Relatório

DOBOT Magician: Experiências Didáticas de Introdução à Inteligência Artificial

Laboratório de Projeto em Engenharia Informática

Daniel Filipe Morais Oliveira al74575

Orientador:

Paulo Oliveira

Coorientador:

Vítor Filipe

Resumo

Este relatório destina-se a exposição e explicação do projeto, "*DOBOT Magician*: Experiências Didáticas de Introdução à Inteligência Artificial". O mesmo foi concebido no âmbito da unidade curricular "Laboratório de Projeto em Engenharia Informática", durante o segundo semestre do ano letivo 2022/2023.

Neste relatório será descrito o robô utilizado, maneiras de o utilizar e possíveis experiências que podem ser desenvolvidas com a ajuda do mesmo. Irá também ser demonstrado todo o processo de desenvolvimento da experiência principal, onde o robô no final da mesma será capaz de jogar ao "Jogo do Galo" contra um adversário humano.

Conteúdo

1. Introdução

Nos dias atuais, a Robótica e a Inteligência Artificial desempenham papéis cada vez mais importantes e fundamentais nas nossas vidas. Estas tecnologias estão presentes na indústria, medicina, educação e até mesmo nas nossas casas, tornando-se parte do nosso cotidiano. Perante este cenário, o presente projeto tem como objetivo central a utilização de robôs de pequeno porte, com destaque para o robô Dobot Magician[1], a fim de despertar o interesse e motivar os alunos para os campos da Inteligência Artificial e Robótica.

O projeto visa realizar um estudo das funcionalidades do robô Dobot Magician, explorando as suas capacidades e aplicando-o em experiências práticas. A principal experiência consiste em desenvolver a habilidade do robô jogar o famoso jogo "Tic-Tac-Toe" (conhecido em Portugal como "Jogo do galo") contra um utilizador. Através desta experiência, procura-se instigar e motivar os alunos mais jovens pelas áreas de Inteligência Artificial e Robótica.

Por meio deste projeto, pretende-se criar um ambiente propício a participação dos alunos, incentivando-os a explorar e compreender as possibilidades oferecidas pela combinação de robótica e inteligência artificial. Mediante a interação com o robô Dobot Magician e do jogo "Tic-Tac-Toe", espera-se que os alunos adquiram conhecimentos práticos e desenvolvam um interesse duradouro por estas áreas em crescimento, preparando-os para um futuro cada vez mais tecnológico e promissor.

2. *DOBOT* Magician

O robô Dobot Magician (Figura 1)[1] foi o primeiro braço robótico de "secretária" com 4 eixos no mundo. É preciso, multifuncional, extensível e de pequeno porte acabando por ser a ferramenta ideal para aprendizagem de robótica. O mesmo consegue fazer diversas tarefas tais como, impressão 3D, gravação a laser, desenhar, escrever e pegar em objetos, com uma pega ou então com uma ventosa.

Figura 1- Robô Dobot Magician

As ferramentas para realizar as tarefas anteriormente referidas são de fácil montagem. Este robô pode ser usado em casa, num laboratório ou até mesmo em competições, basta ter imaginação. Para programar o Magician, pode-se usar as aplicações que nos são fornecidas pela *DOBOT,* como por exemplo o DobotLab, DobotStudio, podemos ainda usar o DobotBlockly que é uma ferramenta de "arrastar" código que proporciona a utilizadores com poucas bases de programação uma aprendizagem simples e eficaz. A linguagem mais usada na programação deste robô é Python, até porque a própria aplicação DobotLab tem a funcionalidade de correr código em Python.

Este braço robótico pesa no total 8kg e tem uma base de tamanho 158 milímetros por 158 milímetros, sendo de fácil deslocação para todos e posicionamento em qualquer secretária ou mesa. O Magician apresenta ainda 13 portas de interface que servem de suporte para projetos de laboratório, desenvolvimento secundário e para cursos de Robótica. Tem um alcance máximo de 320 milímetros, consegue comunicar através de USB, Bluetooth e WLAN. Apresenta um consumo máximo de 78 Watts, funcionando entre as temperaturas de -10ºC a 60ºC.

Na Figura 2 podemos ver todo o equipamento que constitui o Magician, as peças que se encontram a verde são extras e devem ser adquiridas à parte.

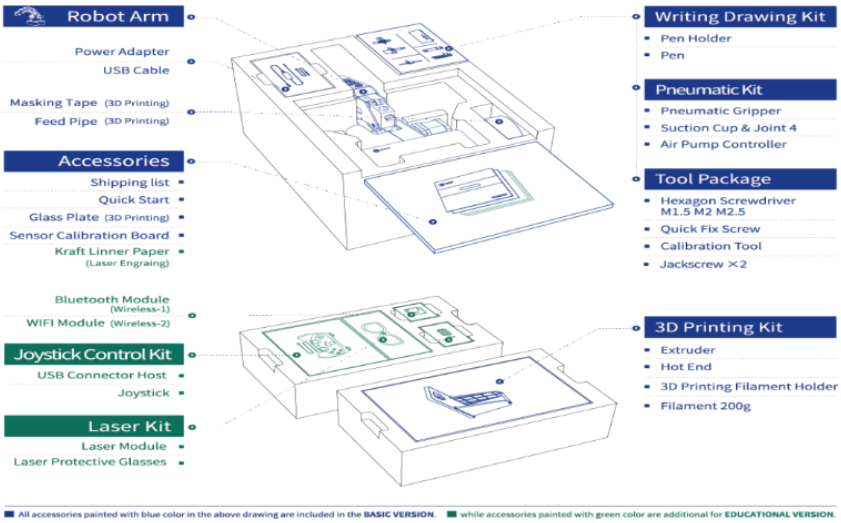

Figura 2- Peças incluídas na embalagem (Extraída de [1])

2.1. Utilização do *DOBOT* Magician

Antes de realizar qualquer experiência com o robô, temos inicialmente que fazer toda a montagem do mesmo, assim como a instalação do software e *drivers* que vamos necessitar. A montagem é muito simples, a maior parte do robô já vem montado, sobrando apenas escolher a peça que queremos utilizar no fim do braço e fazer a montagem da mesma.

No que toca a software, de início convém criar conta no website dos robôs *DOBOT*, de modo a ter acesso as todas as ferramentas disponibilizadas pelo fabricante. De seguida basta escolher os mais adequados em conformidade com a atividade que for a realizar.

O software que acaba por ser necessário a todo o tipo de experiências é o DobotLink[1], que serve de driver para que o robô seja detetado pelo computador como o robô e não um dispositivo USB. Infelizmente este software ou driver vêm com um pequeno *bug*, pois sempre que é iniciado outro software, como por exemplo o DobotLab, é nos indicado que existe uma atualização para o DobotLink, após realizar a mesma voltará a aparecer que ainda existe uma atualização. Este *bug* não prejudica tanto como parece: Para evitar este problema basta fazer o seguinte: antes de aceitar atualizar, carregar em cancelar que o mesmo funciona perfeitamente, até porque está atualizado.

Relativamente as experiências que se desenvolveram a aplicação que mais usada é o DobotLab pois é a mais recente para este robô e que ainda sofre algumas atualizações nos dias atuais. Em muitos vídeos sobre o Magician ainda é possível observar que os utilizadores usam o DobotStudio. Isto porque a maior parte dos vídeos já são um pouco antigos e na altura o DobotLab ainda não tinha sido criado. Mas mesmo que por algum motivo seja necessário seguir um vídeo de modo a tentar perceber como fazer algo, a

interface do DobotLab é bastante semelhante à do DobotStudio e tem as mesmas funcionalidades, sendo assim fácil de acompanhar.

2.2. DobotLab

Como o nome indica, esta aplicação é o laboratório dos robôs *DOBOT*, nela temos diversas formas de utilização, umas mais simples outras mais avançadas, de modo a que seja possível com este robô fazer experiências acessíveis, mas também de grande complexidade.

O *Dobot*Lab(Figura 3) apresenta como funcionalidades o "DobotBlock Lab", onde é possível arrastar linhas de código já definidas de modo a deslocar o braço do Magician e fazer algumas experiências simples, o "Writing and Drawing Lab", que como designa o nome serve para fazer o Magician escrever e desenhar, isto utilizando a caneta do mesmo. Temos também o "Laser Engraving Lab", serve este para fazer gravação a laser em objetos. Parecido com esta ferramenta temos o "3D Printing Lab" que serve para impressões a 3D de pequena escala e bastante simples, estas duas últimas ferramentas referidas são únicas para os *DOBOT* Magician.

No *DOBOT* Lab é possível também utilizar o "Virtual Simulation Lab", uma ferramenta incrível para experimentar todas as funcionalidades do Magician sem na realidade estar a usar o Magician. Neste laboratório são apresentadas várias experiências entre elas a representação de um supermercado, onde após seleção desta podemos escolher onde programar este cenário, através do "DobotBlock Lab", já referida anteriormente, ou "Python Lab" que como o nome indica programar em Python para que o robô faça os movimentos necessários para ser independente neste cenário. O "Python Lab" pode também ser usado para programar movimentos no *Magician real* e não apenas nestas simulações.

Por fim, a ferramenta mais simples, mas que serve de excelente introdução para todo este mundo *DOBOT,* a "Teaching and Playback Lab", onde é possível realizar movimentos no Magician, selecionando os eixos que desejam mover, fazer com que tanto a pega ou ventosa façam o levamento de objetos, tudo isto com meros cliques na interface que é de simples interpretação.

Figura 3- Interface do Software DobotLab

2.3. Experiências realizadas inicialmente

Devido a simplicidade do DOBOT Magician, as experiências iniciais são muito fáceis de realizar, podemos começar logo a executar movimentos no robô, através da aplicação "DobotLab".

Umas das primeiras experiências que realizei foi movimentar o braço e fazer com que o mesmo segura-se uma pequena peça de ferro que tinha a mão. Para isto utilizei o "Teaching and Playback Lab" onde controlei os movimentos do braço através da interface simples deste software. Realizei também alguns testes de movimentação do robô e escrevi as minhas primeiras linhas de código que fariam o robô realizar movimentos, através do "DobotBLock Lab" e do "Python Lab".

A simplicidade e facilidade das primeiras experiências que desempenhei só aumentaram a vontade e motivação para a realização da experiência principal.

3. Experiência principal

Relativamente à experiência final que tenho vindo a referir ao longo deste relatório, a mesma consiste em programar o robô DOBOT Magician, em *Python*, para que este seja capaz de jogar contra um utilizador o "Jogo do Galo", de maneira autónoma e precisa.

O "Jogo do Galo", também conhecido como "Tic-Tac-Toe", é um dos jogos de estratégia mais antigos e populares do mundo. Com origens que remontam à Roma Antiga, este jogo simples, mas desafiador, cativa jogadores de todas as idades até hoje.

O jogo é disputado num tabuleiro 3x3, onde dois jogadores alternam entre colocar a sua marca (X ou O) numa posição vazia. O objetivo é formar uma linha reta com três marcas iguais, seja na horizontal, vertical ou diagonal. [2]

Para a realização deste ensaio, optei por usar a ventosa do robô ao invés da caneta ou até mesmo do gripper, isto pois a ideia seria ter as marcas (X e O) em cortiça para depois serem sugadas pela ventosa e colocadas no devido quadrado no tabuleiro (Figura 4).

Figura 4- Moldes iniciais de cortiça

No que toca ao tabuleiro para o jogo, este foi construído na mesa onde o robô iria atuar, com fita cola preta, de modo a representar para o utilizador o desenho do tabuleiro 3x3 do "Jogo do Galo" (Figura 5). A análise do tabuleiro será feita por uma câmara externa ao robô.

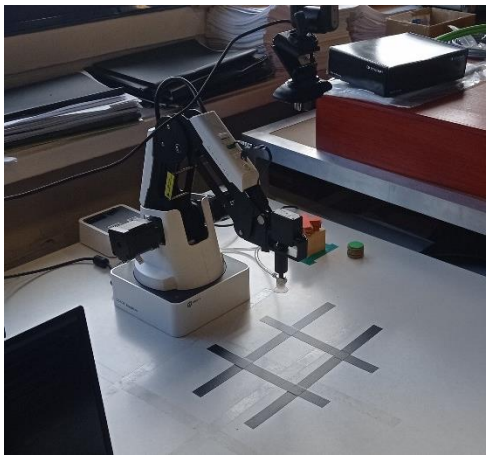

Figura 5- Tabuleiro e posição inicial do DOBOT Magician

Por fim de modo a ser possível a identificação das peças não só pela forma, mas também pela cor, coloquei EVA (Etileno Acetato de Vinila), de cor vermelha nas peças X e de cor verde nas peças O. Isto pois mais tarde na implementação do programa que fará o robô jogar o jogo, uma câmara, como é possível observar na Figura 5, irá devolver as posições das peças do utilizador. Sendo estas verdes e redondas leva ao processo através de OpenCV [3] mais simples e eficiente (Figura 6).

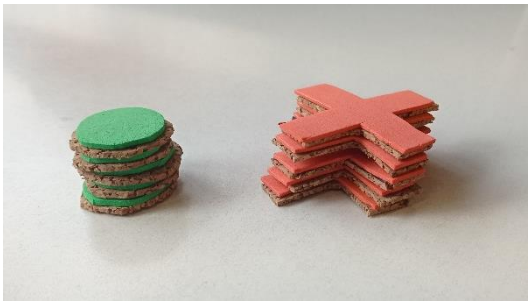

Figura 6- Peças finais

4. Implementação[1]

No que diz respeito a implementação do código necessário para realizar a experiência, todo foi escrito em Python, o IDE utilizado foi o *Spyder* [4] . No início pesquisei código fonte para operacionalizar o robô, este é disponibilizado pela DOBOT de modo a tornar mais simples a aprendizagem. Na pasta onde este código se encontra, está presente também outros códigos com a mesma funcionalidade, funções iniciais de movimento, só que em outras linguagens como por exemplo, Java, CSharp, Arduino entre outros. Para perceber as funções e o código, a DOBOT fornece, em conjunto com o referido anteriormente, um "Demo Description" onde é explicado o conteúdo e funcionamento dos diferentes códigos.

Usando então o código demo em Python, *DobotControl.py*, realizei alguns testes para perceber o que teria de alterar de modo a obter o pretendido para o meu problema. Como tal através do que aprendi criei quatro funções, num novo ficheiro de seu nome, *Functions.py*, neste criei uma função que realiza a recalibração do robô, de modo a que este tenha a posição inicial correta e pretendida. De seguida, escrevi a função que executa o processo de sucção da ventosa. Imediatamente escrevi as últimas duas, servindo estas para definir o movimento do robô, a posição e modo de chegar a mesma, as diferenças entre as duas é que uma utiliza o chamado modo salto, *PTPJUMP* (Figura 7)[5]. Este modo foi o que mais utilizei ao longo da experiência, visto ser o mais indicado para o que pretendia fazer.

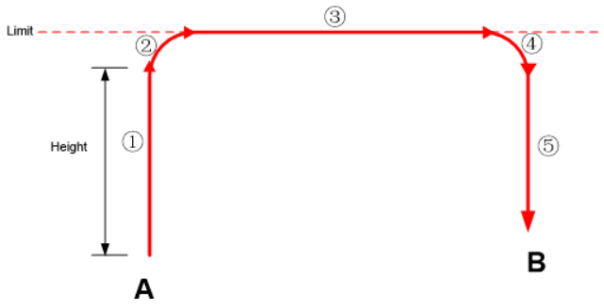

Figura 7- Modo Salto

Na última função usei o *PTPMOVJ* (figura 8)[5], de modo a realizar movimentos no robô menos precisos, para que com um conjunto de movimentos reproduzir uma dança. [2]

[1] Todo o código descrito neste capítulo, localiza-se no repositório GIT criado para o efeito: github.com/df01mo/Project_DobotMagician_TicTacToe

[2] O PTP nestes movimentos significa "point to point movement", movimento ponto a ponto.

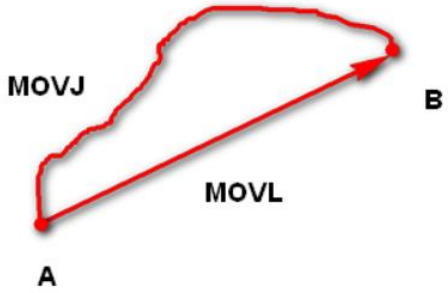

Figura 8- Modo MoveL e MoveJ

Com estas funções criadas concretizei testes de modo a perceber se todas as funções estavam funcionais e a fazer o planeado. Estes testes encontram-se no ficheiro *teste_posicoes.py*. Com todo o processo de movimento planeado e percebido segui definindo as posições que o robô teria que ir para realizar diferentes jogadas, para tal decidi definir cada quadrado do tabuleiro como um número (Figura 9).

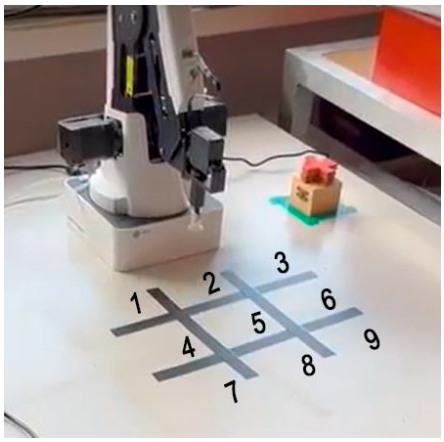

Figura 9- Representação dos números no tabuleiro

Já com as coordenadas definidas para cada número e também para onde o robô irá buscar as peças, realizei outro teste, desta vez a simular uma jogada aleatória, o código usado pode ser revisto no ficheiro *teste_jogada_aleatoria.py.* Neste programa o robô pega numa peça coloca num quadrado e repete.

Com estas etapas concluídas, decidi avançar para a parte do algoritmo que iria usar a fim de fazer com que o DOBOT Magician jogasse o "Jogo do Galo" de maneira perfeita. Sendo assim usei o algoritmo de decisão Minimax[6](Figura 10), que é um método

utilizado para determinar a jogada perfeita num jogo de dois jogados alternados. O método segue os seguintes passos de modo a indicar a melhor jogada:

1. Geração da árvore completa do jogo, explorando todas as possíveis jogadas até os estados terminais.
2. Avaliação da função de utilidade para cada estado terminal, atribuindo um valor numérico que represente o resultado do jogo (por exemplo, +1 para vitória, 0 para empate e -1 para derrota).
3. Retroceder pela árvore, propagando os valores das folhas até o nó raiz, alternando entre maximização e minimização.
4. Seleção da jogada ótima para o jogador atual com base nos valores propagados pela árvore.

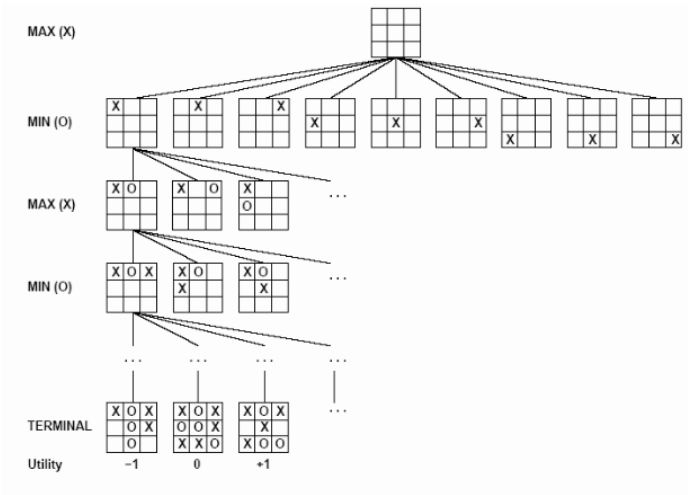

Figura 10- Árvore de decisão parcial para o "Jogo do Galo"

Este algoritmo foi inicialmente implementando num "Jogo do Galo" em consola para testar se estava operacional, após isso acrescentei a parte da movimentação do robô conforme o valor recebido do quadrado a inserir. Após alguns testes verifiquei que já era possível jogar contra o robô, mas ainda não era de uma forma autónoma, pois apesar de ele jogar as peças no sítio certo e ser capaz de ganhar o jogo, os movimentos do utilizador estavam a ser inseridos na consola[3]. Este problema já estava pensado e como tal coloquei

[3] O código referente a esta parte encontra-se no ficheiro *Final_Program.py,* apesar que este como o nome indica já é o programa final. No momento destes testes, o mesmo apresentava-se ainda sem algumas funções implementadas.

uma câmara, externa ao robô, no cimo do tabuleiro, usando outro braço robótico que tinha a disposição, o UR5 da Universal Robots(Figura 11)[7]. Este braço é bastante mais complexo e difícil de operar e devido a já estar parado e desligado há algum tempo, não iniciava.

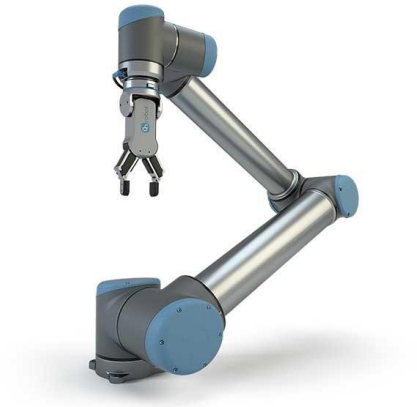

Figura 11- UR5 da Universal Robots

Mas sem abdicar da solução óbvia para a colocação da camara, pesquisei e encontrei o que poderia ser o erro[8]. O problema estaria na bateria da BIOS, que necessitava de ser substituída, ou então como resolução temporária realizar uma ligação direta na motherboard, foi o que fiz, segui o manual que se encontrava no fórum e com uma chave de fendas pressionei os pins indicados (Figura 12), fazendo assim uma ligação direta no UR5.

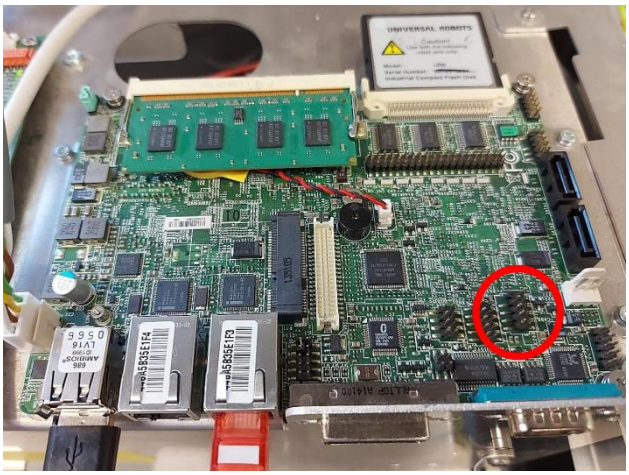

Figura 12- Indicação dos pins a pressionar na motherboard do UR5

Após todo este processo operei o robô e em conjunto com o grappler instalado coloquei a câmara num ângulo de 90 graus em relação ao tabuleiro (figura 13).

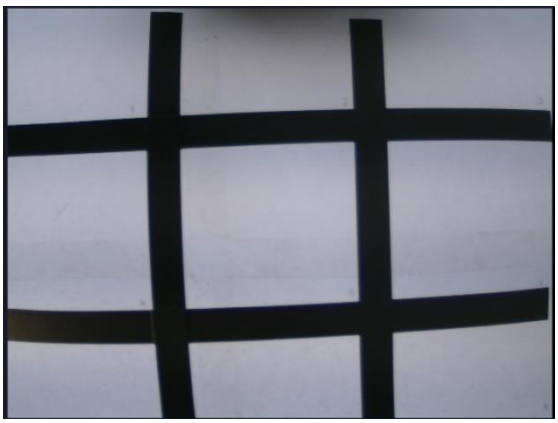

Figura 13- Visão da camara

5. Setup experimental com explicação de como foi colocada a câmara

Em relação ao aspeto de visão computacional utilizado neste projeto, foi utilizado uma câmara externa ao robô, como referido anteriormente, de modo a que o mesmo tivesse conhecimento onde foram colocadas as peças do adversário. Para tal usei OpenCV[9] uma biblioteca desenvolvida para este tipo de tarefas, isto é desenvolvida para "fornecer uma infraestrutura comum para aplicações de visão computacional e acelerar o uso da perceção da máquina em produtos comerciais". De início realizei a calibração da câmara, seguindo o tutorial que pode ser encontrado no site da biblioteca[10](Figura 14).

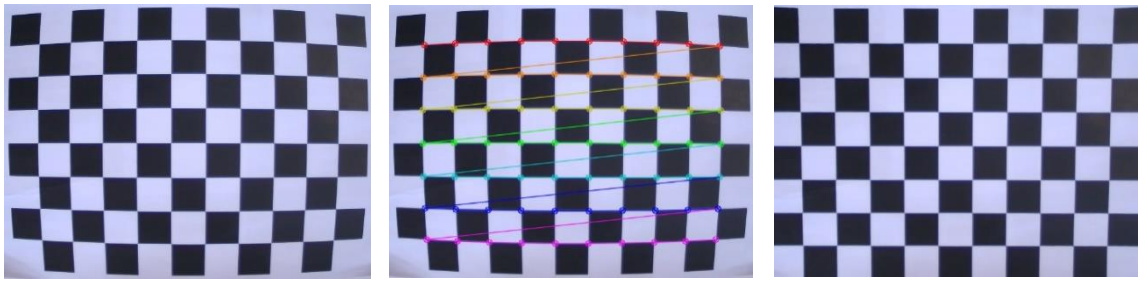

Figura 14- Imagens da Calibração da câmara

Com a câmara calibrada e posicionada conforme o necessário, passei para o passo de identificar as peças do jogador e onde tinham sido colocadas. Para realizar a identificação, num novo ficheiro, *camaratabuleiro.py,* usando OpenCv, defini que devia capturar os objetos de cor verde e entre um tamanho semelhante a peça O, de modo a evitar interferências. Após isto implementei uma divisão imaginária do mesmo estilo que o tabuleiro do jogo. Resumidamente como a câmara apenas vê o tabuleiro dividi a captura da mesma em 9 quadrados, de modo a facilitar o processo de recolha de posição da peça (Figura 15).

Com esta divisão imaginária, o processo de recolha da posição passa por em qual dos 9 quadrados está a peça inserida, retornado depois um valor de 1 a 9 conforme o quadrado.

Por fim adicionei um array que armazena os valores retornados pela captura, de modo a evitar repetições e possíveis erros.

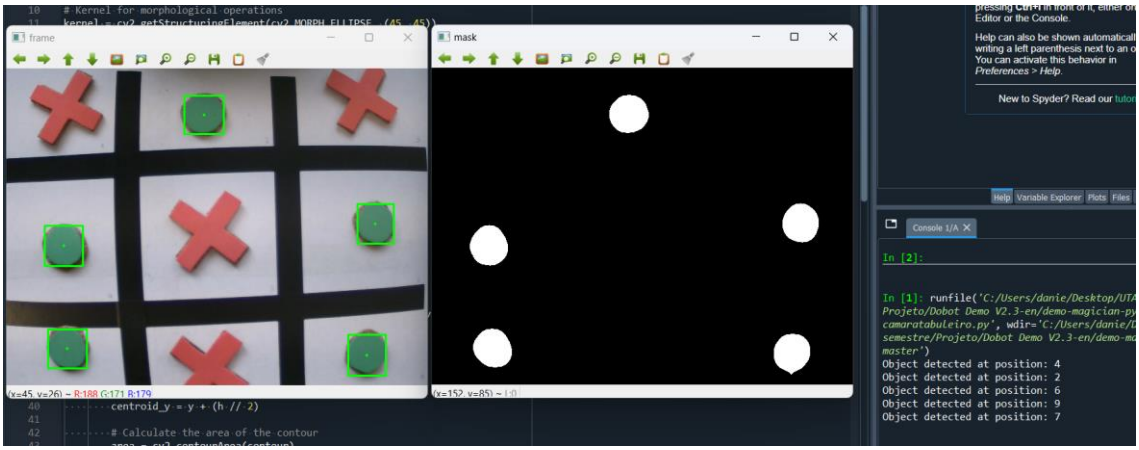

Figura 15- Resultados do programa camaratabuleiro.py

Para concluir injetei todo este código ao código que anteriormente só tinha o "Jogo do Galo" na consola e os movimentos do robô, ficando assim concluindo o objetivo da experiência. Todo o processo era autónomo e eficiente faltando agora só realizar testes e possíveis melhoramentos.

6. Testes de Validação

Como referindo anteriormente, em seguida a implementação da experiência principal realizei alguns testes, com o propósito de verificar se todo o processo estaria completo e sem erros.[4] Tudo correu como previsto e o DOBOT Magician era neste momento capaz de jogar ao "Jogo do Galo" contra um humano, sem que houvesse qualquer intervenção por parte do utilizador, a exceção de colocar a sua peça no respetivo lugar desejado (Figura 16).

Estes testes foram de grande importância, pois com os mesmos corrigi alguns erros de posicionamento, adicionei a possibilidade de ser o jogador a jogar primeiro e para concluir acrescentei que o robô calibrasse sempre antes de jogar, com o objetivo de evitar mais erros de posicionamento.

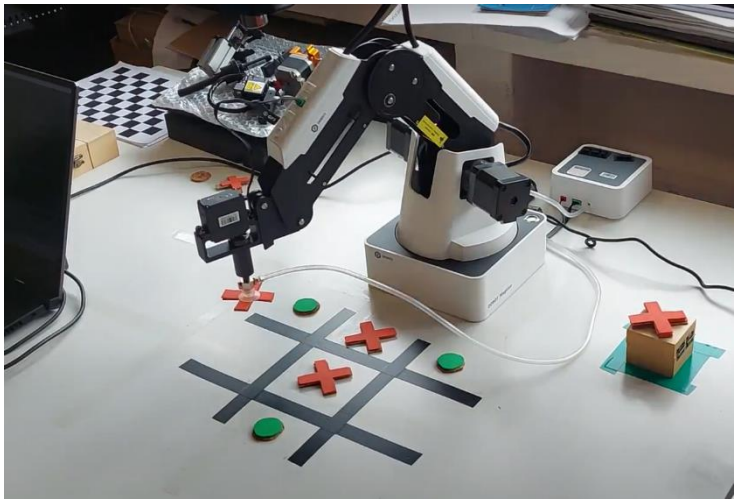

Figura 16- Foto retirada do vídeo

[4] É possível assistir à realização dos testes e produto final neste vídeo: https://www.youtube.com/watch?v=CDU4nE1jcMU&t=9s

7. Conclusão

Dou por concluído o desenvolvimento deste projeto, expressando uma grande satisfação pela oportunidade de o desempenhar e concluir da maneira prevista. Para além disto creio que o uso do robô DOBOT Magician para jogar "Tic-Tac-Toe" é uma forma eficaz de envolver os alunos e despertar o seu interesse pela Inteligência Artificial e Robótica.

Referências

[1]- *DOBOT Magician | An all-in-one STEAM Education Platform*. (n.d.). Acedido em março 10, 2023, de https://www.dobot-robots.com/products/education/magician.html

[2]- Tunguturi, M., & Tunguturi, M. (2022). AI Analysis for Tic-Tac- Toe Game. *Transactions on Latest Trends in Artificial Intelligence*, *3*(3). https://ijsdcs.com/index.php/TLAI/article/view/93

[3]- *Reading and Writing Videos using OpenCV | LearnOpenCV #*. (n.d.). Acedido em abril 7, 2023, de https://learnopencv.com/reading-and-writing-videos-using-opencv/#read-from-web-cam

[4]- *Spyder :: Anaconda.org*. (n.d.). Acedido em maio 1, 2023, de https://anaconda.org/anaconda/spyder

[5]- *2.3.4.2 Point to Point Mode (PTP)*. (n.d.). Acedido em maio 15, 2023, de https://www.dobot.cc/online/help/dobot-m1/16.html

[6]- Artigo *Lua Programming Gems*. (n.d.). Página 258.

[7]- *Braço de robô colaborativo de mesa UR5e que automatiza quase tudo*. (n.d.). Acedido em junho 10, 2023, de https://www.universal-robots.com/pt/produtos/ur5-robot/

[8]- *UR5 not Booting: 2. Digital Input, No Cable - Technical Questions - Universal Robots Forum*. (n.d.). Acedido em maio 16, 2023, de https://forum.universal-robots.com/t/ur5-not-booting-2-digital-input-no-cable/24128

[9]- *OpenCV - Open Computer Vision Library*. (n.d.). Acedido em maio 19, 2023, de https://opencv.org/

[10]- *OpenCV: Camera Calibration*. (n.d.). Acedido em maio 19, 2023, de https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

[11]- Iated. (2019). *2TH INTERNATIONAL CONFERENCE OF EDUCATION, RESEARCH AND INNOVATION*. https://doi.org/10.21125/iceri.2019