DOBOT Magician: Didactic Experiences for Introduction to Artificial Intelligence

Daniel Oliveira
Advisor: Paulo Oliveira, Vítor Filipe
School of Science and Technology, University of Tras-os-Montes e Alto Douro (UTAD), 5000-801 Vila Real, Portugal

***Abstract*—In current times, Robotics and Artificial Intelligence play increasingly important and fundamental roles in our lives. These technologies are present in industries, medicine, education, and even in our homes, becoming part of our daily lives. Given this scenario, the present project aims to utilize small-scale robots, with emphasis on the Dobot Magician robot, in order to spark interest and motivate students in the fields of Artificial Intelligence and Robotics. The main objective is to program and develop the Magician in such a way that it is capable of playing the game of Tic-Tac-Toe.**

***Index Terms*—DOBOT Magician, Robotics, Artificial Intelligence**

I. Introduction

Robotics and artificial intelligence (AI) are two of the fastest growing industries that are changing the way we interact with technology. Robotics is concerned with the design, construction and operation of intelligent machines known as robots, while artificial intelligence focuses on systems that can recognize, think, learn and make decisions like human intelligence. The combination of robotics and AI has opened up endless possibilities across industries and industries. Robots can be found in manufacturing facilities, in hospitals, in space exploration projects, and even in our homes. It is designed to perform tasks efficiently, accurately, often in hazardous or inaccessible areas. Artificial intelligence plays a key role in the scalability of robots. Using advanced algorithms and machine learning techniques, AI enables robots to analyze large amounts of data, make predictions, adapt to new situations and even learn from their experiences This allows robots to behave themselves great, without constant human intervention Able to perform complex tasks. In addition to commercial applications, robotics and AI have made significant contributions to the arts. Artists and technologists are exploring the creative potential of robotic AI systems to create innovative and inspiring works of art. From robotic sculptures, interactive installations and AI-generated music and visuals, this collaboration between human creativity and machine intelligence is redefining the boundaries of artistic expression. However, the rapid advances in robotics and AI also raise ethical and social issues. Questions of work exclusion, privacy, and the impact on human decision-making processes. The objective of the project is to study the performance of the Dobot Magician Robot, analyze its capabilities and apply it in practical applications. The main test involves enabling the robot to play the popular game of ”Tic-Tac-Toe” against the user. Through this experience, the goal is to motivate and inspire young students in artificial intelligence and robotics.

This project seeks to create the right environment for student engagement, encouraging them to explore and understand the possibilities offered by the combination of robotics and artificial intelligence. Through interaction with the Dobot Magician robot and some ”tic-tac-toe” games, students are expected to gain practical knowledge and develop a lasting interest in these growing areas, preparing them for them a future that is gaining technology and promise.

II. DOBOT Magician

The Dobot Magician robot[1] (Figure 1) was the world’s first 4-axis desktop robotic arm. It is precise, multifunctional, extensible, and compact, making it the ideal tool for learning robotics. The robot is capable of performing various tasks such as 3D printing, laser engraving, drawing, writing, and object manipulation, either with a gripper or a suction cup. The tools required to perform these tasks are easy to assemble. This robot can be used at home, in a laboratory, or even in competitions—imagination is the only limit. To program the Magician, various applications provided by DOBOT can be used, such as DobotLab and DobotStudio. Another option is DobotBlockly, a drag-and-drop coding tool that provides a simple and effective learning experience for users with limited programming knowledge. Python is the most commonly used language for programming this robot, and the DobotLab application itself allows running Python code. This robotic arm weighs a total of 8kg and has a compact base measuring 158 millimeters by 158 millimeters, making it easy to move and position on any desk or table. The Magician also features 13 interface ports that support laboratory projects, secondary development, and robotics courses. It has a maximum reach of 320 millimeters and can communicate via USB, Bluetooth, and WLAN. The robot has a maximum power consumption of 78 Watts and operates within a temperature range of -10°C to 60°C.

Fig. 1. DOBOT Magician

A. First time using Magician

Before conducting any experiments with the robot, we need to first assemble it and install the necessary software and drivers. The assembly process is straightforward, as most of the robot comes pre-assembled, leaving only the choice of the end-effector and its assembly. Regarding software, it is advisable to create an account on the DOBOT robots' website in order to access all the tools provided by the manufacturer. Then, choose the most suitable software according to the specific activity being carried out. The essential software for all types of experiments is DobotLink, which serves as a driver to ensure that the robot is recognized by the computer as the robot itself, rather than a USB device. Unfortunately, this software or driver comes with a minor bug: whenever another software, such as DobotLab, is launched, a notification for a DobotLink update appears. Even after performing the update, the notification reappears. However, this bug does not significantly impact functionality. To avoid this issue, simply click "Cancel" instead of accepting the update, as the software is already up to date. In terms of conducting the experiments, the most commonly used application is DobotLab, as it is the latest software for this robot and still receives updates to this day. In many videos showcasing the Magician, users may still be seen using DobotStudio. This is because most of those videos are slightly older, and at that time, DobotLab had not yet been created. Nevertheless, even if it is necessary to follow an older video to understand how to do something, the interface of DobotLab is quite similar to that of DobotStudio, with the same functionalities, making it easy to follow along. Thus, prior to conducting experiments with the Dobot Magician robot, it is crucial to properly assemble it and install the necessary software and drivers, ensuring an environment conducive to successful activity execution.

a.1) DobotLab

As the name suggests, this application is the laboratory of DOBOT robots. It offers various levels of usage, ranging from simple to advanced, allowing for accessible yet complex experiments with the robot.

DobotLab encompasses several functionalities. The "Dobot-Block Lab" enables users to drag predefined lines of code to move the Magician's arm and perform simple experiments. The "Writing and Drawing Lab" allows the Magician to write and draw using its pen. Additionally, the "Laser Engraving Lab" facilitates laser engraving on objects. Similar to this tool, the "3D Printing Lab" enables small-scale and straightforward 3D printing. These last two mentioned tools are unique to the DOBOT Magician.

Within the DOBOT Lab, there is also the "Virtual Simulation Lab," an incredible tool for experimenting with all the Magician's functionalities without actually using the physical robot. This lab presents various experiments, such as simulating a supermarket scenario. Users can program this scenario using the previously mentioned "DobotBlock Lab" or the "Python Lab," which allows programming in Python to make the robot perform the necessary movements within this scenario. The "Python Lab" can also be used to program movements on the physical Magician, not just in these simulations.

Finally, there is the simplest tool, yet an excellent introduction to the DOBOT world—the "Teaching and Playback Lab." It allows users to perform movements on the Magician by selecting the desired axes and controlling the gripper or suction cup to manipulate objects, all with easy-to-understand clicks in the interface.

Overall, DobotLab offers a range of functionalities, from basic to advanced, providing an excellent platform for exploring the capabilities of the DOBOT Magician and delving into the world of robotics.

III. Implementacion

Regarding the implementation of the necessary code to perform the experiment, everything was written in Python using the Spyder IDE [2]. Initially, I researched the source code provided by DOBOT [3] to operationalize the robot, which aimed to simplify the learning process. In the folder where this code is located, there are also other codes with the same functionality, but written in different languages such as Java, CSharp, Arduino, among others. To understand the functions and the code, DOBOT provides a "Demo Description" that explains the content and operation of the different codes. Using the Python demo code, DobotControl.py, I conducted some tests to understand what modifications I needed to make to achieve the desired outcome for my problem. Based on what I learned, I created four functions in a new file named Functions.py. In this file, I developed a function to recalibrate the robot, ensuring it has the correct and desired initial position. Then, I wrote the function that activates the suction process of the suction cup. Lastly, I defined the two functions that determine the robot's movement, specifying the position and the method of reaching it. The only difference between the two functions is that one uses the jump mode, PTPJUMP (Figure 2). Throughout the experiment, I relied heavily on this mode as it was most suitable for my purposes.

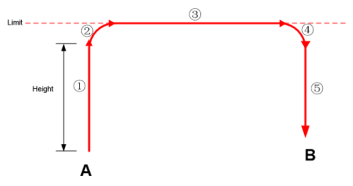


Fig. 2. PTP JUMP recreation

In the last function, I used PTPMOVJ to perform less precise movements on the robot in order to create a dance routine by combining a series of movements.[1]

With these functions created, I conducted tests to ensure that all functions were working correctly and achieving the intended goals. These tests can be found in the file teste_posicoes.py. Once I had planned and understood the movement process, I proceeded to define the positions that the robot would need to reach for different moves. To do this, I decided to assign a number to each square on the game board (Figure 3).

[1]PTP stands for "point to point movement," which refers to the robot's ability to move from one specific point to another in a direct manner.

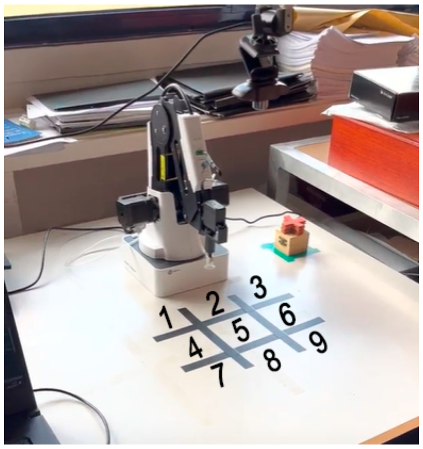


Fig. 3. Game Board with numbers representing each square

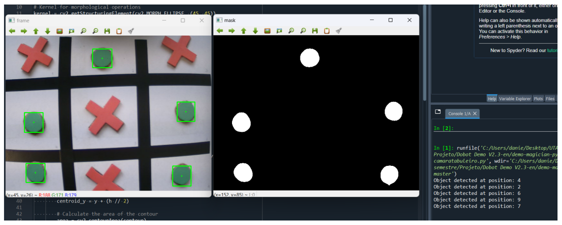
With the coordinates defined for each number and also for the positions where the robot would pick up the pieces, I performed another test, simulating a random move. The code used for this can be reviewed in the file teste_jogada_aleatoria.py. In this program, the robot picks up a piece and places it in a square, repeating the process.

With these steps completed, I moved on to the algorithmic part that I would use to make the DOBOT Magician play Tic-Tac-Toe perfectly. I chose to use the Minimax decision algorithm [4], which is a method used to determine the optimal move in a turn-based two-player game. The algorithm follows these steps to determine the best move:

Step 1: Generate the complete game tree, exploring all possible moves until reaching terminal states.
Step 2: Evaluate the utility function for each terminal state, assigning a numerical value that represents the game's outcome (e.g., +1 for a win, 0 for a draw, and -1 for a loss).
Step 3: Backpropagate through the tree, propagating the values from the leaves to the root node, alternating between maximization and minimization.
Step 4: Select the optimal move for the current player based on the propagated values in the tree.

This algorithm was initially implemented in a console-based ”Tic-Tac-Toe” game to test its functionality. After that, I added the part that handles the robot's movement based on the received square value. After conducting several tests, I found that it was possible to play against the robot, but it was not yet fully autonomous. Although the robot was placing the pieces in the correct positions and capable of winning the game, the user's moves were still being entered through the console. I had already anticipated this issue and came up with a solution: I placed an external camera on top of the game board using another available robotic arm, the UR5 from Universal Robots [5]. However, operating the UR5 was more complex and challenging, and it had been inactive for some time, so it didn't start up. But unwilling to give up the obvious solution of using the camera, I conducted some research and found what could be the issue [6]. It appeared to be a problem with the BIOS battery, which needed to be replaced. As a temporary solution, I performed a direct connection on the motherboard by following the instructions in a forum thread. Using a screwdriver, I pressed the indicated pins, establishing a direct connection on the UR5. After completing this process, I operated the robot and, with the installed gripper, positioned the camera at a 90-degree angle relative to the game board.

Regarding the computer vision aspect of this project, an external camera was used, as mentioned earlier, to provide the robot with knowledge of where the opponent's pieces were placed. For this purpose, I used OpenCV [7], a library developed specifically for computer vision tasks, providing ”a common infrastructure for computer vision applications and accelerating the use of machine perception in commercial products.” First, I performed camera calibration following the tutorial available on the library's website. With the calibrated and properly positioned camera, I moved on to the step of identifying the player's pieces and their respective locations. In a new file called camaratabuleiro.py, I used OpenCV to define capturing objects of green color and approximately the size of the ”O” piece to avoid interference. Additionally, I implemented an imaginary division of the camera's field of view, similar to the game board. Essentially, since the camera only sees the board, I divided the captured image into nine squares to facilitate the process of determining the piece's position (Figure 4).



```
In [2]:

In [1]: runfile('C:/Users/danie/Desktop/UTA
Projeto/Dobot Demo V2.3-en/demo-magician-py
camaratabuleiro.py', wdir='C:/Users/danie/D
semestre/Projeto/Dobot Demo V2.3-en/demo-ma
master')
Object detected at position: 4
Object detected at position: 2
Object detected at position: 6
Object detected at position: 9
Object detected at position: 7
```

Fig. 4. Camera detection and output results

With this imaginary division, the process of collecting the position involved determining which of the nine squares the piece was placed in, and returning a value from 1 to 9 based on the square. Finally, I added an array to store the values returned by the capture process to avoid repetitions and potential errors. To conclude, I integrated all of this code into the existing code that previously only had the console-based ”Tic-Tac-Toe” game and robot movements. By doing so, I achieved the objective of the experiment. The entire process became autonomous and efficient, with only tests and potential improvements remaining to be done.

A. Validation Tests

As previously mentioned, after implementing the main experiment, I conducted some tests to verify if the entire process was complete and error-free. Everything went as planned, and the DOBOT Magician was now capable of playing ”Tic-Tac-Toe” against a human without any intervention, except for the user placing their piece in the desired position.

These tests were of great importance because they allowed me to correct positioning errors, add the possibility for the robot to play first, and, finally, ensure that the robot always calibrated before playing to avoid further positioning errors.

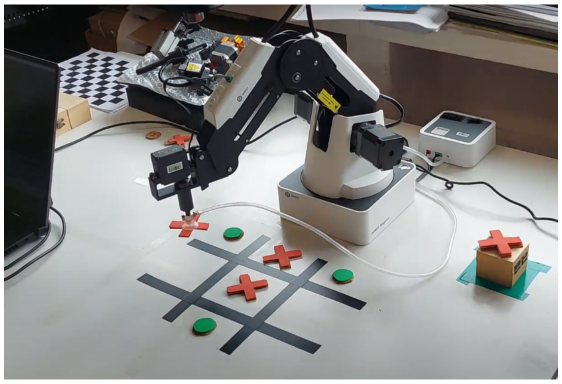


Fig. 5. Magician playing Tic-Tac-Toe

[2] [3]

IV. Related Work

There was a work done by O. Trull-Dominguez, A. Peiró-Signes, J.A. Cabrero-Hernández, J. Rodríguez-Rico, in IES Gonzalo Anaya (SPAIN) and Universidad Politécnica de Valencia (SPAIN), similiar to the one I did. [8]

This paper describes an experiential learning activity (ELA) using DOBOTs, which are learning robots based on Arduino cores. The purpose of the activity is to allow students to experience real-world applications and internalize basic robotics concepts. The activity uses a Tic-Tac-Toe game as a means to develop knowledge in robot arm positioning, tolerancing, sensor positioning, and hysteresis understanding.

The methodology of the activity follows the principles of experiential learning, which involve the transformation of experiences through a four-step cycle: experience, reflexive observation, conceptualization, and active experimentation. The activity is divided into three sessions, each lasting about three hours, and is carried out in groups. The planning, introduction, development, and execution of the activity are carefully organized and guided by the teacher.

In the first session, students are introduced to the DOBOT Magician robot and the Tic-Tac-Toe game. They learn about robot arm positioning, tolerance, and the use of suction cups to pick up game pieces. They also work on programming the robot using Scratch or C++ depending on their level of knowledge.

In the second session, students collaborate with another group and engage in communication and negotiation to play a game of Tic-Tac-Toe between the robots. This session reinforces previous knowledge and introduces new concepts related to communication between robots.

The paper emphasizes the development of transversal competencies such as teamwork, negotiation, and problem-solving throughout the activity. It also discusses the material requirements, planning, and assessment of the activity. The authors conclude that experiential learning activities provide a positive and effective way for students to acquire knowledge and overcome conceptual misunderstandings.

[2]It is possible to watch the execution of the tests and the final product in this video: https://www.youtube.com/watch?v=CDU4nE1jcMUt=9s

[3]All the code described in the chapter ”Implementacion” is located in the dedicated GIT repository: github.com/df01mo/Project_DobotMagician_TicTacToe.

V. Conclusion

I consider the development of this project concluded, expressing great satisfaction with the opportunity to undertake and complete it as planned. Furthermore, I believe that using the DOBOT Magician robot to play ”Tic-Tac-Toe” is an effective way to engage students and ignite their interest in Artificial Intelligence and Robotics.

References

[1] *DOBOT Magician — An all-in-one STEAM Education Platform*. URL: https://www.dobot-robots.com/products/education/magician.html.

[2] *Spyder :: Anaconda.org*. URL: https://anaconda.org/anaconda/spyder.

[3] Shenzhen Yuejiang. “User Guide Dobot Magician User Guide”. In: (). URL: www.dobot.cc.

[4] Mahesh Tunguturi and Mahesh Tunguturi. “AI Analysis for Tic-Tac- Toe Game”. In: *Transactions on Latest Trends in Artificial Intelligence* 3 (3 Dec. 2022). URL: https://ijsdcs.com/index.php/TLAI/article/view/93.

[5] *UR5e Lightweight, versatile cobot*. URL: https://www.universal-robots.com/products/ur5-robot/.

[6] *UR5 not Booting: 2. Digital Input, No Cable - Technical Questions - Universal Robots Forum*. URL: https://forum.universal-robots.com/t/ur5-not-booting-2-digital-input-no-cable/24128.

[7] *OpenCV - Open Computer Vision Library*. URL: https://opencv.org/.

[8] Iated. “12TH INTERNATIONAL CONFERENCE OF EDUCATION, RESEARCH AND INNOVATION, SEVILLE (SPAIN) CONFERENCE PROCEEDINGS”. In: (2019). DOI: 10.21125/iceri.2019.