UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Project and Professionalism

(6CS020)

A1: Project Proposal

Student Id	: 2049822
Student Name	: Sashank Dulal
Supervisor	: Sangay Lama

Submitted on	: Tuesday, November 16,2021

Acknowledgements

My sincere gratitude goes towards University of Wolverhampton and Herald College Kathmandu for this programme and course availability. Secondly, I would like to thank module leaders Mr. Nirmal Thapa and Mr. Biraj Dulal for such experience regarding the project and for much needed guidance. Finally, special thanks to our supervisor Mr. Sangay Lama for his time and effort to make this project better with his knowledge and support.

Title and Declaration Sheet

Abstract

Table of Contents

Table of Figures

Introduction

Chess is a 2-player strategy game played on an 8x8 board with 16 pieces for each player. With the aim of trapping the opponent's king piece (Also called "Checkmate"), the players devise their own strategies and moves to get to that aim. Since the invention of chess as a form of a mind game, people have been trying to assert their dominance in terms of mental abilities through chess (Chess.com, 2021). And with the advent of chess theories, the game have become more and more difficult for a normal person to understand. From a business standpoint, the global chess market has been generally rising in most of the countries. The North American industry is expected to hit 40 million US dollars by 2022. These all statistics point to the fact that the equipment and resources related to chess are on demand (Statista.com, 2021).

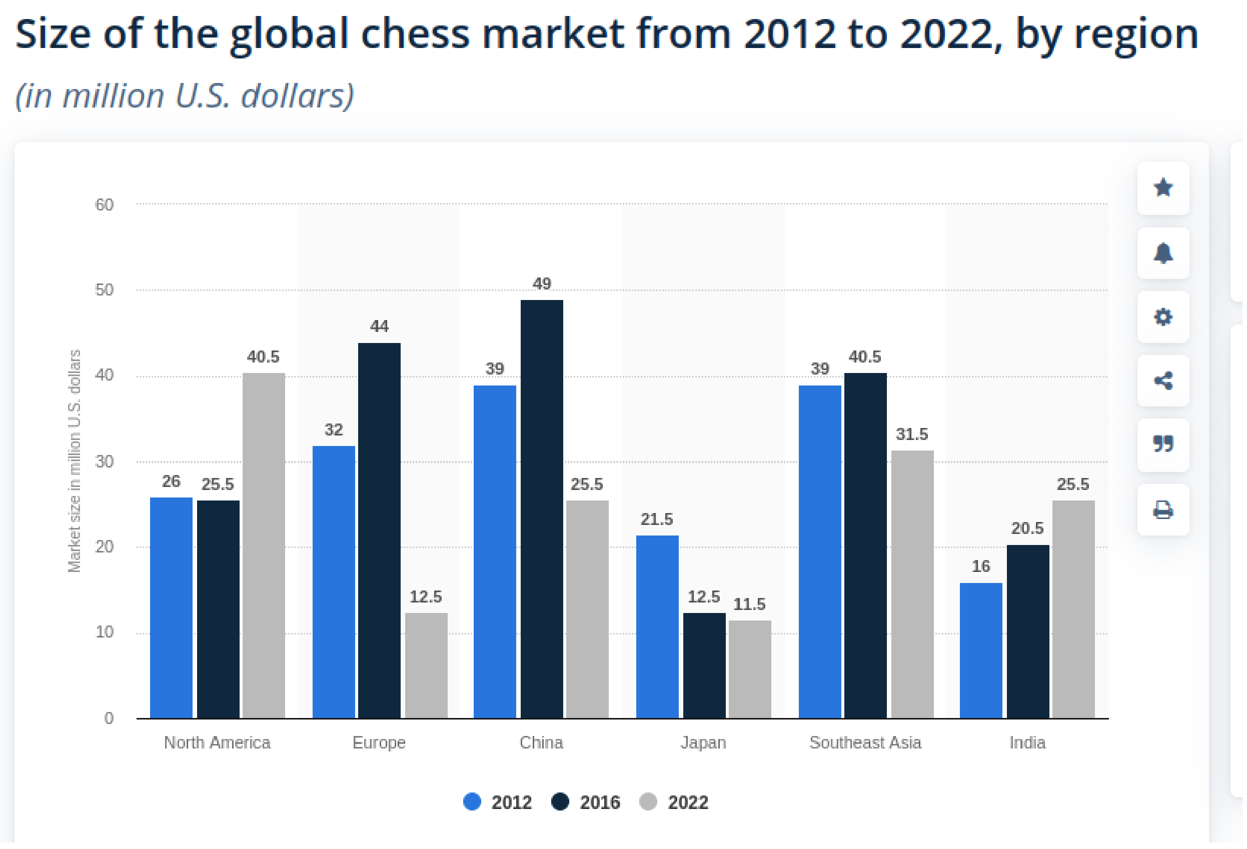


Figure 1: Global Chess Market Size Comparison from 2012 to 2022 (Statista.com, 2021)

With the increase in the whole industry, the availability of intelligent and specialized equipment for chess is very low. Similarly, the published resources are also vague to understand and expensive to manufacture. The main objective of this research is to perform an in-depth analysis on implementation of Artificial Intelligence (AI) to detect and recognize chess moves, analyze different methods to physically move the chess pieces with the help of a robotic system and determine the best way to move the pieces based on the analysis of different aspects of those methods and then construct the robot based on the research findings. At the end of this research, a fully functioning robot capable of playing chess against a user will be constructed with the help of the research materials and findings.

Problem Domain

I. No Proper method of training by playing physical match for newcomers

   If a newcomer wants to gain proper experience at chess, he/she have to physically play that game and since they are not well-trained, playing with another professional can be discouraging for them and it is difficult to find the person who agrees to play with an unexperienced player

II. Costly trainings and coaches

   In order to get good at chess, proper training and regular practice is required and the training programs and coaches are expensive to maintain.

Project as a Solution

I. Even the newest inexperienced players get to play with the standard opponent.

II. The cost of getting trained drastically decreases as it is a one-time investment. Once the bot has been bought, users can play whenever they like. Similarly, the bot would be constructed out of locally-sourced parts, repairing and even upgrading would not be difficult or expensive.

Aims and Objectives

Aims

- Investigate the feasibility of different robotic methods for movement of small objects in a confined space.
- Perform research on different algorithms used for moving the robots within a fixed area.
- Construct a robot using research findings that will be able to play chess against a user opponent.

Objectives

- A robotic system will be designed and developed using the analysis of research findings.
- Implement different image processing algorithms to detect and track movements.
- The robot will be made in such a way that it will be communicating with custom made/ open-source API that can generate chess moves for any given move of the user.

Artefacts to be developed

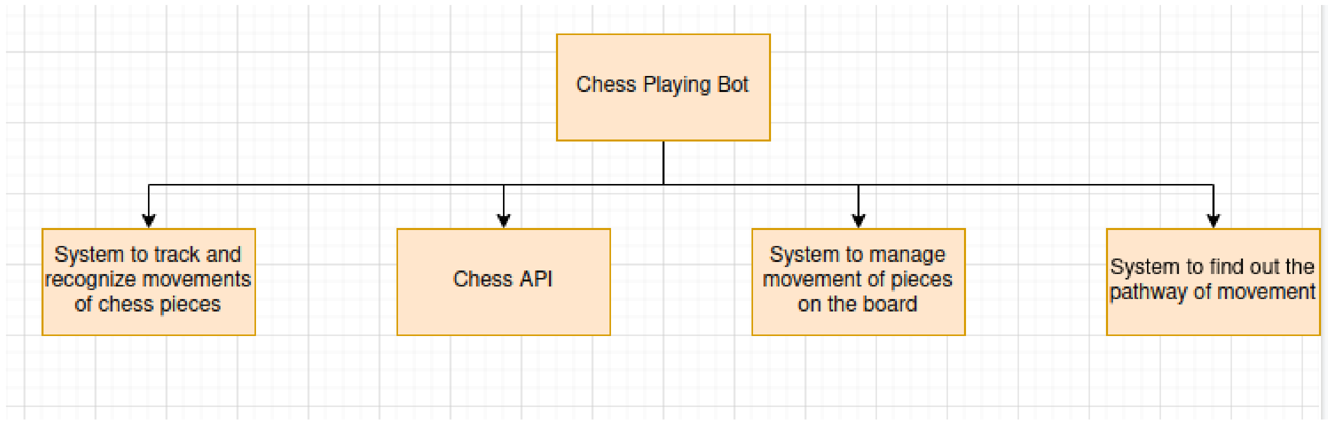

Chess Playing Bot
System to track and recognize movements of chess pieces
Chess API
System to manage movement of pieces on the board
System to find out the pathway of movement

Figure 2: Functional Decomposition Diagram (FDD) of the proposed system

- System to detect movements of chess pieces placed by the user on the board.
  This system is responsible for tracking the position of chess pieces on the board and detecting the moves of the user.

- Chess API

  This system generates best possible moves based on the moves done by the opponent user.

- System for movement of pieces in the board.

  This system takes in the recommendations made by the chess engine to physically move the chess pieces on the chess board.

- System to find out the pathway of the movement.

  This system finds out the optimal pathway for the chess piece to move from one place to another.

Academic Question

- How will the bot get the best possible movement for the move produced by the user?
- How will the chess pieces be moved using the arm?
- What will be used to track the movement of chess pieces?

Literature Review

1.1.1 Gambit: An Autonomous Chess-Playing Robotic System

This paper presents a 6-DoF chess playing robot system named Gambit which is capable of playing chess on a physical board against humans. It includes a low-cost sensor for perception, custom-made robot arm and learning algorithms for detection and recognition of objects on the board. The authors of the project mainly view this project as a way of exploring the perception and manipulation in a “noisy, less constrained real-world environment”. Gambit does not require the chess pieces to be exactly modelled or instrumented as it monitors the state of the board and tracks the movement of the pieces that the opponent has made and communicates with the human opponent using spoken-language interface.

The perception system tracks the position of pieces on the board and the board is not fixed relative to the board and the board is calibrated continuously throughout the game. On the other hand, the system makes use of an open-source arm design as it was moderate in cost (about $18000 for parts) and their major aim was to enable smooth motion and interaction. It consisted of a 6-DoF arm with a gripper attached ad one end. As shown in figure. 1, the DoFs 1, 2 and 3 are used for positional control whereas 4, 5 and 6 are used for orientation control with the help of roll-pitch-roll spherical wrist.

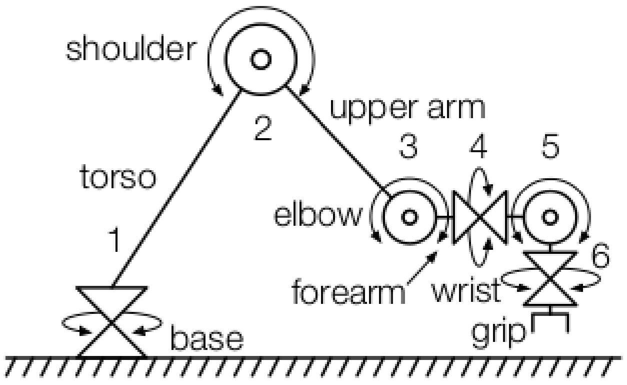


Figure 3: Schematic of the arm construction (Matuszek, et al., 2011)

For sensing purposes, the system consisted of a shoulder-mounted depth-sensing camera (similar to Xbox Kinect) along with a camera attached to the gripper. The depth-sensing camera provided color information along with depth of each pixel and worked within a range of 0.5m to 5m. Since the minimum working range was very high, it presented a problem that the camera had to be mounted high but the authors solved this issue by mounting the camera facing backwards of the torso of the arm so that the distance is increased.

The driver software for the system ran on a dedicated Intel Atom net-top PC with CAN and RS-485 PCI cards whereas controlling of the arm was done using a separate computer that handled ROS operations. And the system for detection and recognition of board consisted of four hierarchical classifiers mainly for detecting squares, pieces/backgrounds and two for recognition of types of chess pieces. Each piece was labelled out of pre-defined pieces {B, N, K, P, Q, and R}. The hierarchy of the vision algorithm is depicted in the figure below.

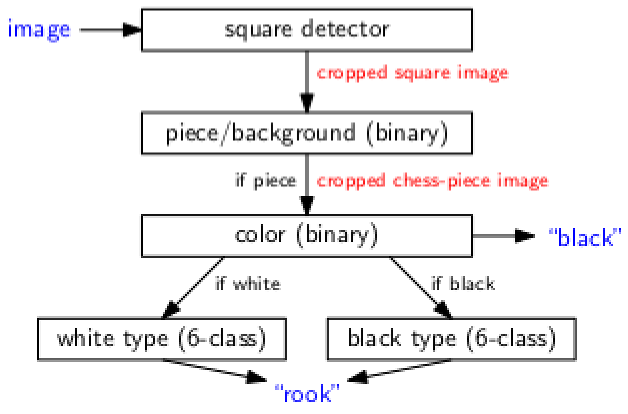


Figure 4: Hierarchy of the chess-piece recognition algorithms (Matuszek, et al., 2011)

(Matuszek, et al., 2011)

1.2 MarineBlue: A Low-cost Chess Robot.

This paper focuses on the details of a chess robot named MarineBlue. Emphasis has been given on the algorithms involving computer vision and robot manipulation in this paper. MarineBlue is an autonomous robot that can recognize moves placed by the user, generate moves for that scenario and move that particular piece with the help of an arm. Development of a compact robot on low cost was prioritized in this paper.

During the construction of the system, many factors such as chess board and construction of chess pieces were considered such that it could be easier to replicate. First the authors had thought of making the chess pieces uniform in dimensions in order to make it easier to move them but in the end, the gripper was constructed in such a way that it could pick and move every type of chess pieces. On the other hand, the chess board had to be modified in such a way that the length of the squares was 30 millimeters. Similarly, disjunctive colors were chosen to make sure that the algorithms detect which squares were occupied and track the objects.

For the vision side of the project, a Sony DFW-VL500 camera was used to gather high quality images and it was mounted 1 meter high above the chessboard. Next, for the movement of chess pieces, a Robix RC-6, a portable and highly configurable robot was used. The robot consisted of a set of separate units that could be put together to form a bigger unit. The units could move in conjunction to other units to form one smooth motion with the help of servos. The controllers controlled the servos and the controllers is linked to the parallel port of the chess computer and the servo controller gets the servo commands from the computer in the form of Robix-dependent scripting language. In order to make the robot sway less because of the heavy end, the gripper was made lighter and a degree of freedom (rotation) was removed from the gripper.Similarly, the gripper was made longer so that it would not collide with other pieces when a piece is being picked up and the gripper was made hemispherical to accommodate for small positioning errors.

For handling chess gameplay, an application that analyzes the situation of the board, recommend moves and execute them was developed in C/C++ using MS Visual Studio and was running on Windows 2000 computer.

To detect the current game situations, the video feed was processed through three layers i.e. pixel classification layer, board layer and chessboard layer. The pixel classification layer determines the class of every pixels according to the color properties of the pixels. There were four defined classes: light square, dark square, light piece and dark piece. Since the white piece on white square could not be easily recognized by the algorithm, the authors used hue, saturation and brightness color space instead of RGB space.

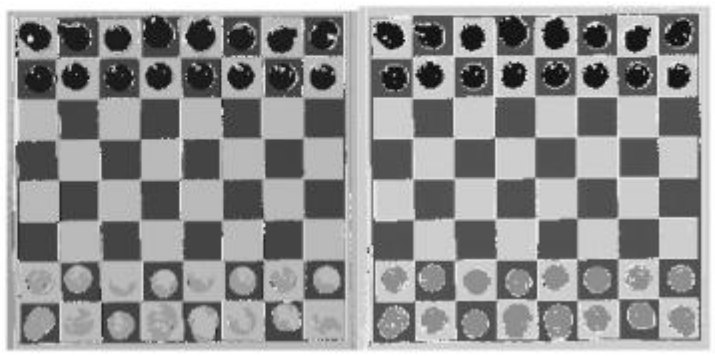


Figure 5: Comparison between RGB Space (on the right) and HSB (on the left)

Then the board layer uses the matrix containing the states of each pixel classes acquired from the previous stage to determine the position of board and pieces inside image and to find out which squares are occupied by the pieces or not. A reference image containing the color of dark square is used and the algorithm searches or the corners of the image by searching the ends of the pixels relating to the dark square category. Then interpolation is used to calculate the position f each square in the image. After that, the number of pixels in the classes “dark piece” and “light piece” is calculated and if the calculated number crosses predefined limit, then the algorithm declares that the square is filled. The ratio of surface area of the piece and the surface area of the square determines the threshold value. The chessboard layer is then fed with the output of previous stage: the list of squares occupied by a chess pieces of a color. This layer also keeps tracks of previous data which can be used to track the movements easily.

To compute the angle for the servos based on the destination of the gripper and the arm configuration, inverse kinematics was used.

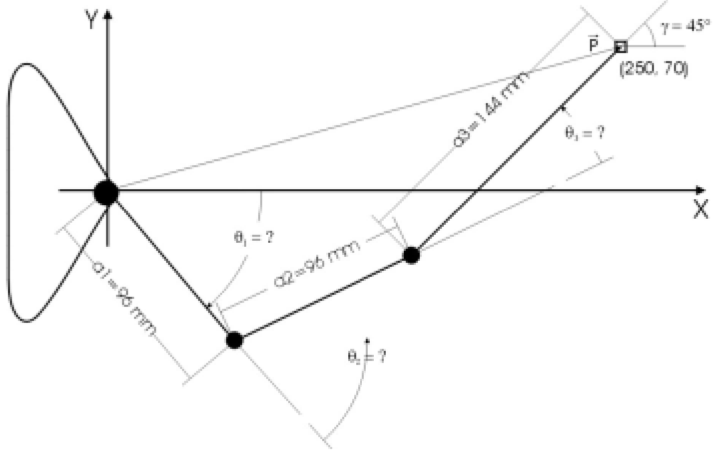


Figure 6: Top view of the Inverse Kinematics for the arm

Since solving the inverse kinematics involve heavy computational power, it is not practical to solve the algorithm for more than six joints. For this paper, the analytical approach was considered along with an assumption that the fourth segment was always in line with the third segment was made.

3.1: $\gamma = \theta_1 + \theta_2 + \theta_3$
$P = a1 + a2 + a3$

The second formula can be rewritten as formula 3.2:

$$x = a_1 c_1 + a_2 c_{12} + a_3 c_{123}$$
$$y = a_1 s_1 + a_2 s_{12} + a_3 s_{123}$$

with

$$c_1 = \cos(\theta_1) \qquad s_1 = \sin(\theta_1)$$
$$c_{12} = \cos(\theta_1 + \theta_2) \qquad s_{12} = \sin(\theta_1 + \theta_2)$$
$$c_{123} = \cos(\theta_1 + \theta_2 + \theta_3) \qquad s_{123} = \sin(\theta_1 + \theta_1 + \theta_1)$$

Figure 7: Formula for the inverse kinematics

When the destination for gripper position (x,y, γ) is given, the objective is now to find out the three theta values by solving the above system of equations. As there can be infinite number of solutions for the equations if the γ is considered not important, the method used increases the angle γ in small increments and calculate two solutions to this angle if those solutions exist. After that, the best solution that requires the least movement for the arm is chosen.

(URTING & BERBERS, 2003).

1.3 Multi-Object Recognition and Tracking with Automated Image Annotation for Big Data Based Video Surveillance

In this paper, an improved region based scalable convolution neural network (IRS-CNN) based multiple object tracking model has been discussed. It uses Automatic Image Annotation (AIA) in order to improve the detection capacity and reduce computation time.

In the IRS-CNN method, the anomaly detection takes place in three stages namely AIA model to annotate images, RPN(Region Proposal Networks) and R-CNN. During the first stage, the image annotations are generated. Then a CNN creates regions in the image and finally at the last stage, the regions created in previous stage is used to detect the anomalies in the image. The IRS-CNN model is significant in size and consists of smaller sized sub-networks to detect anomalies. Similarly, the scaling awareness with AIA enables the IRS-CNN model to detect anomalies efficiently no matter the size and distribution.

Furthermore, CNN with WARP (Weighted Approximated Ranking) method was used to improve the time complexity of the annotation process. Five convolutional layers with 3 connected layers was used from CNN model. Similarly, the Region Proposal Networks

(RPN) was made in such a way that it inputs images with varying dimensions and outputs groups of rectangular objects as proposals with their objectless values. In order to facilitate faster resource sharing with R-CNN, each network was made to share similar convolution layers and for region proposal, a smaller network gets an input of n*n spatial window where the sliding windows get mapped to lesser dimensional feature set. To train the RPN, a binary label (normal or abnormal) for each of the anchors (rank of every region box) was assigned.

Hence the paper concluded that the proposed IRS-CNN based MOT system was better than the existing RS-CNN method with the use of AIA to increase efficiency in detection and decrease computation complexity.

(Vijiyakumar, et al., 2021)

1.4 Analysis of the inverse kinematics for 5 DOF robot arm using D-H parameters

This paper discusses on the drawbacks of using brute force method in iteration to solve the equations involving D-H parameters of the arm and focuses on reducing the RMSE (Root Mean Squared Error) to get accurate solution to the non-linear equations. Then the results was implemented to grip objects using DOF robotic arm and the methods used in this paper could be used for robotic arms up to 6 DOF. Since the location and orientation of the base of the arm are already known for every case, finding out the location of the end effector must be done by calculating the transformation (relationship between the base and the end effector). The transformation matrix ${}^{(i-1)}T_i$ is calculated with the help of four parameters: a_i,

α_i, θ_i and d_i where a_i is the distance between z_i and z_{i+1} along x_i , α_i is the angle between z_i and z_{i+1} about x_i , d_i is the distance between x_{i-1} and x_i along z_i and θ_i is the angle between x_{i-1} and x_i abouts z_i.x. Then the system of equations to be explored is given below:

$$^{i-1}T_i = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)D_z(d_i)$$

where,

$$R_z(\theta_i) = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_x(a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x(\alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 8: Formula to get the transformation matrix

Hence, the formula for the transformation matrix is given by:

$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & -c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figure 9: Formula for the transformation matrix

Inverse kinematics could be achieved through two mainly two methods; analytical method which requires solving closed equations and numerical method where an iterative approach is used to get an approximation by solving he system of non-linear equations. The second method is computationally complex. The analysis is done on the numerical method in this paper which is conducted in 8 steps. First of all a line diagram with the joints and connections between joints is created, then the joints are assigned frames and D-H parameters are calculated, transformation matrix between two successive links is constructed, the transformation matrices are multiplied to get the total transformation matrix, in the next step, the numeric transformation matrix is constructed for the target location, then an equation is formed by equating the two obtained matrices. After that step, the equations are solved using iterative methods to get the angles for the joints and finally RMSE for the solutions is calculated using forward kinematics to get the most suitable solution.

The paper concluded that the test was done with over 20 test cases and the authors found the algorithm to be working efficiently and results can be used for the robotic arms up sto 6 DOF.

2 Analysis of Literature Review

The Gambit robot makes use of off-the-shelf equipments such as a camera that is similar to Xbox Kinect, a PC equipped with intel atom processor and an open-source arm design to build a capable yet slightly cheaper chess playing bot using inverse kinematics for generating movements of the bot. The system was able to keep track of the initial position of the board and then track the movement of the chess pieces as they were moved throughout the board so the chess pieces would not have to be modelled correctly.

The MarineBlue system presented all of the aspects to be considered while building a chess robot from scratch and with price considerations in mind. Since the paper was from the early 2000s, machine learning techniques found today were either not invented or not that sophisticated as it is now due to computing bottlenecks and so on. So the paper described using of surface areas to recognize chess pieces and track them on the chess board. For the movement of the arm, the inverse kinematics algorithm was used which can be found in similar systems till now. Similarly, the algorithm can detect all valid chess moves, even the castling move (which involves a displacement of the king and the rook), given that the user adheres to the chess playing rules. Some cheating actions, such as swapping two chess pieces of the same color would go unnoticed by this algorithm, since it will not detect any changes on the chessboard in that case. The reason for this is that the board layer only recognizes the color of a piece, not its type.

The third paper discusses use of multiple object tracking using technologies such as neural networks and Automatic Image Annotation in order to improve the accuracy of detection of objects in real time.

The fourth paper presented with the unique aspect of using analytical approach with iterative method and avoiding errors while calculating inverse kinematics for DOF robot arms up to 6 joints.

With the analysis of these different algorithms and techniques, it can be deduced that inverse kinematics is one of the most important algorithm while working with robotic arms that involves the use of various DOFs.

Research on Similar Systems

Raspberry Turk

Raspberry Turk is an open-source chess playing bot that uses Raspberry Pi as the main computer. The robot was inspire by the Mechanical Turk, a chess playing machine from the 18th century. It uses computer vision technique to keep track of the situation of the board and then makes moves based on the changes on the board. A raspberry pi camera is used as the sensor to track the movements and in order to train the computer vision algorithm, the author wrote a custom script and then captured every possible move on the board manually and he took about 4 images of a chess piece in each square varying the position and orientation within the square and changing the lighting scenario. Then the image were processed in a series of steps starting from creating individual folders for each iteration, then breaking the images and then creating a labelled folder containing 60x60 pixel images and then preparing the dataset out of it.

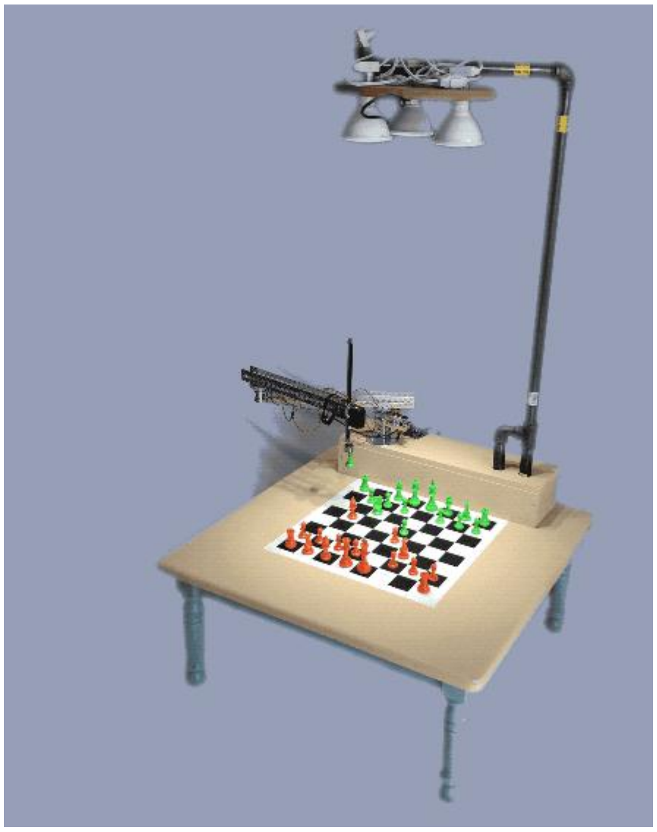


Figure 10: Picture of the Raspberry Turk System (Meyer, 2017)

In order to move the chess pieces, a SCARA (Selective Compliance Articulated Robot Arm) was used and it could move only in x and y axes. Then for picking and placing the chess piece, an electromagnet was attached to horizontal beam that could move up and down and the chess pieces contained a piece of metal attached on a drilled hole on top of them. For handling the chess gameplay, stockfish engine was used.

Analysis of the system

As the image processing techniques could not easily identify the pieces on the board if the white piece is on the white square and black piece is on the black square so the author used green and orange as the color of the pieces. That was one of the main highlights of the system. Similarly, to allow for promotion of rook to other pieces, the author has trained a convolutional neural network to train the system classifying pieces (rook, bishop or knight) in the image to find out which piece the player promoted the pawn to. So this made the end result comparable to a chess game against real human opponent.

(Meyer, 2017)

3 Artefacts

The whole project is divided into 3 major subsystems and the Functional Decomposition Diagram (FDD) of the system can be seen below:

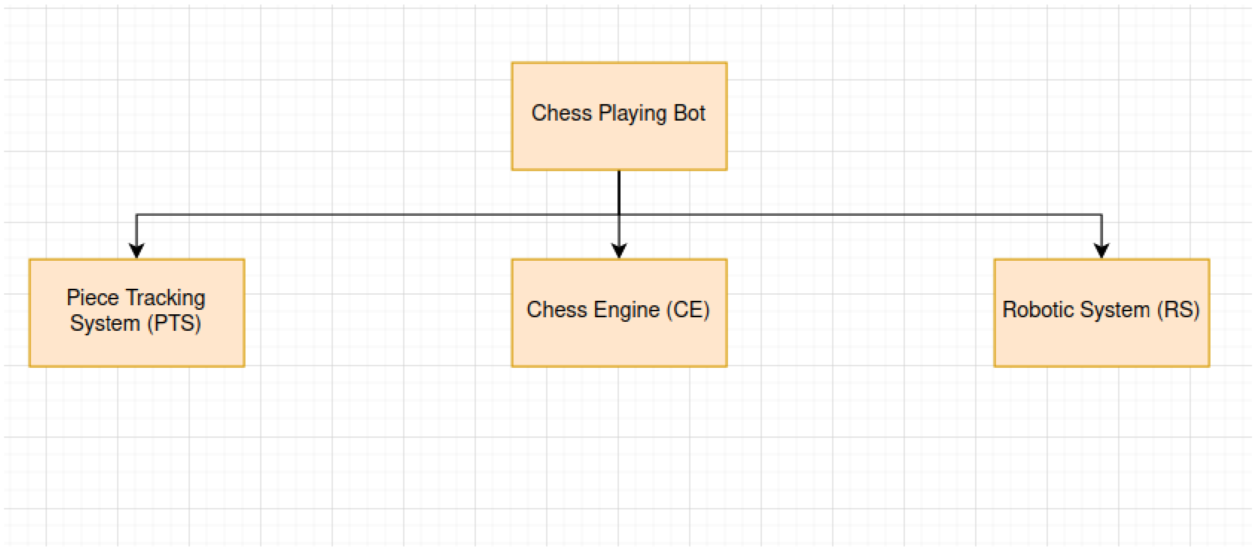


Figure 11: Proposed Artefacts of the System

3.1 Chess Engine

3.1.1 SRS

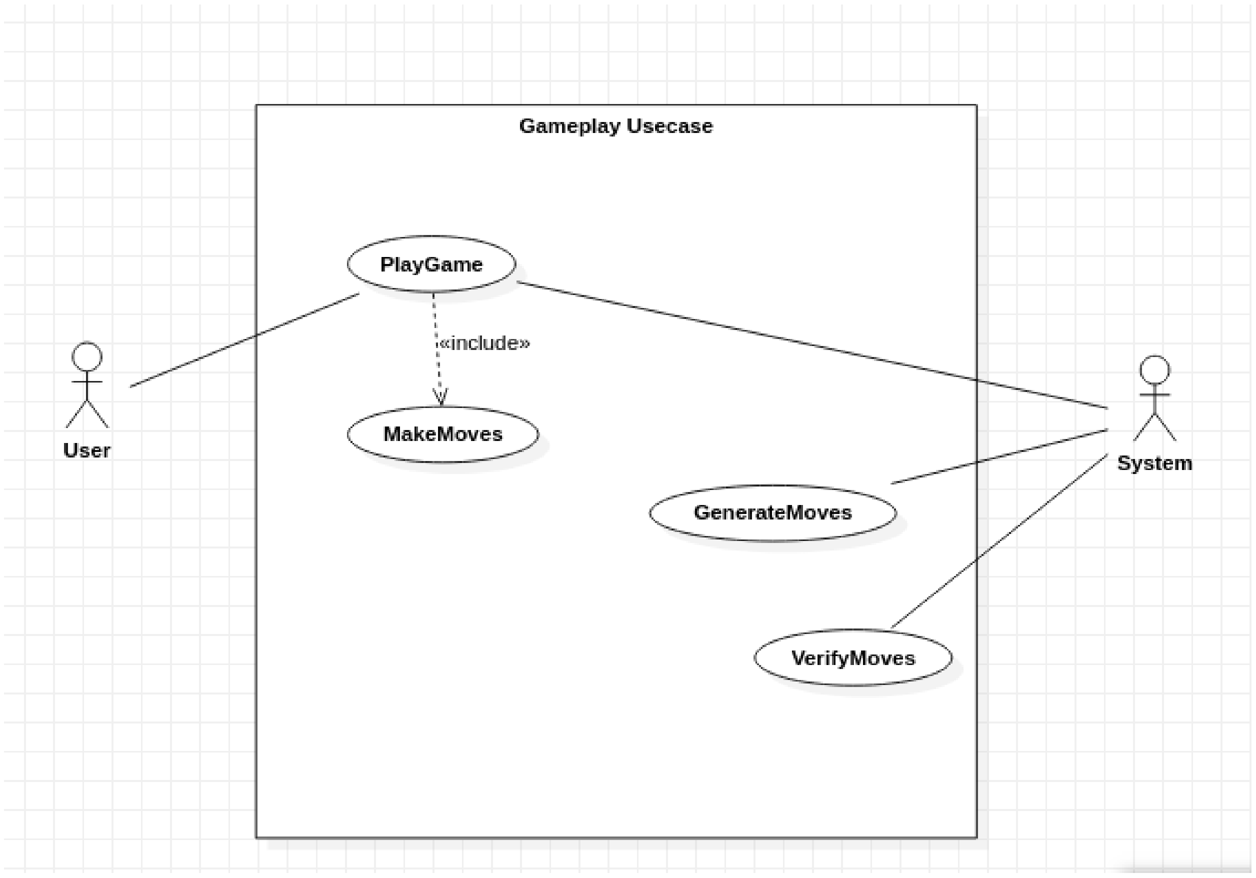
CE - F - 1.0

Types of Requirements:
F : Functional Requirements
NF : Non-Functional Requirements
UR : Usability Requirements

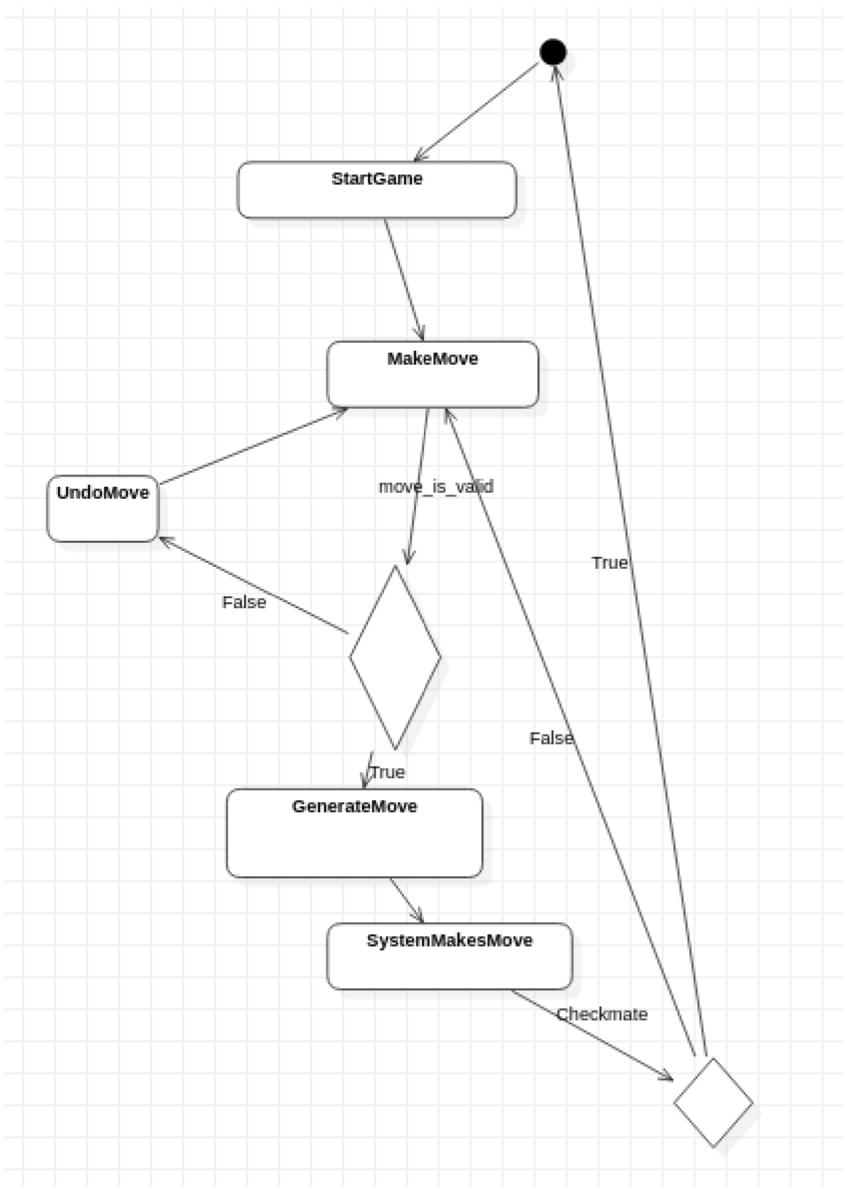
Sub System
CE : Chess Engine

Req. code	Requirements Description	Use Case
CE-F-1.0	The user should be able to play games against the system and the moves should be validated.	Chess Engine
CE-NF-1.1	Castling functionality should be available for both the user and the bot.	
CE-U-1.2	Once the castling has been done, the user and bot should not be able to castle again.	
CE-U-1.3	Whenever there is an invalid move placed by the user, the system should inform the user that they have placed invalid moves.	
CE-F-1.4	The system should have level selection functionality.	
CE-NF-1.5	The difficulty of gameplay should be based on the level selected.	

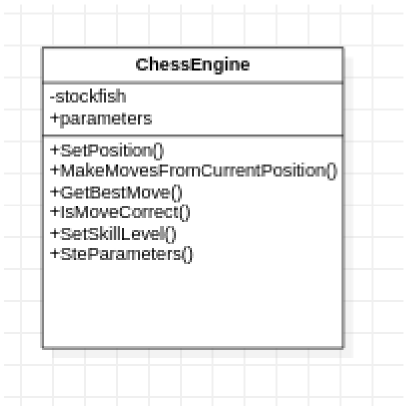
Use Case Diagram



Activity Diagram



Class Diagram



3.2 Piece Tracking System

3.2.1 SRS

PTS - F - 1.0

Types of Requirements:
F : Functional Requirements
NF : Non-Functional Requirements
UR : Usability Requirements

Sub System
PTS: Pieces Tracking System

Req. code	Requirements Description	Use Case
PTS-F-1.0	The system should be able to detect squares and their colors.	Detect Squares
PTS-NF-1.1	The system should keep track of the squares, their color and their co-ordinate within the image.	
PTS-F-1.2	The system should be able to identify chess pieces.	Detect Pieces
PTS-F-1.3	The system should be able to detect changes occurred in the chess board and identify what changes have occurred since the last time.	Track Pieces
PTS-NF-1.4	Depending on the difference in the images, the system should be able to track the movements of the pieces on the chess board.	Track Pieces
PTS-U-1.5	The system should also track the pieces taken by the user which is usually placed outside of the chess board.	

3.2.2 Algorithm Approach

3.3 Robotic System

3.3.1 SRS

RS - F - 1.0

Types of Requirements:
F : Functional Requirements
NF : Non-Functional Requirements
UR : Usability Requirements

Sub System
PTS: Pieces Tracking System

Req. code	Requirements Description	Use Case
RS-F-1.0		

4 References

Chess.com, 2021. What Russia Taught the World About Chess. [Online]
Available at: https://www.chess.com/article/view/russia-world-chess

Matuszek, C. et al., 2011. Gambit: An Autonomous Chess-Playing Robotic System. 2011 IEEE International Conference on Robotics and Automation, pp. 1-7.

Meyer, J., 2017. Raspberry Turk. [Online]
Available at: http://www.raspberryturk.com/
[Accessed 10 12 2021].

Statista.com, 2021. Global Chess Market. [Online]
Available at: https://www.statista.com/statistics/809953/global-chess-market-size/

URTING, D. & BERBERS, Y., 2003. MarineBlue: A Low-Cost Chess Robot. pp. 1-7.

Vijiyakumar, K., Govindasamy, V. & Akila, V., 2021. 4. Multi-Object Recognition and Tracking with Automated Image Annotation for Big Data Based Video Surveillance. IEEE, pp. 1-5.