Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

06/12/2024

Máquina de resolução de cubos mágicos

Amélie Julie Dufour

PUC RIO

Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Máquina de resolução de cubos mágicos

Aluno: Amélie Julie Dufour

Orientador(es): João Ricardo Cortes Nunes

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Em primeiro lugar, gostaria de agradecer à Pontifícia Universidade Católica do Rio de Janeiro por me receber em seu estabelecimento e, em particular, ao Professor Eduardo Costa da Silva, coordenador da especialidade de Controle e Automação, por me permitir realizar este projeto, e ao meu orientador, Professor João Ricardo Cortes Nunes, por sua ajuda e apoio durante todo o projeto.

Também, gostaria de agradecer ao Jan Krueger Siqueira por sua ajuda neste projeto, ao Wouter Caarls pelos elementos do curso de robótica industrial utilizados e pelo equipamento emprestado, e aos técnicos do laboratório por seu apoio pessoal e profissional.

Em seguinte, gostaria de agradecer a todas as pessoas que contribuíram direta ou indiretamente para o desenvolvimento deste projeto e, em particular, a Gabriella Cristina Bastos Rodrigues, por seu apoio e ajuda na revisão da redação deste trabalho.

Além disso, gostaria de expressar meus sinceros agradecimentos a Sébastien Delprat e Florian Pradrelle por terem me ajudado a realizar essa dupla graduação.

Agradeço também à Universidade politécnica Hauts-de-France e a INSA Hauts-de-France por ter me concedido apoio financeiro, o que possibilitou a realização deste duplo diploma.

Por fim, gostaria de fazer uma menção especial à minha família, que me apoiou durante todos os meus estudos e me ajudou a realizar meu sonho.

Resumo

Desde 1974, o cubo mágico também chamado "*Rubik's cube*" fascina gerações. Alguns tentam resolver o cubo durante anos, porém sem sucesso. E outros, um pouco mais apaixonados, resolvem ele cada vez mais rápido, otimiza-se cada movimento para se aproximar do denominado "número de ouro". Essa busca por aperfeiçoamento leva dezenas de cientistas e engenheiros a trabalhar em um robô de resolução desse quebra-cabeça cada vez mais eficiente.

Nesse contexto, o objetivo deste trabalho foi construir um robô capaz de resolver um cubo mágico (3x3) e competir com um humano (< 2min). Para realizar esse projeto foi necessário detectar o embaralhamento do cubo mágico a partir duma câmara, identificar a resolução otimizada e resolver o cubo mágico de forma eficiente. A máquina foi apta a gerar embaralhamento via um aplicativo. Esses movimentos poderiam ser aleatórios ou comandados pelo usuário. Com isto, o dispositivo tinha que ser fácil de usar e bastante compacta.

Para alcançar esses objetivos, foi utilizada a visão computacional via um aplicativo por meio de ferramentas como "*OpenCV*". Para montar o quebra-cabeça, foi aplicado um método de resolução computacional chamado o algoritmo de duas fases ou "*Kociemba*" auxiliado por servomotor.

Em suma, o robô podia solucionar o cubo mágico de maneira rápida, eficiente e otimizada.

Palavras-chave: Cubo mágico; Robô solucionador; visão computacional; OpenCV; Kociemba;

Abstract

Magic cube solving machine

Since 1974, the Rubik's cube has fascinated generations. Some have tried to solve the cube for years, but to no avail. Others, a little more passionate, solve it faster and faster, optimizing each movement to get closer to the so-called "golden number". This quest for improvement has led dozens of scientists and engineers to work on a robot that can solve this puzzle more and more efficiently.

In this context, the aim of this work was to build a robot capable of solving a magic cube (3x3) and competing with a human (< 2min). To carry out this project, it was necessary to detect the shuffling of the magic cube from a camera, identify the optimized resolution and solve the magic cube efficiently. The machine was able to generate shuffles via an application. These movements could be random or commanded by the user. The device had to be easy to use and very compact.

To achieve these goals, computer vision was used via an application using tools such as "OpenCV". To assemble the puzzle, a computational solving method called the two-phase algorithm or "Kociemba" was applied, aided by a servomotor.

In short, the robot was able to solve the magic cube quickly, efficiently and optimally.

Keywords: Magic cube; Robot solver; computer vision; OpenCV; Kociemba;

Sumário

1. Introdução 7
2. Estado da arte 8
3. Notações 10
   a. Vocabulário 10
   b. Notações dos movimentos 11
   c. Notação para o embaralhamento 11
4. Algoritmo 12
   a. A escolha do algoritmo 12
   b. O algoritmo de duas fases 13
5. Métodos 13
   c. Mecânica 13
   d. Eletrônica 16
   e. Programação 17
6. Solução final 22
   a. A montagem 22
   b. O Software 22
7. Resultado e discussões 23
8. Trabalhos futuros 24
9. Conclusão 25
10. Referências 26
11. Apêndices 28
   a. Desenhos técnicos das peças impressas em 3D 28
   b. Códigos *Python* 32
   c. A solução final: a interface 32

Lista de figuras

Figura 1: Robô solucionador de cubos mágico do MIT 8
Figura 2: Robô solucionador de cubo mágico com motor de passo 9
Figura 3: Robô solucionador de cubo mágico "MindCub3r" 9
Figura 4: Robô solucionador de cubo mágico de GANCube 10
Figura 5: Ilustração nomeando cada parte do cubo mágico 10
Figura 6: Esquema da notação dos movimentos [13] 11
Figura 7: Esquema da notação para o embaralhamento [14] 11
Figura 8: Foto de um servomotor AX-12A 14
Figura 9: Modelização 3D de um suporte motor 15
Figura 10: Esquema representando os ângulos alcançáveis pelo robô 15
Figura 11: Esquema representando os ângulos alcançáveis pelo robô com uma rotação de 45° 16
Figura 12: Modelização 3D do eixo para girar o cubo mágico 16
Figura 13: Foto da placa [22] 17
Figura 14: Esquema elétrico do sistema 17
Figura 15: Diagrama de componentes 18
Figura 16: Esquema que apresenta o sistema HSV 18
Figura 17: Vista frontal do AX-12A de DYNAMIXEL com seus ângulos de alcance 20
Figura 18: Diagrama de caso de uso. 21
Figura 19: Foto do robô montado 22
Figura 20: Imagem não exaustiva da interface do usuário 22
Figura 21: Captura da tela mostrando as diferentes opções disponíveis na guia 32
Figura 22: Captura da tela mostrando uma resolução em tempo real 32

Lista de tabelas

Tabela 1: Medidas e Cálculo do Torque 14

Glossário

CAD (em inglês: Computer Aided Design) Desenho Assistido por Computador

PLA Poliácido Láctico

RGB (em inglês: Red Green Blue) Vermelho Verde Azul

HSV (em inglês: Hue saturation Value) Matriz Saturação Valor

FTDI Future Technology Devices International (empresa especializada em projetar e fabricar circuitos integrados para conectividade e conversão USB) [1]

TTL (em inglês: Transistor-Transistor Logic) Lógica transistor-transistor

CC DCCorrente contínuo

WCA (em inglês: World Cube Association) Associação Mundial de Cubo Mágico

ID Identificador

RAM (em inglês: Random-Access Memory) Memória de acesso aleatório

EEPROM Electrically-Erasable Programmable Read-Only Memory

CFOP Cross, F2L, OLL, PLL

1. Introdução

Inventado pelo Ernő Rubik em 1974, o cubo magico também chamado "*Rubik's cube*", é um dos quebra-cabeças mais conhecidos do mundo. Ele foi inicialmente criado para ensinar arquitetura, mas foi rapidamente bem-sucedido e foi vendido mais de 450 milhões de unidades desde a década de 1980. [2]

Atualmente, amplamente disponível em várias formas (*2x2, 3x3, piraminx, megaminx etc.*), o *Rubik's cube* continua a fascinar gerações. Alguns tentam resolver o cubo no canto de suas mesas há anos, sem sucesso. Outros, um pouco mais apaixonados, estão tentando resolvê-lo cada vez mais rápido e criaram o esporte chamado "". Notamos que o atual recorde mundial nessa disciplina é de Max Park, que resolveu o quebra-cabeça em somente 3,13 s em 2023. [3]

Devido à essa popularidade, cientistas particularmente matemáticos estudaram a questão. Eles calcularam que o número de embaralhamentos possíveis é 43 252 003 274 489 856 000. Desta forma eles tentam otimizar a resolução do cubo, a fim de se aproximar do denominado "*God's number*" [4] (número de Deus). Ou seja, o número máximo de movimentos necessários para resolver qualquer combinação do cubo. Em 2010, foi comprovado que é possível resolver um cubo magico em 20 movimentos. [5]

Ao estudar o cubo mágico, eles conseguiram estabelecer várias teorias e propriedades matemáticas como a teoria dos grupos que permite resolvê-lo com algoritmo computacional em quase 20 movimentos.

Isso atraiu dezenas de cientistas e engenheiros para trabalhar em um robô de resolução cada vez mais eficiente. Essas máquinas são capazes de analisar o cubo e girar as faces para resolvê-lo em tempo recorde, sem intervenção exterior.

Este projeto consistiu em desenvolver um robô capaz de identificar o embaralhamento e resolver um cubo magico de forma eficiente, rápida e totalmente autônoma.

Inicialmente, serão apresentadas uma breve visão geral do estado das artes e uma análise das diferentes tecnologias de máquina usadas nos últimos dez anos, assim como os diferentes métodos de resoluções de um cubo magico que inclui desde métodos tradicionais de solução de problemas até os usados por computadores.

Além disso, este trabalho será dividido em três categorias principais: mecânica, computação e eletrônica.

Para a parte mecânica, foi desenvolvida uma máquina de resolução com seis eixos motorizados com desempenho capaz de competir com um humano (<2 min). Para isso, foi usado um software de Desenho assistido por computador (CAD) e, em seguida impressos os vários elementos em 3D.

Sobre a programação, foi usada a visão computacional por meio de ferramentas como "*OpenCV*" [6] e de duas câmeras posicionadas em perspectiva isométrica. Isso nos permitiu detectar as cores de cada face do cubo e, assim, deduzir seu embaralhamento. Essas informações são apresentadas por meio de uma interface gráfica no computador.

Em seguidante, foi aplicado um método de resolução computacional chamado o algoritmo de duas fases ou "*Kociemba*" [7] que vai permitir encontrar a solução otimizada rapidamente. Desta forma, a máquina é capaz de enviar os movimentos necessários para resolver o cubo mágico para os seis motores.

Finalmente, foi desenvolvido o circuito eletrônico que controla os seis motores por meio de um microcontrolador. A máquina também é capaz de gerar embaralhamento via uma interface gráfica. Esses movimentos podem ser aleatórios ou comandados pelo usuário.

Em suma nesse relatório, vamos tratar na primeira parte do estado das artes, as notações usadas e as metodologias de resolução. Depois, vamos abordar as metodologias usadas para esse projeto incluindo

as partes mecânica, de programação e eletrônica. Então, serão estudados os resultados e a discussão das diferentes metodologias. E este documento será encerrado com a conclusão deste trabalho.

2. Estado das artes

Para poder escolher todos os componentes e dimensionar a máquina, analisamos o que já foi feito no passado. Essas máquinas têm objetivos diferentes. Alguns são projetados para serem rápidos e outros para ajudar na resolução.

De fato, desde 2024, o recorde mundial de resolução de um cubo mágico com uma máquina é de 0,305 s. Esse desempenho realizado pelo robô da empresa japonesa Mitsubishi Electric [8]. A máquina bateu o recorde anterior de 0,38s estabelecido por um outro robô em 2018 [9]. Aquele foi realizado pelo Ben Katz, estudante de pós-graduação do MIT assistido de Jared Di Carlo. Ele otimizou o mais possível a parte hardware e software. Para identificar cores, ele usou duas câmeras posicionadas em perspectiva isométrica. Isso significa que é possível ver todos os lados do quebra-cabeça sem usar uma câmera para cada face. Por isto, torna a detecção de cores um pouco mais complicada, pois faz-se necessário uma fonte de luz homogênea com bom contraste, e é por isso que ele coloriu um dos lados de preto.

O algoritmo de *Kociemba* também chamado algoritmo de dois fases é utilizado para resolver o quebra-cabeça. Este algoritmo é um dos mais populares para esse tipo de projeto devido à sua eficiência e à otimização do número de movimentos (mais ou menos movimentos para resolver).

Optou-se por usar seis motores CC "*Kollmorgen ServoDisc U9/N9*" com *encoders* posicionados na peça central do cubo para mais velocidade. Essa técnica exige que a face da peça central seja removida para que o eixo do motor possa inserir no eixo do cubo. Pois, há uma pequena modificação do *Rubik's cube*.

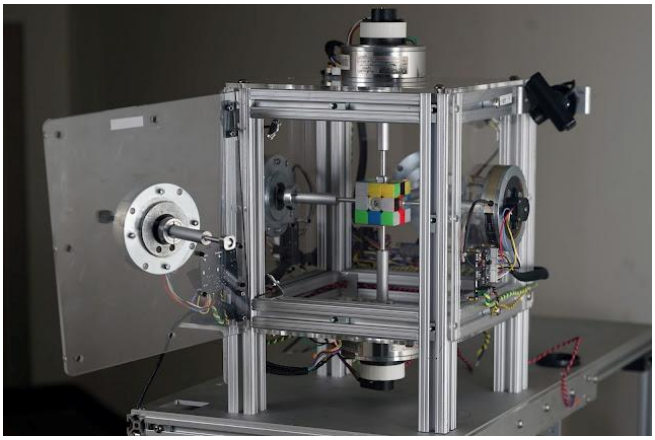


Figura 1: Robô solucionador de cubos mágico do MIT

Outros exemplos de máquinas feitas para velocidade como esta máquina impressa em 3D de seis eixos que usa motores de passo "Nema 17" [10]:

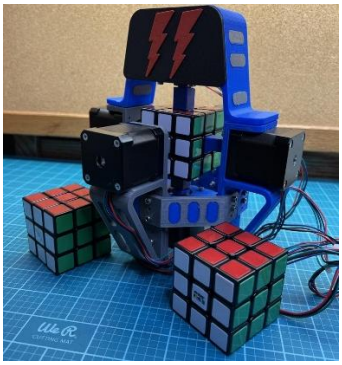


Figura 2: Robô solucionador de cubo mágico com motor de passo

Essa máquina não é tão otimizada quanto a anterior, mas é mais fácil de construir e usa os mesmos princípios gerais. Ela consegue resolver um cubo em menos de 10 s, no entanto o equipamento não permite a detecção do embaralhamento.

Há também robôs feitos de *LEGO*® MINDSTORMS® que são muito mais acessíveis. Por exemplo, um dos robôs mais conhecidos é o "*LEGO MINDSTORMS EV3*". [11]

Este robô não é rápido, mas foi projetado para ser construído por um público mais jovem. Ele permite a rotação do cubo no eixo z, usando um servomotor e rotação no eixo x, por meio de um braço.

Ele pode detectar cores de uma forma não otimizada. A máquina usa um sensor de cores para identificar cada quadrado, um por um. Isso demora mais que duas câmeras.

Figura 3: Robô solucionador de cubo mágico "MindCub3r"

Por fim, a empresa *GANCube* comercializou uma máquina chamada "*GAN robot*" [12] com servomotores posicionados em cinco eixos. É uma máquina compacta e de alto desempenho, ela usa inteligência artificial e pode resolver um cubo em menos de 5 segundos, sendo usada para auxiliar as pessoas em sua resolução ou treinamento.

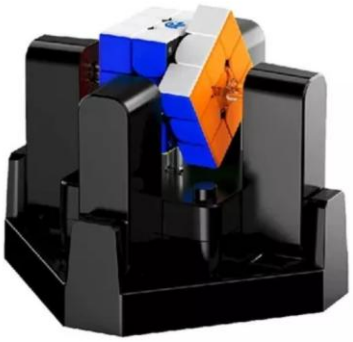


Figura 4: Robô solucionador de cubo mágico de GANCube

Esse robô não exige que você remova a face central do cubo o que é uma vantagem real, mas tem que usar um cubo mágico que possa entrar no robô.

A única desvantagem dessa máquina é que ela tem apenas cinco eixos. Portanto, se você tiver um movimento na parte superior, terá de encontrar uma série de movimentos alternados, o que acaba sendo um pouco menos eficiente. No entanto, há uma economia de custo e energia com esse método.

3. Notações

Para entender melhor as diferentes notações usadas neste relatório, é importante revisar brevemente alguns conceitos.

a. Vocabulário

Nessa parte será explicado, o vocabulário usado nesse relatório em relação com o cubo mágico.

Rubik's cube, refere-se ao outro nome do cubo mágico.

Face refere-se a um único lado do cubo composto por 9 adesivos.

Aresta refere-se aos cubos que têm apenas duas cores anexadas a eles.

V*értice* refere-se aos cubos que têm 3 cores anexadas a eles.

C*entro* se refere aos cubos que têm apenas uma cor anexada a eles.

Movimento é o deslocamento de uma determinada face em 90, 180 ou -90° graus.

Rotação é o giro de todo o cubo sem mover nenhuma face.

S*peedcubing* é a atividade de resolver um cubo mágico os seus derivados o mais rápido possível.

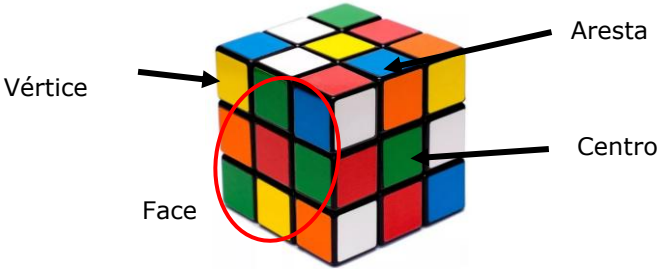


Figura 5: Ilustração nomeando cada parte do cubo mágico

b. Notações dos movimentos

Para entender melhor, esse projeto, tem que entender as notações dos movimentos usadas.

Nesse relatório será usado as notações oficiais de *WCA*, o organismo internacional que regulamenta a disciplina do *speedcubing*.

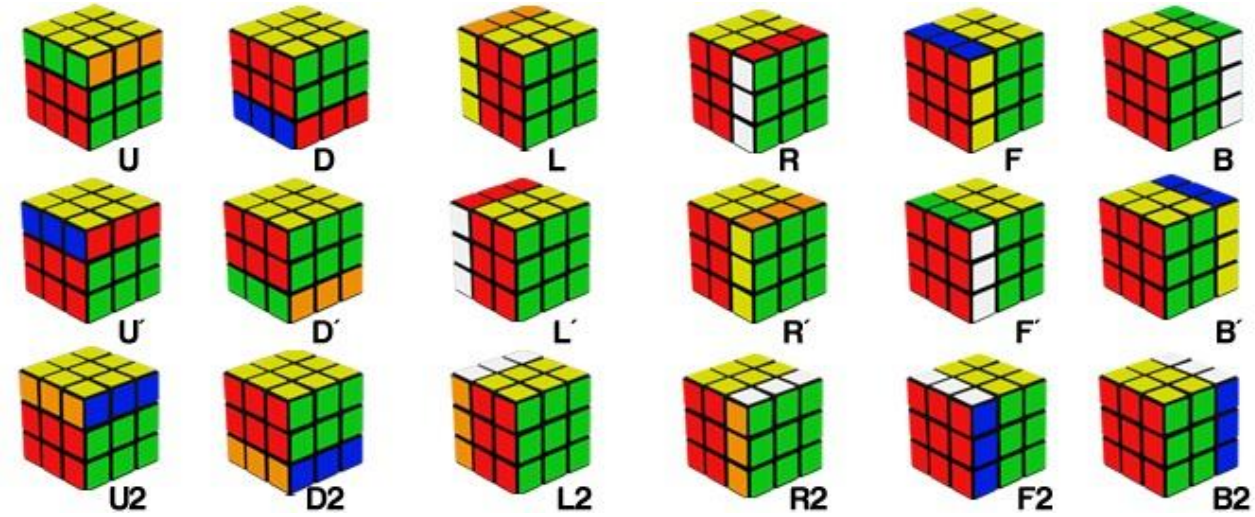


Figura 6: Esquema da notação dos movimentos [13]

Notamos que um movimento X, corresponde a girar a face X de 90° no sentido horário. Um movimento X' corresponde a girar a face X de 90° no sentido anti-horário. Por fim, um movimento X2 corresponde a girar a face X de 180°. As letras significam *Up* – Em cima, *Left* - Esquerda, *Front* – A frente, *Right* – a direita, *Back* -atrás e *Down* -sob.

c. Notação para o embaralhamento

Para determinar como o cubo mágico é misturado, cada faceta corresponde a uma face e um número representando a posição da peça na face. Isso facilitará o uso de algoritmos de resolução de computador. Os nomes das posições das peças do cubo são as seguintes:

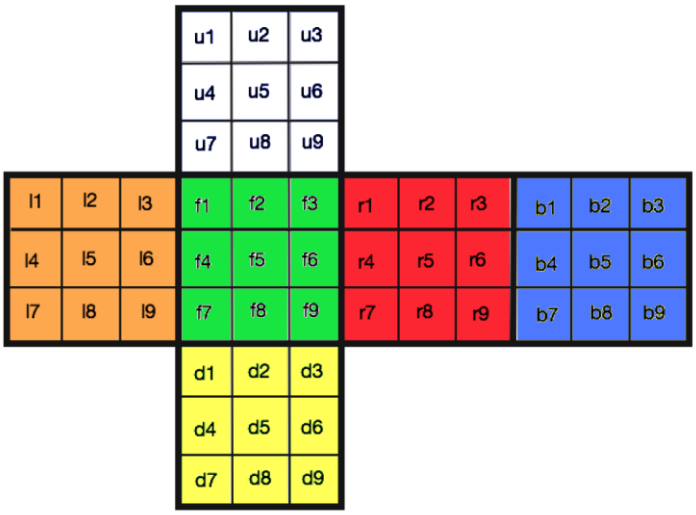


Figura 7: Esquema da notação para o embaralhamento [14]

Com isso, podemos formar uma cadeia de caracteres correspondente ao embaralhamento atual, o que facilitará a leitura pelo algoritmo. Esta cadeia de caracteres tem a seguinte ordem: u1, u2, u3, u4, u5, u6, u7, u8, u9, r1, r2, r3, r4, r5, r6, r7, r8, r9, f1, f2, f3, f4, f5, f6, f7, f8, f9, d1, d2, d3, d4, d5, d6, d7, d8, d9, l1, l2, l3, l4, l5, l6, l7, l8, l9, b1, b2, b3, b4, b5, b6, b7, b8, b9.

Dessa forma, uma cadeia de caracteres de definição de cubo "UDL" significa que na posição u1 temos a cor U, na posição u2 temos a cor R, na posição u3 temos a cor D e entre outros.

Notamos que nesse relatório as notações das faces têm sempre as mesmas cores: F = Verde, U = Branco, R = Vermelho, L = Laranja, D = Amarelo e B = Azul.

Isso corresponda a posição oficial para embaralhar da *WCA*.

4. Algoritmo

a. A escolha do algoritmo

Conforme mencionado na introdução, há 43.252.003.274.489.856.000 maneiras possíveis de embaralhar um *Rubik's cube*, portanto, é difícil encontrar uma solução sem nenhum método, muito menos conhecer todas as combinações possíveis. É por isso que foram inventados diferentes métodos para facilitar a solução do cubo mágico.

Atualmente, a maioria dos melhores *speedcubers* usa o método CFOP [15] (*Cross, F2L, OLL, PLL*), que leva menos de seis segundos se executado por um especialista, mas custa cerca de 55 movimentos por resolução depois de dominado. Esse tipo de algoritmo não é adequado para resolver um cubo mágico com o menor número possível de movimentos, e é por isso que métodos como ROUX [16]ou PETRUS [17] são mais adequados para a disciplina FMS. Com um pouco de reflexão, o método Roux ou o método Petrus pode ser usado para resolver um cubo mágico com cerca de 28 movimentos.

Entretanto, no processo de resolução de um cubo usando um robô, é desejável otimizar o número de movimentos para resolver o cubo no menor tempo possível. Para isso, foram desenvolvidos algoritmos de solução baseados em computador que permitem que o cubo seja resolvido em menos movimentos do que os métodos tradicionais usados por humanos. De fato, graças ao poder da computação, os métodos de solução mais avançados podem ser usados em menos tempo. O objetivo é chegar o mais próximo possível do número de ouro (20 movimentos) e resolver o cubo o mais rápido possível.

Além da aprendizagem profunda e da força bruta, há três métodos clássicos usados para resolver um cubo mágico: o algoritmo de duas fases (*Kociemba*),o método de *Thistlethwaite* e o método de *Korf*.

Neste trabalho, a ênfase será dada aos métodos clássicos de solução. De fato, a aprendizagem profunda e a solução por força bruta (tentar todas as combinações até encontrar a correta) exigem recursos desnecessários para esse projeto.

Um desses algoritmos é o método de *Thistlethwaite* que pode ser resolvido em 4 etapas e com cerca de 52 movimentos.

O método de *Kociemba* reduz esse número de movimentos para ≤ 30 movimentos para a primeira solução encontrada e pode ser reduzido para ≤ 20 movimentos. O tempo de resposta é rápido e esse método permite que uma solução seja encontrada todas as vezes. Ambos desses métodos usam a teoria de grupo que será mais explicada na parte seguinte. [18]

O método de *Koft*, por outro lado, usa o princípio da pesquisa A* com heurística, consome muitos recursos e o tempo de cálculo é exponencial.

Concluindo, por motivos de praticidade, tempo de resposta e otimização do número de movimentos, optamos por trabalhar com o algoritmo *Kociemba* de duas fases.

b. O algoritmo de duas fases

Esta seção descreve a operação básica do algoritmo de duas fases. Como mencionado acima, esse método é baseado na teoria de grupos. Ele pode ser dividido em duas fases.

A primeira consiste em encontrar uma sequência de movimentos que leve uma posição arbitrária do cubo a qualquer posição no subconjunto H. Esse subconjunto é formado por todas as posições do cubo.

Para fazer parte desse subconjunto o cubo precisa que todos os vértices e arestas estejam orientados corretamente. Os cubos de borda que pertencem à linha do meio estão localizados na linha do meio. Essa fase pode ser resolvida em um máximo de 18 movimentos.

Essas características são preservadas pelas jogadas U, U2, U', D, D2, D', R2, L2, F2, B2 chamadas de conjunto A. Observamos que esses movimentos são suficientes para transformar cada posição de H em um estado resolvido.

Depois de reduzir o número de soluções possíveis graças à simetria, e aplicamos a segunda fase para resolver o cubo mágico em menos de 12 movimentos. [19]

5. Métodos

Para concluir o projeto, o trabalho foi dividido em três partes: eletrônica, mecânica e programação

c. Mecânica

Para começar, é importante definir corretamente o número de eixos usados na máquina. De acordo com o estado da arte, os melhores resultados em termos de tempo são aqueles obtidos com um robô de 6 eixos.

Por esse motivo, decidimos trabalhar com 6 eixos de motor afim de otimizar o tempo de resolução. Com isso em mente, a parte mecânica foi dividida em cinco subseções.

- Escolha do motor

Nesta seção, nos concentramos na escolha do motor. Para este projeto, precisamos escolher um motor CC que seja adequado às nossas necessidades. Ele precisa ser preciso, rápido e, acima de tudo, ter torque suficiente para girar uma face.

Para este projeto, usaremos um cubo mágico da marca *Moyu* vendido em qualquer loja de brinquedos. Esse cubo é bastante permissivo e permite que você gire uma face mesmo que ela não esteja perfeitamente alinhada. Estamos buscando uma precisão de 0,5°. Também queremos ser capazes de resolver o cubo mágico em menos de 1 minuto, com, na pior das hipóteses, 30 movimentos,

$$v = \frac{N_{movimentos}}{t} = \frac{30}{60} = 0{,}5\ movimento/s\ ; \quad (1)$$

Para simplificar, presumimos que um movimento será menor que uma revolução completa. Portanto, no mínimo, queremos uma velocidade de rotação de $\pi\ rad/s$.

Por fim, para descobrir a quantidade de torque necessária, foram utilizados os dados medidos por C. Wen. [20]

Ele mediu a massa para um cubo mágico original, profissional e genérico. Para isso, ele colocou uma massa na ponta do cubo mágico até que ele virasse.

Com essas medidas, podemos calcular o torque necessário para girar uma face. Sabemos que os cubos mágicos têm o tamanho de 57mm. Assim,

$$T[N.m] = m[kg] * \frac{L_{Cubo}[m]}{2} * g = m * \frac{0{,}057}{2} * 9{,}81 = m * 0{,}28 \quad (2)$$

Com os valores medidos pelo C.Wen temos a seguinte tabela.

Tipo de Cubo	Teste 1 [Kg]	Teste 2 [Kg]	Teste 1 [Kg]	Massa Média[Kg]	Torque Médio[N.m]
Profissional	0,066	0,101	0,088	0,085	0,0238
Original	0,231	0,209	0,241	0,227	0,06356
Genérico	0,090	0,034	0,087	0,07	0.0196

Tabela 1: Medidas e Cálculo do Torque

Portanto, decidiu-se usar o servomotor AX-12A da *Dynamixel*. Esse servomotor tem uma precisão de 0,29°, um torque de 1,5 N.m a 12 V e 1,5 A e uma velocidade de rotação sem carga de 59 rev/min 12V. Portanto, ele atende a todas as especificações exigidas. [21]

Deve-se observar também que, no modo de controle de ângulo, ele opera somente entre 0° e 300°.

Figura 8: Foto de um servomotor AX-12A

- A caixa

Para construir a estrutura, precisávamos de uma caixa larga o suficiente para que as câmeras pudessem ver todo o cubo. Por isso, decidimos usar barras de metal de um kit da *Vexrobotics* para fazer um cubo de 42 cm x 42 cm x 42 cm. Isso permitirá reutilizar peças.

- Suportes dos motores

Para suportar os motores, era necessário criar uma estrutura capaz de ajustar a distância entre o cubo e os motores permitindo também remoção do cubo com facilidade. Assim, decidiu fazer um suporte em duas partes. A primeira é fixada na estrutura e a segunda no motor, sendo as duas partes conectadas por parafusos. Isso permite que a distância seja ajustada deslizando as duas partes juntas e, em seguida, fixando-as com os dois parafusos. O suporte do motor foi modelado usando o software *Solidworks* e impresso na impressora 3D em material PLA. O desenho técnico pode ser encontrado no apêndice.

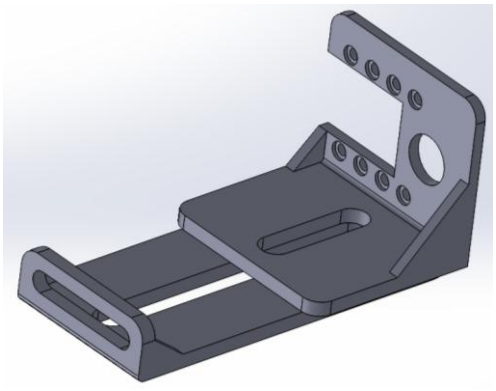


Figura 9: Modelagem 3D do suporte do motor

- Os eixos

Para mover as faces do cubo mágico, foram criadas duas soluções. A primeira é segurar o cubo pela lateral e a segunda pelo eixo central. Como a primeira solução não permitia que todas as faces girassem livremente, optou-se por girar uma face por meio do eixo central. Para isso, levamos em conta os recursos específicos do eixo central do cubo e o motor para desenvolver um eixo personalizado. Como o motor não gira 360° com controle de posição, os eixos tiveram de ser modificados para que as faces do cubo mágico pudessem ser giradas nas 4 posições possíveis.

Em um círculo trigonométrico definido em $]-\pi;\pi]$, o motor não pode alcançar posições entre

$]\frac{5\pi}{6};\pi] \cup]-\pi;-\frac{5\pi}{6}[$ Conforme mostrado no desenho abaixo.

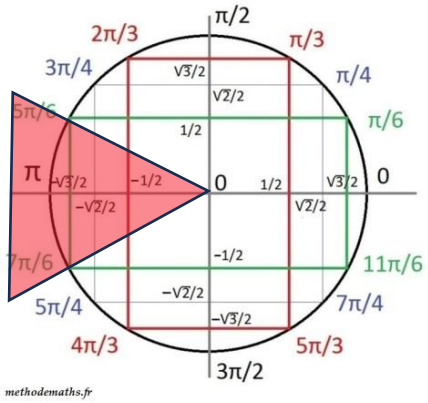


Figura 10: Esquema representando os ângulos alcançáveis pelo robô

Como precisamos acessar os ângulos $0, \pi, \frac{\pi}{2}\ e\ -\frac{\pi}{2}$, tudo o que precisamos fazer é colocar o ponto cego entre essas duas posições, conforme mostrado abaixo.

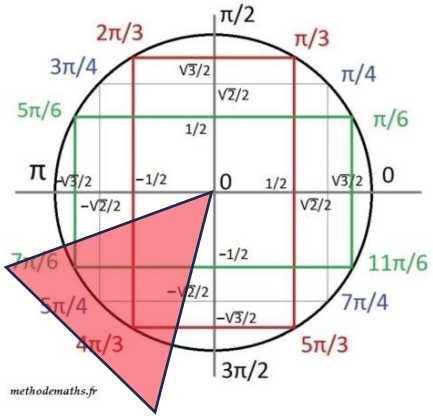


Figura 11: Esquema representando os ângulos alcançáveis pelo robô com uma rotação de 45°

Assim, entre o eixo do motor e o eixo central do cubo, foi feita uma rotação para evitar esse problema.

O seguinte eixo foi modelado usando o software *CAD SolidWorks* e foi impresso em uma impressora 3D em PLA. O desenho técnico pode ser encontrado no apêndice.

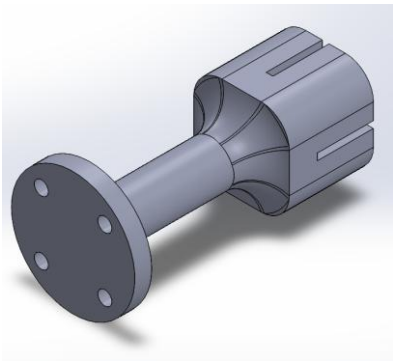


Figura 12: Modelagem 3D do eixo para girar o cubo mágico

- Montagem

Para montar a caixa, os diferentes elementos foram fixados com parafusos e porcas. Foi necessário atenção ao alinhamento correto dos eixos.

d. Eletrônica

Para os componentes eletrônicos, usamos uma placa *Arbotix-M*, seis motores AX-12A da *Dynamixel* e duas câmeras.

Para o controle dos motores AX-12A da *Dynamixel* usamos uma placa *Arbotix-M* projetada especificamente para controlar esses motores. Essa placa tem um microcontrolador AVR de 16 MHz (ATMEGA644p) e foi escolhida porque possuir 3 portas do tipo TTL adequadas aos motores. Os motores foram ligados em série, de modo que apenas uma porta TTL foi necessária para controlar todos os seis motores. Veja abaixo uma foto da placa.

Figura 13: Foto da placa [22]

Por meio de um cabo USB/Serial foi possível programar e integrar o controle do motor ao programa Python, enviando as informações necessárias pela porta serial do *Arbotix-M.*

Além disso, para poder identificar as cores, decidiu-se usar câmeras. Após algumas pesquisas, a visão computacional pareceu ser a solução mais adequada. Isso reduz consideravelmente o número de sensores e/ou o tempo de detecção (se for usado um sensor de cor).

Para otimizar o número de câmeras e o tempo de detecção, decidimos colocar duas câmeras em visão isométrica. Isso significa que podemos capturar três faces ao mesmo tempo, de modo que não precisamos nos mover para capturar todas as cores ou usar uma câmera por face. A desvantagem desse método é que o eixo do motor pode ocultar um lado, dificultando assim a detecção.

Abaixo, o esquema elétrico do sistema.

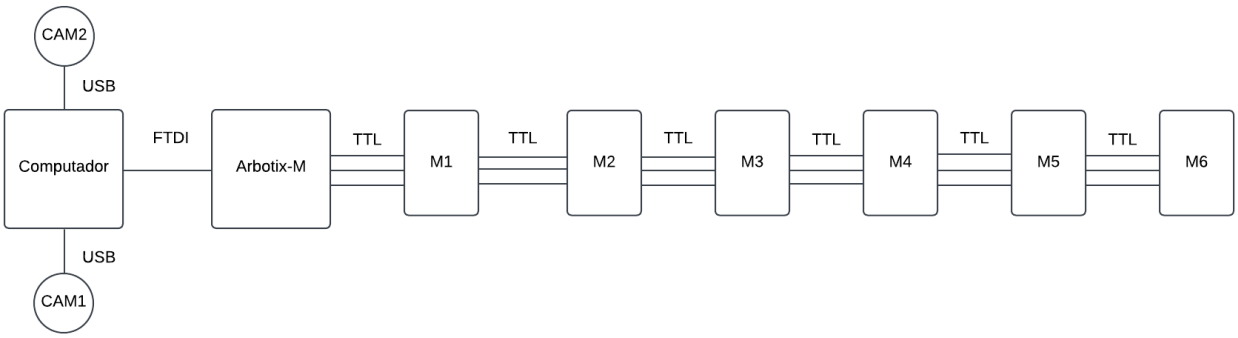


Figura 14: Esquema elétrico do sistema

e. Programação

Finalmente, para a parte de programação, o programa foi dividido em várias partes. O programa inclui a parte de visão assistida por computador, na qual detectaremos o embaralhamento do cubo mágico. Em seguida, falaremos sobre o algoritmo de resolução usado. Depois que o algoritmo tiver sido calculado, o controle do motor será discutido. Por fim, o software foi desenvolvido para facilitar a interface do usuário.

- Arquitetura geral

Para facilitar a programação, cada parte foi dividida em arquivos. A arquitetura usada é a seguinte.

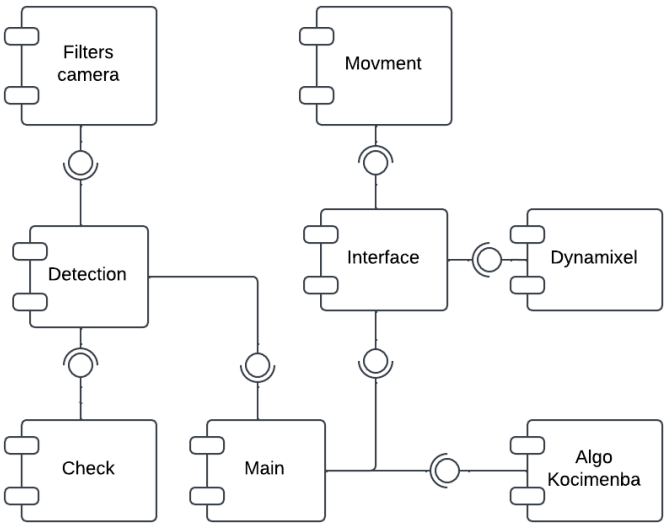


Figura 15: Diagrama de componentes

- Os componentes "*detection*", "*check*" e "filters_camera" se ocupam da parte da visão computacional.
- Os componentes "*Movment*" e "*Dynamixel*" se ocupam da parte de movimento.
- "Algo *kociemba*" permite de converter as cores do cubo mágico em uma cadeia de caracteres emendável pelo algoritmo de *kociemba*.
- A parte interface corresponde ao software, ela também manda os movimentos (convertido pelo componente "*movment*") ao motor usando as funções do componente "*Dynamixel*".
- O componente "*Main*" permite de reunir todos os componentes e de executar o programa na sua globalidade.

Todos esses componentes serão explicados em baixo e os códigos podem ser encontrados no apêndice.

- Visão computacional

Para esta seção, o objetivo foi detectar as cores em cada face do cubo mágico. Como abordado antes, as câmeras fazem a aquisição das imagens em vista isométrica. Assim foi possível detectar 3 faces por câmeras. Para fazer isso, foi usada a biblioteca *OpenCV* em *Python*.

Para começar, foram criadas máscaras para reconhecer cada cor. Inicialmente converte-se a imagem recebida no sistema *HSV* no código principal.

Nota-se que com esse sistema é possível identificar as cores independentemente da saturação e do brilho. Portanto, a máscara facilitará a identificação das cores vermelha, laranja, amarelo, azul, verde e branco. A imagem abaixo apresenta o sistema *HSV*:

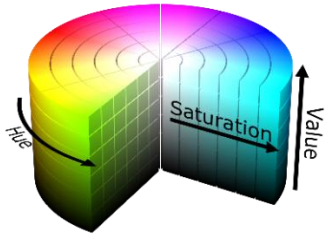


Figura 16: Esquema que apresenta o sistema HSV

Para criar as máscaras, é necessário apenas observar a tonalidade, que é expressa entre 0 e 255 no Python (equivalente a 0 e 2π no círculo trigonométrico). Também, sabemos que o 0 corresponde a uma tonalidade do vermelho. Com isso, foi criada as funções para fazer as máscaras para todas as cores.

Para melhor a detecção com filtros morfológicos. Isso permite uma melhor segmentação da imagem. Esta segmentação fecha os buracos dentro do objeto (fechamento) e depois remove a segmentação errada fora do objeto (abertura). Como abaixo:

$F = Abertura(Fechamento(I)) = Dilatação(Erosão(Erosão(Dilatação(I))))$:

Essas máscaras são calculadas no arquivo *filters_camera*.py

Tendo em vista que as cores podem ser detectadas independentes umas das outras, foram definidas as zonas de deteção de cada quadradinho do cubo. Em cada área, foi calculada a cor mais presente. Com isto, lembrando que os eixos escondem a cor atrás deles, foi desenvolvido um algoritmo para corrigir essa cor. Ele consiste a ter as duas cores presentes nas outras faces do vértice e possibilita a dedução da cor escondida seguindo a ordem das duas cores detectadas. O resultado é armazenado em um dicionário composto de uma matriz para cada face.

Esse dicionário é completado com a função *detection.py* e as correções das cores são feitas com o arquivo *check.py.*

- Algoritmo de resolução

Como explicado acima, o algoritmo usa o algoritmo de duas fases de H. *Kociemba*. Aqui será usado a biblioteca "*Kociemba*" de *Python* [23].

Para resolver o cubo mágico foi usada a função solve(embaralhamento) que recebe como parâmetro o embaralhamento em forma de cadeia de caracteres.

Notamos que a função solve() retorna uma cadeia de caractere correspondendo as movimentos para terminar o cubo com a notação internacional visto acima (exemplo: R2, U' D etc.).

Para usar esta função, é necessário converter o dicionário composto das matrizes de cada face em uma cadeia de caracteres. Isso é feito chamando função do arquivo *algo_kociemba.py* no "*main*".

Essa informação é atualizada na interface que se ocupa de chamar as outras funções para resolver o cubo.

- Movimento dos motores

Antes de poder movimentar os motores, precisa-se converter a solução retornada pela função "*solve*()" vista na parte "Algoritmo de resolução" em angulo.

Para isso, é associado um movimento (uma letra da notação internacional) ao ID do motor. Exemplo: ID 2 = R, ID 1 = D.

Assim, foi construído um dicionário que associa o ID motor correspondente a letra a um ID positivo se o movimento for no sentido horário (como R), ao ID negativo se o movimento é no sentido anti-horário (como R') ou alterar o tipo de ID para *float* se o movimento for seguido de um 2 (como R2).

Exemplo: R: 2; R' : -2; R2: 2.0 e D : 1; D':-1; D2: 1.0

Isso permite de armazenar o sentido e quantos graus são necessários para girar o motor do ID correspondente. Assim é calculado o angulo certo a mandar pegando em conta o angulo morto. Para efetuar esses cálculos será usado o componente *"movment.py".*

Depois, para mandar as informações de movimento para os motores, será usada a linguagem Python associada as bibliotecas *serial* [24] e *dynamixel* (modificada) [25] . Para enviar informações aos motores, é necessário usar o cabo USB/Serial, que converte o barramento serial em TTL. Dessa forma, é possível enviar as informações necessárias para o controle do motor pela porta serial do computador, que será convertida para o barramento TTL dos motores.

Esses motores têm uma tabela de controle armazenada na EEPROM, na qual é possível modificar e/ou ler os parâmetros armazenados nessa tabela, que são os parâmetros gerais do motor. Por exemplo, é possível modificar a ID do motor para ter 6 IDs diferentes para diferenciar os 6 motores do sistema. Isso possibilitará, por exemplo, enviar a posição desejada para cada motor de forma independente. Os limites de ângulo e temperatura também podem ser definidos.

Tem-se uma segunda tabela de controle na RAM que, por exemplo, pode ser usada para enviar a posição desejada para o motor entre 0 e 1023.

Trabalha-se no intervalo $]-\pi;\pi]$ do círculo trigonométrico. Desta forma deve-se converter o valor do ângulo calculado em rad em um número entre 0 e 1023.

Para isso, é necessário consultar ao diagrama apresentado na figura 17, o ângulo $-\frac{5\pi}{6}$ corresponde a 0 e o ângulo $\frac{5\pi}{6}$ corresponde a 1023.

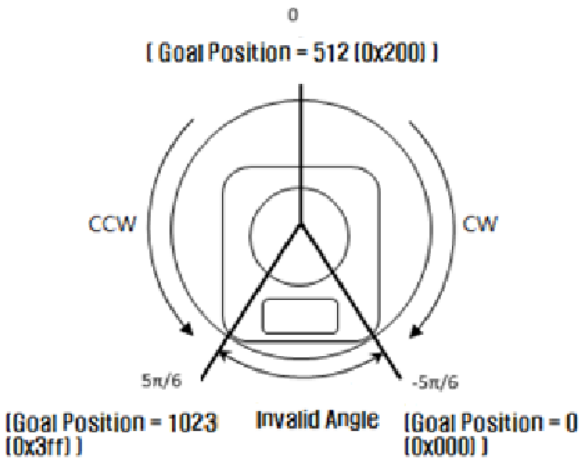


Figura 17: Vista frontal do AX-12A de DYNAMIXEL com seus ângulos de alcance

Como explicado anteriormente, para evitar o ponto cego, foi decidido girar o eixo de 45° ($\frac{\pi}{4}$ rad) para poder colocar o ponto cego entre $\frac{-\pi}{2}$ e $-\pi$.

Esse deslocamento deve ser levado em consideração para encontrar a posição a ser enviada ao motor. Sabe-se que existem entre 0 e 1023 posições possíveis e que temos um intervalo de 300 graus ($\frac{5\pi}{3}$ rad). Com isso, foi aplicado um produto cruzado para descobrir a que corresponde o deslocamento de 45° no intervalo [0,1023], realizamos o seguinte cálculo $1023 * 45/300 = 123$. Assim, um deslocamento de 45° corresponde à posição 123.

Agora, tem que calcular a nova posição correspondente ao angulo 0 rad. Antes o angulo 0 rad era alcançado na posição 512. Portanto, para encontrar a nova posição, calculamos $512 - 123 = 359$. Assim, o angulo 0 rad corresponde à posição 359.

Por fim, para converter a posição enviada pelo motor em radiam, usamos a seguinte fórmula.

$$\hat{A}ngulo = \frac{pos - 359}{1023} * \left(\frac{5\pi}{3}\right) \quad (3)$$

Com isso, também podemos converter o angulo da posição em radiano na posição enviada ao motor usando a seguinte fórmula.

$$Pos = \frac{\hat{a}ngulo}{\frac{5\pi}{3}} * 1023 + 359 \quad (4)$$

Então, o pacote que contém as informações de posição desejadas para o motor n é enviado pela placa *Arbotix-M* com a ajuda da biblioteca dynamixel.py.

- Interface

Foi desenvolvida uma interface gráfica para visualização e configuração do sistema. Para tal, foi utilizada a biblioteca *Python Tkinter*. Esta biblioteca foi utilizada para apresentar o padrão do cubo mágico e permitir ao usuário acessar várias funções por meio de um menu suspenso.

Nessa interface é possível:

- Embaralhar o cubo aleatoriamente ou na ordem desejada e mudar as cores do cubo
- Utilizar a câmara para preencher o padrão com a mistura atual e resolvê-lo
- Resolver o cubo magico
- Monitorar a mistura ou a resolução efetuada em tempo real e o algoritmo de resolução para resolver o cubo.
- Reinicializar o embaralhamento

Essas funcionalidades são apresentadas no seguinte diagrama de caso de uso.

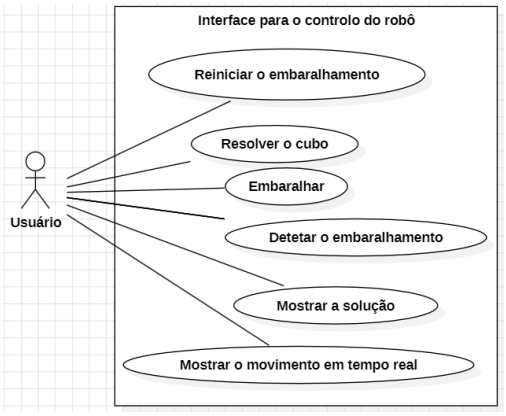


Figura 18: Diagrama de caso de uso.

Quando um movimento é enviado para o cubo, a interface solicita a conversão do movimento em notação internacional fornecido pelo usuário ou pela função solve() do algoritmo de *Kociemba* para o componente "*Movment*". Uma vez feito, ele envia as informações aos motores usando a biblioteca dynamixel.py. A interface apresenta os movimentos executados em tempo real.

O código dessa interface está detalhado no arquivo interface.py

6. Solução final

a. A montagem

A montagem final é apresentada na figura 19. Pode-se observar a caixa metálica, os suportes de motor, os motores Dynamixel e uma caixa para abrigar a placa Arbotix-M

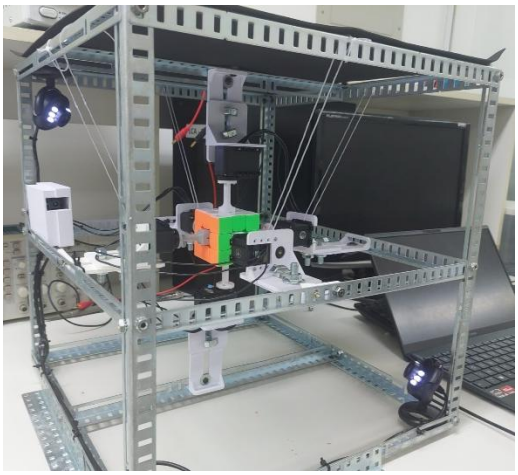


Figura 19: Foto do robô montado

b. O Software

Foi desenvolvido um programa para manipular o cubo na máquina. A interface pode ser vista na figura 20.

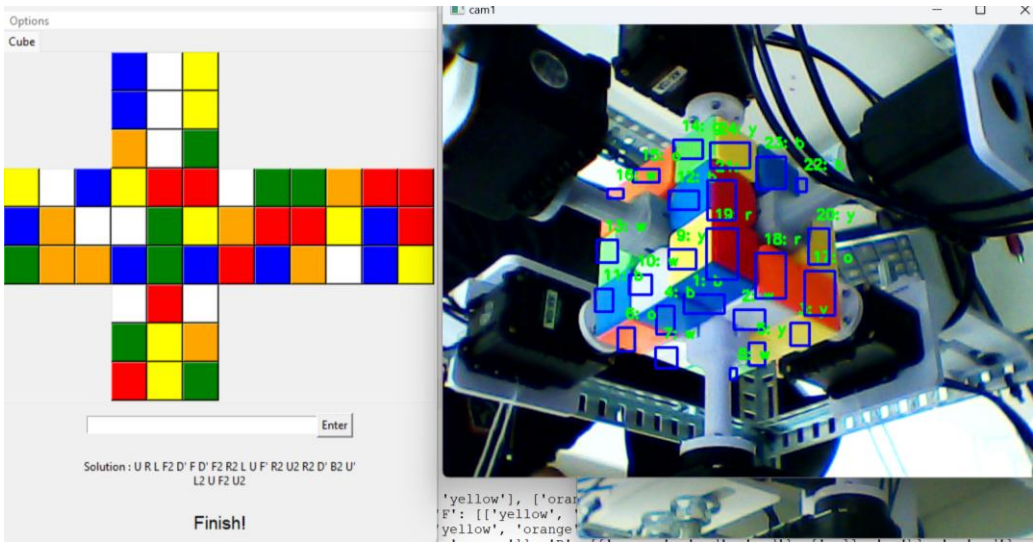


Figura 20: Imagem da interface do usuário

O programa é capaz de solucionar um cubo mágico, detectando o embaralhamento por meio de visão computacional. Também se pode embaralhar o cubo aleatoriamente ou inserir uma série de movimentos de acordo com as notações oficiais. Durante a execução do movimento, a interface exibe o movimento em andamento. É possível corrigir as cores de cada face usando uma paleta de cores. Isso permite inserir um embaralhamento sem usar as câmeras, corrigindo possíveis erros de detecção de cores. As cores não detectadas são exibidas em preto.

Outras fotos ilustrativas as são presentes em Apêndices.

Ele resolve um cubo mágico em menos de 20 segundos. Seu melhor tempo foi de 13 segundos.

7. Resultado e discussões

Deve-se observar que o objetivo de competir com um ser humano na resolução de problemas foi alcançado, pois ele consegue resolver o cubo mágico em menos de 20 segundos, o que equivale a um *speedcuber* de nível avançado.
Além disso, no que diz respeito aos motores, o desempenho geral em termos de tempo poderia ser aprimorado com o uso de outro tipo de motor, como um motor de passo ou um motor CC com controle de posição, para melhorar significativamente o tempo de resolução.

Os eixos do motor permitem que o cubo mágico gire totalmente sem atingir um ponto morto. Esse deslocamento foi corretamente compensado na parte de programação, mas isso leva a movimentos excessivos (em vez de 90° para determinadas posições, são 270° por causa do ponto cego).

Acrescentamos que os suportes do motor permitem que a posição do cubo mágico seja ajustada conforme necessário, mas eles tendem a vibrar quando o motor está em funcionamento, fazendo com que os eixos se movam ligeiramente, de modo que, com o tempo, os eixos não ficam mais exatamente alinhados e o cubo emperra ou o centro gira no vazio. Para corrigir isso, foi necessário adicionar fios tensionados aos suportes do motor para evitar qualquer movimento indesejado. Isso permitiu que o sistema se movesse menos, mas não é suficiente, por isso é necessário repensar o sistema de fixação em geral. Também se pode acrescentar um sistema de detecção de problemas ao sistema de visão auxiliado por computador. Isso evitaria danificar as peças do mecanismo.

Além disso, para evitar o ofuscamento das câmeras, foi adicionado um teto preto, o que reduziu os erros de detecção da câmera, mas a variação de luz ainda tem um leve impacto na detecção nas extremidades. As cores ocultas atrás dos eixos são corrigidas usando o algoritmo de correção de cores, mas isso ainda pode deixar um ou dois erros durante a detecção se houver uma mudança significativa na luz do ambiente. No entanto, esse não é um problema grave, pois pode ser facilmente corrigido manualmente por meio da interface. A detecção de cores é instantânea.

Para melhorar a detecção de cores, o sistema poderia ser mais bem protegido da luz externa e luzes artificiais poderiam ser adicionadas para proporcionar uma iluminação mais constante. A caixa tem 42x42x42cm, o que é bastante grande, portanto, poderíamos considerar diminuí-la, levando em conta a distância focal da câmera e se ela conseguiria ver todo o cubo mágico.

Em suma, os principais objetivos deste projeto foram alcançados, mesmo que ele possa ser aprimorado de modo geral para aumentar seu desempenho.

8. Trabalhos futuros

Para melhorar o desempenho da máquina, as seguintes linhas de pesquisa podem ser desenvolvidas:

- Parte mecânica:
  - Refazer os suportes do motor para estabilizar e reduzir as vibrações causadas pelo motor
  - Aprimorar o sistema de ajuste da posição dos suportes para facilitar a instalação do cubo mágico
  - Reduzir o tamanho do corpo, prestando atenção à distância focal da câmera
  - Escolher motores melhores e mais rápidos
  - Desenvolver um controle eficiente do motor

- Parte elétrica:
  - Desenvolver sua própria placa de controle de motor
  - Usar um *RasberryPi* para miniaturizar o robô.
  - Adicionar luz artificial para controlar as diferenças de luminosidade.
  - Escolher novas câmeras para reduzir o corpo o máximo possível.

- Parte de computação:
  - Melhorar a interface do usuário e por que não criar um aplicativo móvel
  - Adicionar uma verificação de erros (caso o cubo fique preso ou não se mova corretamente).
  - Aprimorar a detecção de cores e automatize a detecção de facetas.

9. Conclusão

Em suma, esse projeto tem objetivo de construir um robô capaz de solucionar o cubo mágico de forma autônoma, eficiente e otimizada, controlada com uma interface do utilizador.

Inicialmente, foi apresentado uma breve revisão do estado das artes e uma análise das diferentes tecnologias de máquina usadas nos últimos dez anos. Também, foram apresentados os diferentes métodos de resoluções de um cubo magico falando dos métodos tradicionais até os usados por computadores.

Foi desenvolvida uma máquina de resolução com seis eixos motorizados com desempenho capaz de competir com um humano em menos de 20 segundos. Isso foi bem melhor que a expectativas iniciais.

Para montar a máquina, decidiu-se usar servomotores AX-12A da *Dynamixel* controlados pela placa . Para criar a estrutura, foi projetada uma caixa de metal com elementos reutilizados e suportes e eixos modelados com o software de CAD e impressos em 3D.

Sobre a programação, foi usada a visão computacional por meio de ferramentas como *OpenCV* e de duas câmeras posicionadas em perspectiva isométrica. Isso nos permitiu detectar as cores de cada face do cubo e, assim, deduzir seu embaralhamento. Notamos que biblioteca *dynamixel* foi usada para enviar os movimentos aos motores.

Em seguinte, foi aplicado um método de resolução computacional chamado o algoritmo de duas fases ou *Kociemba* que demostrou um bom desenho com soluções rápidas e otimizadas (abaixo de 30 movimentos.)

Também, uma interface de usuário foi projetada para permitir movimentos e tarefas aleatórios ou controlados pelo usuário, como usar as câmeras ou resolver o cubo magico.

Por fim, os principais objetivos deste trabalho foram alcançados. Entretanto, em trabalhos futuros, será possível aprimorar os pontos levantados na seção de discussão.

10. Referências

[1] PCM-cable, "Para que serve um FTDI," 20 jun. 2023. [Online]. Available: https://pt.pcm-cable.com/info/what-is-an-ftdi-used-for-83442579.html. [Acesso em 06 dez. 2024].

[2] Ruwix Twisty Puzzle Wiki, "Rubik Ernő, the inventor of the Rubik's Cube," [Online]. Available: https://ruwix.com/the-rubiks-cube/the-inventor-rubik-erno-1974-budapest-hungary/. [Acesso em 25 jun. 2024].

[3] World Cube Association, "Records," [Online]. Available: https://www.worldcubeassociation.org/results/records. [Acesso em 25 jun. 2024].

[4] Ruwix Twisty Puzzle Wiki, "God's Number," [Online]. Available: https://ruwix.com/the-rubiks-cube/gods-number/. [Acesso em 25 jun. 2024].

[5] T. Rokicki, H. Kociemba, M. Davidson, J. Dethridge e L. Garron, "God's Number is 20," [Online]. Available: https://cube20.org/. [Acesso em 25 jun. 2024].

[6] OpenCV team, "OpenCV," [Online]. Available: https://opencv.org/. [Acesso em 06 dez. 2024].

[7] H. Kociemba, "Herbert Kociemba's Homepage," 2020. [Online]. Available: https://kociemba.org/. [Acesso em 06 dez. 2024].

[8] "Japon : un robot bat le record du monde de Rubik's cube," Cnews, 29 maio 2024. [Online]. Available: https://www.cnews.fr/monde/2024-05-29/japon-un-robot-bat-le-record-du-monde-de-rubiks-cube-1504691. [Acesso em 06 dez. 2024].

[9] B. Katz, "The Rubik's Contraption," 07 mar. 2018. [Online]. Available: https://build-its-inprogress.blogspot.com/2018/03/the-rubiks-contraption.html . [Acesso em 06 dez. 2024].

[10] aaedmusa, "Rubik's Cube Solver," 16 mar. 2024. [Online]. Available: https://www.instructables.com/Rubiks-Cube-Solver-2/. [Acesso em 06 dez. 2024].

[11] MindCuber, "How to build MindCub3r for LEGO MINDSTORMS EV3," 20 abr. 2021. [Online]. Available: http://mindcuber.com/mindcub3r/mindcub3r.html . [Acesso em 20 jun. 2024].

[12] GanCube, "GAN Cube Robot Auto Scramble and Solving," [Online]. Available: https://www.gancube.com/products/gan-speed-cube-robot?srsltid=AfmBOoq10ctn7pXfGWREkageT8zZ0tBZUMCG2YJJywMhKCbTI45fjzcU. [Acesso em 06 dez. 2024].

[13] R. Cinoto, "Notação dos movimentos," 12 fev. 2020. [Online]. Available: https://cinoto.com.br/cubomágico/notacao-dos-movimentos/ . [Acesso em 06 dez. 2024].

[14] H. Le Thanh, "Optimally Solving a Rubik's Cube Using," Imperial College London, 2015.

[15] Speedsloving.com, "CFOP method," 31 jul. 2024. [Online]. Available: https://www.speedsolving.com/wiki/index.php/CFOP_method . [Acesso em 06 dez. 2024].

[16] Speedsolving.com, "Roux method," 09 ago. 2024. [Online]. Available: https://www.speedsolving.com/wiki/index.php?title=Roux_method. [Acesso em 06 dez. 2024].

[17] Speedsolving.com, "Petrus Method," 28 ago. 2024. [Online]. Available: https://www.speedsolving.com/wiki/index.php?title=Petrus_Method. [Acesso em 06 dez. 2024].

[18] S. McAleer, F. Agostinelli, A. Shmakov e P. Baldi, Solving the Rubik's Cube Without Human Knowledge, 2018.

[19] P. Baldi, "Twenty-Five Moves Suffice for Rubik's Cube," 2008.

[20] W. Cesar e K. G. Francisco, "Robô Para Solução do Cubo de Rubik," POLI USP, 2020.

[21] Robotis e-manual, "AX-12A," 2024. [Online]. Available: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/. [Acesso em 06 dez. 2024].

[22] Redohm, "Présentation et Utilisation de la carte ArbotiX-M," jun. 2020. [Online]. Available: https://www.redohm.fr/2020/06/presentation-et-utilisation-de-la-carte-arbotix-m/ . [Acesso em 06 dez. 2024].

[23] "Python/C implementation of Herbert Kociemba's Two-Phase algorithm for solving Rubik's Cube," Pypi, 26 jun. 2019. [Online]. Available: https://pypi.org/project/kociemba/. [Acesso em 06 dez. 2024].

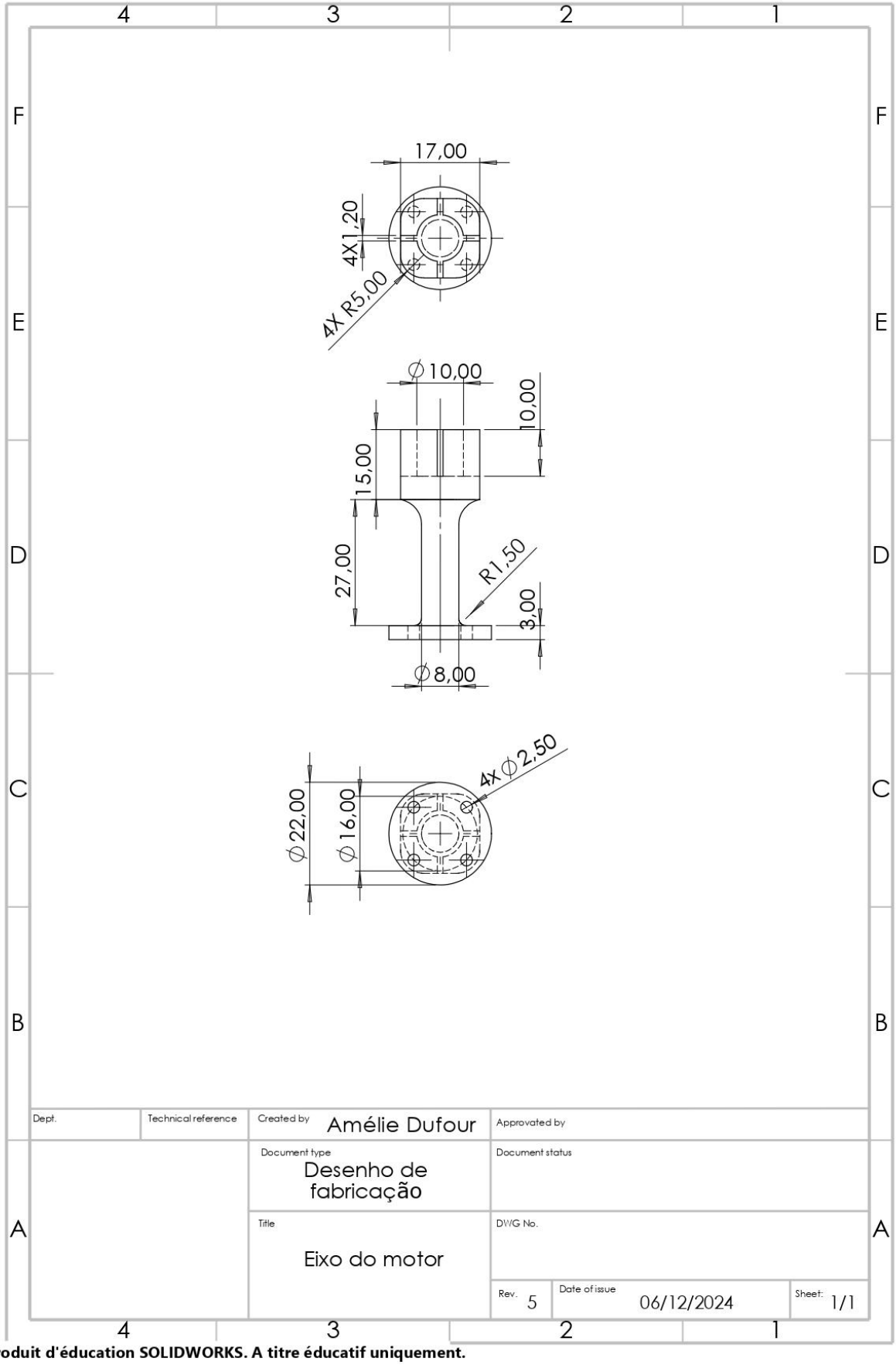
[24] Cliechti, "Python Serial Port Extension," Pypi, 23 nov. 2020. [Online]. Available: https://pypi.org/project/pyserial/. [Acesso em 06 dez. 2024].

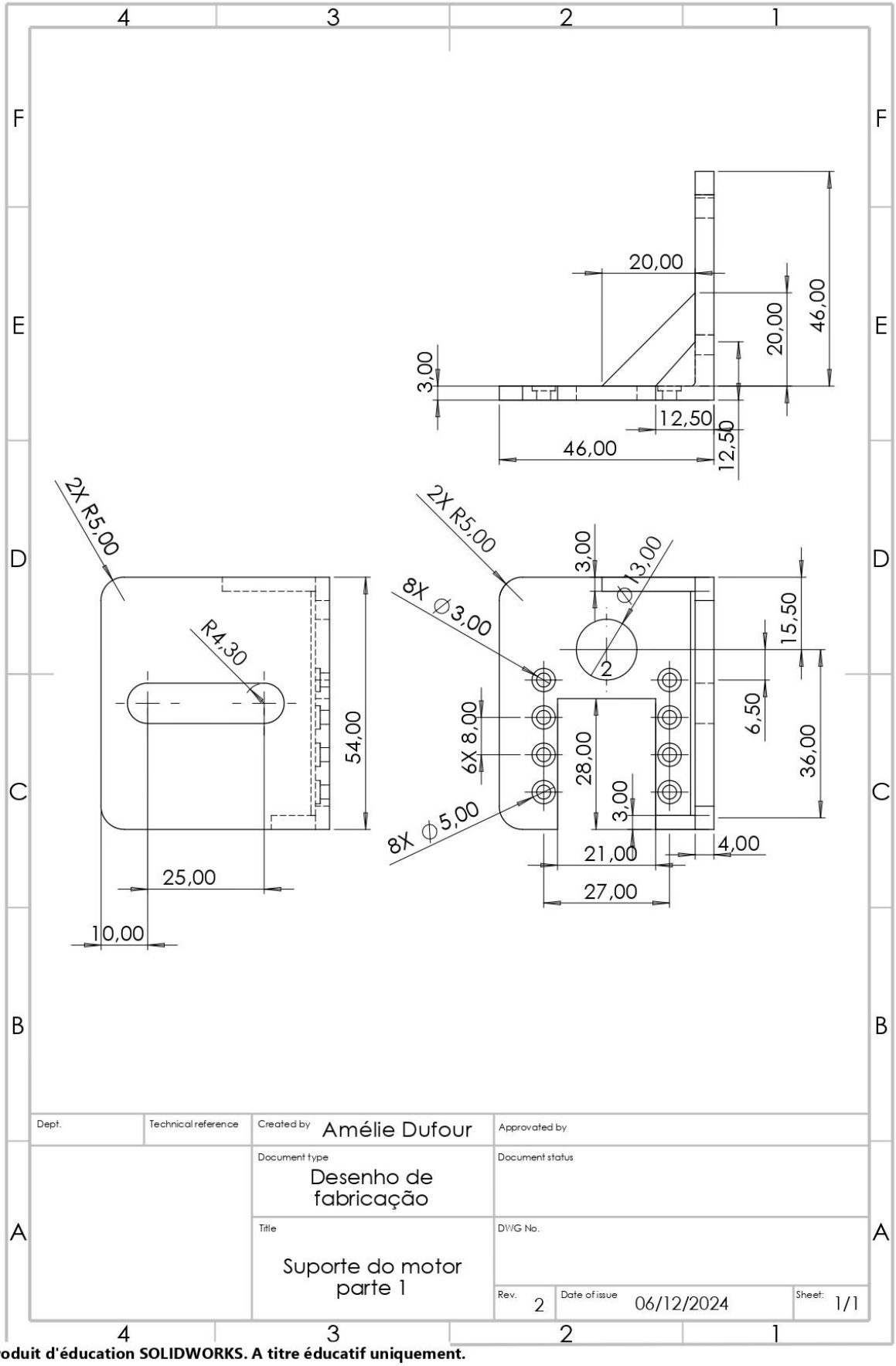
[25] C. Wouter, *Aula Robótica Industrial,* 2024.

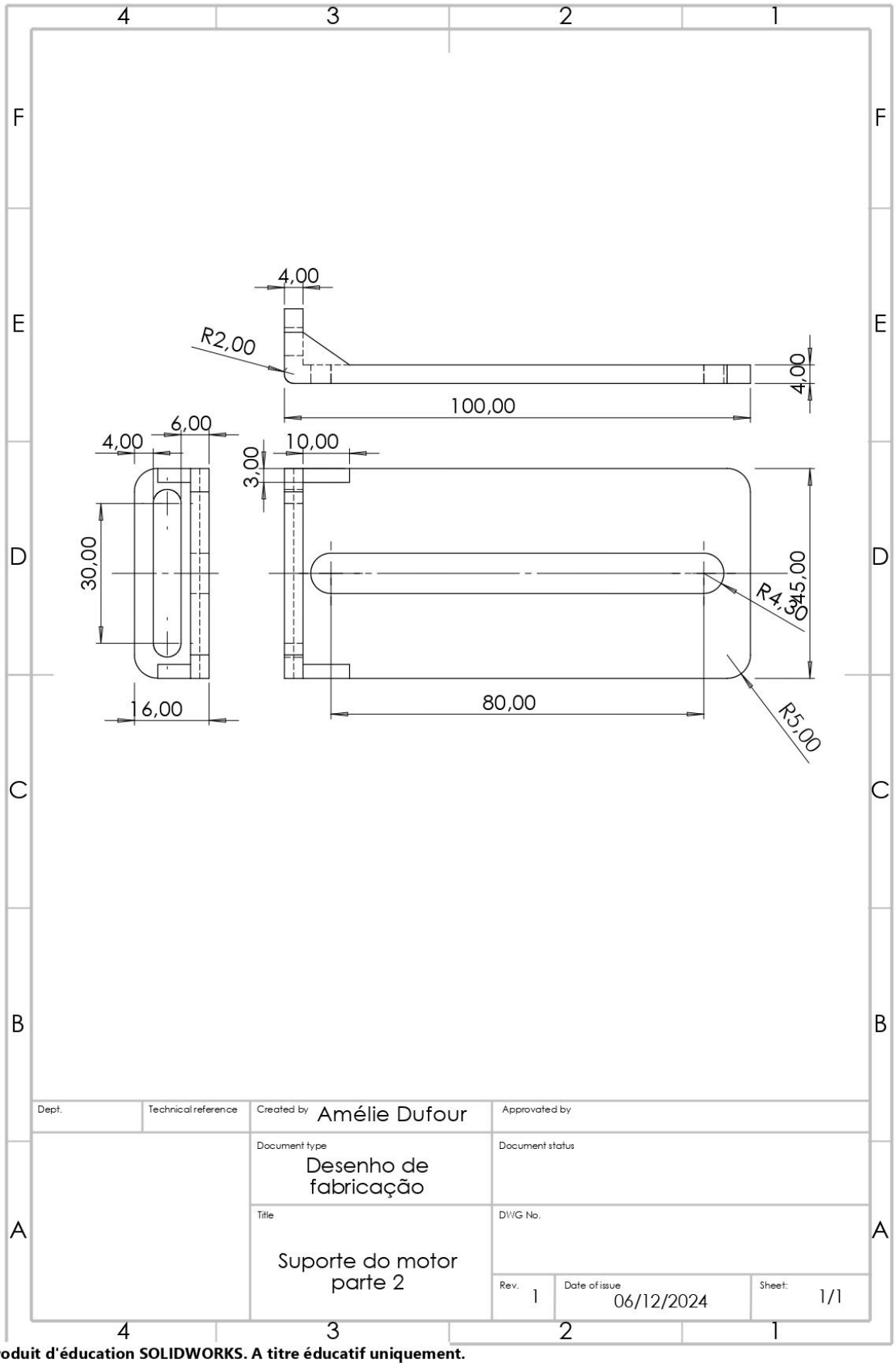
11. Apêndices

a. Desenhos técnicos das peças impressas em 3D

Abaixo, os desenhos técnicos correspondentes ao eixo do motor e ao suporte do motor, respectivamente.







Dept.	Technical reference	Created by Amélie Dufour	Approvated by
		Document type: Desenho de fabricação	Document status
		Title: Suporte do motor parte 2	DWG No.
			Rev. 1 / Date of issue 06/12/2024 / Sheet: 1/1

Produit d'éducation SOLIDWORKS. A titre éducatif uniquement.

b. Códigos *Python*

Neste link, está o diretório do *Github* que contém todo o código usado

c. A solução final: a interface

A imagem abaixo é uma captura de tela da interface, ela mostra todas as opções disponíveis na guia.

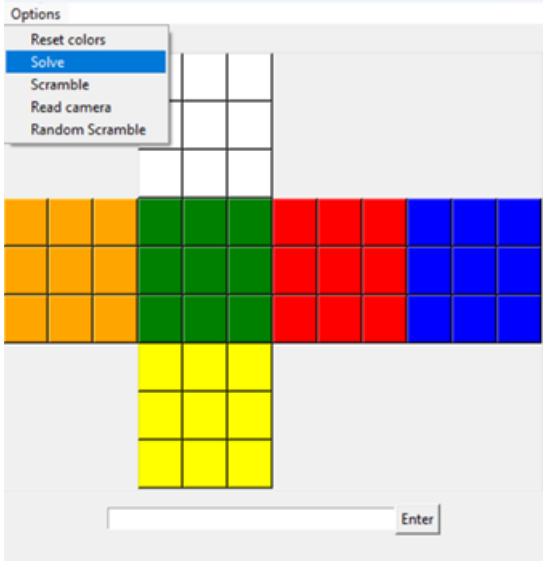


Figura 21: Captura da tela mostrando as diferentes opções disponíveis na guia

Da mesma forma, essa imagem mostra um cubo sendo resolvido pela máquina. O usuário inseriu o embaralhamento do cubo mágico com uma cadeia de caracteres e, em seguida, clicou na guia "*solve*" (resolver). A solução completa é exibida e o movimento pode ser visto em tempo real.

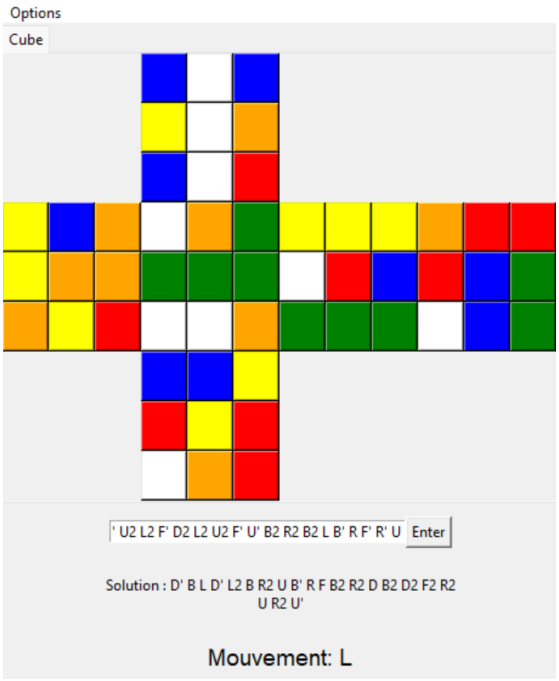


Figura 22: Captura da tela mostrando uma resolução em tempo real