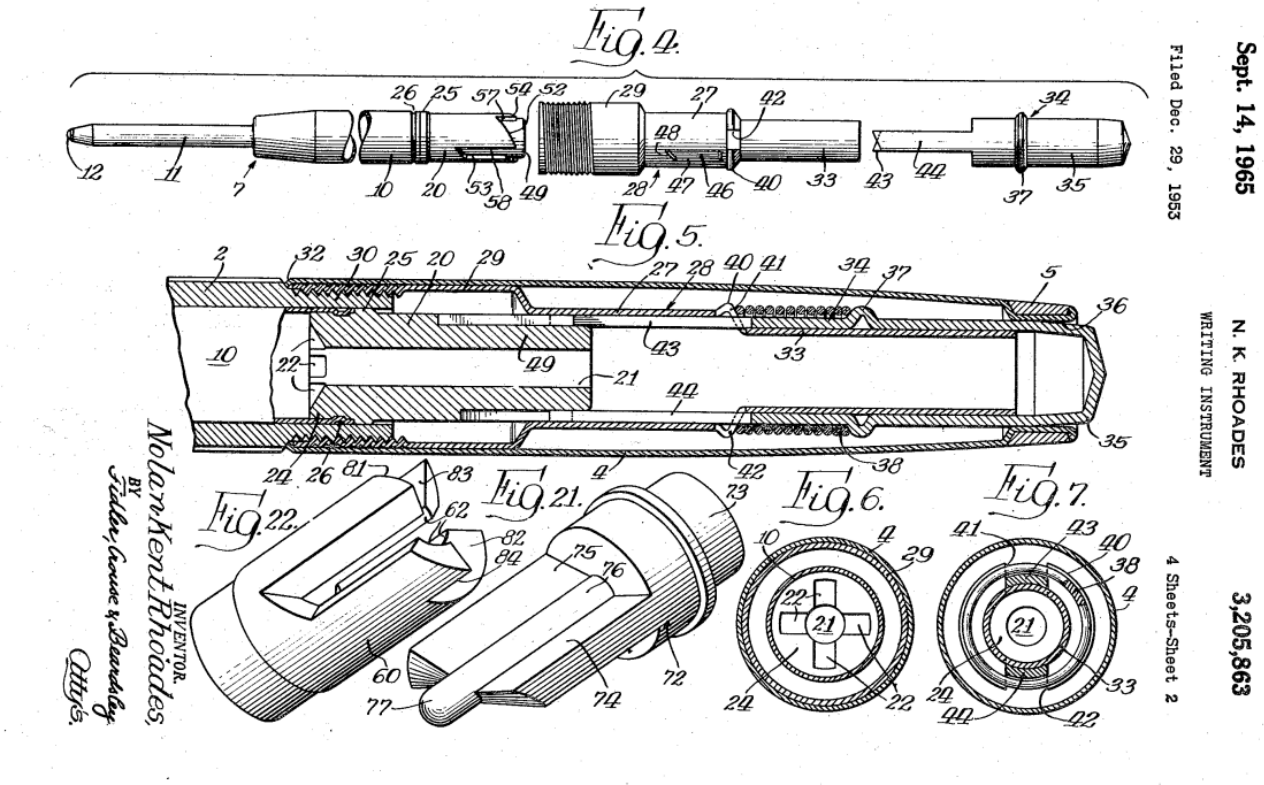


Figure 18 - Parker Pen Company retractable ballpoint pen

Source: [7]

Due to the difficulty of manufacturing the housing to repurpose the pen's mechanism and a lack of time, it was decided to remove the need for a mechanical gripper and use an electromagnet instead.

3D printing

Additive manufacturing is the process of making objects by adding material layers at a time. For most of human history manufacturing has involved removing material. 3D printing is a type of additive manufacturing typically used for prototyping. 3D printing allows digital models to be made into three-dimensional solid objects of any shape and dates back to 1972 when Mastubara proposed curing photopolymer resin cured under a mercury vapour lamp [8]. The real birth of 3D printing occurred in 1984 and 1991 with the inventions of stereolithography (SLA) and fused deposition modelling (FDM). Many innovations have been made since, such as growing organs from 3D printed scaffolds to improvements in prosthetics and medicine. Today there are many materials that can be used in 3D printing and several different technologies which primarily focus on how layers are built up. An FDM Ultimaker S5 was made available for this project. Printing with FDM comes with some disadvantages in comparison to SLA, such as resolution, warping and supports leaving marks however the cost per part is significantly less [9]. The S5 is capable of printing industrial grade parts and a 0.4mm nozzle was used. In order to print the desired parts a slicing engine is required to convert an STL file into G-code, which

represents the path the 3D printer will follow. Additionally, G-code contains information about print speed, geometric dimensioning and tolerancing. Due to the absence of Mr. Bilbrough it was required to learn how to use slicing software and choose parameters. Ultimaker recommends Cura as a slicing engine and most of the recommended settings were changed to prioritise speed over quality as the tolerances were within what was required. The orientation of the parts minimised the amount of bridges and overhangs, however support structures were still required. It was recommended to add support for overhangs of over 45 degrees and bridges greater than 5mm. Cura has the functionality to automatically add support everywhere, however this setting was changed to avoid imperfections. The support density was set at 30% and the pattern was set to Zigzag as it is the simplest to remove. In retrospect the density should have been lower as the support was difficult to remove with the rack section of a part. Joint distance and support distance in all 3 axes were left at their default settings as increasing it would only make it harder to remove and it was not known what it could be reduced to before there is a loss in detail. To balance between beam deflection requirements and minimising material used, the infill density was set to 20% with a honeycomb pattern. After the closure of the University, an online service was used to manufacture and dispatch remaining parts. In order to have more control over the printing process, support structures were integrated into the design as the manufacturer did not allow control over their slicing software.

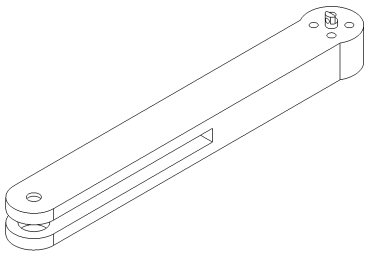


Figure 19 - Without Support Structure

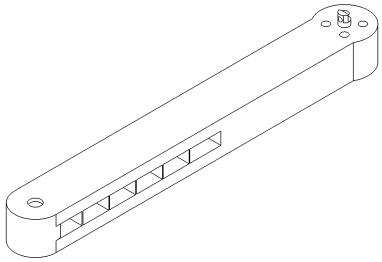


Figure 20 - Integrated Support Structure

Figure 21 - Support Structure Removed

Electronic systems:

Motor choice:

There are many different types of electric motor choices that will meet the minimum specification requirements for this project. Usually brushless servo motors or electro-hydraulic actuators are used in industrial robots, whilst research robots use stepper or DC motors. The specification of this project requires a focus on accuracy and repeatability over speed and load capabilities. Accuracy is a measure of the distance error associated between the target point and achieved point.

$$A_p = \sqrt{(\bar{x} - x)^2}$$

A_p – positional accuracy

$\bar{x}$ – mean of positions attained in one dimension

x – target position

Repeatability is a measure of consistently reaching a target point in space.

$$S_1 = \sum_{i=1}^{n} \sqrt{\frac{(1_i - \bar{1})}{n-1}} [10]$$

S_1 – Standard deviation for repeatability

$\bar{1}$ – mean positional repeatability

n – Number of repetitions

Goswami et al. [11] explores how both of these qualities are linked to some degree and that different applications will often prioritise one of these qualities. For example, a surgical robot may require high accuracy, where as a manufacturing robot will be optimised for repeatability. In the scope of this project this will mean choosing between stepper motors and positional servo motors.

Stepper motors are synchronous motors which can be divided into 3 main types, seen in figure 22. Variable reluctance stepper motors have magnetically soft iron rotors which means they cannot match the holding torque of the other types, however they do maintain their torque longer at higher speeds.

Permanent stepper motors are the simplest to manufacture, however the step size is limited by the number of stator windings and the absence of gears on the rotor limit its resolution in comparison to the other types. Hybrid stepper motors have the advantage of smaller step angles, higher torques and higher holding torques, this makes them highly desirable and widely available.

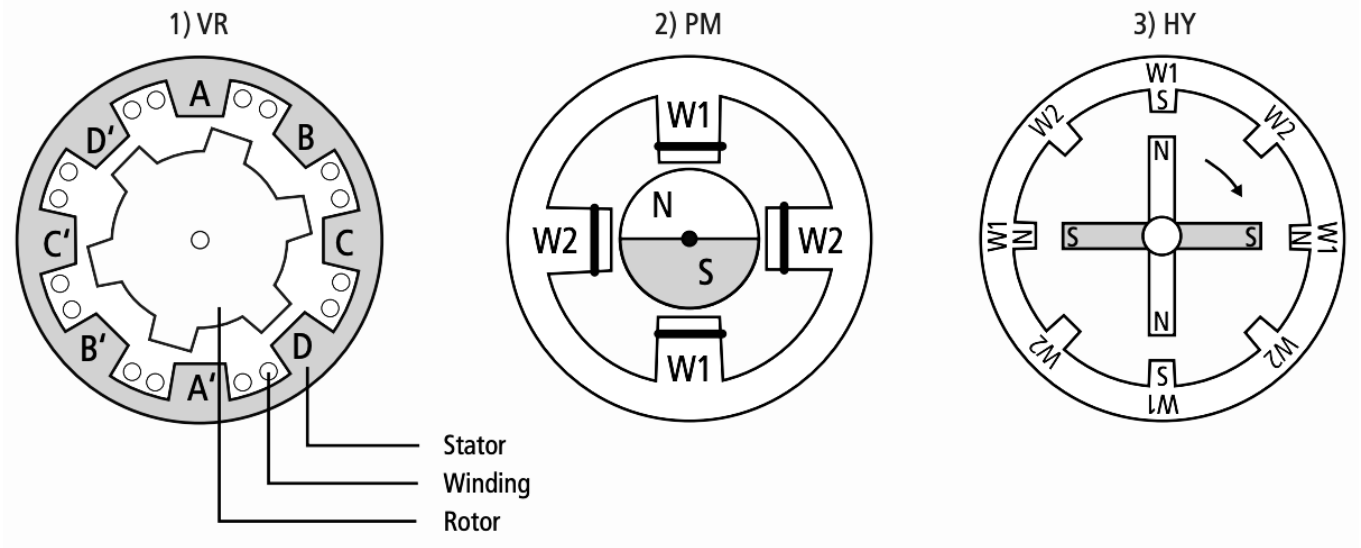


Figure 22 - Stepper Motor types: Variable reluctance (1), Permanent magnet(2) and Hybrid(3)

Source: [12]

Within hybrid stepper motors, there is the choice of unipolar and bipolar. Unipolar motors are similar to bipolar motors with the addition of a central tap, allowing each coil to be split into two smaller coils. However, if the central tap were to be disconnected a unipolar motor could be converted into a bipolar configuration. Bipolar motor windings are larger, and the dual polarity means that all phases can be active simultaneously, creating larger magnetic fields. The need for dual polarity makes bipolar motors more complex to drive, with the requirement of an H-bridge or a driver chip.

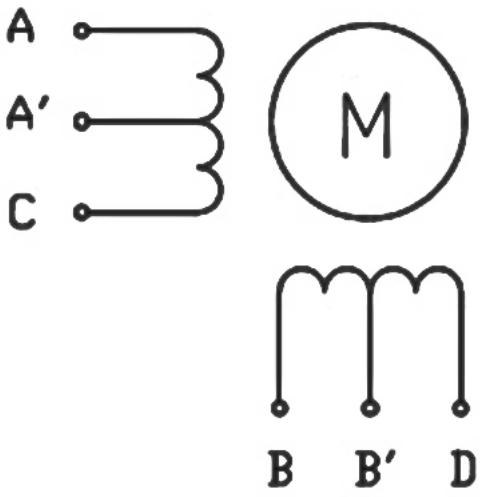


Figure 23 - Bipolar stepper motor

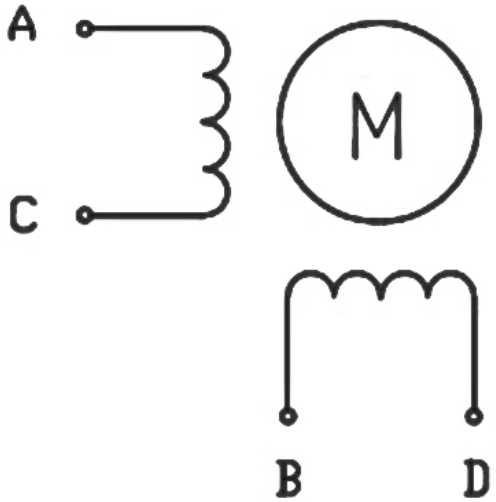


Figure 24 - Unipolar stepper motor

Source: [13]

A hobby positional servo motor is a closed loop system consisting of a DC motor, a potentiometer and a control circuit. They require power, ground and a signal line. They are controlled by modifying the duty cycle of the pulse width modulation (PWM) signal, with larger and smaller duty cycle moving the output shaft towards the rotational limit and 0 degrees respectively. The control for a servo motor occurs on an

integrated chip which contains an H-Bridge, pulse stretcher, pulse generator and pulse width comparator as demonstrated in appendix A of [14]. The pulse generator converts the resistance value from the potentiometer to a PWM signal and the pulse width comparator outputs an error waveform and error direction based on which pulse was wider. A pulse stretcher will then elongate the error signal, allowing the motor to spin for longer. The direction signal is sent to the H-Bridge, dictating which direction the motor will spin. For the Parallax Standard Servo this process occurs 50 times per second until the steady state error is within the acceptable tolerance (dead-band). The designer chooses a dead-band magnitude 'to both satisfy system specifications and outperform a similarly designed open-loop system' chapter 4.9 of [14]. Additionally, because back electromotive forces (EMFs) are proportional to motor speeds, a designer can tune the level of damping by feeding the EMF signals into the pulse generator to influence the PWM signal output.

A NEMA 17 bipolar hybrid stepper motor and Parallax Standard Servo were provided for testing for this project. The torque at low speeds were 0.268 Nm for the servo motor and 0.260 Nm for the stepper motor meaning both motor types are well within the specification. It was decided to test different methods of driving the stepper motor but given a longer project time the integrated circuit of the Parallax servo could have been optimised as well. For example, replacing resistor R_{DB} with a smaller resistance to increase the magnitude of the dead-band could prevent jittering that occurs when rotating the inertial load. Also speed control and damping can be tuned by changing the values of R_s/C_s and R_f respectively.

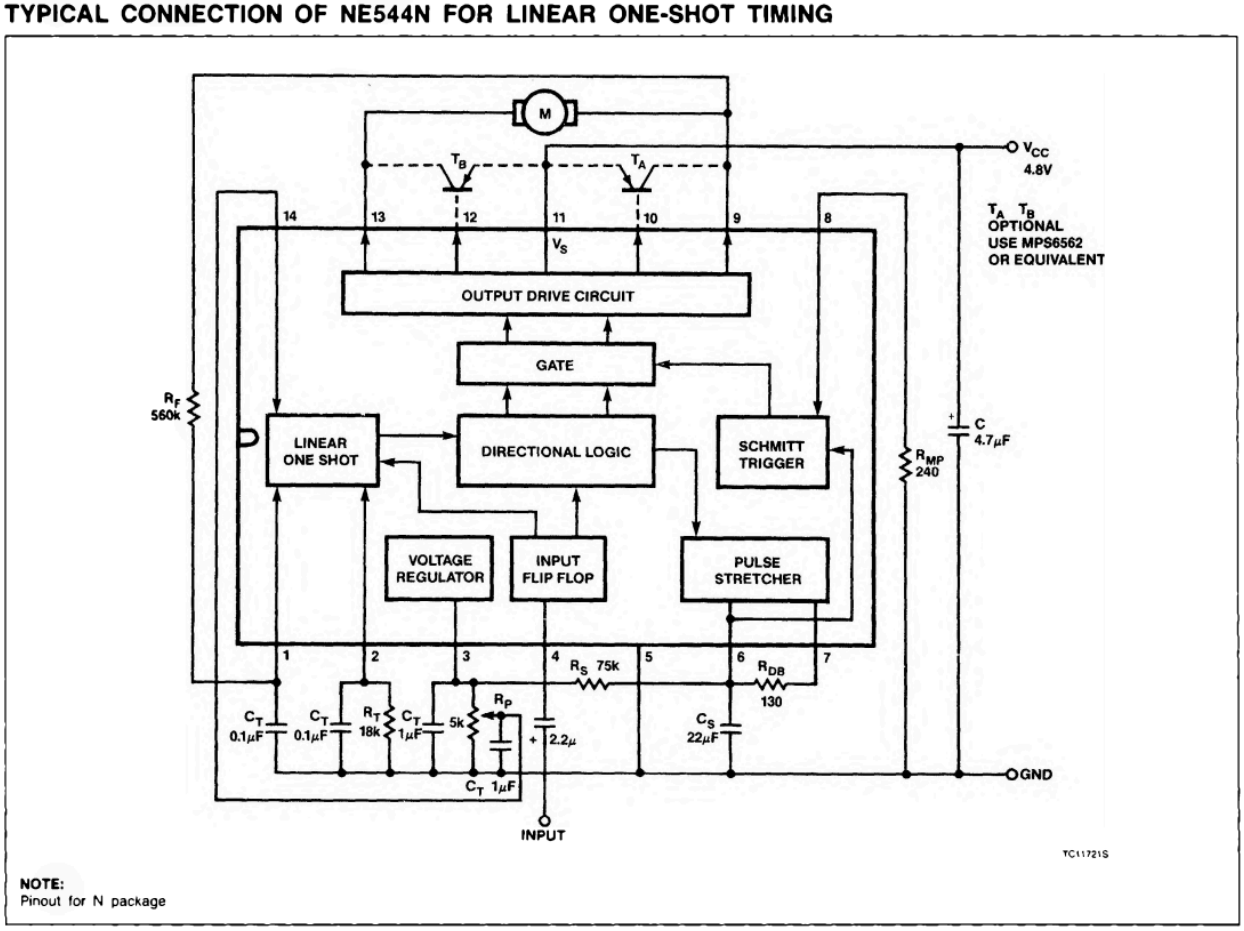


Figure 25 - Electrical block diagram for Signetics NE544

Source: [15]

There are several ways of driving the NEMA 17 stepper motor. The fundamental feature of all the drive circuits are dual H bridges, each consisting of 4 MOSFET transistors that act as switching mechanisms. Enhancement MOSFETs are used over bipolar and IGBT transistors because they are easier to control as they are voltage control devices and offer faster switching times which allows for faster step times and smaller losses. Diodes are placed in parallel in order to protect the MOSFETs from the EMFs which occur when the magnetic field around the windings of the motor.

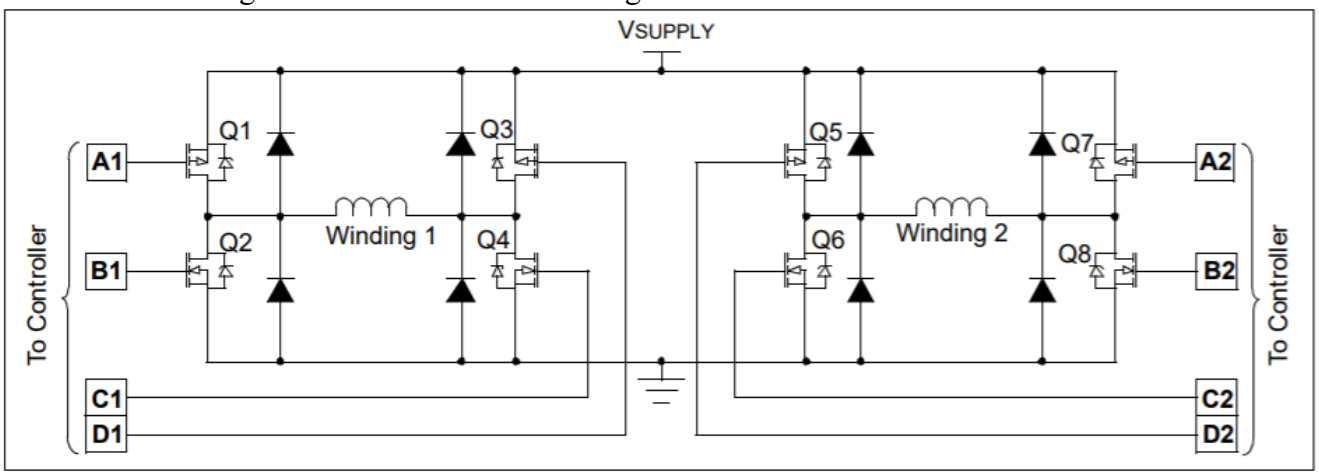


Figure 26 - H-Bridge

Figure 26 shows that each winding is controlled by two N-channel and two P-channel MOSFETs requiring positive and negative bias respectively. P-channel MOSFETs control the high side because they do not need to be driven above the source, however it is harder to find P-channel MOSFETs with the same low levels of $R_{DS(on)}$ that are found with N-Channel.

Due to the limitations of the amount of ports on the selected microcontroller an integrated circuit board with dual H-Bridges was chosen instead of building the circuit from discrete transistors. This allows us to step a motor with 4 signal lines instead of 8 per stepper motor. The L298N was chosen because it can handle 35 Volts at 3 amps which is suitable for the NEMA 17.

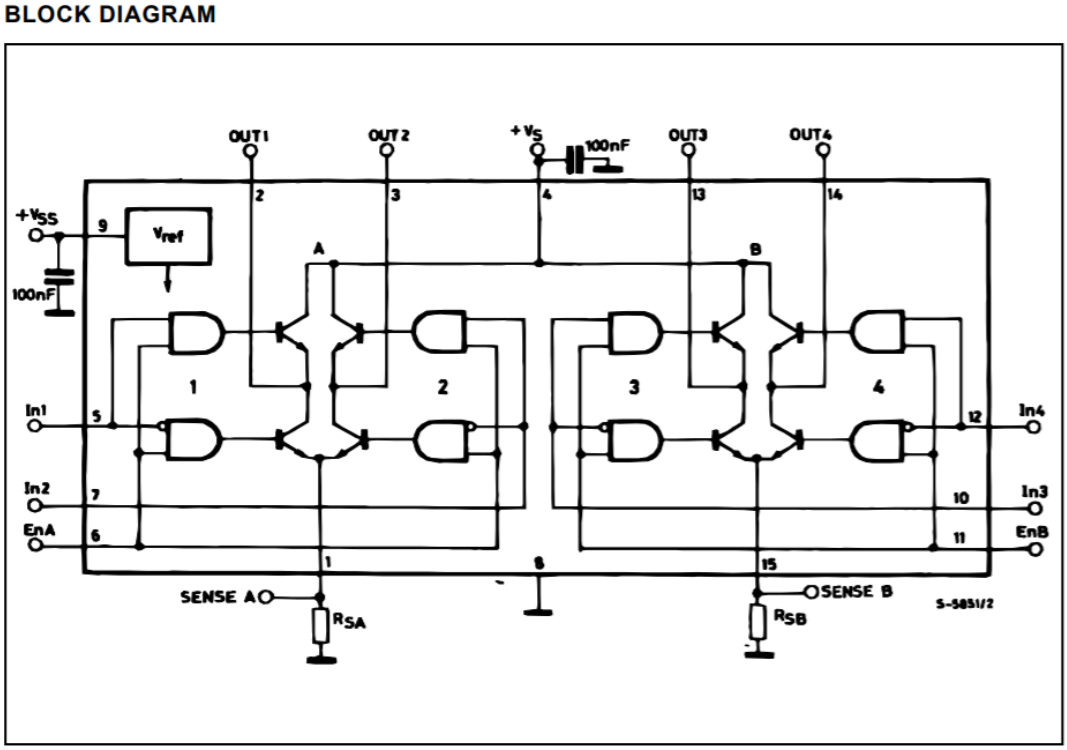


Figure 27 - Electrical block diagram for L298 circuit

Source: [16]

Driving two stepper motors with the L298N is close to the selected microcontrollers processing limit. If the all-revolute design were to have been chosen it would not have been possible to drive 3 stepper motors simultaneously without replacing high level methods or using a dedicated controller such as the A4988. The logic for stepping the motor occurs within the controller reducing the processing requirements especially when microstepping.

Electromagnet:

Plain steel dowel pins were inserted into all the chess pieces. The dowel pins are a length of 12mm, and after being inserted into the chess pieces all terminate at the same height, regardless of the chess piece heights.

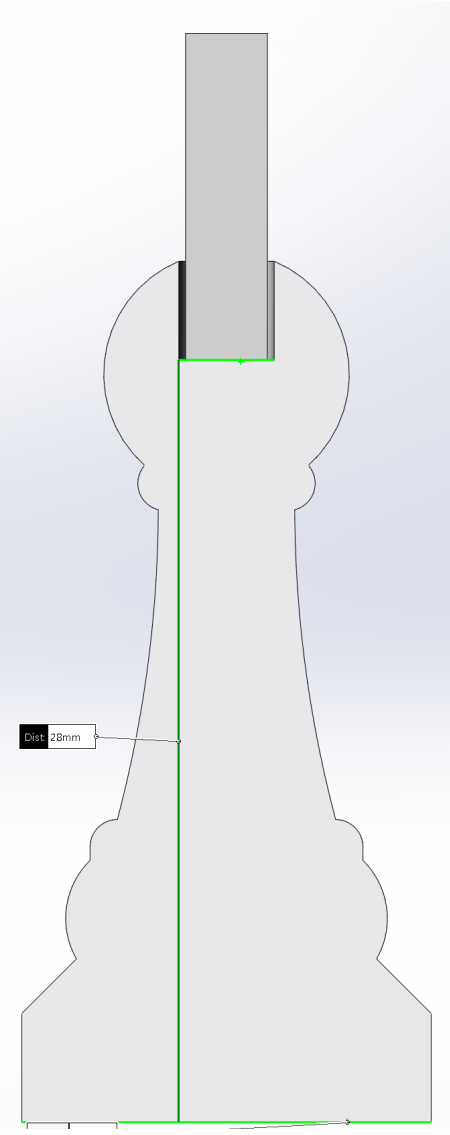


Figure 28 - Pawn

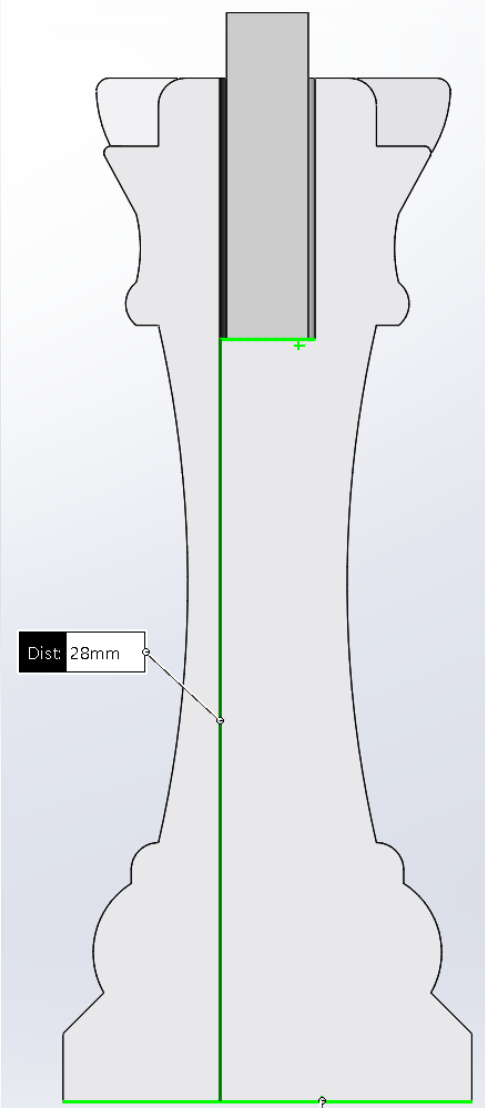


Figure 29 - Queen

For the robot to reliably pick up and place the chess pieces, the magnetic force of the robotic end effector must at least match the weight of the heaviest chess piece at an arbitrary distance of 0.5 centimetres. Too strong of a force wastes power, not ideal if this system was to be revised to be a battery-operated product. An electromagnet with a manufacturer rated performance of 1.2 Watt, 1.8 kilogram pull was chosen. Pull is how much weight is needed to pull the electromagnet off a steel surface and assumes the steel is thick enough to have no saturation. Other qualities which limit the force include permeability, shape and

temperature. Additionally, the force between the electromagnet and the steel dowels is inversely proportional to the square of the distance.

$$F = \frac{\mu_0 A (NI)^2}{2d^2}$$

N – Number of coils in the solenoid

I – Current

μ_0 – permeability of space

A – area of the surface

d – distance between electromagnet and dowel pin

The electromagnet requires 12 Volts, 0.1 Amp of direct current. This is more than the microcontroller can output from and therefore an enhancement-mode N-channel MOSFET was used to switch the electromagnet on and off. The 2N7000G MOSFET was selected because it can handle the required gate source voltage, has a relatively low on-resistance compared to the electromagnet and has a gate threshold voltage within the microcontroller's output voltage range. Additionally, a flyback diode was connected across the electromagnet to prevent the sudden voltage spike and damage to the MOSFET. This occurs because the solenoid creates a relatively high inductive load.

Microcontroller choice:

A microcontroller board was required to interface between MATLAB and the project's electronic components. The project required nine digital output pins, one of which has to be PWM capable and one analog input pin. The Texas Instruments mixed signal microcontroller (MSP) and Arduinos were shortlisted, both having advantages over each other. Arduino was chosen because there was a support package for MATLAB, making it easy to integrate the I/O results into the project's program without compiling. The 32-bit, 48 MHz arm cortex Arduino Zero was capable of driving more stepper motors simultaneously than the 8-bit, 16 MHz AVR Arduino Uno but the Uno was chosen based on price. Furthermore, with a longer project time the high-level functions which have unnecessary safety checks which take up valuable clock cycles could be rewritten. For example, the digital write function checks if the pin is a valid one and ends previous PWM outputs before writing a high or low value, which are processes that do not need to happen more than once.

Chess clock:

A chess clock consists of two clocks that keep track of a player's time. The purpose is to ensure that neither player overly delays the game and since their inception in 1883 many types of time limited games have been popularised, most notably Blitz [17]. For the scope of this project a chess clock is additionally being used to trigger the image captures needed for the computer vision software. A LEAP PQ9907S Chess Clock uses a reed switch to determine which side of the switch is active. An input wire is fed from the reed switch in series with a 1k ohm resistor into the microcontroller.

Computer science and Control systems:

This section will give a brief overview of the work done in the Autumn semester, and the progress made since then. As shown the interim report, an 'indication to the perceived occupancy' [3] is needed in order to deduce the chess algebraic notation for the chess engine. This requires knowing which squares are occupied and what colour the pieces that occupy them are. Computer Vision was used to detect the chess squares and Euclidean distance method was replaced with a simple convolutional neural network as it led to an improvement in accuracy.

Computer Vision

Square Detection

The first step of this process is to detect the squares of the chess board. This was done by removing the board of any chess pieces and using Harris corner detection, which assigns corners where vertical and horizontal gradients are simultaneously large. A square structuring element was used to erode any information smaller than a square, to avoid false corners from being detected around letters and numbers on the chess board. However, because the square structuring element is applied vertically the advantage

of the Harris corner detection being able to detect corners even when the board was rotated was no longer effective.

Figure 30 - Rotated RGB image

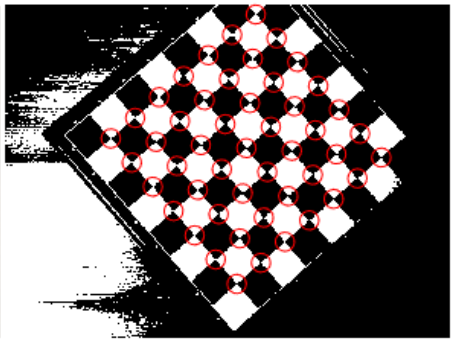


Figure 31 - Rotated without structuring element

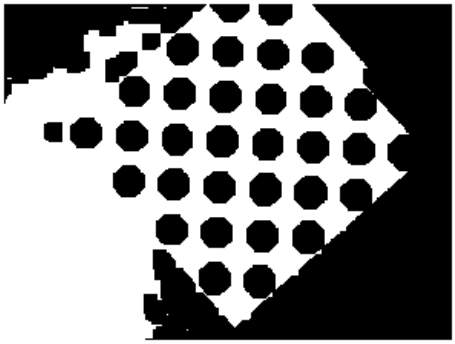


Figure 32 - Rotated with structuring element

Figure 33 - Parallel RGB image

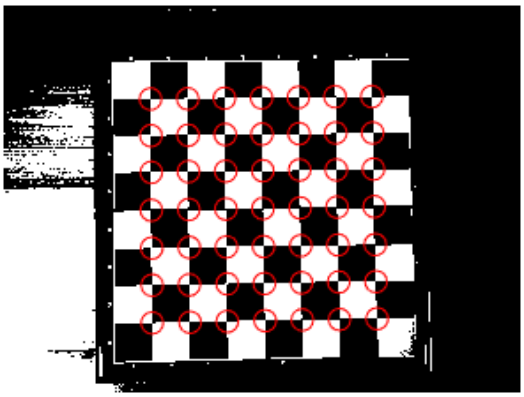


Figure 34 - Parallel with structuring element

It was not necessary to make the program redundant with large angles of the chess board, so the focus was put on accuracy instead. The next section of the of the program calculates the centres of the squares using the detected corners and estimated the centres of the 28 remaining squares. These centres were used in order to crop and label every individual square. The MATLAB scripts can be found in the appendix.

Square Classification using an Artificial Neural Network

The interim report uses a Euclidean distance equation with the CIELAB colour space to compare previous and current photos of all chess squares. This method was accurate when the chess pieces were placed in the centre of the square, however false positives occurred in adjacent squares when pieces were placed off centre.

Figure 35 - Intrusion onto an Empty Chess Square

One solution to fix this was prioritise pixels in the centres of squares as described [18]. Alternatively, it was decided to expand on the lab work of the Robotics and Machine Intelligence module and attempt to train a more complicated artificial neural network (ANN). The most challenging task in the laboratory was to program an ANN capable of representing an XOR gate, whilst the requirements for designing an optimised ANN for this project are significantly more difficult.
There are many variants of neural networks, most of which are derived from a multilayer perceptron (MLP) neural networks. This project could have used an MLP network, however this would have been

highly inefficient to train and the following subsection will explain how an MLP network works and why a 2-D convolutional layer neural network is more suitable for image recognition.

Multilayer perceptron (MLP) neural network

A neural network is remotely related to its biological counterpart, both are highly interconnected computational devices and the strength of connection between neurons determine the function of a neuron. A perceptron shown in figure 36, takes a scalar input p and multiplies it by a weight w and adds it to a bias b. This is referred to as the net input which goes through a activation function f, which outputs a scalar output a.

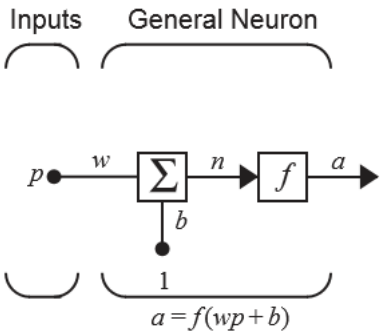


Figure 36 - Perceptron

Source: [19]

An MLP consists of several neurons, divided into several layers that can be described as matrices. 'Each layer has a weight matrix W, its own bias vector b, a net input vector n and a output vector a'.

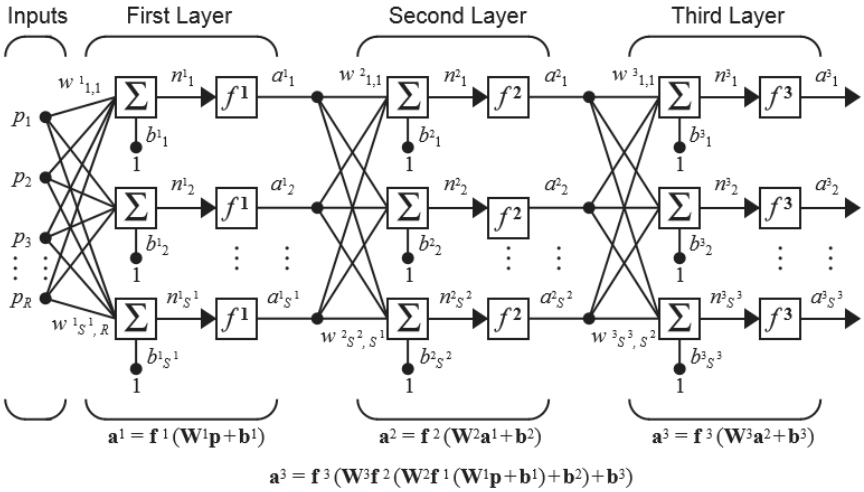


Figure 37 - Multilayer Perceptron Neural Network

Source: [19]

In a typical feedforward MLP the activation function is chosen by the designer and the weights and bias are refined during the training process. The number of neurons in the input layer for this project is determined by the resolution of the pictures, which are square photos with a width of 46 RGB pixels. The project's MLP network would require $3*46^2$ or 6,348 input neurons. The project requires a picture to be classified into one of three categories, therefore we require three output neurons. After being trained these output neurons describe the likelihood that any given test picture belongs to a category. Determining how many hidden layer neurons are required is still an active field of research however most MLPs have two or three hidden layers, as shown in chapter 11 of [19]. Additionally, this network is referred to as a

feedforward network because the output is computed directly form the input, it was chosen because it is an efficient type of network for pattern recognition.

To train this feedforward MLP, a backpropagation algorithm is often used. The first step of this process assigns random values to all the weights and biases. The net input vector *n* is fed forward resulting in an output vector *a*. A cost function is used to the output vector *a* to the desired output.

$$C_p = \frac{1}{2}\sum_j (t_j - a_j)^2$$

t_j – Desired outputs

a_j – actual output

C_p – Cost for a single output

The derivatives of this function with respect to the weights and biases are calculated using the chain rule, and it is therefore necessary to start with the last layer.

$$\frac{\partial C_p}{dw} = \frac{\partial net}{dw}\frac{\partial a}{\partial net}\frac{\partial C}{\partial a}$$

This process is then repeated with the previous layer until all the derivatives in all layers are computed. These derivatives represent how sensitive the cost function is to small changes in the weights and biases. In this project the cost represents whether any given picture is in the correct category (such as a red chess piece). By changing the weights and biases in the opposite direction of the gradient, the algorithm preforms gradient descent which slowly decreases the size of the cost, making it more likely an image is correctly categorised. Eventually the algorithm finds a minimum and depending on the number of neurons and hence the order of the cost function this minimum will either be a local or a global one. Using backpropagation between every layer would not be suitable for this project because of the very long training times. Assuming we use 3 hidden layers with 47 neurons and 3 output neurons.

Number of Weights: $3(46^2) + (46 * 46)^2 + 3 * 46 = 4{,}483{,}942$

Number of biases: $4 * 46 + 3 = 187$

Total of $4{,}484{,}129$ weights and biases

Even with techniques that accelerate convergence this would require significant computer processing power and time to find a local minimum and the network may memorise data from the training set but fail to correctly classify new images. Additionally, the activation function for the neurons would have to be limited to something differentiable such as the Log-Sigmoid function. Therefore, it is more suitable to use a 2-D convolutional layer neural network (ConvNet).

ConvNets usually contain three types of layers, convolutional layers, pooling layers and fully connected layers. A ConvNet is useful when the size of the input vector *n* is large. This is because an ANN made entirely of fully connected layers does not consider the relationship between space and pixels i.e., pixels that are close together in pictures are typically more correlated than pixels which are far apart. Therefore, because space matters, we only need to connect neurons to inputs which are close to those neurons.

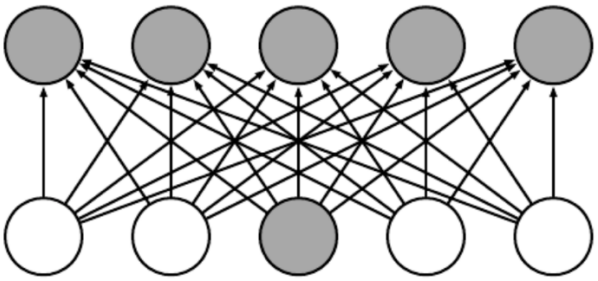


Figure 38 - Feedforward MLP neural network

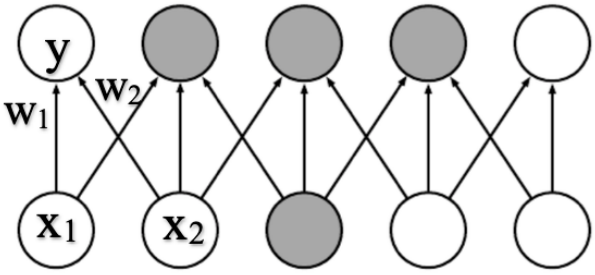


Figure 39 - Convolutional layer of a1-D ConvNet

Source: [20]

The remaining weights that go to each neuron are called filters because they only relate to a specific area of a picture and the weights w_1 and w_2 can be set to detect features.

$$y = w_1 x_1 + w_2 x_2$$

$$\text{If } (w_1, w_2) = (1, -1)$$
$$y = x_1 - x_2$$
$$0 \leq x_n \leq 1$$
$$\text{y is max when } (x_1, x_2) = (1,0)$$

This shows that weights can be configured so the output of y is at its maximum when a strong edge between two pixels is present. In the real-world pictures contain noise, which produce bigger differentials than the edges we are trying to detect. The solution is to average the pixel values which are close to each other using a smoothing filter. Then when we differentiate the edges can be easily detected. This process is called convolution. In this project we are dealing with 2-D images with 3 channels (red, green and blue), therefore we can use 2-D convolution because the ordering of the channels has no meaning with regards to space i.e., the filters are only moving in two dimensions of space. The convolution layer with 2 filters and a stride of 2 for a 5 by 5 picture can be seen in figure 40, where the input is represented in blue, the weights in red and the output in green. Additionally, padding was added around the border of the input picture, this keeps the dimension of the output the same size of the input (if the stride was 1).

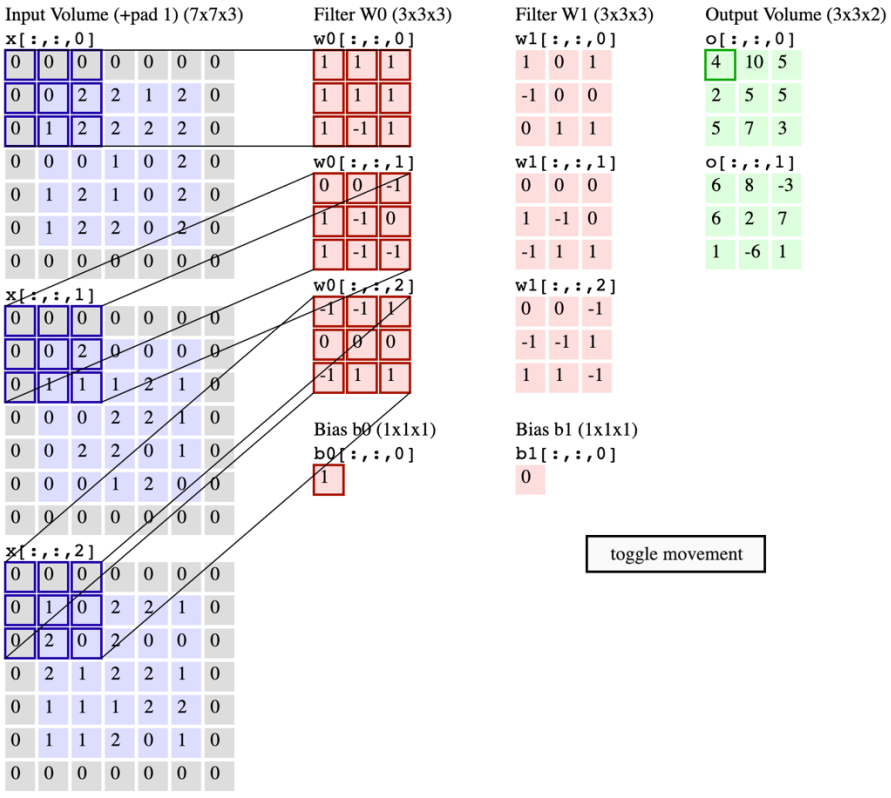


Figure 40 - Convolution Layer with 2 Filters

Source: [21]

As the filter is convolved over the input matrix, it produces a 2-D activation map that outputs the response of that filter at every spatial position. During training the filters' values in the first layer will change to be more responsive to features such as an edge or a spot of colour. If the project had several convolution layers, the higher layers of the network would be able to recognise higher level features. For example, if we were attempting to classify the type of chess piece without knowing its previous position on the board.

This project only requires the distinction low level features, such as if there is a high number of red pixels towards the centre of a given image. Therefore, the inputs which need to be set before the learning process known as hyperparameters can be set manually after experimentation. In state-of-the-art

applications additional programs use Bayesian optimisation to tune the hyperparameters, because a brute force method such as a grid search scales poorly [22]. Additionally, Bayesian optimisation uses prior outputs to search through the parameter space, whereas frequentist approach does not. For this project because the parameters were adjusted through human trial and error, it could be considered as a simple Bayesian optimisation.

It was decided to only experiment with one convolution layer as this will be able to accurately distinguish low level features. The first stage of this process was to amass enough image data. A single green and red chess piece were photographed on every chess square on the board and the computer vision software was used to automatically crop the images as shown in the appendix. The images were manually separated into 3 folders, empty, red chess piece and green chess piece. Within MATLAB the images are divided further into training and validation data sets. Each category in the training set contains 100 images whilst the validation sets contain 5 each. Next the size of the input layer was set to 46-by-46-by-3 with 3 classes. Finally, the net was trained with the default training options and hyperparameters were changed based on experimentation.

A filter size of 5 was found to produce accurate results, therefore the size of the zero-padding was set to 2 in order to keep the size of the output width and height the same as the input (for a stride of 1). This is important because within the training set, there were pictures which were labelled as empty even though they contained a small segment of a chess piece from an adjacent square shown in fig. This way a higher-level layer can be taught to prioritise features towards the centre of chess squares. The depth (number of filters) was experimented with and, ceteris paribus, it was found that at least 2 filters were required to give accurate results. The accuracy with only 1 filter usually oscillates around 83% after enough epochs whereas the 2 filters can achieve 100% accuracy.

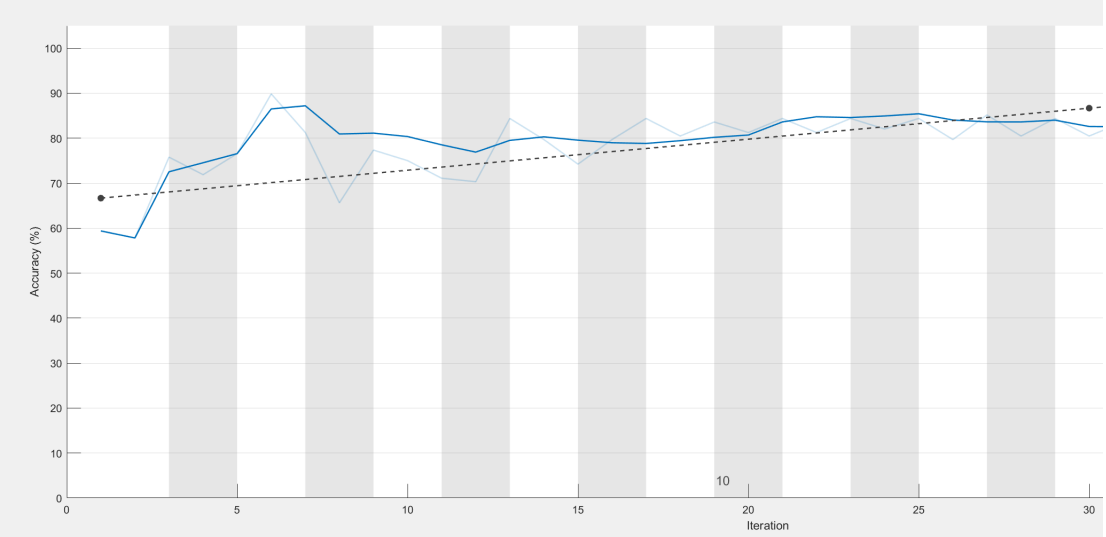


Figure 41 – Accuracy against Iterations for ConvNet with 1 Filter

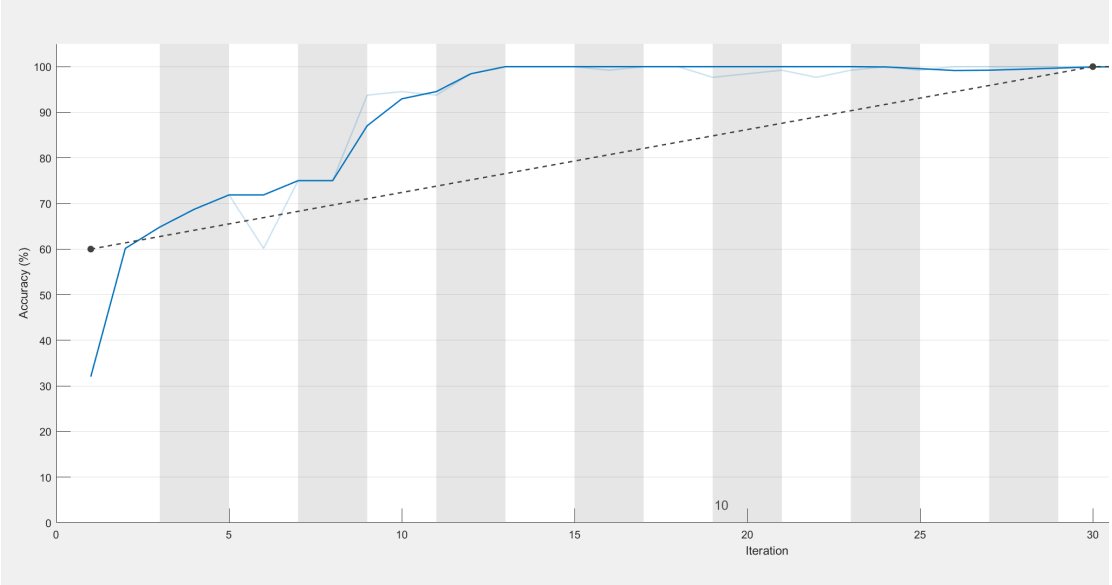


Figure 42 - Accuracy against Iterations for ConvNet with 2 Filters

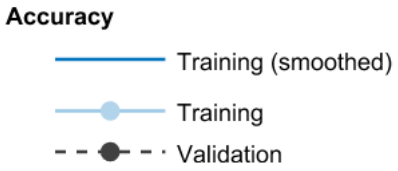


Figure 43 - Accuracy Legend

The stride was set to 1 to detect smaller features. The stride decides how the filter convolves around the input volume. A stride of 1 shifts the filter over the input one unit at a time. The stride is usually set at 1 or 2, with a stride of 2 producing a smaller output volume as seen in figure 40, known as downsampling.

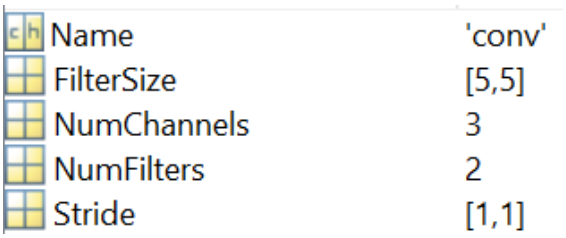


Figure 44 - Convolution Layer Hyperparameters

A batch normalisation layer is placed between the convolution layer and activation function layer in order to decrease the sensitivity to the initial weights and biases. This allows the net to be trained in a smaller number of epochs. When this layer was removed the accuracy was more likely to oscillate in comparison to when this layer was included, in other words this layer increases the stability as shown in [23].

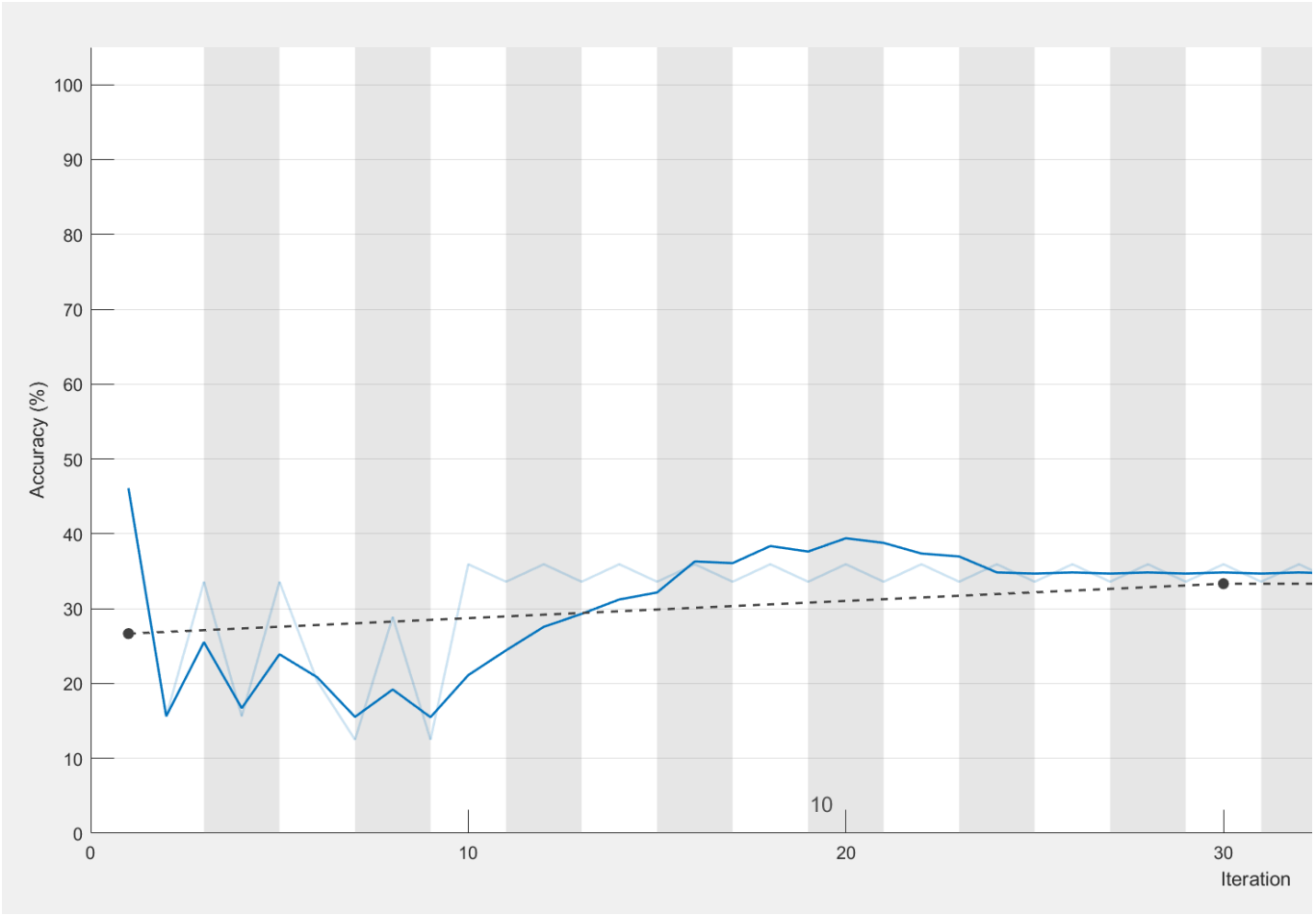


Figure 45 - Accuracy against Iterations for ConvNet without batch normalisation

The batch normalisation layer works by subtracting a batch mean from the output of the convolution layer and dividing by a batch standard deviation. The parameters for the mean and standard deviation are stored as separate variables instead of changing all the weights. The function to initialise the channel scale factors was kept at its default of 'ones'.

The Rectified Linear Units (ReLULayer) was selected as the activation function (layer). As mentioned earlier, whatever nonlinear function is chosen it must be differentiable.

$$\text{ReLULayer function} : f(x) \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The ReLULayer is one of the most computationally efficient to train because it sets all the negative activations to 0 which increases the nonlinear properties of the model as shown in [24].

The fully connected layer outputs a vector of size 3, representing the probability that an image's features belong to a class. The SoftmaxLayer normalises all the probabilities so that the probability of every class sums to 1. For example if the figure 35, might create an output of [0 0.1 0.9]. This means the net believes there is a 0% probability the square contains a red piece, 10% probability it contains a green piece and a 90% probability that the square is empty. The fully connected layer looks for features which strongly correlate to a given class. After being trained the weights and biases will have been adjusted so that when you multiply the weights with the previous layer you get the correct probability for each class. MATLAB allows you to visualise features that the network learns. Figure 46 shows the features that filters in the convolution layer learnt to respond to the red and green chess pieces towards the middle of the image, whilst the SoftmaxLayer represents the combination of these low-level features.

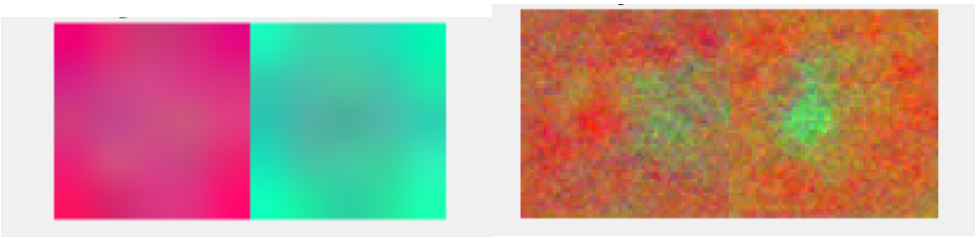


Figure 46 - Convolution Layer Features Figure 47 - Softmax Layer Features

After the ConvNet was trained additional images with different lighting conditions were tested and the net has yet to misidentify an image.

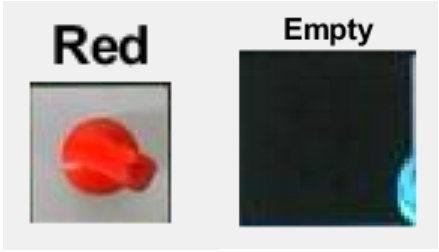


Figure 48 - Correct Identifications of Squares

Code overview

When the chess clock of the player is pressed, a photo is taken and divided and labelled into chess squares. The already trained neural net classifies all the squares and if any squares change classification this will be used to create the chess algebraic notation. This is fed into a chess engine, which then outputs a move for the robot to make. This is fed to the inverse kinematics program which is fed into the steps program which calculates how a piece should be moved. If a piece needs to be removed this will occur first. The last step is for the manipulator to press the chess clock and return to a position out of frame.

Chess engine

A chess engine is a program which analyses a chess move and returns a move. Stockfish was used in this project because the rating can be adjusted to match the level of any human player, it is open source and it works with the Universal Chess Interface (UCI) protocol. The MATLAB Add-On, Chess Master was then used to interface between Stockfish and portable game notation (PGN). Plain text format is desirable because the chess moves are given in algebraic chess notation, and the import format describes data that may have been prepared by hand, so it has some leniency on what it is able to detect. The movetext describes the actual moves and is given as Standard Algebraic Notation (SAN). SAN follows a few basic rules which are described in [25].

Project management issues and improvements

Although many of the objectives of this project have been achieved, the robot is still not able to repeatably displace chess pieces. An improvement in planning would have made more use of the University's 3D printer's during the second semester. This would have reduced the number of days between revisions in the design, as the University had a much smaller turnaround time in comparison to the online manufacturer. Additionally, the Udacity [26] course should have been moved to coincide with the neural network section of the Robotics and Machine Intelligence course.

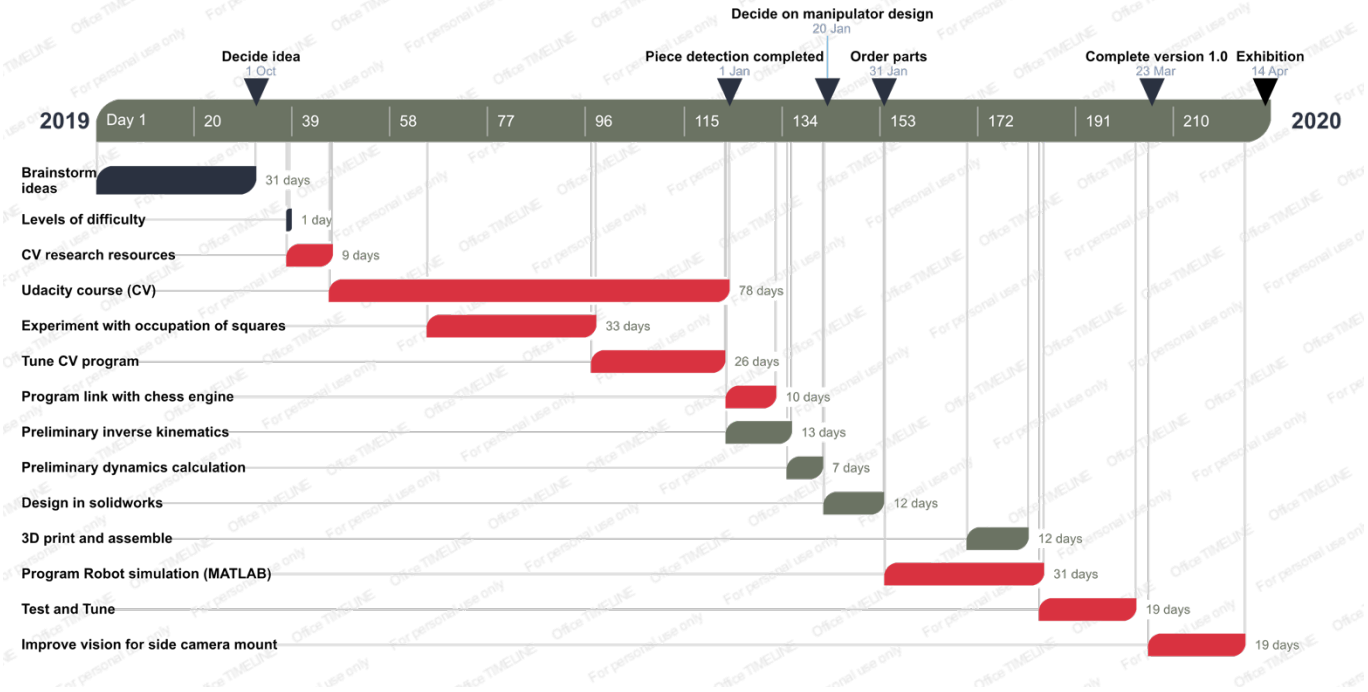


Figure 49 - Gantt Chart

The project assessment day has been set to the 9th of June and getting the manipulator to accurately and repeatably play a game of chess will be achievable as new parts arrive.

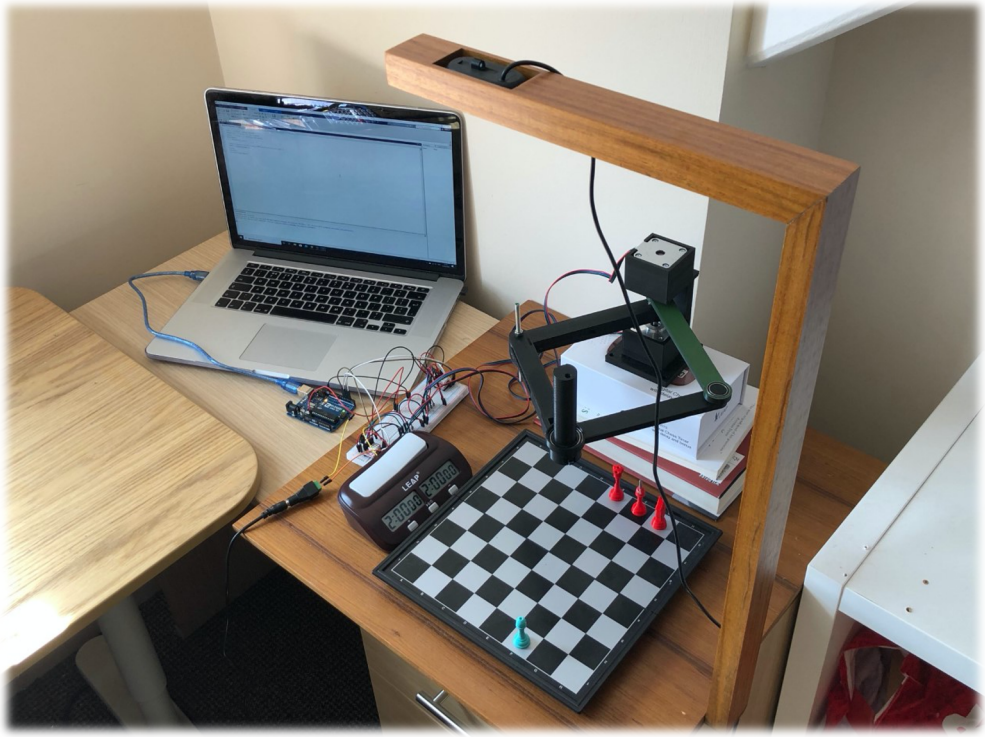


Figure 50 - Picture of Current assembly

Conclusion

This project has presented the design process for an autonomous chess playing robot capable of responding to moves in real-time. It can accurately track all moves that occur between 2 consecutive images from a mounted RGB webcam. The use of a ConvNet meant that lighting conditions did not affect the accuracy of classifications as much as the Euclidean distance approach, given a large enough image training set. Future iterations of this project's software could be exported onto a neural network compatible microcontroller in order to allow for a portable, battery-operated version. The camera could be integrated into the unit, and more complex computer vision techniques would be used to deal with the occlusion and distortion problems. Additionally, with a second unit, both human players would have the option to play on physical boards.

Bibliography

[1] F. I. d. Échecs, “Handbook,” 1 March 2020. [Online]. Available: https://handbook.fide.com/.

[2] Chess.com, “Chess.com Members,” 22 November 2017. [Online]. Available: https://www.chess.com/article/view/how-many-chess-players-are-there-in-the-world.

[3] S. Donald, “A System for Live Analysis of Chess Gameplay and Feedback from a Chess Engine,” University of Canterbury, Christchurch, 2018.

[4] H. Maki K and J. P. Davim, Interdisciplinary mechatronics : engineering science and research development, London: ISTE, 2013.

[5] S. A , R. Muhammad, N. L. Thomas and V. V. Silberschmidt, “Mechanical and thermal characterisation of poly (l-lactide) composites reinforced with hemp fibres,” Journal of Physics, 2013.

[6] robotshop.com, “Large Robot Gripper RB-Dfr-358”.

[7] P. P. Company, “Writing Instrument”. USA Patent 3205863, 14 9 1965.

[8] B. Dutta, S. Babu and B. . H. Jared, Science, Technology and Applications of Metals in Additive Manufacturing, ELSEVIER, 2019.

[9] N. J. Mardis, “Emerging Technology and Applications of 3D Printing in the Medical Field,” Missouri Medicine, 2018 .

[10] “American National Standard for Industrial Robots and Robot Systems,” ANSI, 1990.

[11] A. Goswami, A. Quaid and M. Peshkin, “Complete parameter identification of a robot from partial pose information,” IEEE, Atlanta, GA, USA, 1993.

[12] B. N. A. Technology, “Application Note DK9222-0410-0014,” Beckhoff, Verl, Germany.

[13] D. W. Jones , Stepping Motors, THE UNIVERSITY OF IOWA Department of Computer Science, 1995.

[14] S. M. Tobin, DC Servos: Application and Design with MATLAB, CRC Press, 2017.

[15] Signetics, “NE544 Servo Amplifier Data Sheet”. 28 January 1988.

[16] STMicroelectronics, “L298 datasheet”. 2000.

[17] T. F. I. d. Échecs, “FIDE LAWS of CHESS,” Dresden, Germany, 2008.

[18] C. Koray and E. Sumer, “A Computer Vision System for Chess Game Tracking,” Department of Computer Engineering Baskent University , Ankara, 2016.

[19] M. T. Hagan, H. B. Demuth and M. H. Beale, Neural Network Design, Frisco, Texas: Martin Hagan, 2014.

[20] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Cambridge: MIT Press, 2016, pp. 326-335.

[21] S. University, “Convolutional Neural Networks for Visual Recognition (Notes from Stanford CS231 class),” Stanford University, Stanford, California.

[22] P. Murugan, “Hyperparameters Optimization in Deep Convolutional Neural Network / Bayesian Approach with Gaussian Process Prior,” Nanyang Technological University, Singapore, 2017.

[23] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Cornell University, Ithaca, New York, 2015.

[24] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” University of Toronto, Toronto, 2010.

[25] N. Sfetcu, The Game of Chess Digital, Lulu.com, 2011.

[26] Udacity, “Udacity,” 2019. [Online]. Available: https://www.udacity.com/course/introduction-to-computer-vision--ud810.

Appendices

MATLAB script written in order to find the coordinates of the chess squares:

```
clc;    % Clear command window
close all;  % Close all the figures
clear;  % Erase all existing variables
workspace;
%%
cam = webcam('Logitech');
I = snapshot(cam); %Takes picture of board
img = imrotate(I,180,'bilinear');  %Rotates image 180
[rows, columns, numberOfColorChannels] = size(img); %notes dimensions of picture
%%
gray = rgb2gray(img);

%%
level = 0.5;
Ithresh = im2bw(gray,level);
imshowpair(img,Ithresh,'montage');


%%
se = strel('square',20);
Iclosed =imclose(Ithresh,se);
```

It is then passed through Harris corner detection and another function which finds the midpoints of the squares:

```
function s = midPoints(p,q)

p1x = p(1);
p1y = p(2);

p2x = q(1);
p2y = q(2);

midx = (p1x + p2x)/2;
midy = (p1y + p2y)/2;

s.midPointx = midx;
s.midPointy = midy;
s.Pointx = p1x;
s.Pointy = p1y;
s.Distx = abs(((p1x + p2x)/2) - p1x);
s.Disty = abs(((p1y + p2y)/2) - p1y);

```

These coordinates are used to capture images in order the train the CovNet:

```
clc;    % Clear command window
close all;  % Close all the figures
clear;  % Erase all existing variables
workspace;
%%


cam = webcam('Logitech');

I = snapshot(cam); %Takes picture of board
img = imrotate(I,180,'bilinear');  %Rotates image 180
[rows, columns, numberOfColorChannels] = size(img); %notes dimensions of picture

b = 0;
%% Saves 46 by 46 images used to train the ConvNet
load('C:/Users/Pierre/Documents/MATLAB/Detection/points.mat') % Load coordinates of the chess square corners


for a = 1:41
    Z = imcrop(img,[s(a).Pointx s(a).Pointy 45 45]);
baseFileName = sprintf('%d.jpg',a + b);
fullFileName = fullfile('C:/Users/Pierre/Documents/MATLAB/Photos', baseFileName);
imwrite(Z, fullFileName);

end
b = b + 41;
```

The CovNet consits of 7 layers:

1x1 ImageInputLayer
1x1 Convolution2DLayer
1x1 BatchNormalizationLayer
1x1 ReLULayer
1x1 FullyConnectedLayer
1x1 SoftmaxLayer
1x1 ClassificationOutputLayer

A MATLAB script then takes the coordinates derived from the output of the chess engine and decreases the increments between coordinates, creating smooth coordinates:

```
while i < rawsize(1,1)

  while  (pdist([proc(end,:);raw(i,:)],'euclidean') > resolution || pdist([proc(end,:);proc(end-1,:)],'euclidean') > resolution)

    while pdist([proc(end,:);proc(end-1,:)],'euclidean') > resolution
    x1=proc(end-1,1);
    y1=proc(end-1,2);
    x2=proc(end,1);
    y2=proc(end,2);
    curr = [x1,y1;x2,y2];
    mp = [(x1+x2)/2, (y1+y2)/2];
    proc(end,:)=[];
    proc = [proc;mp];
    end
    proc = [proc;raw(i,:)];
  end
  i=i+1;
  proc = [proc;raw(i,:)];
end

```

coordinates.csv

	A	B
	coordinates	
	x	**y**
	Number	Number
1	x	y
2	0	0
3	-110	40
4	110	40
5	110	260
6	-110	260
7	0	230

	A	B
	smoothcoordinates	
	VarName1	**VarName2**
	Number	Number
1	0	0
2	-3.4375	1.25
3	-6.7676	2.4609
4	-9.9936	3.634
5	-13.119	4.7705
6	-16.146	5.8714
7	-19.079	6.9379
8	-21.921	7.9711
9	-24.673	8.972
10	-27.339	9.9416
11	-29.923	10.881
12	-32.425	11.791
13	-34.849	12.672
14	-39.546	14.38
15	-43.95	15.982
16	-48.078	17.483
17	-51.948	18.89
18	-55.576	20.209
19	-58.978	21.446
20	-62.166	22.606
21	-65.156	23.693
22	-67.959	24.712
23	-70.586	25.668
24	-73.05	26.564
25	-77.669	28.243
26	-81.71	29.713
27	-85.246	30.999
28	-88.34	32.124
29	-91.048	33.108
30	-93.417	33.97
31	-97.563	35.477
32	-100.67	36.608
33	-105.34	38.304
34	-110	40

MATLAB script written in order to simulate the inverse kinematics for the design with 6 links and 6 joints for a single coordinate:

```
clc;    % Clear command window
close all;  % Close all the figures
clear;  % Erase all existing variables
workspace;
%Specify arm lengths
length_a = 150;
length_b = 150;
length_c = 150;
length_d = 150;



%%
%Specify steppermotor coordinates
r0_x = 0;
r0_y = 0;

r1_x = 0;
r1_y = 0;

%Specify end effector coordinate
x = 115;
y = 40;

angle_b = CALC_ANG_b(x,y,length_a,length_b);
angle_d = CALC_ANG_d(x,y,length_c,length_d);

angle_a = CALC_ANG_a(x,y,length_a,length_b,-angle_b);
angle_c = CALC_ANG_c(x,y,length_c,length_d,angle_d);

[D_x,D_y,B_x,B_y] = JOINT_POS(length_c,angle_c,length_a,angle_a);


plot([r0_x B_x x], [r0_y B_y y]);
hold on
plot([r1_x D_x x], [r1_y D_y y]);

axis equal


```

The inverse kinematic formulae:

```
function [angle_a] = CALC_ANG_a(x,y,length_a,length_b,angle_b)
%CALC_ANG_a Preforms inverse kinematics

angle_a = (atan(y/x)) + (atan((length_b * sin(angle_b))/(length_a + (length_b*cos(angle_b)))));


end
```

```
function [angle_b] = CALC_ANG_b(x,y,length_a,length_b)
%CALC_ANG_b Preforms inverse kinematics

angle_b = -acos((x^2 + y^2 - length_a^2 - length_b^2)/(2*length_a*length_b));


end

```

```
function [angle_c] = CALC_ANG_c(x,y,length_c,length_d,angle_d)
%CALC_ANG_c Preforms inverse kinematics

angle_c = (atan(y/x)) - (atan((length_d * sin(angle_d))/(length_c + (length_d*cos(angle_d)))));


end
```

```
function [angle_d] = CALC_ANG_d(x,y,length_c,length_d)
%CALC_ANG_d Preforms inverse kinematics

angle_d = acos((x^2 + y^2 - length_c^2 - length_d^2)/(2*length_c*length_d));


end
```

```
function [D_x,D_y,B_x,B_y] = JOINT_POS(length_c,angle_c,length_a,angle_a)
%JOINT_POS Calculates position of joints 2 and 4
D_x = length_c * cos(angle_c);
D_y = length_c * sin(angle_c);

B_x = length_a * cos(angle_a);
B_y = length_a * sin(angle_a);


end
```

Example plot:

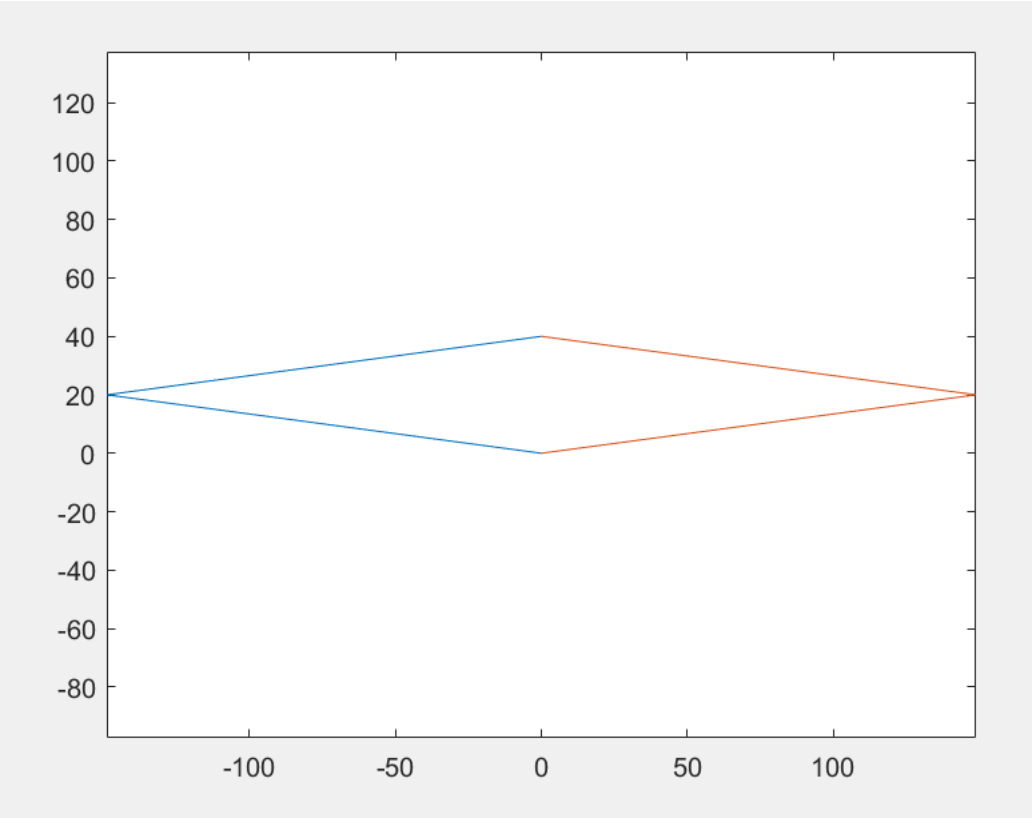


Figure 51 – MATLAB Plot of End Effector between the near King and Queen pieces

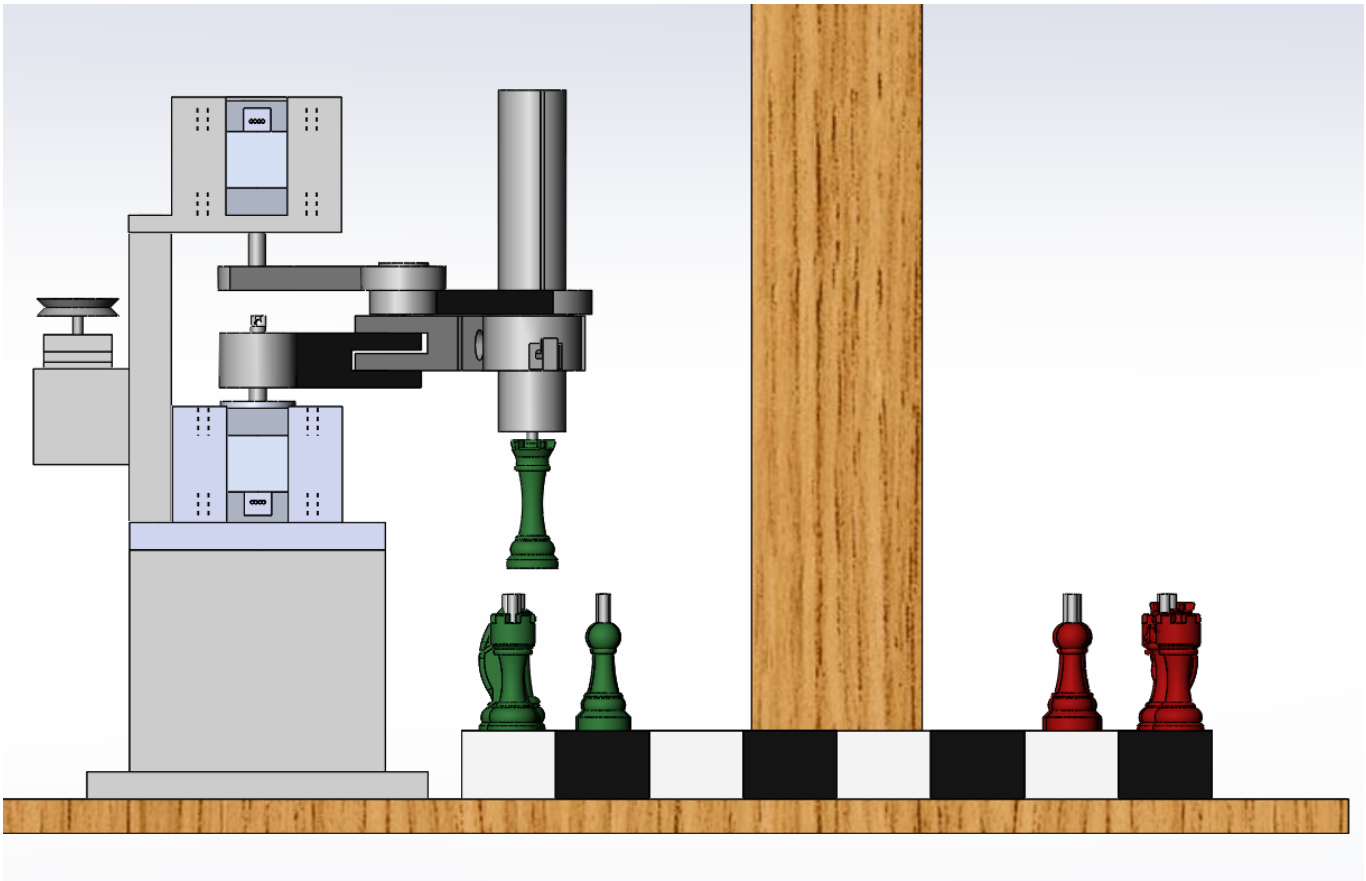


Figure 52 - SolidWorks Plot of End Effector between the near King and Queen pieces

The inverse kinematic script is inserted into a *for loop* which outputs the angles for the two stepper motors. This is then converted into steps:

	A	B
	Angles	
	VarName1	VarName2
	Number	Number
1	0	0
2	1	-2
3	3	-3
4	4	-5
5	6	-6
6	7	-7
7	8	-9
8	10	-10
9	11	-11
10	12	-12
11	13	-13
12	14.0	-14
13	16	-17
14	18	-19
15	20	-20
16	22	-22
17	24	-24
18	25	-25
19	27	-27
20	28	-28
21	29	-30
22	31	-31
23	32	-32
24	34	-34
25	36	-36
26	38	-38
27	39	-39
28	40	-40
29	41	-42
30	43	-44
31	45	-45
32	47	-47
33	49	-50
34	49	-47

	A	B
	Steps	
	VarName1	VarName2
	Number	Number
1	0	0
2	0	0
3	31.831	-63.662
4	63.662	-31.831
5	31.831	-63.662
6	63.662	-31.831
7	31.831	-31.831
8	31.831	-63.662
9	63.662	-31.831
10	31.831	-31.831
11	31.831	-31.831
12	31.831	-31.831
13	31.831	-31.831
14	63.662	-95.493
15	63.662	-63.662
16	63.662	-31.831
17	63.662	-63.662
18	63.662	-63.662
19	31.831	-31.831
20	63.662	-63.662
21	31.831	-31.831
22	31.831	-63.662
23	63.662	-31.831
24	31.831	-31.831
25	63.662	-63.662
26	63.662	-63.662
27	63.662	-63.662
28	31.831	-31.831
29	31.831	-31.831
30	31.831	-63.662
31	63.662	-63.662
32	63.662	-31.831
33	63.662	-63.662
34	63.662	-95.493

```
angles = csvread('Angles.csv',0,0);
angles = [[0,0];angles];

stepSize = 1.8;

absoluteSteps = angles/(deg2rad(stepSize));
relativeSteps = [0,0];

for k = 1:(length(angles))-1;
    relativeSteps = [relativeSteps;(absoluteSteps((k+1),:) - absoluteSteps(k,:))];
    disp(k)
end



csvwrite('Steps.csv',relativeSteps);
```

MATLAB script to step the stepper motor:

```
steps = csvread('Steps.csv',0,0);
steps = [[0,0];steps];

col1 = steps(:,1);
col2 = steps(:,2);

a = arduino('COM3','Uno');



a.writeDigitalPin('D4', 0);

for i = 1:(length(col1))-1;
y = steps(i,1);
 for x = 0:y
a.writeDigitalPin('D3', 1);
pause(0.02);
a.writeDigitalPin('D3', 0);
pause(0.02);
 end
end

a.writeDigitalPin('D4', 1);

for i = 1:(length(col2))-1;
y = steps(i,2);
 for x = 0:y
a.writeDigitalPin('D5', 1);
pause(0.02);
a.writeDigitalPin('D5', 0);
pause(0.02);
 end
end

```

The arm would then proceed to step: