Hikaru Nakarmsen (DOBOT Magician)

Course

Fundamentals of Robotics – RB310

Students Names

Hazem Mohamed - 211001615
Mohamed Youssef - 211001821
Mohamed Abdelraheem - 211002827

Table of Contents

Introduction

Summary

- The main purpose of the project is to fully automate Dobot Magician to play chess against a human player using a chess engine. The expected result of the project is to have a robotic arm with 4 degrees of freedom and a camera that detects the player's move on the chess board and performs the best move for the current situation. The total anticipated budget is 1000 EGP.

Motives

- After careful consideration, we realized that exploring the DOBOT Magician capabilities of playing chess was not a common undertaking. However, we were enticed by the prospect of a project that would simultaneously test our coding skills and hardware proficiency. It became evident that this project was the ideal fit for our team.

- Organizing a chess tournament on our campus, offering an opportunity for the winner to compete against our robotic arm. This tournament aims to showcase the capabilities of our advanced robotic arm, designed specifically for playing chess. By organizing this event, we seek to engage chess enthusiasts and provide them with a platform to test their skills against our technology. The tournament promises to be an exciting and competitive occasion, serving as a testament to the progress achieved in the field of robotics.

Hardware Specifications

- Dobot Magician Robotic Arm:

  The project incorporates the Dobot Magician robotic arm to execute optimal moves determined by the chess engine.

DoF	4
Payload	500 g
Max. Reach	320 mm
End Effector Used	Vacuum Suction Cup

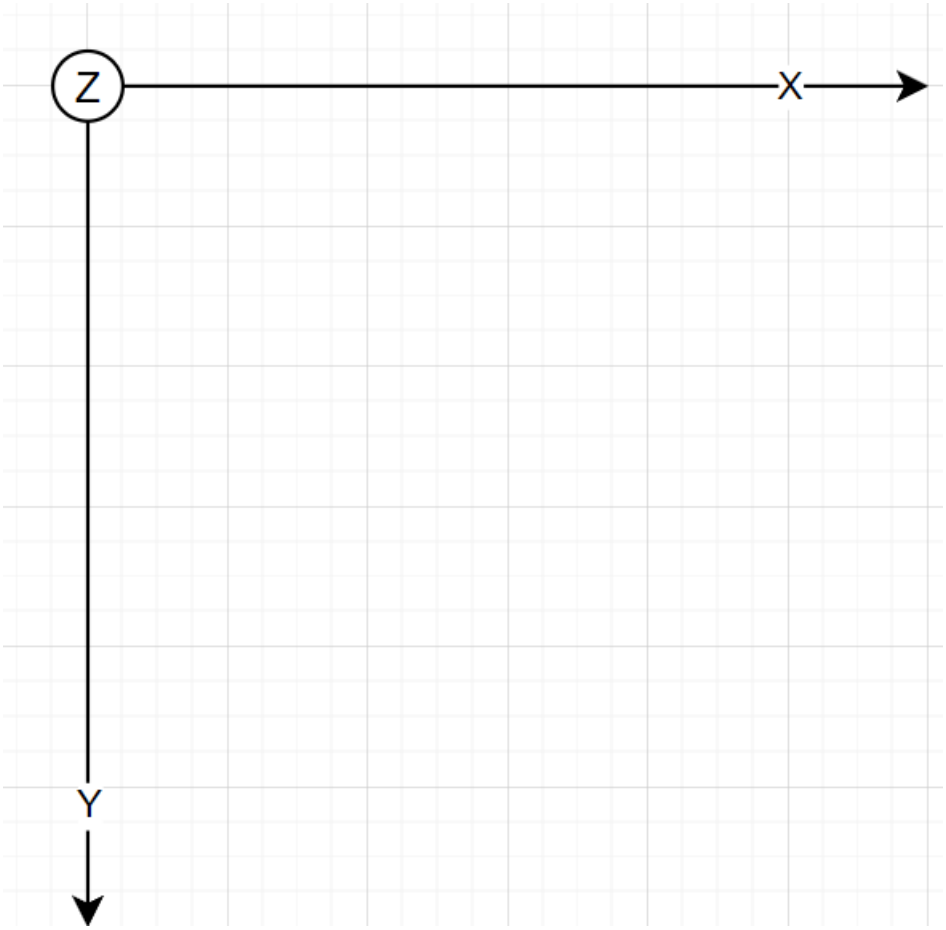
- Logitech C270 HD Webcam:

  The Webcam provides high-definition, real-time video capture of the chessboard, which allows the detection of the player's moves, using computer vision techniques.

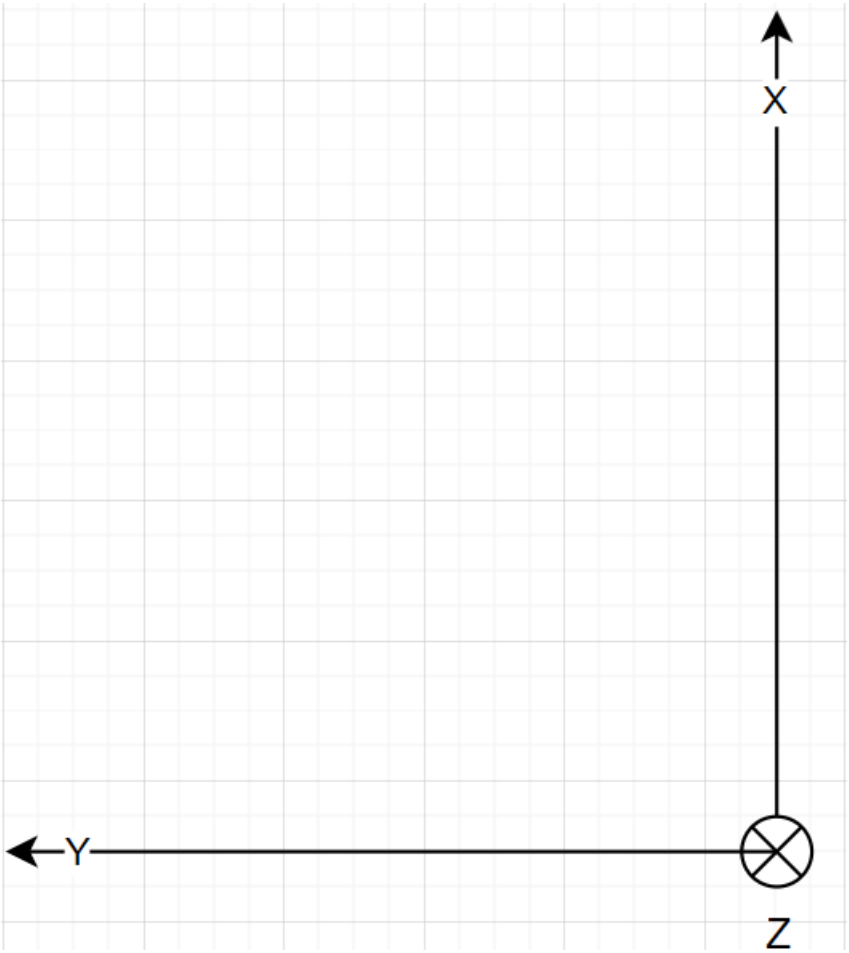
Resolution	720
Frames per Second	30
Mega Pixels	0.9
Diagonal field of view (DFoV)	55°

DOBOT Kinematics

The DOBOT Magician is a robotic arm that has a range of reach of 320mm (32cm). This means it can reach up to 32 centimetres from its base. To control the DOBOT's orientation we can either change the cartesian coordinates (x, y, z, r), or change the angles of the joints (j1, j2, j3, j4). However, we don't know the values of the cartesian coordinates of any of the cells. This is where we utilize the camera for its second purpose.



This is the camera coordinate frame, where the origin (0,0,0) is the first top left pixel of the camera's POV.



This is the DOBOT's coordinate frame, where the origin (0,0,0) is at the base of the DOBOT. Using these two coordinate frames, we can find a simple homogenous transformation matrix that can help us translate the coordinates that we get from the camera to the DOBOT.

Let's start with finding the homogenous transformation matrix. First, we need to calculate the rotations around the axes. The following image shows the rotation matrix for each axis:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For simplicity, we are going to denote the camera's coordinate frame with subscript 'C' and the DOBOT with subscript 'D'. Now, we need to translate the X_C, Y_C, and Z_C to X_R, Y_R, and Z_R. There are several ways to do such translation, but this is how we did it. X_C will be rotated 180º and Z_C will be rotated 90º. Then, we must calculate the displacement from the origin of the camera coordinate frame to the DOBOT coordinate frame. After measuring it, we found that the D_x is 20.5 centimeters and D_y is 36 centimeters. Now that we have all our variables let's make the transformation matrix. The rotation matrices will be as following:

$$R_z = \begin{bmatrix} \cos(90) & -\sin(90) & 0 \\ \sin(90) & \cos(90) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(180) & -\sin(180) \\ 0 & \sin(180) & \cos(180) \end{bmatrix}$$

Multiplying these matrices, we get the result of:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

We append to it the Displacement vector which is as following:

$$Displacement = \begin{bmatrix} 20.5 \\ 36 \\ 0 \end{bmatrix}$$

And finally, we add a column vector of:

$$[0 \ 0 \ 0 \ 1]$$

Resulting in the following transformation matrix:

$$Transformation = \begin{bmatrix} 0 & 1 & 0 & 20.5 \\ 1 & 0 & 0 & 36 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now that we have our transformation matrix, we need to process the coordinates that we get from the camera. The camera's coordinates are represented in pixels, which is inconsistent with DOBOT's coordinates. Therefore, we need to translate the coordinates from pixels to centimeters. This can be easily done using a simple conversion from pixels to centimeters by dividing the camera width POV by the number of pixels on the X-axis.

$$pixel\ 2\ cm\ = \frac{camera\ width_{cm}}{camera\ width_{pixels}}$$

Then we multiply the coordinates that we get from the camera with the previous ratio, and finally we get the cartesian products for the DOBOT.

The DOBOT has pre-implemented DLL files that calculate the inverse kinematics by giving it the cartesian coordinates and receiving the joint angles. However, to calculate the inverse kinematics we need to have the DH Model of the DOBOT to calculate the forward kinematics first.

The DH Model of the DOBOT is as follows:

Joint	α_{i-1}	a_{i-1}	d_i	Θ_i
1	180	0	L_1	Θ_1
2	0	L_2	0	Θ_2
3	90	0	L_3	$\Theta_3 + 90$

Where:

L_1 = 13.8 cm
L_2 = 13.5 cm
L_3 = 14.7 cm

Using the previous DH Model, Inverse Kinematics is implemented to get the joint angles.

Software Implementation

- Used libraries:

  - Chess: The chess library in Python is used for representing the chess board and pieces, generating possible moves, and parsing and formatting chess moves from Portable Game Notation (pgn) files.

- Torch: The torch library, also known as PyTorch, provides a wide range of features and functionalities that facilitate the development and training of neural networks. It is used in this project to create and train the DQN model, it is also used to evaluate chess positions or to select moves after training the model.

- NumPy: NumPy is a fundamental library in Python for numerical computations. It is used in this project for working with arrays, matrices, and mathematical functions, which is needed in the process of building the AMCTS and the DQN.

- OpenCV: Provides a suite of computer vision functionalities. It's mostly used for image processing, object detection, and pattern recognition. In the context of this project, it is needed for real-time image processing for chessboard detection and move recognition.

- Project Files:

  - C2M.py: This file includes necessary functions for getting the played chess move by using the camera, it is also needed to get the cartesian coordinates of each chessboard cell.

  - CE.py: This file includes necessary functions that interact with the chess engine to get the best moves according to the state of the game at that moment.

  - DrDRA.py: This file includes necessary functions for getting the chess engine's predicted best move's cartesian coordinate and applying the move on the board using the robotic arm.

  - GUI.py: This file includes the implementation of the GUI on which the chess game is streamed through.

References

- https://markelsanz14.medium.com/introduction-to-reinforcement-learning-part-3-q-learning-with-neural-networks-algorithm-dqn-1e22ee928ecd
- https://web.archive.org/web/20180623055344/http://mcts.ai/about/index.html
- https://towardsdatascience.com/alphazero-chess-how-it-works-what-sets-it-apart-and-what-it-can-tell-us-4ab3d2d08867
- https://github.com/OmarShalash/Dobot
- https://github.com/Momad-Y/Hikaru-Nakarmsen-Chess-Robot
- https://www.dobot-robots.com/products/education/magician.html
- Project's Source Code: https://github.com/Momad-Y/Hikaru-Nakarmsen-Chess-Robot