Centro Federal de Educação Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Kesley Roberto Ferreira Silva

Reconhecimento e Simulação da Manipulação de Cubos por um Robô Industrial Utilizando Visão Computacional

Divinópolis
2019

Kesley Roberto Ferreira Silva

Reconhecimento e Simulação da Manipulação de Cubos por um Robô Industrial Utilizando Visão Computacional

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Robótica - Computação e Controle.

Orientador: Prof. Dr. Renato de Sousa Dâmaso

Divinópolis
2019

Kesley Roberto Ferreira Silva

Reconhecimento e Simulação da Manipulação de Cubos por um Robô Industrial Utilizando Visão Computacional

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Áreas de integração: Robótica - Computação e Controle.

Comissão Avaliadora:

Prof. Dr. Renato de Sousa Dâmaso
CEFET-MG / *Campus* Divinópolis

Prof. Dr. Álisson Marques da Silva
CEFET-MG / *Campus* Divinópolis

Prof. Dr. Lucas Silva de Oliveira
CEFET-MG / *Campus* Divinópolis

Divinópolis
2019

Aos amigos de curso que levo para a vida.

Agradecimentos

Agradeço,

- à minha família, que forneceu o apoio necessário para que fosse possível cursar o ensino superior em período integral, mesmo com todas as dificuldades envolvidas;
- Ao meu professor orientador Dr. Renato de Sousa Dâmaso pelo apoio durante todo o curso de Engenharia Mecatrônica, resultando em uma relação de confiança para além do meio acadêmico;
- à Turma 8, pelos diversos momentos que pudemos estar juntos interagindo, divertindo, sendo amigos, esperando futuramente encontrá-los novamente no mercado de trabalho;
- aos membros da Coordenação de Engenharia Mecatrônica, que tornaram possível minha conclusão do curso de forma regular, mesmo com os atrasos gerados em razão de um acidente;
- à minha amiga Maria Vitória, a dupla que se tornou bem mais que companheira dos trabalhos. A afinidade no domínio do conteúdo das matérias acabou se transformando em um relacionamento de amizade e confiança que espero levar para o resto da vida;
- aos meus amigos Gaio, Lucas e Wanderson que estiveram comigo ao longo de todas as festas (2 no total) e acima de tudo, puderam ser um ponto de apoio em todos os momentos difíceis ao longo do curso;
- e aos demais companheiros de CEFET que de alguma forma, colaboraram para minha formação através do companheirismo e respeito sempre presente.

Deve-se aprender sempre, até mesmo com um inimigo.

Isaac Newton

Resumo

A utilização de simuladores tem por principal característica a possibilidade de reprodução contínua de um ambiente de forma que os estados iniciais se mantêm os mesmos. Além disso, a simulação proporciona a aplicação de conteúdo teórico com baixo ou nenhum investimento financeiro permitindo com que sejam estudados diversas fontes de erros em eventuais aplicações reais. Aliada à técnica de visão computacional tais sistemas podem ser utilizados para validar diversos conteúdos abrangendo a área de robótica, envolvendo principalmente técnicas de controle e computação. Este trabalho teve por objetivo o desenvolvimento de uma célula robótica simulada implementada através de um conjunto de linguagens de programação onde o robô, através de visão computacional e uma garra acoplada, realiza o empilhamento com correção de orientação de três cubos iguais com faces coloridas. Foi possível elaborar um modelo funcional do robô Comau Smart5 SiX acionado de forma manual no *software* V-REP e, em seguida, utilizar a API remota via Python para o cálculo da posição das juntas via cinemática inversa pelo método geométrico. Também foi realizada a integração com visão computacional para que os objetos na simulação fossem posicionados de acordo com as informações extraídas das imagens.

Palavras-chave: Manipulação Robótica, Visão Computacional, Smart5 SiX.

Abstract

The use of simulators has as main feature the possibility of continuous reproduction of an environment so that the initial states remain the same. Moreover, the simulation provides the application of theoretical content with little or no financial investment allowing to study various sources of errors in any real applications. Allied to the computer vision technique such systems can be used to validate various contents covering the area of robotics, involving mainly control and computation techniques. This paper aimed the development of a simulated robotic cell implemented through a set of programming languages where the robot, through computer vision and a coupled tool, performs the orientation correction stacking of three equal cubes with colored faces. It was possible to elaborate a functional model of the Comau Smart5 SiX robot manually activated in the V-REP software. It was used the remote API via Python to calculate the position of the joints via inverse kinematics by the geometric method. The integration with computer vision was also performed so that the objects in the simulation were positioned according to the information extracted from the images.

Key-words: Robotic Manipulation, Computer Vision, Smart5 SiX

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Notação

ABS	*Acrylonitrile butadiene styrene*
APC	Acopus PC
API	*Application Programming Interface*
CC	Corrente contínua
CLP	Controlador lógico programável
COD	Extensão de arquivo desenvolvida pela Byte Craft Code Development Systems
DH	Denavit-Hartenber
GDL	Graus de Liberdade
HSV	*Hue, Saturation, Value*
PDL2	Linguagem de programação para controladora Comau
ROS	*Robot Operating System*
TCC	Trabalho de Conclusão de Curso
TP	*Teach Pendant*
USB	*Universal Serial Bus*
V-REP	*Virtual Robot Experimentation Platform*

P	matriz de relação entre os pontos no espaço e na imagem
u	indica a coordenada horizontal de um *frame*
v	indica a componente vertical das coordenadas de uma imagem
ω	representa uma constante de peso utilizada na mudança de base
k_{ij}	termo da matriz de transformação de coordenadas
$G(x, y)$	função de máscara gaussiana bidimensional
σ	desvio padrão a ser considerado
H	matriz de transformação homogênea
R_n^0	matriz de rotação de n à 0
o_n^0	vetor de translação de n à 0
A_i	matriz de DH referente à transformação i
$U_{att,i}$	intensidade do campo atrativo
ζ_i	constante de peso atrativo
d	raio de atuação do campo
η_i	constante de peso repulsivo
ρ_0	distância de influência de um obstáculo
$\rho(o_i(q))$	menor distância entre o_i e o obstáculo
V_{CE}	tensão de coletor-emissor
V_B	tensão de base

Capítulo 1

Introdução

O interesse em métodos de processamento de imagens digitais decorre de duas áreas principais de aplicação: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para percepção automática através de máquinas (GONZALEZ, 2000). No segundo caso, a aplicação de técnicas de visão computacional proporciona a um sistema robótico a capacidade de se adaptar a situações em que transdutores comuns são insuficientes. Em uma linha de produção por exemplo, torna-se possível reorganizar os produtos que estejam fora de sua posição e orientação ideal para sequência na fabricação.

No laboratório de robótica do CEFET-MG / Unidade Divinópolis está disponível para pesquisa de ferramentas e aplicações o robô industrial Comau Smart5 SiX®. O equipamento consiste em uma estrutura antropomórfica de 6 graus de liberdade com repetibilidade de +/- 0,05 mm (COMAU, 2014). Para o acionamento do robô, é utilizada a unidade controladora C5G, onde estão contidas as malhas de controle de posição, velocidade e aceleração do Smart5 SiX, possibilitando sua programação através do *Teach Pendant* TP5 ou via execução de um arquivo .COD traduzido do código fonte PDL2.

Um diferencial do sistema robótico disponível no laboratório da instituição é a disponibilidade da plataforma Comau C5G Open®. Este sistema torna possível a comunicação em tempo real entre a unidade controladora e o computador industrial Acopus PC 910 através do protocolo *Ethernet Powerlink*. Desta forma, pode ser realizada a programação do robô em vários níveis, incluindo a aquisição da leitura bruta dos *encoders* dos motores e adição de sensores externos, como câmeras para visão computacional (FERRARA, 2013). A fim de buscar extrair todo o potencial da ferramenta disponível, foi montado na instituição um grupo de pesquisadores visando investigar a escassa documentação disponível de modo a permitir que os testes iniciais envolvendo a modalidade Open sejam realizados.

Para que possam ser desenvolvidos trabalhos utilizando visão computacional aliados ao robô industrial disponível de forma independente do andamento das pesquisas envolvendo a plataforma Open, as simulações de células de produção robóticas surgem como possibilidade de aplicação. Deste modo, os algoritmos de movimentação e tomada de decisão a partir das imagens capturadas podem ser testados e estudados para que futuramente sejam aplicados

de forma simultânea na simulação e no robô.

Dentre os diversos ambientes de simulação robótica, destaca-se o V-REP desenvolvido pela empresa Coppelia Robotics. Através da API fornecida pela desenvolvedora ou utilizando a integração com o *framework* ROS, torna-se possível realizar a programação de robôs dos mais diversos tipos em C, C++, Python, dentre outras linguagens. A modularidade do programa faz com que possam ser realizadas simulações levando em consideração diversos tipos de geometria e interação entre objetos da cena. Desta forma, torna-se possível a integração com visão computacional para realização de manipulação de cubos com faces de cores distintas, tema deste trabalho.

1.1 Definição do Problema

Devido à crescente automatização dos processos produtivos, busca-se fazer com que os sistemas computacionais e de robótica sejam capazes de tornar automática a execução de tarefas complexas. O reconhecimento de posição e orientação de objetos pode influenciar a tomada de decisão de um robô de montagem (RUDEK; COELHO; JUNIOR, 2001).

Visando aplicar técnicas de manipulação de sinais de imagem e computação em ambientes robóticos, foi proposto o desenvolvimento de uma célula robótica simulada. Serão dispostos cubos com faces de cores distintas, representando um sistema a ser organizado utilizando como referência as posições identificadas através de visão computacional. O robô utilizado será do modelo igual ao disponível no laboratório para que os resultados possam ser comparados com os reais numa eventual aplicação utilizando a plataforma Open.

1.2 Motivação

A utilização de softwares de simulação é de extrema importância no desenvolvimento de técnicas para aplicações robóticas. É de grande importância na atualidade a formação de pessoas capacitadas nesta área, com conhecimento sobre o tema e capazes de elaborar projetos e aplicações robóticas (BERRI; GRASSI JR.; OSORIO, 2015).

A manipulação robótica é uma área de vasta utilização no meio industrial. Sua união com sistemas de visão computacional ainda torna possível a adaptação à imprevisibilidade, tornando tarefas robóticas autônomas como organização de estoque, mais bem preparadas para lidar com interferências externas.

A falta do modelo do robô Comau Smart5 SiX para o V-REP impede que estudos de simulação mais próximos do real sejam desenvolvidos na instituição utilizando a versão gratuita do programa. Assim, após a finalização do trabalho, o modelo será disponibilizado para *download* a quem interessar.

1.3 Objetivos do Trabalho

São objetivos do trabalho a ser desenvolvido:

1.3.1 Objetivos gerais

Este trabalho teve por objetivo o desenvolvimento de uma célula robótica simulada no ambiente V-REP onde o robô, através de visão computacional e uma garra acoplada, realiza o empilhamento com correção de orientação de três cubos iguais.

1.3.2 Objetivos Específicos

- Obtenção de um modelo funcional do robô Comau Smart5 SiX 6-1.4 para utilização ao longo da simulação;
- Modelagem de uma garra robótica funcional para o *software* V-REP baseada em uma ferramenta anteriormente desenvolvida e disponível no laboratório;
- Finalização do desenvolvimento da Garra robótica acionada por motor de corrente contínua;
- Realização do empilhamento de três cubos considerando também sua orientação;
- Elaboraração de um algoritmo de geração de trajetória para aplicação na simulação;
- Estudos da biblioteca OpenCV para o aplicativo V-REP visando a utilização em um sistema real;
- Estudos da API de comando externo do V-REP;
- Identificação de posição e orientação dos cubos através da manipulação da imagem da câmera digital;
- Unificação dos sistemas em um único computador, realizando as movimentações do robô simulado a partir de imagens capturadas externamente no ambiente real.

1.4 Estado da Arte

A necessidade de simular sistemas é antiga, muito anterior às máquinas, surgindo das representações de campos de batalha em tabuleiros. Os sistemas de simulação tiveram um grande impulso durante a Segunda Guerra Mundial. Neste período, os supercomputadores eram utilizados para realizar cálculos balísticos pelo exército norte-americano, visando simular o lançamento de mísseis (BALADEZ, 2009). Após este período, os simuladores continuaram sendo desenvolvidos. Entretanto o alto poder de processamento e o custo dos

equipamentos fez com que os mesmos estivessem acessíveis apenas para grandes corporações e universidades.

Com o tempo de desenvolvimento e consequente percepção por parte das grandes empresas da economia gerada pelos processos simulados, a indústria automobilística passou a adotar métodos de simulação para resolver problemas de segurança e otimizar os meios de produção, sendo utilizados também em ambiente de negócios (BALADEZ, 2009).

Com o grande avanço da tecnologia computacional, tornou-se possível simplificar os procedimentos de simulação. A redução do custo dos sistemas mais potentes fez com que desenvolvedores investissem mais tempo no projeto de sistemas de simulação, incluindo nestes, os jogos eletrônicos. Sendo o mercado de entretenimento bastante lucrativo, cada vez mais as empresas buscaram proporcionar maior realismo nos *games*, o que colaborou com a evolução das mecânicas de simulação presentes nos aplicativos. A utilização de simuladores de voo é o exemplo mais conhecido do uso militar de simulações extremamente precisas no treinamento de pessoal (SCHITCOSKI, 2009), que também está bastante presente no mercado dos jogos.

De modo semelhante, os ambientes robotizados passaram a ser simulados visando verificar diversos protocolos de segurança, testar sensores e equipamentos, dentre outros dispositivos. Desta forma, universidades e desenvolvedoras independentes passaram a produzir simuladores com alta confiabilidade visando a economia de recursos (HARRIS; CONRAD, 2011).

Existem diversos simuladores no mercado, muitos deles elaborados pelas próprias fabricantes de robôs, o que os torna um produto adicional ao se realizar a compra do equipamento. Existem também os programas com versões gratuitas para estudantes, que permitem o estudo do comportamento de determinado algoritmo sem ser necessário alto recurso financeiro.

O V-REP (*Virtual Robot Experimentation Platform*) surgiu de um esforço na tentativa de conciliar um ambiente de desenvolvimento modular e versátil (ROHMER; SINGH; FREESE, 2013). Com um conjunto de elementos interativos, o mesmo torna possível a programação de forma livre de diversos tipos de robô.

Diversos trabalhos envolvem a aplicação de técnicas de visão computacional a robôs. Alguns deles, utilizam ambientes de simulação para a validação da estratégia estudada, enquanto outros são aplicados diretamente nos sistemas reais.

FARIAS *et al.* (2018) utiliza o software de simulação V-REP em conjunto com o processamento de imagem externo via OpenCV para realizar a navegação do robô Khepera IV em um ambiente controlado. É realizada a comunicação entre as bibliotecas através de um *script* em Python responsável por gerar os parâmetros de velocidades de cada um dos motores.

Em MEDINA (2015) é utilizada visão computacional para a identificação de posição e orientação de objetos para posterior manipulação através de um robô acionado pneumaticamente. É planejado um sistema de geração de trajetória para que o robô possa posicionar as peças em um local predeterminado.

A produção de OLIVARES-MENDEZ; KANNAN; VOOS (2015) aborda a utilização de visão computacional em conjunto com controladores PID para realizar o pouso de um veículo

aéreo não tripulado do tipo quadricóptero em uma área demarcada. Todo o desenvolvimento é simulado também no ambiente V-REP em conjunto com o *framework* ROS, se assemelhando bastante ao trabalho a ser desenvolvido.

Com base nos estudos anteriormente citados, visa-se desenvolver uma célula simulada em que um robô industrial modelo Comau Smart5 SiX realiza o reconhecimento de posição e orientação de cubos com faces coloridas através de visão computacional. Após a geração da trajetória para a manipulação do cubo, o mesmo deverá ser posicionado em uma plataforma de posição fixa permitindo o empilhamento dos objetos de forma semelhante. Para esta tarefa, será utilizado o ambiente de simulação V-REP, em conjunto com a biblioteca OpenCV.

1.5 Organização do Documento

Este documento está dividido em 5 capítulos. O presente capítulo, aborda os conceitos introdutórios responsáveis pela escolha do tema a ser estudado. É definido o problema e, em seguida, são demonstrados os objetivos do trabalho e a motivação da realização da investigação. Na sequência é realizada uma breve apresentação dos trabalhos correlatos.

No Capítulo 2 está contido o conteúdo teórico a ser utilizado como base para o desenvolvimento do trabalho. São definidas as principais técnicas a serem utilizadas e apresentados os *softwares* que serão estudados.

O terceiro capítulo demonstra a ideia principal da proposta e como a mesma será realizada. São definidas as estratégias de abordagem de cada módulo do trabalho para, em seguida, serem executadas. Também são elencados os passos para execução do trabalho, demonstrando de forma detalhada todos os procedimentos realizados ao longo do desenvolvimento deste documento.

No quarto capítulo, são obtidos os resultados atingidos com o desenvolvimento, mostrando através de imagens e tabelas a parte gráfica e análise da movimentação gerada.

Já no quinto e último capítulo são apresentadas as considerações finais referentes ao trabalho desenvolvido. Também são listadas as propostas de continuidade para o aperfeiçoamento do trabalho em questão.

Capítulo 2

Fundamentos

Nesta seção, serão apresentados os conceitos e formulações necessárias para o desenvolvimento do trabalho em questão. Os mesmos estão divididos em quatro grandes grupos, sendo eles:

- Ambiente de Simulação
- Visão Computacional
- Robótica
- *Softwares*

2.1 Ambiente de Simulação

Com o crescente desenvolvimento dos dispositivos robóticos e popularização de equipamentos de *hardware* com alto poder de processamento, houve também o crescimento do interesse na utilização de ambientes de simulações robóticas (HARRIS; CONRAD, 2011).

As simulações são utilizadas para verificar o comportamento de sistemas robóticos visando evitar possíveis erros de programação e falhas de segurança como por exemplo no caso de uma eventual colisão entre o robô e objetos fixos no ambiente.

Desse modo, para que um ambiente simulado seja fiel ao sistema real, é necessário considerar o maior número de informações possível da célula robótica. Cada parâmetro negligenciado pode reduzir drasticamente a confiabilidade dos resultados obtidos virtualmente.

2.2 Visão Computacional

Entende-se por visão computacional a técnica de extração automatizada de informações a partir de algum tipo de imagem.

Segundo (DÂMASO, 2006), "A visão computacional ou visão de máquina é composta basicamente pelas etapas de aquisição e de processamento de imagens de uma cena, objeti-

vando a extração de informações úteis a serem utilizadas no controle de um sistema robótico, por exemplo."

Desse modo, a visão computacional tem como resultado a execução de determinada tarefa de acordo com a imagem (ou conjunto de imagens) a ser processada, sendo considerada também um sistema de controle.

2.2.1 Aquisição de Imagens

Na primeira etapa, é realizada a aquisição das imagens através de uma câmera. Durante a captura de imagem, que depende da luminosidade do ambiente e objeto, podem ocorrer diversas interferências, que são atenuadas através da utilização de uma série de filtros digitais.

Um estilo de câmera bastante utilizado devido ao seu baixo custo e ampla disponibilidade é a *webcam* USB ou as câmeras contidas em *smartphones*. Neste caso, a imagem já é processada pelo computador de forma digital, evitando a necessidade de um processamento extra. Entretanto, tais modelos de câmeras por serem de baixo custo, possuem maior distorção da imagem ao ser convertida para informações em duas dimensões.

Para que seja possível relacionar as coordenadas da imagem com as coordenadas do espaço faz-se necessário obter os parâmetros de configuração da câmera. Isto é feito utilizando a relação entre pontos de geometrias conhecidas (MEDINA, 2015).

De um modo geral, a matriz P que relaciona os pontos no espaço com os *pixels* de uma imagem pode ser obtida através da Equação 2.1.

$$\begin{bmatrix} u_\omega \\ v_\omega \\ \omega \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.1}$$

$$(u, v) = \left(\frac{u_\omega}{\omega}, \frac{v_\omega}{\omega}\right) \tag{2.2}$$

em que (u, v) são as coordenadas em *pixels*, ω um valor correspondente a um peso e k_{ij} os termos da matriz de transformação P.

Outra forma desta representação é mostrada na Equação 2.3 onde neste caso são considerados os valores intrínsecos da câmera.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

2.2.2 Processamento de Imagens

Uma imagem digital pode ser representada por uma matriz em que cada elemento corresponde a um pixel, sendo armazenado neste suas informações referentes à tonalidade.

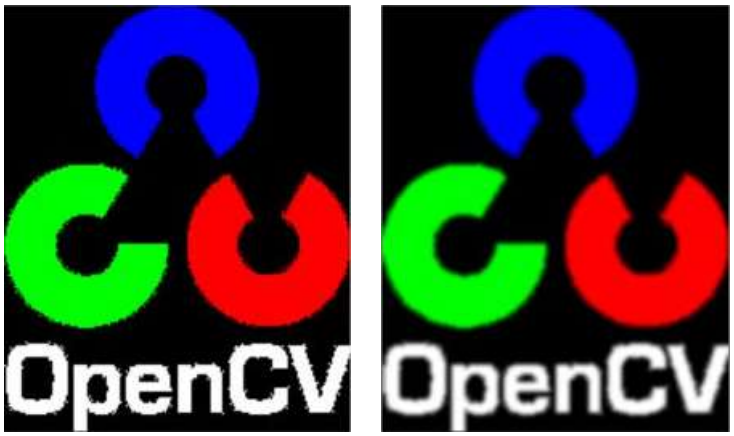
Para que seja possível realizar a extração de informações da imagem, é necessário processar cada um dos *frames* de uma informação em vídeo, por exemplo. Em ambientes reais, as interferências externas como diferença de luminosidade ou ruído de imagem podem fazer com que ocorra um falseamento na informação extraída. Para evitar esses problemas, durante a etapa de processamento de imagem, devem ser observados os fatores que podem influenciar o ambiente em questão para assim definir as técnicas a serem utilizadas.

2.2.3 Filtro Gaussiano

O filtro gaussiano consiste na técnica de convolução de uma máscara gaussiana na imagem a ser processada, como se segue (JESUS; COSTA JR., 2014).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

em que, x é a distância horizontal da origem, y a distância vertical da origem e σ é o desvio padrão da distribuição gaussiana. Como a imagem é definida em duas dimensões, esta máscara também o é e pode ser representada pela Equação 2.4. Esta filtragem é comumente utilizada para a redução do ruído de imagem, como mostrado na Fig. 2.1. A imagem processada possui o aspecto borrado, de acordo com o desvio padrão (σ) definido para a máscara gaussiana.



(a) Imagem original. (b) Imagem Filtrada.

Figura 2.1: Demonstração da aplicação do filtro gaussiano (OPENCV, 2013)

Por este motivo, esta técnica é bastante utilizada na primeira etapa de filtragem durante o processamento de uma imagem.

2.2.4 Espaço de Cores

Um espaço de cores é um espaço que compreende todos os valores possíveis de cores em níveis de intensidade e brilho. A forma como os mesmos são definidos faz com que cada um deles possua aplicações específicas, sendo mais comum a utilização do RGB, do inglês *red, green, blue*.

A definição do espaço de cores a ser utilizado influencia na técnica selecionada para o processamento de imagem. Em algumas bases, como a HSV (do inglês *hue, saturation, value*), exemplificada na Fig. 2.2, uma cor pode ser mais facilmente identificada mesmo que o objeto tenha variação em sua luminosidade, já que a cor é definida apenas pelo primeiro valor.

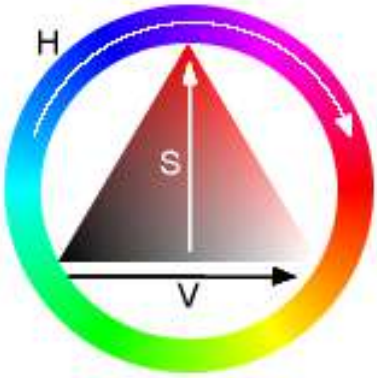


Figura 2.2: Espaço de cores HSV (SCHROEDER, 2013)

Neste caso, é definido um valor de 0 a 360 para cor, e de 0 a 100 para saturação e brilho, que são combinados para se obter a tonalidade de cada *pixel* da imagem.

2.3 Robótica

Robótica é a área de estudos envolvendo diversos setores da engenharia. Entender a complexidade dos robôs requer o conhecimentos em Engenharias Elétrica, Mecânica, Controle e Automação e em Computação.

Segundo SPONG; HUTCHINSON; VIDYASAGAR (2006), um robô é um dispositivo multifuncional reprogramável, desenvolvido para mover materiais, peças, ferramentas ou equipamentos específicos através de trajetórias programadas para a execução de uma variedade de tarefas. Os robôs ainda podem ser divididos de acordo com suas aplicações como móveis, manipuladores ou de função específica. No trabalho em questão, será abordado o robô do tipo manipulador industrial.

2.3.1 Robôs Manipuladores Industriais

De acordo com SCIAVICCO; SICILIANO (2000), a característica essencial que difere um robô industrial dos outros diversos tipos, é a sua aperfeiçoada versatilidade. Este fator ocorre principalmente devido ao seu *end effector*, podendo ser uma ferramenta de diversos tipos como para soldagem, medição, agarre e manipulação. Além disso, manipuladores industriais possuem um amplo espaço de trabalho em comparação com os demais tipos.

Na Fig. 2.3 é representado o *workspace* de um manipulador industrial Comau Smart5 SiX 6-14. No interior de toda área representada, o robô consegue se posicionar com uma orientação arbitrária.

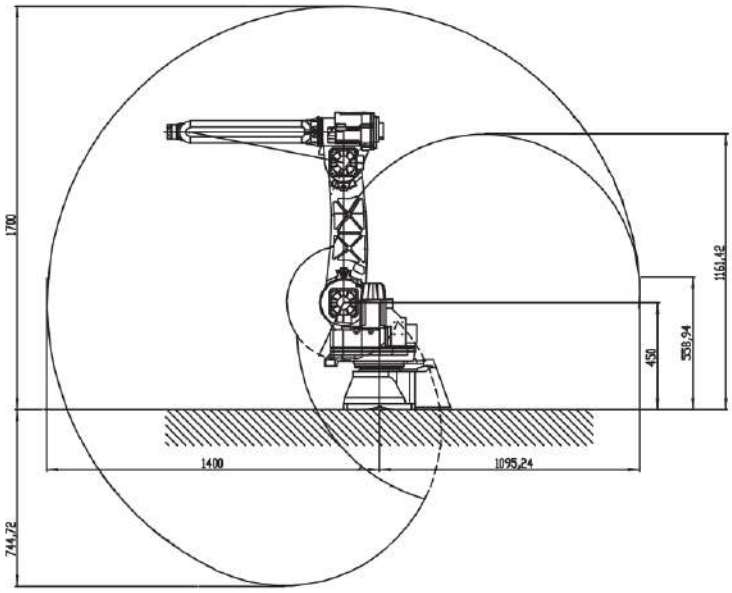


Figura 2.3: Espaço de trabalho de um manipulador industrial Comau Smart5 SiX 6-14. Adaptado de (COMAU, 2014)

2.3.2 Comau Smart5 SiX

Na Unidade Divinópolis do CEFET-MG, encontra-se disponível o robô industrial Comau Smart5 SiX, mostrado na Fig. 2.4. O mesmo consiste em uma estrutura antropomórfica de 6 graus de liberdade movimentado através de servomotores *brushless*, com repetibilidade de +/- 0,05 mm (COMAU, 2014). Para o acionamento do robô, é utilizada a unidade controladora C5G, onde estão contidas as malhas de controle de posição, velocidade e aceleração do Smart5 SiX. Sua operação e programação pode ser feita através do *Teach Pendant* TP5 ou via execução de um arquivo .COD traduzido do código fonte escrito na linguagem PDL2.

Figura 2.4: Robô Comau Smart5 SiX 6-14 do CEFET-MG / Unidade Divinópolis

Na Tab. 2.1 são mostradas as principais características do robô industrial Comau Smart5 SiX 6-1.4.

Tabela 2.1: Características e performance do COMAU Smart5 SiX 6-14 (COMAU, 2014)

VERSÃO	**SIX 6-1.4**
Estrutura / n^o de GDL	Antropomórfica / 6 GDL
Carga no punho	6 kg
Carga adicional no braço	10 kg
Torque do motor 4	11,7 Nm
Torque do motor 5	11,7 Nm
Torque do motor 6	5,8 Nm
Eixo 1	+/- 170^o (140^o/s)
Eixo 2	+155^o/-85^o (160^o/s)
Eixo 3	0^o/-170^o (170^o/s)
Eixo 4	+/- 210^o (450^o/s)
Eixo 5	+/-130^o(375^o/s)
Eixo 6	+/-2700^o (550^o/s)
Alcance máximo hoizontal	1400 mm
Repetibilidade	+/- 0,05 mm
Peso	160 kg
Flange de acoplamento de ferramenta	ISO 9409-1-40-4-M6
Motores	AC *brushless*
Sistema de medição de posição	Encoder
Potência total instalada	3 kVA / 4,5 A
Cor do robô	Vermelho RAL 3020

Um diferencial do sistema robótico disponível no laboratório da instituição é a disponibilidade da plataforma Comau C5G Open. Este sistema faz com que seja possível a comunicação em tempo real entre a unidade controladora e o computador industrial Acopus PC 910 através do protocolo *Ethernet Powerlink*. Desta forma, pode ser realizada a programação do robô em vários níveis, incluindo a aquisição da leitura bruta dos *encoders* dos motores e adição de sensores externos, como câmeras para visão computacional (FERRARA, 2013). Devido às características do robô disponível na instituição, o mesmo foi escolhido como modelo a ser simulado durante o desenvolvimento deste trabalho.

2.3.3 Matriz de Transformação Homogênea

Entende-se por *frame* um conjunto de três versores perpendiculares entre si denominados convencionalmente x, y e z e utilizados como referência para posição e orientação de corpos. Em sistemas robóticos, se faz necessário compreender as características dinâmicas da ponta da ferramenta em relação a um sistema de coordenadas fixo, normalmente posicionado na base do robô.

Para estes estudos, são utilizadas as matrizes de transformações homogêneas, que são desenvolvidas a partir das características de movimentação do robô. Para tal, essa matriz é descrita como se segue:

$$H = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix}, \tag{2.5}$$

em que R_n^0 é a matriz de rotação $\mathrm{R}^{3\times3}$ do n-ésimo *frame* ao *frame* zero e o_n^0 Um vetor coluna de termos x, y e z correspondente à posição do n-ésimo *frame* em relação ao *frame* zero.

2.3.4 Convenção de Denavit-Hartenberg

A cinemática direta de um manipulador consiste na obtenção da posição e orientação da extremidade do robô em relação ao seu sistema de coordenadas de base. Uma estrutura bastante utilizada em aplicações robóticas é a convenção de Denavit-Hartenberg ou simplesmente convenção DH (SPONG; HUTCHINSON; VIDYASAGAR, 2006). Neste caso, cada transformação homogênea A_i é representada pelo produto de quatro transformações básicas de translação e rotação:

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{2.6}$$

Desta forma, a Equação 2.6 resulta em:

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Desse modo, cada matriz A_i representa a mudança de coordenadas relativas a um *link*. Entretanto, para a utilização desta técnica é necessário realizar a atribuição de *frames* cumprindo os requisitos definidos em (SPONG; HUTCHINSON; VIDYASAGAR, 2006).

2.3.5 Cinemática Inversa

Na realização da movimentação de robôs, tem-se na maioria dos casos apenas a posição e orientação final desejada. Para realizar o posicionamento da ferramenta, faz-se necessário então identificar as variáveis articulares de cada uma das juntas para então posicionar o robô de modo que se alcance a posição desejada.

O problema da cinemática inversa consiste na solução desse problema, visando obter a posição de cada uma das juntas para que o robô alcance determinada posição e possa consequentemente seguir uma trajetória definida.

Como as equações de cinemática direta são não-lineares, a resolução algébrica do problema inverso se torna complicada e então é utilizada a aproximação geométrica. No caso de um robô 2R planar, a solução pode ser obtida através do diagrama mostrado na Fig. 2.5.

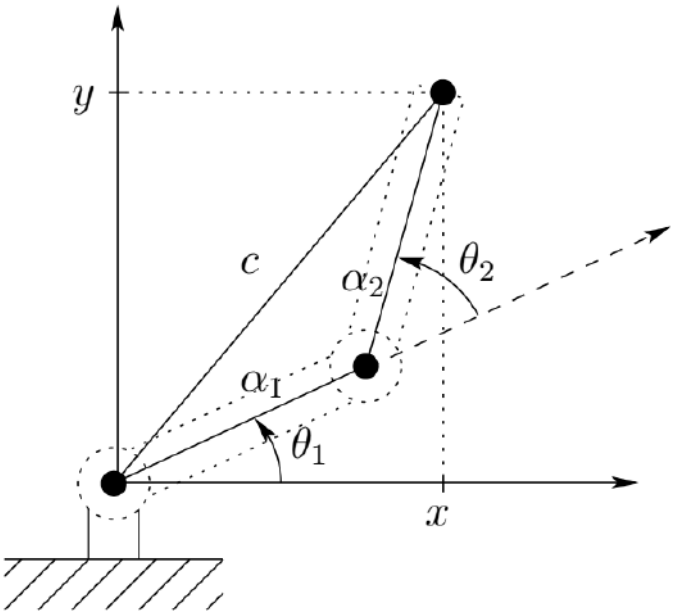


Figura 2.5: Cinemática inversa pelo método geométrico (SPONG; HUTCHINSON; VIDYASAGAR, 2006).

Desta forma, são obtidos através de relações geométrica os valores angulares das juntas utilizando as equações que se seguem.

$$cos\ \theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2} := D \tag{2.7}$$

$$\theta_2 = tan^{-1}\left(\frac{\pm\sqrt{1 - D^2}}{D}\right) \tag{2.8}$$

$$\theta_1 = tan^{-1}\left(\frac{y}{x}\right) - tan^{-1}\left(\frac{a_2 sen\ \theta_2}{a_1 + a_2 cos\ \theta_2}\right) \tag{2.9}$$

em que a_1 e a_2 correspondem ao comprimento dos dois primeiros *links*, x e y se referem às coordenadas espaciais do ponto a ser alcançado e θ_1 e θ_2 os ângulos referentes às juntas 1 e 2

Em robôs com seis graus de liberdade sendo três deles referentes ao punho esférico, como neste trabalho, é utilizada a técnica do desacoplamento cinemático para identificação de todos os valores angulares. Neste caso, são utilizados os três primeiros GDL para o posicionamento do centro do punho, e os três GLD restantes para as correções necessárias em sua orientação. Para isto, é utilizada a Equação 2.10 visando identificar a posição do centro do punho para os cálculos iniciais através do método geométrico.

$$o_c^0 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.10}$$

em que o representa as coordenadas de posição final, o_c^0 equivale à coordenada do centro do punho esférico, d_6 representa a distância do centro do pinho esférico até o *end effector* e R equivale à matriz de rotação R_6^0.

Esta relação pode ser melhor observada através da Fig. 2.6.

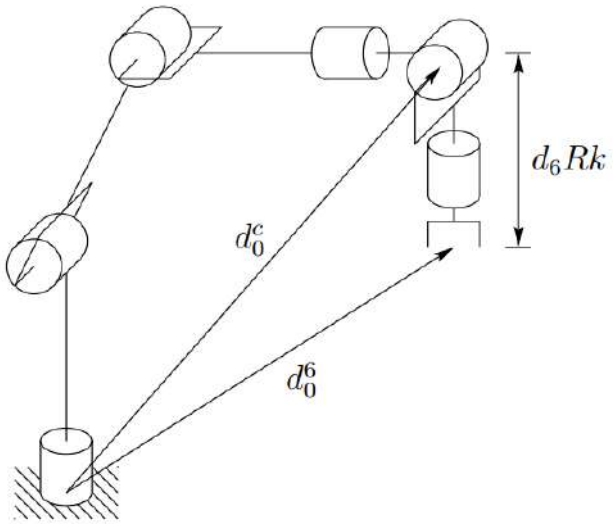


Figura 2.6: Desacoplamento cinemático (SPONG; HUTCHINSON; VIDYASAGAR, 2006)

Em seguida, para a identificação da posição, é utilizada a matriz R_6^3. Essa pode ser obtida através da relação:

$$R_6^3 = (R_3^0)^T R \tag{2.11}$$

neste caso, R_6^3, $(R_3^0)^T$ e R representam as matrizes de rotação do centro do punho esférico ao *frame* final, da origem ao terceiro *frame* e a matriz de rotação completa respectivamente.

Como o lado direito da Equação já é conhecido, torna-se simples a obtenção do termo desejado e a matriz R_6^3, também é conhecida como matriz de ângulos de Euler, é representada com os ângulos:

$$\theta_4 = \phi$$

$$\theta_5 = \theta$$

$$\theta_6 = \psi$$

que resulta em:

$$R_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix} \tag{2.12}$$

2.3.6 Geração de Trajetória

Segundo SCIAVICCO; SICILIANO (2000), o requisito mínimo para um manipulador é a capacidade de se mover de uma posição inicial a uma posição final. Existem infinitas possibilidades de um robô atingir um ponto definido em seu espaço de trabalho. Entretanto,

em um sistema real deve-se considerar os limites de movimentação devido à geometria e os obstáculos presentes na célula robótica.

Para fazer com que a movimentação entre dois pontos ocorra de forma satisfatória, é necessário realizar o planejamento de trajetória considerando todas as interferências externas que estão presentes no ambiente.

Uma forma de realizar a movimentação articular do robô, evitando possíveis obstáculos, ocorre através da utilização de campos potenciais artificiais (SPONG; HUTCHINSON; VIDYASAGAR, 2006). Neste caso, o ponto de destino é definido como campo atrativo atuando na extremidade do robô, enquanto os obstáculos são considerados campos repulsivos atuando sobre todo o corpo da ferramenta. Tais campos são formulados por:

$$U_{att,i}(q) = \begin{cases} \frac{1}{2}\zeta_i ||o_i(q) - o_i(q_f)||^2, & \text{se } ||o_i(q) - o_i(q_f)|| \leq d \\ d\zeta_i ||o_i(q) - o_i(q_f)|| - \frac{1}{2}\zeta_i d^2, & \text{caso contrário.} \end{cases} \tag{2.13}$$

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2}\eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0}\right)^2, & \text{se } \rho(o_i(q)) \leq \rho_0 \\ 0, & \text{caso contrário.} \end{cases} \tag{2.14}$$

2.3.7 Softwares

A seguir são apresentados os principais aplicativos e sistemas a serem utilizados ao longo do desenvolvimento do trabalho.

2.3.8 V-Rep

V-REP é um *software* de simulação de ambientes robotizados com uma arquitetura de controle distribuído. Cada objeto ou modelo pode ser controlado individualmente via rotina independente ou acessado através de complementos de terceiros (COPPELIAROBOTICS, 2019a). O programa é distribuído em versões comerciais (V-REP PRO) e disponibiliza sua edição completa para projetos educacionais (V-REP PRO EDU) não sendo necessário comprovar as finalidades de utilização para o *download*. Com estas características, se torna possível desenvolver dentre outras aplicações, sistemas de controle para robôs e algoritmos de seguimento de trajetória. A tela inicial do aplicativo é mostrada na Fig. 2.7.

Uma das diversas formas de programação do *software* é a partir da utilização da API (*Application Programming Interface*, conjunto de padrões para acesso a informações de um *software*) remota, cuja documentação é disponibilizada pela própria desenvolvedora. Desse modo, é possível estabelecer uma comunicação cliente-servidor e controlar o ambiente de simulação via algoritmo externo. Uma função de exemplo em linguagem Python é mostrada a seguir:

```
vrep.simxSetJointPosition(junta, angulo, modo)
```

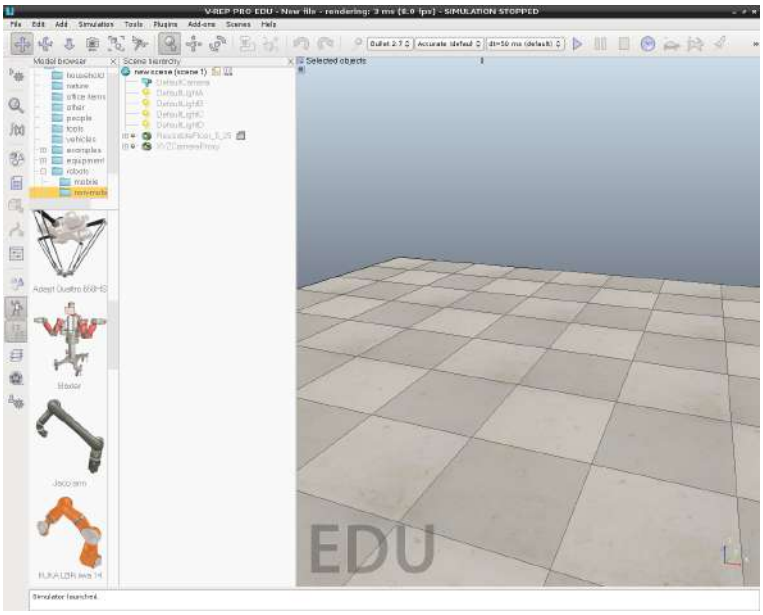


Figura 2.7: Captura de tela do *software* V-REP.

Neste comando exemplificado é utilizada a API externa para alterar o posicionamento de uma junta no ambiente simulado.

Além das funções anteriormente citadas, o programa ainda possui um complemento responsável pela representação em gráficos de diversos parâmetros selecionáveis como distâncias, posições e velocidades. Desaa forma, torna-se possível analisar o comportamento dinâmico da simulação utilizando a geração e comparação de dados graficamente em tempo real.

2.3.9 ROS

ROS (*Robot Operating System*) é um *framework* bastante flexível para o comando de sistemas robóticos. O ambiente é as vezes denominado pseudo-sistema operacional, uma vez que o mesmo executa várias funções de um sistema operacional, mas requer um sistema operacional como o Linux para ser executado (FAIRCHILD; HARMAN, 2016).

O componente possui um conjunto de ferramentas, bibliotecas e convenções para simplificar a tarefa de criar comportamentos complexos com elevada robustez para diferentes plataformas robóticas. Além disso, o *framework* foi criado do zero visando encorajar o desenvolvimento colaborativo do sistema *open source* por empresas e usuários. (QUIGLEY *et al.*, 2009).

2.3.10 OpenCV

A OpenCV é uma biblioteca de visão computacional escrita em C e C++ disponível para os sistemas operacionais Windows, Linux e Mac OS X. De acordo com BRADSKI; KAEHLER (2008), a ferramenta foi desenvolvida tendo como objetivo proporcionar uma estrutura de visão computacional que tornasse possível produzir estruturas sofisticadas de forma simplificada.

A mesma possui uma série de métodos de manipulação de imagens que proporcionam uma acurácia elevada nas identificações necessárias. Além disso, diversas referências fazem o uso da mesma devido a sua simplicidade de utilização em diferentes tipos de imagens.

2.3.11 Autodesk EAGLE

O Autodesk EAGLE é um *software* de automação de projetos eletrônicos que permite ao projetista de placas de circuito impresso conectar diagramas esquemáticos, definir o posicionamento de componentes, diagramar mais de uma face cobreada e acessar conteúdos de várias bibliotecas. Possui uma versão gratuita para usuários ocasionais que inclui duas folhas esquemáticas, duas camadas de sinal e uma área de placa com 80 cm^2. Os recursos ofertados pela versão gratuita foram suficientes para execução deste trabalho.

2.3.12 Linguagem PDL2

PDL2 é a linguagem de programação baseada em Pascal e utilizada no robô industrial Comau Smart5 SiX 6-14. Através dela, se faz possível dentre outras tarefas, mover braços robóticos, enviar e receber arquivos de informação, monitorar eventos e estados e implementar o tratamento de exceções (COMAU, 2005).

Ao longo do algoritmo, também é possível utilizar estruturas de repetição e decisão, o que permite desenvolver sequencias bem definidas de instruções de acordo com variáveis lidas externamente. Um exemplo de linha de código em PDL2 é mostrado a seguir.

```
MOVE TO {alpha, beta, gamma, delta, omega, theta}
```

em que: *alpha*, *beta*, *gamma*, *delta*, *omega* e *theta* equivalem aos ângulos correspondentes às juntas 1, 2, 3, 4, 5 e 6 respectivamente.

Capítulo 3

Desenvolvimento

Neste capítulo são elencados os procedimentos realizados para que os objetivos principais fossem alcançados ao longo da execução do presente trabalho. São apresentados as etapas sequenciais para que o trabalho possa ser reproduzido futuramente por outro interessado.

3.1 Metodologia

De modo a possibilitar uma progressão eficiente no desenvolvimento do trabalho, a metodologia abordada visou uma evolução gradativa no aprendizado das tarefas. Inicialmente, foram levantadas diversas referências para o estudo ao longo do desenvolvimento do trabalho. Como se trata de uma aplicação específica, se fez necessário categorizá-las em quatro subdivisões: Visão Computacional, V-Rep, ROS e C5G Open. Além disso, foram também listadas referências para posterior consulta envolvendo técnicas de controle e processamento de imagem, linguagens de programação e eletrônica.

Após a identificação e separação das principais referências, deu-se início aos estudos envolvendo a utilização isolada da biblioteca OpenCV e o *software* V-REP. Simultaneamente em conjunto com a disciplina optativa Laboratório de Sensores e Atuadores para Mecatrônica, foi realizado um breve estudo direcionado para a utilização da plataforma C5G Open.

Para permitir a execução do trabalho de forma modular, uma das formas de abordagem do problema consistiu na adoção de uma estratégia em que o mesmo desenvolvimento pudesse ser aproveitado futuramente para o sistema real e o simulado. O esquema proposto é representado na Fig. 3.1.

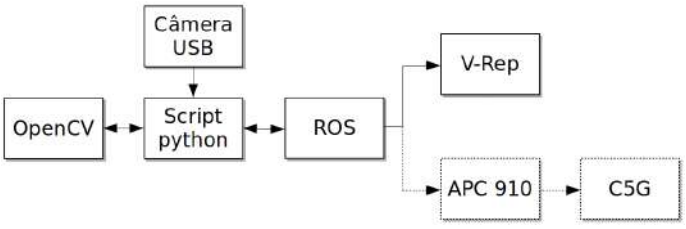


Figura 3.1: Diagrama de funcionamento do sistema completo.

Inicialmente, a imagem captada pela câmera será processada pela biblioteca OpenCV para a aquisição de informações relativas às cores identificadas e posicionamento dos cubos. Em seguida, essa informação é processada pelo *framework* ROS que será responsável pela interpretação dos dados recebidos e conversão dos mesmos em uma posição para o deslocamento do robô. Neste ponto, o trabalho é subdividido em dois casos: no primeiro, as coordenadas espaciais serão enviadas ao *software* V-Rep e então o modelo do robô será movimentado para a execução da tarefa; no segundo, ocorre a mesma movimentação anterior, mas simultaneamente os dados serão enviados para o APC 910 que, em seguida, os encaminha para a controladora C5G que então, realiza a movimentação do robô.

Durante os estudos iniciais, verificou-se a possibilidade de realização de tais etapas sem a necessidade da utilização do pacote ROS para a comunicação entre sistemas. Como no estado atual dos desenvolvimentos não está prevista a utilização do sistema C5G Open, a fase inicial definida consistiu na utilização de um *script* em Python para a unificação entre o *software* V-REP e o ambiente real via câmera USB. O novo diagrama a ser executado é mostrado na Fig. 3.2.

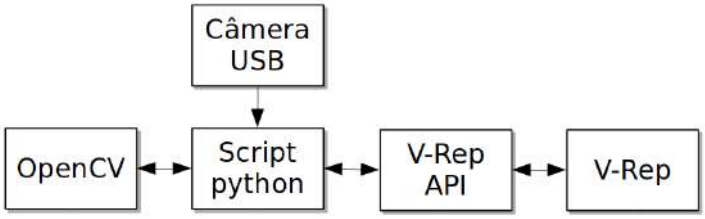


Figura 3.2: Diagrama de funcionamento para o presente trabalho.

A vantagem nesta abordagem se dá no fato de que nesta situação, pode-se utilizar versões mais recentes da linguagem de programação Python3 e biblioteca OpenCV 3.6.4. No caso em que é realizada a captura de imagem virtualmente pela própria simulação, os *plugins* disponíveis suportam apenas versões antigas da biblioteca (OpenCV 2).

Portanto, para a finalização do trabalho proposto, foram realizadas as seguintes etapas:

1. Estudos teóricos dos *softwares* a serem utilizados;
2. Estudo das adaptações na garra para a funcionalidade proposta;
3. Modelagem do robô funcional para a simulação de acordo com os requisitos de projeto;
4. Projeto do sistema de comunicação entre ferramenta e controladora;
5. Impressão das peças e aquisição dos componentes para a montagem da ferramenta;
6. Modelagem da garra no *software* de simulação;
7. Finalização e teste do funcionamento da garra no robô real;
8. Elaboração do algoritmo de geração de trajetória no V-REP;

9. Desenvolvimento do algoritmo de calibração de câmera e identificação dos cubos em Python;

10. Unificação dos sistemas para execução no computador a partir de um ponto no espaço;

11. Testes e comparações de resultados.

3.2 Modelagem

Para realização da simulação via *software*, foi necessária a elaboração do ambiente no V-REP. Essas etapas foram divididas em:

- Modelagem do Robô;
- Modelagem da Garra;
- Montagem da Célula de Simulação.

3.2.1 Modelagem do robô

Inicialmente, foram realizadas buscas por modelos dinâmicos do robô industrial Comau Smart5 SiX 6-1.4 compatíveis com o *software* V-REP. Consultou-se o site do fabricante e da desenvolvedora, bem como autores de trabalhos semelhantes que pudessem tê-lo desenvolvido. Como não foi possível obter informações referentes à disponibilidade do modelo para simulação compatível, tornou-se necessário elaborar um componente suportado pelo aplicativo em questão.

Os desenhos em três dimensões do robô foram obtidos no site do fabricante (COMAU, 2019) e tornou-se necessário apenas realizar a exportação das peças para o formato .stl.

Dessa forma, a etapa de modelagem teve seu início diretamente no V-REP. O *software* possui uma ferramenta para criação de juntas a partir dos parâmetros de DH de um robô qualquer. A Fig. 3.3 mostra o diagrama de arame do Smart SiX elaborado, utilizando a convenção para atribuição de *frames* possibilitando a extração dos parâmetros necessários. Cabe ressaltar que os *frames* 0 e 6 foram definidos de acordo com o manual do equipamento.

A partir do diagrama elaborado, foram obtidos os parâmetros de DH mostrados na Tab. 3.1.

Tabela 3.1: Parâmetros de Denavit-Hartenberg

Link	θ_i	d_i	a_i	α_i
1	$-\theta_1$	450	150	-90^o
2	$\theta_2 - 90^o$	0	510	0^o
3	θ_3	0	130	90^o
4	θ_4	$-647{,}07$	0	-90^o
5	θ_5	0	0	90^o
6	$\theta_6 + 180^o$	-95	0	180^o

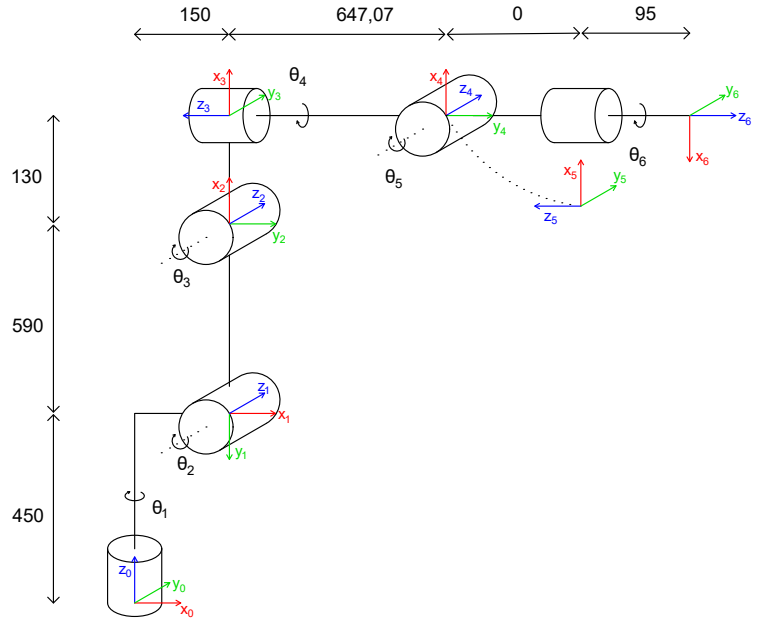


Figura 3.3: Diagrama de arame do robô Comau Smart5 SiX 6-1.4

Estes valores foram utilizados para a primeira etapa da modelagem do robô no programa de simulação, posicionando inicialmente as representações das juntas com a posição, orientação e sentido de rotação correspondentes à movimentação do robô real. Na Fig. 3.4 é mostrada uma captura de tela da ferramenta com as juntas posicionadas a partir das informações da Tab. 3.1.

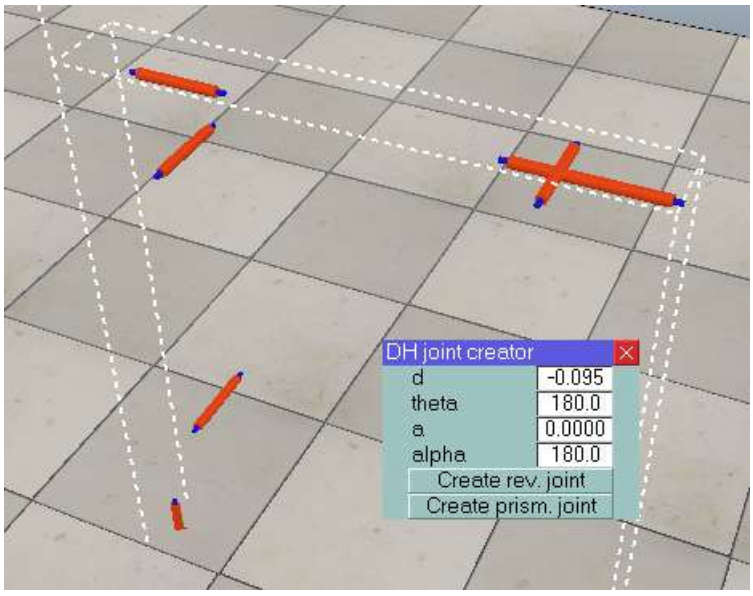


Figura 3.4: Juntas inseridas com a ferramenta DHjointcreator.

Em seguida, foram importados os arquivos STL gerados a partir do desenho fornecido pela fabricante do robô. Os mesmos foram posicionados de modo que as juntas anteriormente inseridas estivessem posicionadas no eixo de rotação correspondente ao *link* importado. Deste modo, a variação angular na junta resultaria na movimentação adequada do *link*.

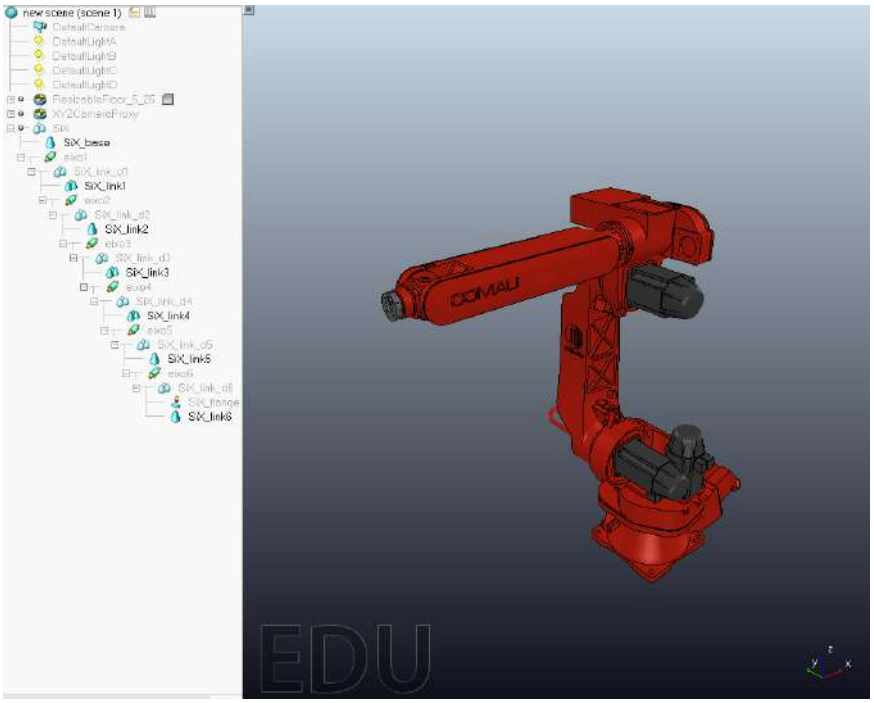


Figura 3.5: Modelo do Smart5 SiX elaborado no V-REP.

Os passos seguintes da elaboração do modelo foram realizados de acordo com as instruções fornecidas em COPPELIAROBOTICS (2019b). Também foram realizadas cópias simplificadas do modelo tridimensional para que os cálculos do programa fossem realizados de forma otimizada, tendo como base uma geometria com menor número de pontos. Além disso, foram configurados os parâmetros dinâmicos de cada uma das juntas de modo que as mesmas passaram a possuir as informações apresentadas anteriormente na Tab. 2.1. A Fig. 3.5 mostra uma captura de tela com o modelo elaborado.

O modelo desenvolvido também foi configurado para estar apto a acoplar uma ferramenta e também ser acoplado a uma base. Deste modo, é possível alterar o ambiente de acordo com as variações no sistema real.

Para permitir a realização de testes na movimentação dos links de forma dinâmica em tempo real, também foi elaborado uma pequena interface utilizando *sliders* para ajustar a orientação de cada um dos servomotores de forma independente. Conforme é mostrado na Fig. 3.6 a interface desenvolvida até então é composta por seis *sliders* podendo variar de acordo com as limitações das juntas anteriormente apresentadas na Tab. 2.1 e o código fonte em lua encontra-se na página do GitHub[1].

Deste modo, obteve-se um modelo funcional do robô para ser comandado ao longo do desenvolvimento do trabalho através da técnica de cinemática direta. Um dos principais pontos observados diz respeito ao comportamento de velocidade de movimentação das juntas. Como não foi definido um algoritmo de geração de trajetória, o simulador realiza a movimentação angular com aceleração instantânea muito elevada.

[1]https://github.com/kesleyroberto/Tcc

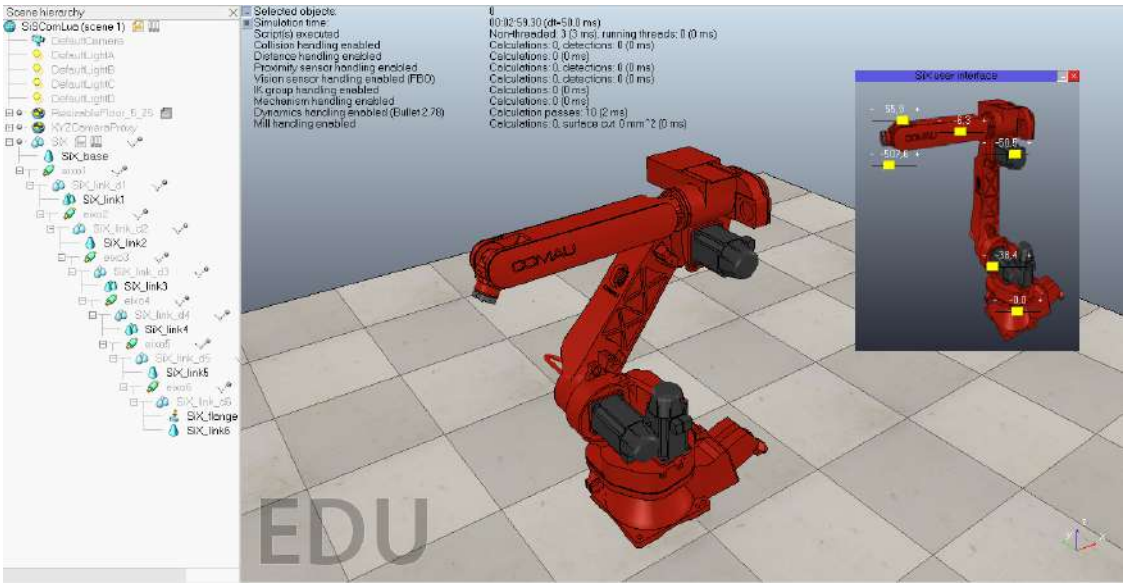


Figura 3.6: Captura de tela da interface elaborada com as posições das juntas alteradas.

Para verificar este comportamento, foi elaborado um gráfico da velocidade angular da extremidade do robô em função do tempo, mostrado na Fig. 3.7. O mesmo foi obtido realizando a variação do ângulo definido para a junta 3 de forma manual utilizando o *slider* da interface desenvolvida e a ferramenta *graph* disponível no V-REP.

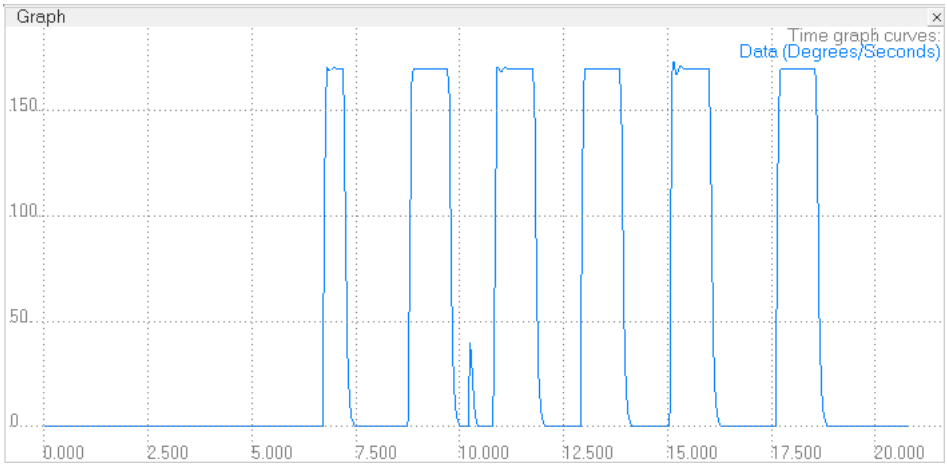


Figura 3.7: Gráfico da velocidade instantânea do flange do robô.

Conforme foi definido na etapa de configuração das juntas, a resposta em módulo da velocidade da extremidade do robô atingiu em seu máximo o valor de $170^o/s$, mostrando que a simulação neste aspecto respeitou as limitações reais do servomotor em questão.

3.2.2 Modelagem da Garra

De modo semelhante ao desenvolvimento do Smart SiX, a ferramenta do tipo garra foi montada virtualmente através dos desenhos realizados em Solidworks. Entretanto, neste caso, tais desenhos tiveram que ser elaborados pelo autor e convertidos para funcionamento no ambiente de simulação.

As etapas de montagem foram realizadas desta vez de forma manual para definição dos componentes específicos da ferramenta, o que não seria possível através da matriz de transformação homogênea, neste caso. Para a movimentação semelhante à real, optou-se pela utilização de duas juntas prismáticas paralelas. Deste modo, a movimentação simulada ocorre a partir de atuadores diferentes dos reais, entretanto tendo como resultado a mesma ação resultante. Na Fig. 3.8 é mostrada a garra montada no V-REP, já com sua interface gráfica de testes de movimentação.

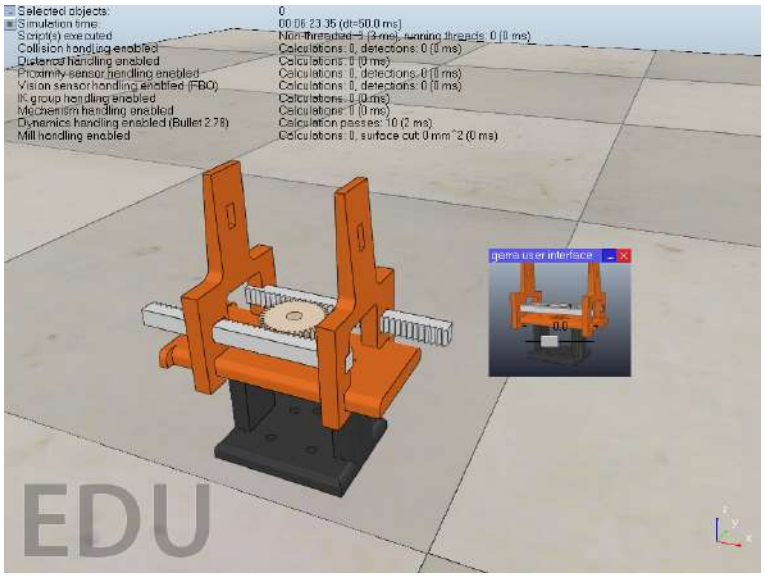


Figura 3.8: Ferramenta para manipulação dos cubos montada no V-REP.

Neste caso, a diferença da forma de atuação não gera interferência numa eventual comparação com o mundo real futuramente pois, do mesmo modo, o acionamento da garra ocorrerá de forma independente das modelagens de cinemática inversa. Na aplicação, o sistema de controle estará contido em um microcontrolador. Enquanto que na simulação, tal sistema está embutido no *script* da ferramenta.

3.2.3 Montagem da Célula de Simulação

Para a finalização do ambiente a ser utilizado no trabalho, foi inserida a mesa de suporte e posicionamento dos cubos, mostrada na Fig. 3.9. Trata-se de um objeto disponível no V-REP com dimensões personalizáveis para cada aplicação. Neste caso, para representar a área real na qual é possível obter informações a partir da câmera superior foram definidas as dimensões de 45 x 45 cm de área e 50 cm de altura do chão. Esta altura se torna suficiente pois o robô não foi posicionado em sua base fixa.

Na quina superior esquerda do modelo da mesa, foi adicionado um elemento *dummy*, com a função de *frame* de referência para as movimentações. Desta forma, a etapa de conversão entre a imagem e o espaço tridimensional ocorre de forma simplificada.

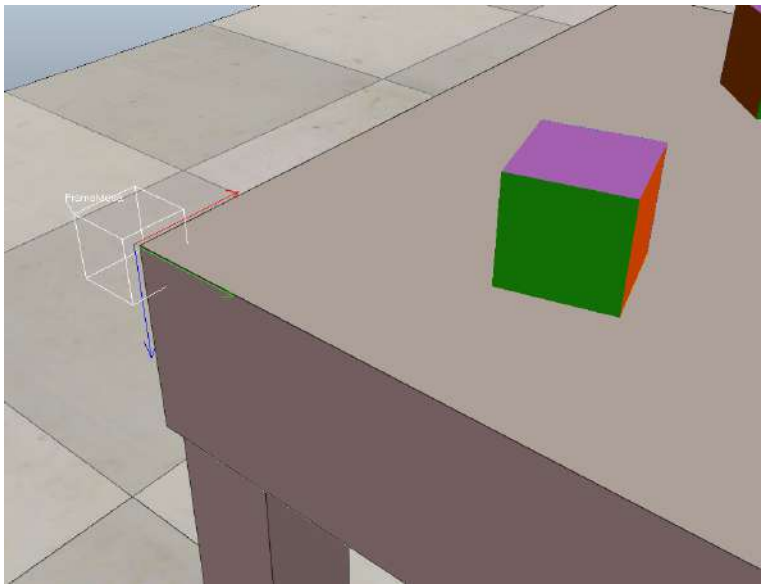


Figura 3.9: Ferramenta para manipulação dos cubos montada no V-REP.

3.3 Garra Robótica

Para a sequência do trabalho, sendo utilizada a plataforma C5G Open, foi necessário finalizar a construção da garra manipuladora para os cubos (Fig. 3.10). A mesma, que estava disponível no laboratório de robótica, se encontrava sem o motor de acionamento e placa de comunicação com a unidade de controle. Após a verificação do comportamento da garra para a aplicação desejada em uma bancada de testes, verificou-se também a necessidade de ajustes no posicionamento dos sensores de fim de curso.

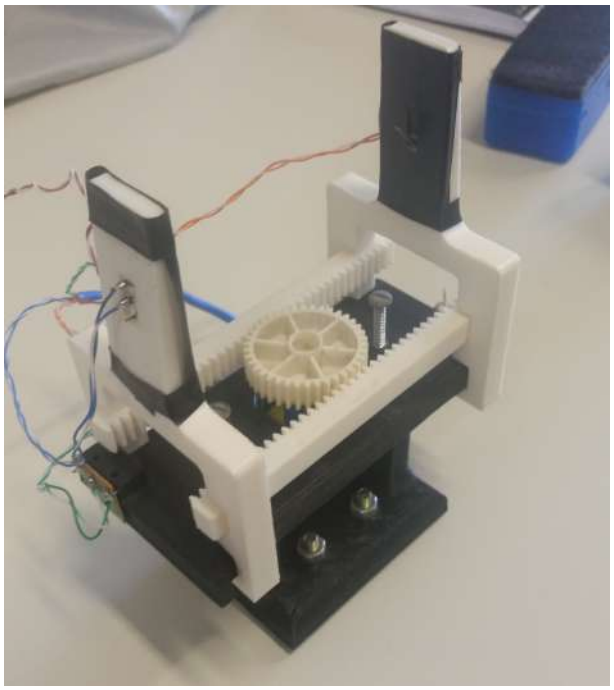


Figura 3.10: Ferramenta para manipulação dos cubos.

Foi adquirido um motor CC com caixa de redução para permitir a movimentação precisa

com a realimentação de duas chaves de fim de curso. Portanto, foi elaborado um novo desenho ajustando o posicionamento dos mesmos para que seu funcionamento se torne satisfatório na execução da tarefa proposta. Além das alterações mecânicas da garra, também foi necessário a elaboração de um circuito de comunicação e proteção para seu acionamento.

A comunicação da garra com a controladora se dá através das entradas e saídas disponíveis no CLP (COMAU, 2008). Como já foram desenvolvidas ferramentas para o robô anteriormente, já é prática comum no laboratório de robótica, a utilização da comunicação em paralelo utilizando as saídas digitais de 0-24V reguladas para 0-5V. Deste modo, foi necessário a utilização de duas entradas e saídas digitais para a abertura e fechamento da garra com confirmação da movimentação, uma vez que o sistema de controle com realimentação dos sensores de fim de curso se dá de forma embarcada na mesma.

Para isto, foi utilizado o microcontrolador Arduino Uno, que está conectado a uma ponte-H para o acionamento do motor de corrente contínua. O dispositivo aguarda pela interrupção vinda da controladora para então realizar o acionamento do motor e consequente fechamento da garra. Ao final da movimentação, o microcontrolador retorna um sinal informando que a ação solicitada foi realizada.

Portanto, para a finalização da ferramenta foram necessários os seguintes componentes:

- Arduino Uno R3
- Motor CC Pololu 6V 100 RPM
- Módulo ponte-H L298N
- Chaves de Fim de Curso
- Conectores em geral

3.3.1 Melhorias na Garra

Ao ser realizada a análise do comportamento da estrutura da ferramenta já existente, foram observados diversos pontos a serem alterados para uma futura implementação do sistema no robô industrial. Todos os dispositivos eletrônicos contidos na garra foram removidos, uma vez que os componentes estavam danificados e foi adotado um novo padrão de conexão com a controladora C5G.

Por se tratar de um projeto de longa data elaborado por alunos do curso superior e doado ao laboratório, a garra já passou por diversos ajustes em sua estrutura visando alterações no mecanismo de movimentação. Ao ser desmontada para verificação, constatou-se a necessidade de reconstruir três peças impressas em 3D devido a furos e canais incompatíveis que estavam ocasionando folga e inconsistência nas partes móveis. As peças que foram refeitas são mostradas nas Figs. 3.11 e 3.12.

(a) Vista superior. (b) Vista inferior.

Figura 3.11: Peças que necessitaram de reconstrução.

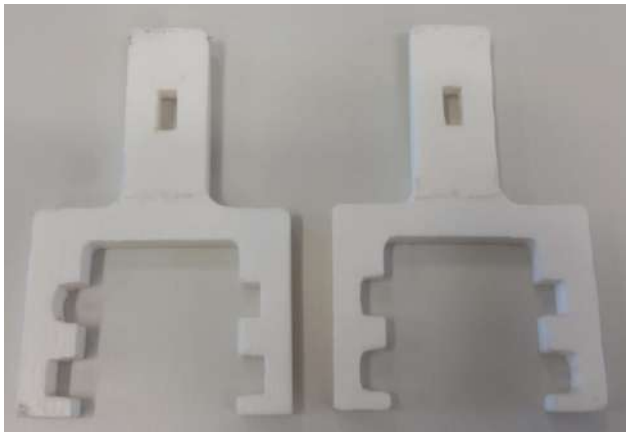


Figura 3.12: Ferramenta de agarre com canais danificados.

Na peça base da estrutura de movimentação, mostrada na Fig. 3.11(a) e 3.11(b) podem ser observados diversos furos além de uma peça plástica fixada com cola. Já nas peças de agarre mostradas na Fig. 3.12, os ajustes realizados anteriormente através do desbaste mostram que o projeto foi inicialmente dimensionado de forma incorreta. Além disso, os furos presentes para a colocação dos sensores de fim de curso possuem dimensões incompatíveis com as chaves disponíveis.

Desta forma, foram reaproveitadas por estarem em bom estado apenas as duas barras dentadas e a base de fixação no robô, mostradas na Fig. 3.13.

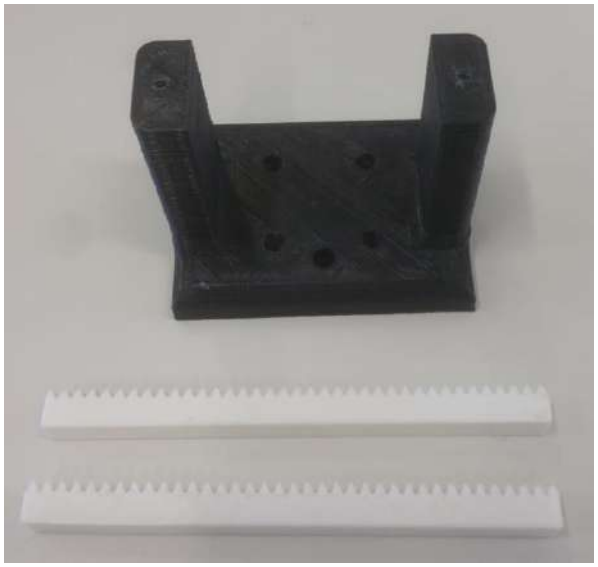


Figura 3.13: Peças a serem reaproveitadas.

A engrenagem utilizada no mecanismo de cremalheira foi substituída por outra de mesma geometria obtida em um depósito de engrenagens do laboratório. Assim se tornou necessário ainda fabricar o suporte para o motor a ser utilizado em conjunto com a nova engrenagem.

Um dos fatores que tornaram a ferramenta sub-utilizada no laboratório era a incapacidade de agarrar objetos diversos com massa semelhante à dos cubos que serão utilizados. Além disso, a superfície de contato lisa e o mal posicionamento dos sensores de fim de curso faziam com que o movimento de fechamento da garra fosse interrompido antes do contato real entre a ferramenta e o objeto com força de atrito suficiente.

Para minimizar este problema, foi proposta a cobertura da superfície de contato com material de alta capacidade de deformação elástica. Na nova garra, as chaves de fim de curso serão mais bem posicionadas e será adicionada uma camada de espuma em cada uma das peças de contato conforme ilustrado na Fig. 3.14. Desta forma, as chaves identificarão o contato com os objetos apenas após a deformação da espuma, que já é suficiente para elevar os cubos.

De modo a possibilitar a impressão das peças com as dimensões corrigidas de acordo com as necessidades atuais, todo o mecanismo foi redesenhado e as peças substituídas, ajustadas. A montagem da nova ferramenta proposta é mostrada na Fig. 3.15.

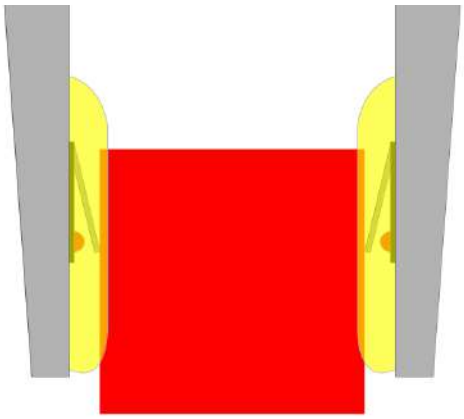


Figura 3.14: Representação da alteração na superfície.

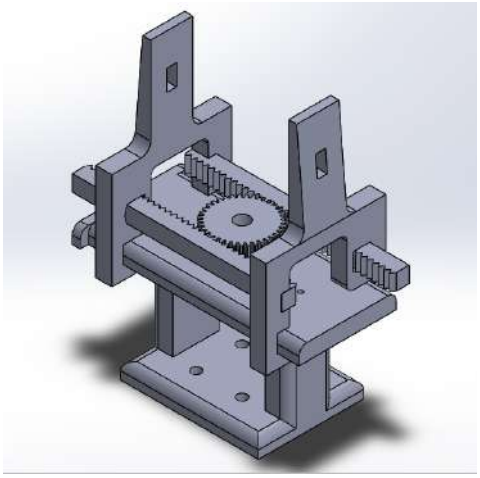


Figura 3.15: Vista isométrica da montagem da garra.

Foram realizados ajustes dimensionais nas partes móveis mantendo suas principais características originais. Além disso, também foi elaborado um acoplador para o eixo do motor à engrenagem e um suporte de fixação para o motor CC. O desenho completo das peças impressas está disponível no Apêndice A.

3.3.2 Circuito de comunicação

Como a garra se encontrava sem seu sistema de acionamento funcional, foi necessário estudar a forma de comunicação a ser utilizada durante a implementação da ferramenta. Foi adotado o padrão definido no laboratório para utilização das ferramentas conectadas à controladora C5G. Neste caso, a comunicação de entrada e saída é realizada através de cabos RJ-45 e o fornecimento de tensão de alimentação feito utilizando uma fonte variável externa.

Conforme informado no Cap. 2, os sinais de entrada e saída digital da controladora operam em níveis de tensão de 0V a 24V. Consultando a folha de dados módulo de entrada (B & R, 2018) verificou-se que um sinal é reconhecido como ativo ($HIGH$) para valores de tensão acima de 15V e considerado inativo (LOW) para tensões abaixo de 5V. Além disso, é informado o valor da impedância de entrada como $6{,}4k\Omega$ para operação em valores nominais.

Partindo destas informações e considerando que o microcontrolador escolhido é capaz de fornecer uma tensão contínua de apenas 5V, foi proposto um pequeno circuito para realizar o envio de informação da garra à controladora a partir de um sinal digital. Sabendo-se que outra produção de ferramenta para o robô estava ocorrendo de forma simultânea com os mesmos requisitos de comunicação, foi realizada uma parceria com a aluna Maria Vitória Pereira Vaz no projeto e desenvolvimento da placa de comunicação, com a divisão dos custos totais de forma igual. O esquema do circuito elaborado é mostrado na Fig. 3.16.

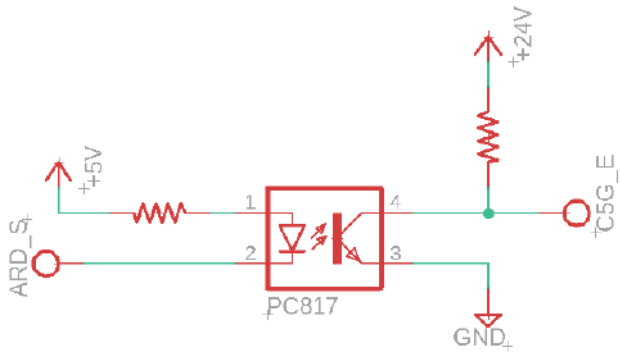


Figura 3.16: Esquemático do circuito de comunicação proposto.

Neste caso, foi proposto um conversor de nível lógico através de um fotoacoplador. Desta forma, é feita a isolação entre os circuitos evitando com que ambos os módulos possam ser danificados durante a operação.

Caso o sinal de acionamento seja de nível lógico baixo, ocorre o fluxo de corrente pelo diodo emissor de luz interno ao dispositivo. Isto faz com que o fototransistor permita a passagem de corrente tornando o sinal em C5G_E igual a 0V. Caso contrário, o fluxo de corrente no fototransistor é interrompido e o mesmo é direcionado para a entrada digital da controladora, fazendo com que o circuito se comporte como um divisor de tensão.

Para o dimensionamento do circuito, inicialmente foi consultada a disponibilidade de circuito integrados fotoacopladores para aquisição com maior facilidade. Optou-se pelo CI EL817 (EVERLIGHT, 2010), cujo funcionamento se adéqua aos parâmetros de operação necessários para aplicação.

Consultando o *datasheet* do fotoacoplador, verificou-se que as características de operação do *led* interno devem ser de aproximadamente $1.2V$ e $20mA$. Utilizando a Equação 3.1, foram obtidos os valores das resistências no segmento de circuito de acionamento do CI para os casos em que $Vcc = 5V$ (Arduino $\rightarrow$ C5G) e $Vcc = 24V$ (C5G $\rightarrow$ Arduino).

$$R = \frac{Vcc - 1{,}2V}{20mA} \tag{3.1}$$

No primeiro caso, a resistência obtida foi de aproximadamente 190Ω. Como não se trata de um valor comercial, optou-se pela utilização do resistor de 220Ω. No segundo caso, o valor calculado foi de 1140Ω, e o adquirido de $1{,}2k\Omega$.

Em seguida, partindo dos parâmetros já obtidos, foi encontrada a potência máxima a ser dissipada pelos componentes utilizando a Equação 3.2.

$$P = (Vcc - 1{,}2V)I \tag{3.2}$$

A potência dos resistores foi então definida como de $0{,}25W$ para os resistores de 220Ω e $1W$ para os componentes de $1{,}2k\Omega$.

Do outro lado do circuito, os resistores foram calculados considerando uma corrente máxima no fototransistor de $10mA$. Desta forma os valores dos componentes adquiridos foram de 510Ω para o sentido C5G $\rightarrow$ Arduino e $1{,}2k\Omega$ no sentido Arduino $\rightarrow$ C5G.

Foi definido a elaboração de uma placa com 6 canais de comunicação em cada sentido. Portanto, para a montagem do circuito foram necessários os seguintes componentes:

- 12 Resistores 1,2kΩ - 1W
- 6 Resistores 510Ω - 1/4 W
- 6 Resistores 220Ω - 1/4 W
- 12 CI's EL817
- 2 Soquetes de CI 24p
- 14 Bornes de 2 vias

Confecção da placa

Para a elaboração da placa de circuito impresso, foi utilizado o método de transferência por calor. Inicialmente, foi desenvolvido o *layout* mostrado na Fig. 3.17, das trilhas no *software* Autodesk Eagle. Neste caso, foi necessário a organização dos componentes de modo que a placa pudesse ser produzida utilizando apenas uma face cobreada e a sequência das conexões de entrada e saída estivesse organizada de forma intuitiva. Também nesta etapa, foi definida a utilização de um módulo regulador de tensão 24V/5V de modo a permitir que toda a alimentação seja fornecida pela controladora. O modelo escolhido para a aplicação foi o LM2596 DC-DC.

Com o projeto virtual realizado, o mesmo foi impresso com uma impressora a laser em papel *high gloss*. Deste modo, foi possível a transferência do *layout* para a placa virgem de fenolite.

Após a transferência, a placa cobreada foi submergida em uma solução de percloreto de ferro no Laboratório de Química da instituição por aproximadamente 10 minutos. Ao final do processo, a placa resultante já com as trilhas bem definidas foi higienizada para remover as sobras de toner no cobre, chegando ao resultado final desta etapa mostrado na Fig. 3.18.

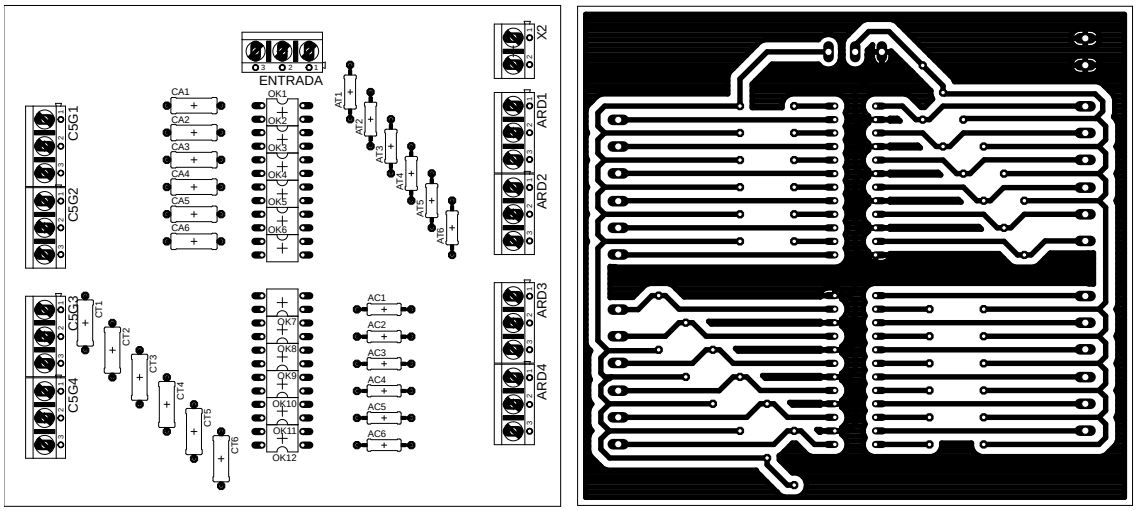


Figura 3.17: Vista superior e Inferior da placa de circuito impresso projetada.

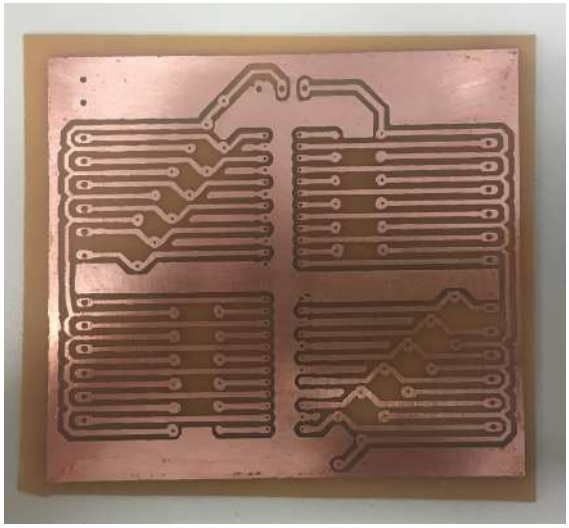


Figura 3.18: Placa cobreada já corroída pronta para etapa de furação.

A etapa seguinte se deu pela furação e soldagem dos componentes, também realizada nos laboratórios do CEFET-MG Divinópolis. Para a furação, foi utilizada a furadeira de bancada disponível no laboratório de protótipos. Foi necessário adquirir uma broca para aço de 1mm de diâmetro devido à falta da ferramenta na instituição.

Em seguida foi realizada a etapa de soldagem dos componentes em suas devidas posições de acordo com o especificado inicialmente via *software*. Na Fig. 3.19, é mostrada a vista superior e inferior da placa finalizada.

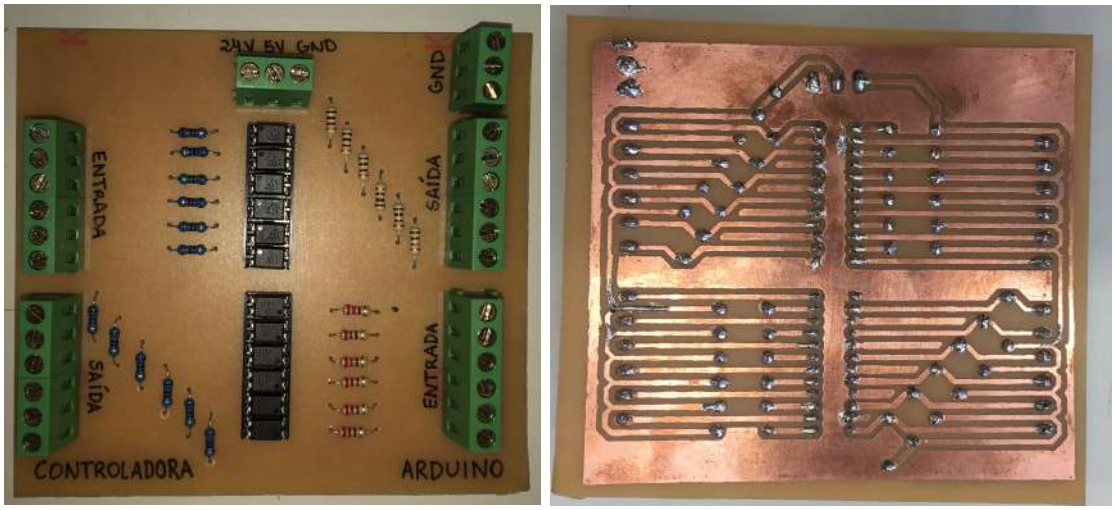


Figura 3.19: Vista superior e Inferior da placa de circuito impresso montada.

3.4 Identificação dos cubos

Em relação à visão computacional, o estudo se deu inicialmente pela identificação da posição de cubos coloridos em um fundo monocromático com a utilização de uma câmera superior. Planejou-se realizar a identificação dos vértices em conjunto com a cor predefinida das faces estimando as coordenadas do objeto. Com a inclusão da segunda câmera, foi possível determinar a orientação dos cubos a ser corrigida no empilhamento. Para estas tarefas, foi utilizada a biblioteca OpenCV programada em Python.

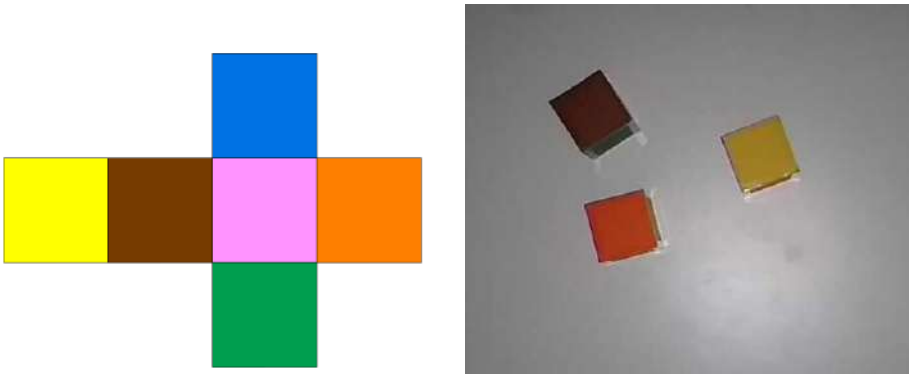
O passo inicial para a elaboração do ambiente a ser simulado deu-se pela montagem de três cubos de mesmo tamanho e orientação de faces com cores posicionadas da mesma forma. Deste modo, tornou-se possível desenvolver um algoritmo para reorientá-los de acordo com o posicionamento definido.

Em seguida, foram realizados os testes iniciais de identificação dos mesmos pela câmera através de imagens capturadas com diversas variações de posicionamento.

3.4.1 Cubos utilizados

Os cubos escolhidos para a utilização já se encontravam disponíveis no laboratório de robótica. Os mesmos são de material ABS impresso em 3D nas dimensões de 5 x 5 x 5 cm na

cor amarela. Como para este trabalho é necessário tratar a posição, bem como a orientação dos cubos, fez-se necessário cobrir as seis faces com material de cores distintas. Para isso foram recortados quadrados de 5 x 5 cm em papel cartão em seis tonalidades distintas e afixados em cada uma das faces. A representação planificada dos cubos a serem utilizados é mostrada na Fig. 3.20(a), enquanto os mesmos já montados podem ser vistos na Fig. 3.20(b).



(a) Representação planificada. (b) Fotografia de teste.

Figura 3.20: Cubos coloridos a serem utilizados.

3.4.2 Calibração da Câmera

Em sistemas que se faz necessário realizar a manipulação de objetos no espaço a partir de uma imagem estática, tem-se como primeira e principal etapa a de calibração da câmera. Isto se dá devido às características intrínsecas de cada lente, que distorce a imagem ao se projetar um ambiente tridimensional em apenas duas dimensões.

Para esta aplicação, foi utilizada uma câmera de um *smartphone* em conjunto com o aplicativo IP Webcam, pela qual se fez possível transmitir as imagens em tempo real através da rede *Wi-Fi*. Desta forma, não foi necessária a aquisição de uma *webcam* USB.

A biblioteca OpenCV possui métodos específicos para a calibração da câmera a ser utilizada. Para isto, foi realizada a impressão da figura denominada *pattern.png* composta de um padrão xadrez em preto e branco utilizado pelo algoritmo de calibração automática para cálculo dos parâmetros da câmera utilizada. O algoritmo utilizado encontra-se disponível no repositório do GitHub.

Nesta etapa são necessárias diversas fotografias com variação da posição e angulação do padrão impresso. Além disso, também é necessário realizar a medição real do lado dos quadrados. Neste caso, esta medida foi de 25 mm. Após esta ação, o algoritmo foi executado, realizando a identificação dos padrões em 30 capturas distintas, as quais são exemplificadas na Fig. 3.21.

Para correta extração dos parâmetros, o algoritmo deve identificar corretamente 54 pontos de conexão nas quinas do quadriculado e ao final, é exibida a informação da matriz da câmera

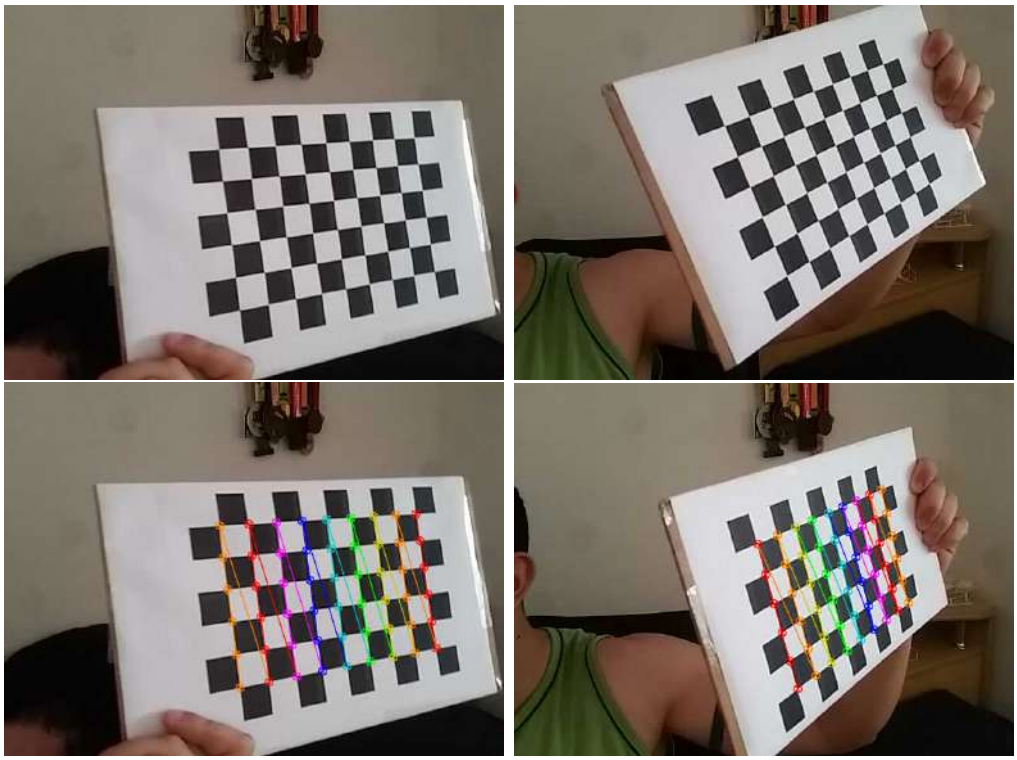


Figura 3.21: Padrão quadriculado utilizado para calibração.

e de distorção.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 807{,}72 & 0 & 477{,}86 \\ 0 & 805{,}57 & 319{,}49 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Essa é ser utilizada na correção da distorção da câmera, conforme demonstrado na Fig. 3.22.

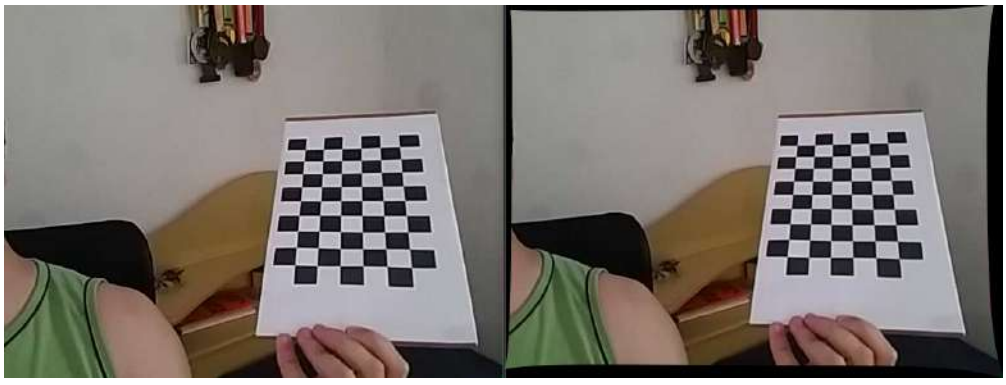


Figura 3.22: Comparação da imagem original e a mesma com correção da distorção

Percebe-se a maior distorção nas bordas da imagem, o que é de se esperar em uma câmera de *smartphone* com lente comum.

3.4.3 Identificação dos Cubos

O início do processo de identificação dos cubos deu-se pela definição do intervalo de cores a ser extraído da imagem. Para isto, foi elaborado um programa em Python que possui seis *sliders* para os ajustes de máximo e mínimo de cada um dos parâmetros H, S e V. Foi utilizada a transmissão em tempo real através da câmera superior para os ajustes referentes à separação dos intervalos de cores, conforme apresentado na Fig. 3.23.

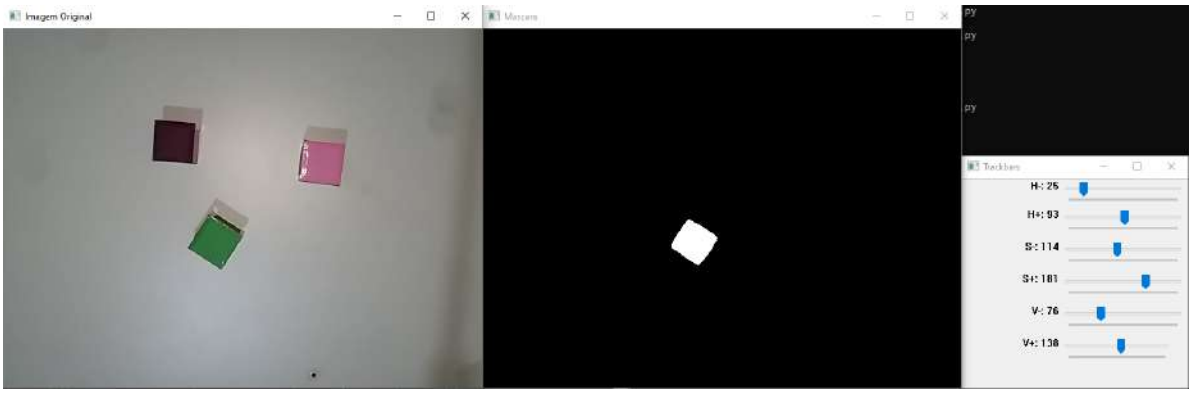


Figura 3.23: Exemplo de definição da mascara para a cor verde.

Nesta etapa, foi inicialmente realizada a filtragem gaussiana da imagem, que em seguida foi convertida para o domínio de cores HSV. Com a aplicação do filtro gaussiano, a binarização da cor selecionada ocorreu de forma mais suave, o que aumentou a precisão das demais identificações.

Após esta etapa, foi utilizada a máscara obtida para a identificação da posição planar da face identificada. Para isto foi elaborado um algoritmo para percorrer todos os *pixels* e identificar os contornos produzidos na imagem binarizada. Foi realizada uma média aritmética durante os testes iniciais para definição dos intervalos HSV e constatou-se que um valor de área abaixo de 400 *pixels* se trataria de um contorno indesejado.

Desta forma, os contornos foram obtidos apenas para áreas maiores que 400 *pixels* de modo a excluir os pontos da máscara onde foram identificados formas errôneas. Em seguida, foi utilizada uma função do OpenCV que utiliza o vetor de contornos identificados e aproxima as formas contidas pelo menor retângulo, retornando os pontos do centro da forma na imagem, largura, altura e sua inclinação, conforme mostrado na Fig. 3.24.

Com estas informações, tornou-se possível a identificação do centro geométrico dos cubos a partir da vista superior, uma vez que a altura dos mesmos é fixa em 5cm. Desta forma, resta-se apenas a conversão do ponto na imagem em duas dimensões para um ponto no plano de ação do robô.

Figura 3.24: Identificação do retângulo na cor laranja,

3.5 Transformação de coordenadas

As informações obtidas na etapa de calibração da câmera foram usadas na Equação 2.3 para a etapa de identificação da matriz de transformação do espaço bidimensional da imagem para o espaço tridimensional do mundo real. Nesta etapa, são definidos pontos previamente definidos no espaço de trabalho a ser utilizado e estes, medidos através das coordenadas em *pixels*. Neste trabalho foram definidos nove pontos e impressos em uma folha A4 para posterior medição no plano de imagem. A Fig 3.25 mostra a imagem utilizada nesta etapa, constituída de nove círculos pretos espaçados impressos em papel branco para facilitar a identificação. Neste caso a imagem foi convertida para tons de cinza e posteriormente foi utilizada a técnica de aplicação do *threshold* binarizando a captura. Utilizando a função para identificação dos contornos, em conjunto com a função *minEnclosingCircle* foram anotadas as coordenadas de cada uma das formas identificadas.

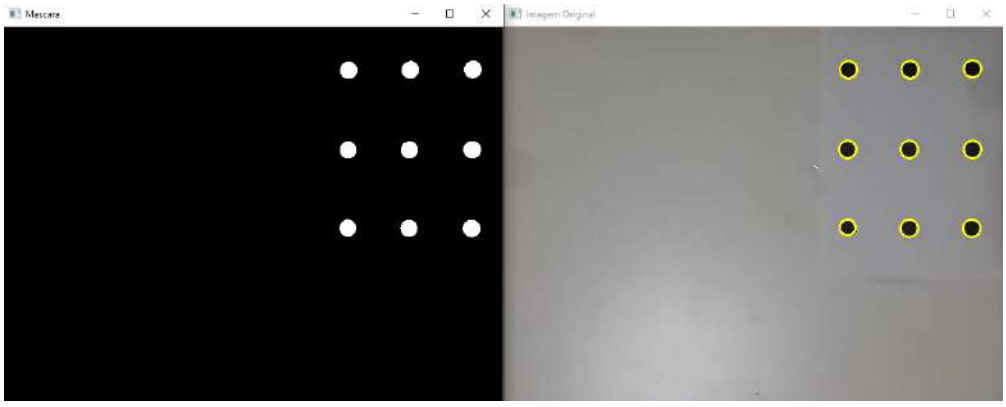


Figura 3.25: Imagens utilizadas para identificação das coordenadas do padrão.

As medições também foram realizadas com uma fita métrica considerando apenas o espaço presente na imagem e os valores obtidos são mostrados na Tab. 3.2

Com estes valores em conjunto com um algoritmo verificado em tempo real com *sliders* para ajuste das variáveis, foi possível aproximar a relação entre a posição em milímetros do objeto e a posição em pixels na imagem através de duas constantes K_u e K_v com valor

Tabela 3.2: Pontos selecionados na imagem e espaço

Número	Coord. Espacial	Coord. Imagem
1	(386,56) mm	(441, 63) px
2	(456, 56) mm	(519, 63) px
3	(526, 56) mm	(599, 62) px
4	(386, 147) mm	(441, 164) px
5	(456, 147) mm	(519, 164) px
6	(526, 147) mm	(600, 164) px
7	(386, 236) mm	(441, 264) px
8	(520, 236) mm	(456, 264) px
9	(526, 236) mm	(601, 264) px

aproximado mostrado na Equação 3.4.

$$K_u = 0.924 \qquad K_v = 0{,}9102 \tag{3.4}$$

3.6 Algoritmo centralizador

O algoritmo do trabalho, desenvolvido em Python tem por finalidade centralizar todas as tarefas em um único ambiente. Através do mesmo, foi elaborada toda a estrutura de identificação dos cubos e movimentação do robô, partindo dos conceitos já citados.

O algoritmo final foi desenvolvido em forma sequencial. Utilizando as bibliotecas disponíveis, foram elaboradas funções intermediárias para cada aplicação. Uma forma de exemplificar os procedimentos do código final é mostrada no fluxograma da Fig. 3.26.

Neste caso, são tratados apenas eventos macro, e cada processo também pode ser representado por um fluxograma individual, passando por todas as etapas de captura e processamento de imagem, definição dos pontos a percorrer e movimentações do robô.

3.6.1 Cinemática Inversa

Para a elaboração do algoritmo de resolução da cinemática inversa do robô manipulador Smart5 SiX, foi utilizado o método geométrico a partir da técnica do desacoplamento cinemático. Desta forma, inicialmente foram considerados apenas os seus três primeiros graus de liberdade, fazendo com que seja possível identificar os valores dos ângulos θ_1, θ_2 e θ_3.

Partindo do diagrama de arame mostrado anteriormente na Fig. 3.3, o robô foi representado em uma posição arbitrária (x_c, y_c, z_c) em sua vista superior na Fig. 3.27.

Assim, como o diagrama do robô em sua posição inicial não possui nenhum deslocamento no eixo y, é fácil obter o valor de θ_1 a partir da relação mostrada na Equação 3.5:

$$\theta_1 = Atan2(y_c, x_c) \tag{3.5}$$

Para os dois ângulos restantes, foi considerado que o robô terá área de trabalho limitada

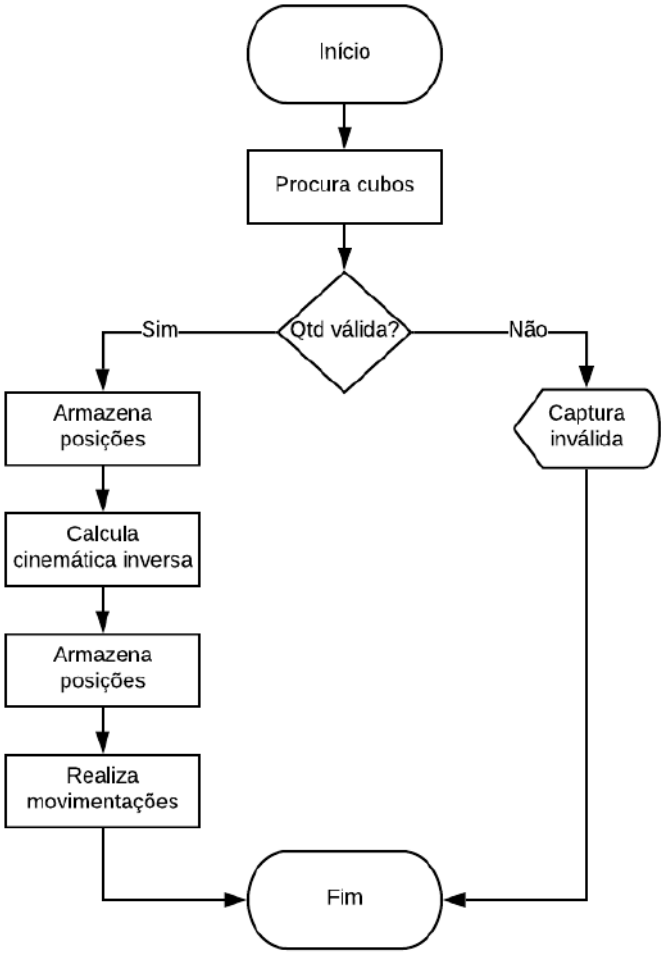


Figura 3.26: Fluxograma do algoritmo final.

à região da mesa. Deste modo, pôde-se desconsiderar algumas das soluções possíveis devido às possibilidades cotovelo superior e inferior da cinemática inversa. Então, a partir da Fig. 3.28 foram estabelecidas as relações para identificação.

Primeiro foram identificadas as seguintes dimensões independentes dos ângulos:

$$r = \sqrt{x_c^2 + y_c^2} \tag{3.6}$$

$$h1 = z_c - l_1 \tag{3.7}$$

$$b1 = r - l_2 \tag{3.8}$$

$$w = \sqrt{h1^2 + b1^2} \tag{3.9}$$

$$l_x = \sqrt{l_4^2 + l_5^2} \tag{3.10}$$

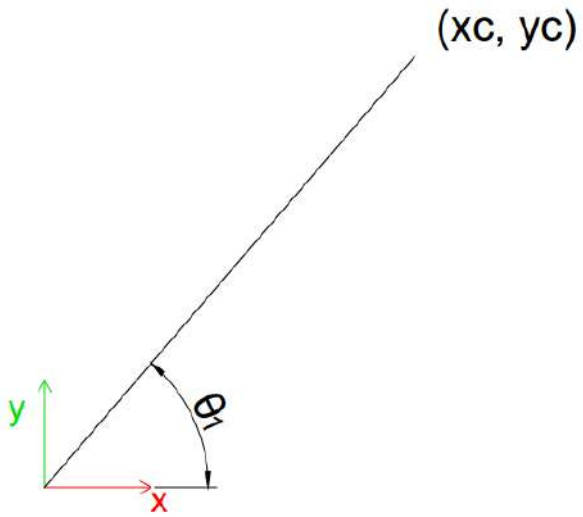


Figura 3.27: Representação superior para cinemática inversa.

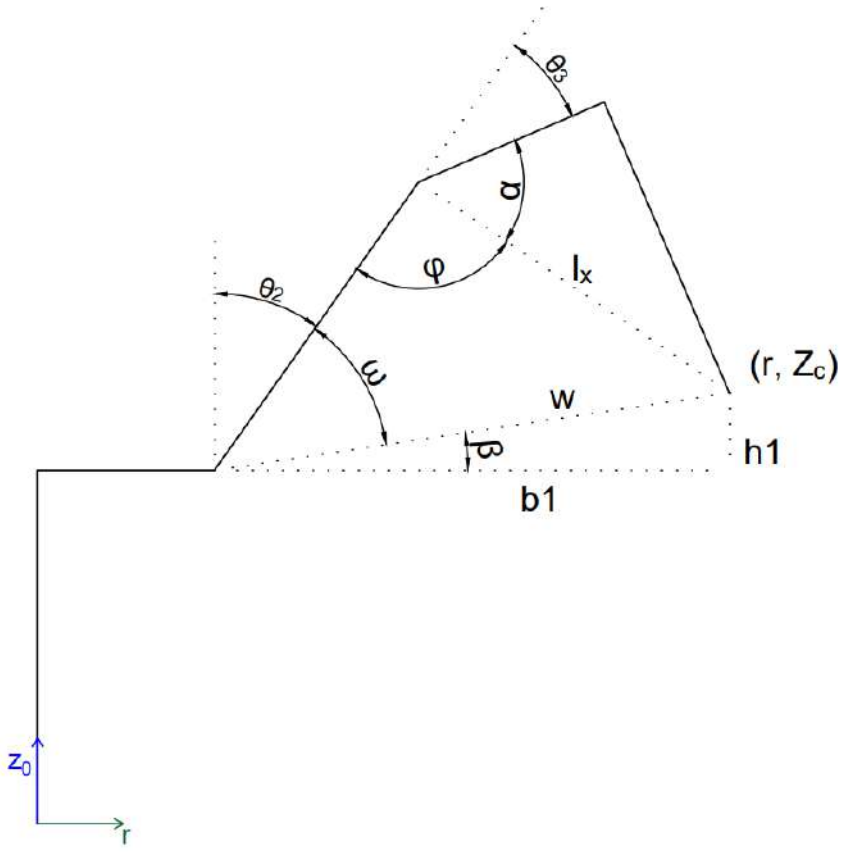


Figura 3.28: Representação lateral para cinemática inversa.

$$\alpha = tan^{-1}(l_5/l_4) \tag{3.11}$$

$$\beta = tan^{-1}(h1/b1) \tag{3.12}$$

Com estes valores já definidos, podem então serem encontrados os termos dependentes a partir das equações das leis dos cossenos e lei dos senos:

$$\phi = cos^{-1}(\frac{l_3^2 + l_x^2 - w^2}{2l_3 l_x}) \tag{3.13}$$

$$\omega = sen^{-1}(\frac{l_x sen(\phi)}{w}) \tag{3.14}$$

E finalmente, chega-se aos ângulos θ_2 e $\theta 3$:

$$\theta_2 = 90^o - \omega - \beta \tag{3.15}$$

$$\theta_3 = 180^o - \alpha - \phi \tag{3.16}$$

Para os três últimos valores angulares, inicialmente foi utilizada a comparação com a matriz de ângulos de Euler, demonstrada em na Equação 2.12.

$$A_1 = \begin{bmatrix} c1 & 0 & s1 & 150c1 \\ -s1 & 0 & c1 & -150s1 \\ 0 & -1 & 0 & 450 \\ 0 & 0 & 0 & 1 \end{bmatrix}; A_2 = \begin{bmatrix} s2 & c2 & 0 & 510s2 \\ -c2 & s2 & 0 & -510c2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

$$A_3 = \begin{bmatrix} c3 & 0 & s3 & 130c3 \\ s3 & 0 & -c3 & 130s3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Realizando as multiplicações necessárias e isolando apenas a matriz de rotação, tem-se que:

$$R_3^0 = \begin{bmatrix} c1s2c3 + c1c2s3 & s1 & c1s2s3 - c1c2c3 \\ -s1s2c3 - s1c2s3 & c1 & -s1s2s3 + s1c2c3 \\ c2c3 - s2s3 & 0 & c2s3 + s2c3 \end{bmatrix} \tag{3.17}$$

Como no trabalho em questão, o eixo z da orientação destino para manipular os cubos foi definida como o sentido negativo do *frame* base, pôde-se simplificar o problema através do método geométrico, chegando nas Equações 3.18 e 3.19:

$$\theta_4 = 0^o \tag{3.18}$$

$$\theta_5 = 90^o - \theta_2 - \theta_3 \tag{3.19}$$

Restando apenas o ângulo θ_6 para correção de orientação da garra, que é obtido diretamente da relação entre θ_1 e a informação de ângulo extraída da imagem via OpenCV.

Capítulo 4

Resultados e Discussões

Neste capítulos são listados os resultados obtidos a partir da execução das tarefas mostradas nos capítulos anteriores. Os mesmos foram subdivididos de acordo com a área abordada, permitindo com que a análise pudesse ser realizadas de forma segmentada.

4.1 Circuito de acionamento da ferramenta

Para permitir a continuidade do trabalho futuramente em uma aplicação real, foi desenvolvida a placa de acionamento mostrada no capítulo anterior. Visando verificar seu comportamento em situações semelhantes às de sua aplicação, foram realizados os testes de modo a validar seu comportamento para a comunicação de entrada e saída entre o microcontrolador Arduino e a controladora C5G.

Inicialmente, o circuito foi montado em bancada utilizando duas fontes de corrente contínua ajustáveis e foi verificado seu comportamento em cada um dos 6 canais de comunicação. A ponta de prova negativa das duas fontes foi curto-circuitada e as tensões de entrada ajustadas nos valores de 5 V e 24 V de acordo com as indicações na placa, mostradas anteriormente na Fig. 3.19.

Na primeira etapa, foi analisado o estágio de comunicação no sentido Arduino → C5G. Foram inseridos sinais de 5 V e 0 V representando os niveis lógico 0 e 1 nos 6 canais de comunicação e os valores medidos nos terminais do lado direito são mostrados na Tab. 4.1.

Como esperado devido às características do circuito eletrônico proposto, foi possível realizar a conversão de nível lógico 1 de forma adequada. No segundo caso, para o nível 0, o valor lido nos terminais de conexão com a controladora foi próximo de 3,3 V. Conforme as características do módulo de entrada e saída da controladora, todo valor abaixo de 5 V é tido como sinal baixo. Portanto, neste caso a placa elaborada se comportou de forma adequada à aplicação.

No sentido contrário, C5G → Arduino, foram realizadas verificações semelhantes alterando o nível alto de entrada desta vez para 24 V. Conforme esperado, foi obtido o valor de aproximadamente 5 V em todos os casos de sinal alto. Os valores medidos foram registrados

Tabela 4.1: Verificação do comportamento no sentido Arduino → C5G

Canal	Arduino (V)	C5G (V)
1	4.99	24.06
2	5.00	24.06
3	4.99	24.07
4	4.99	24.06
5	4.99	24.07
6	4.99	24.06
1	0	3.33
2	0	3.23
3	0	3.11
4	0	3.10
5	0	3.37
6	0	3.11

na Tab. 4.2.

Tabela 4.2: Verificação do comportamento no sentido C5G → Arduino

Canal	Arduino (V)	C5G (V)
1	24.06	4.99
2	24.07	4.99
3	24.07	4.99
4	24.07	4.99
5	24.07	4.99
6	24.07	4.99
1	0	0.66
2	0	0.70
3	0	0.59
4	0	0.64
5	0	0.67
6	0	0.66

Assim como no caso anterior, ao se inserir um sinal equivalente ao nível lógico 0, foi registrado um vestígio da tensão no lado oposto, sendo neste caso de aproximadamente 0,7 V. Em consulta ao *datasheet* do controlador do Arduino, constatou-se que para o caso de entradas digitais, um valor de tensão menor que 1,5 V é considerado nível baixo.

4.2 Ajustes na ferramenta

Ao longo da primeira etapa do trabalho de conclusão de curso, foram estudadas formas de corrigir os problemas apresentados e finalizar a ferramenta que estava disponível no laboratório de robótica do CEFET. Foram realizadas as impressões das peças projetadas visando substituir as que se encontravam defeituosas devido à erros de projeto ou adaptações para outros tipos de motores. As peças foram impressas, montadas e são apresentadas na Fig. 4.1.

Figura 4.1: Ferramenta com as novas peças impressas

Durante a montagem, percebeu-se um problema nas características físicas da forma de movimentação da ferramenta e concluiu-se que o modelo anterior passou por adaptações devido à erros de medição relacionados à engrenagem central. Com as peças redesenhadas simetricamente, as cremalheiras eram posicionadas de modo que pressionassem a engrenagem. Este fator fez com que as mesmas fossem flexionadas, aumentando o torque necessário para abertura e fechamento da ferramenta e limitando seu curso. Além disso foram identificados mais problemas, como o acoplamento da engrenagem principal no eixo do motor de corrente contínua. Este fato fez com que a tarefa, inicialmente apenas complementar, se tornasse bastante onerosa em função do tempo de desenvolvimento. Para a correção dos problemas identificados, seria necessário realizar um novo projeto completo de ferramenta. Este fator se tornou inviável durante a execução do TCC II devido às etapas de simulação que compunham o escopo principal da proposta.

Deste modo, não foi possível finalizar a ferramenta e torná-la utilizável na aplicação da técnica do robô real. Para que isto seja feito, faz-se necessário o projeto e desenvolvimento ou aquisição de uma ferramenta acoplável ao robô com sensoreamento de contato com os objetos a se manipular.

4.3 Transformação de coordenadas

A etapa mais complexa do trabalho foi referente à conversão entre o ponto no espaço bidimensional da imagem e o ponto em R^3. Pequenas variações ou incertezas nas medições podem fazer com que se obtenha um resultado totalmente oposto àquele esperado.

Conforme demonstrado ao longo do desenvolvimento, esta transformação foi realizada por aproximação linear entre a relação dos pontos no espaço e os pontos da imagem. Esta técnica pôde ser utilizada apenas devido ao fato da limitação do campo de atuação do robô. Caso fosse necessário o distanciamento maior entre os cubos, o erro acumulado impossibilitaria o sucesso da execução. Os valores mostrados na Equação 3.4 foram comparados com uma fita métrica na imagem, conforme mostrado na Fig. 4.2.

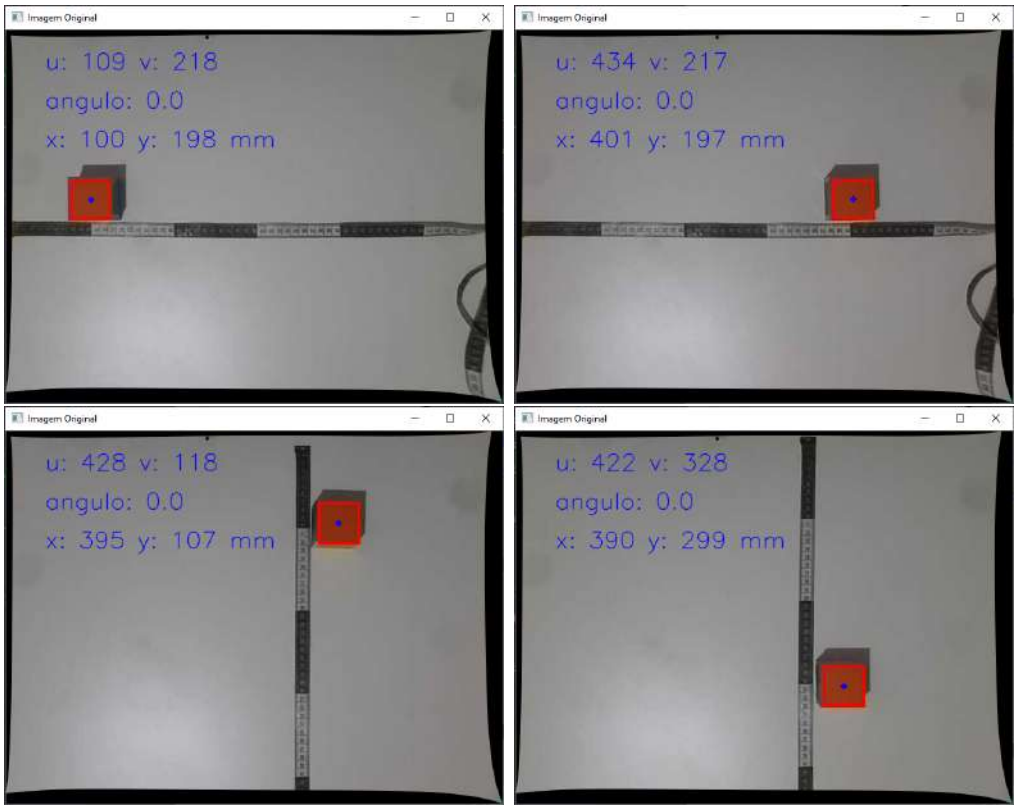


Figura 4.2: Validação da constante de transformação.

Como foi reduzida a distorção da imagem causada pela câmera através da matriz de calibração, o erro de perspectiva foi reduzido para esta aplicação. Como é esperado, o erro da aproximação é máximo nas extremidades da região de interesse. Desta forma, a aproximação linear torna-se indicada apenas para pontos próximos do centro da imagem, em que a interferência da distorção e perspectiva é menor.

4.4 Cinemática inversa e geração de trajetória

Devido às limitações estabelecidas no problema, a etapa de identificação dos ângulos dos motores para cada um dos pontos tornou-se simplificada durante a movimentação dos cubos. Uma limitação encontrada durante o desenvolvimento do trabalho foi em relação à geração de trajetória.

A biblioteca de comunicação impede o envio simultâneo dos ângulos das juntas, então se faz necessário comandar uma a uma. Desta forma, as trajetórias a serem percorridas foram

desenvolvidas a partir da utilização de pontos fixos no espaço. Cada movimento contou com pelo menos quatro pontos fixos após a identificação da posição dos cubos sendo: o primeiro imediatamente acima do cubo; o segundo em altura suficiente para o agarre do cubo; o terceiro imediatamente acima da posição de empilhamento; e o quarto ponto na posição de liberação do cubo. Assim, foi possível desenvolver as lógicas de movimentação para os casos possíveis de serem identificados pela câmera.

Para validar a estratégia de identificação dos ângulos dos motores, foram realizados testes em diversos pontos da mesa e comparadas a posição obtida em relação à posição desejada afim de calcular o erro aproximado do algoritmo desenvolvido. Estes dados são mostrados na Tab. 4.3.

Tabela 4.3: Comparação da precisão do algoritmo de cinemática inversa

	Desejado (mm)	**Calculado (mm)**	**Erro (%)**
1	(1.14200, 0.14037)	(1.14040, 0.13757)	0.17%
2	(1.14200, -0.23463)	(1.14060, -0.23715)	0.07%
3	(0.76695, -0.23463)	(0.76549, -0.23670)	0.10%
4	(0.76695, 0.14038)	(0.76500, 0.13783)	0.30%
5	(0.96698, 0.14038)	(0.96519, 0.13768)	0.22%
6	(0.96698, -0.18461)	(0.96545, -0.18698)	0.11%
7	(0.94198, -0.03461)	(0.94030, -0.03708)	0.17%
8	(0.76698, -0.05962)	(0.76526, -0.06186)	0.20%
9	(1.14200, -0.03463)	(1.14040, -0.03728)	0.13%
10	(0.96703, -0.13463)	(0.96545, -0.13704)	0.13%

Os 10 pontos foram escolhidos manualmente na área onde os cubos poderiam estar posicionados sobre a mesa, e então as informações de posição fornecidas pelo simulador foram inseridas no algoritmo em Python responsável pelo cálculo da cinemática inversa e comunicação com a API remota. Os erros mostrados foram calculados a partir do módulo dos dois pontos, desconsiderando a orientação e elevação que eram fixos para todos os casos. Na Fig. 4.3 é demonstrado um exemplo dos testes realizados.

Afim de verificar a posição final do centro da ferramenta, foi utilizado um elemento *dummy* (frame isolado) em seu centro e registrada a posição para cada ponto analisado.

4.5 Empilhamento dos Cubos

A etapa de empilhamento foi inicialmente realizada de forma comandada entre pontos definidos externamente. Neste caso o objetivo foi de verificar o comportamento da simulação com interação com outros objetos. Foi definido um ponto de referência para a realização do empilhamento sem considerar variações na orientação dos cubos. Neste caso, foi possível analisar a precisão da movimentação simulada com a técnica adotada para cinemática inversa.

Foram posicionados os três cubos na mesa e o algoritmo de comunicação, contando com a posição real de cada objeto realizou a movimentação de forma sequencial, ou seja, com a

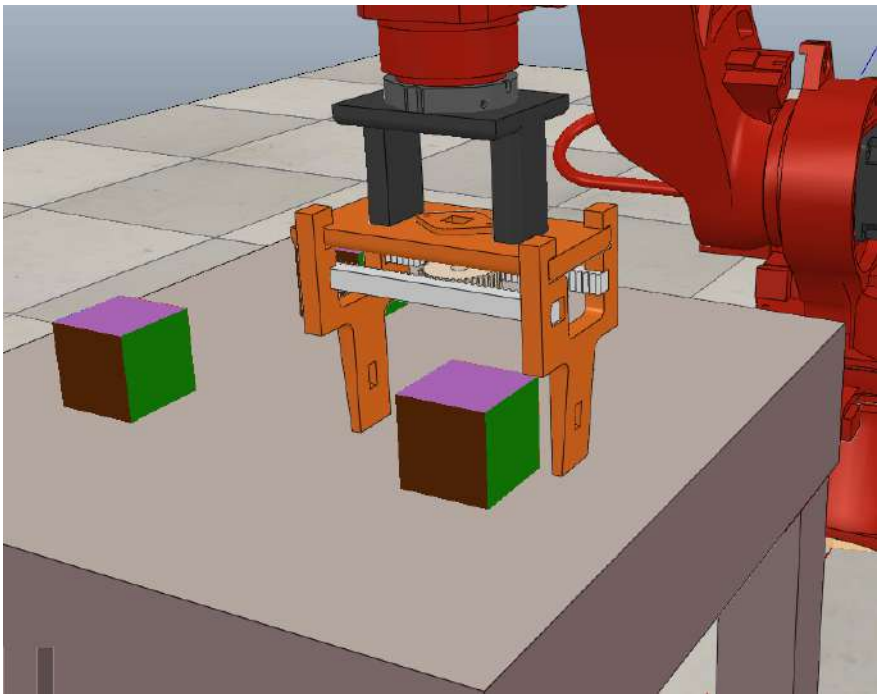


Figura 4.3: Teste do posicionamento do robô.

trajetória definida por pontos fixos no espaço. A Fig. 4.4 mostra uma captura da tela ao longo da movimentação de empilhamento dos cubos.

Como citado, o processo foi realizado através de pontos fixos no espaço. Desta forma, o posicionamento final dos cubos teve como limitação apenas a precisão da técnica de cinemática inversa adotada. Na Fig. 4.5 é possível visualizar a pequena variação da posição dos cubos causada pelo algoritmo do cálculo da posição. Para melhor visualizar o alinhamento dos componentes, também foi realizada a captura de tela considerando apenas o objeto dinâmico tratado sem as texturas de superfície.

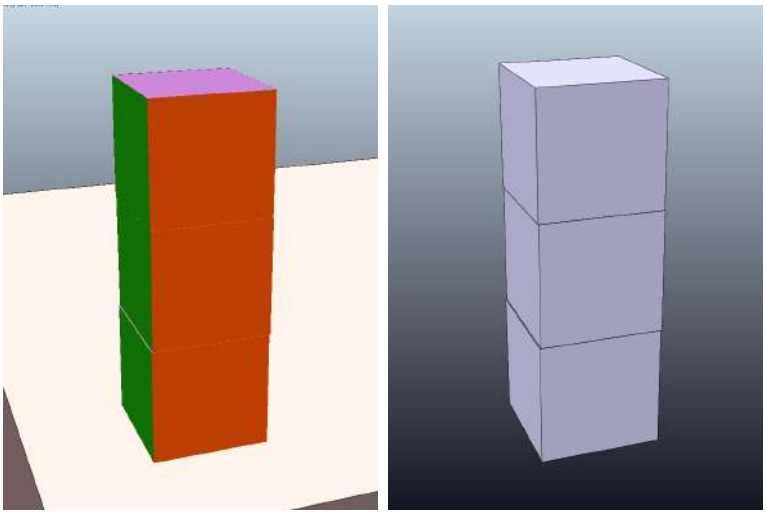


Figura 4.5: Cubos empilhados

Uma limitação encontrada ao longo desta tarefa foi referente ao contato entre a ferramenta e os cubos. Embora fosse possível atribuir um nível de atrito para cada superfície, durante a

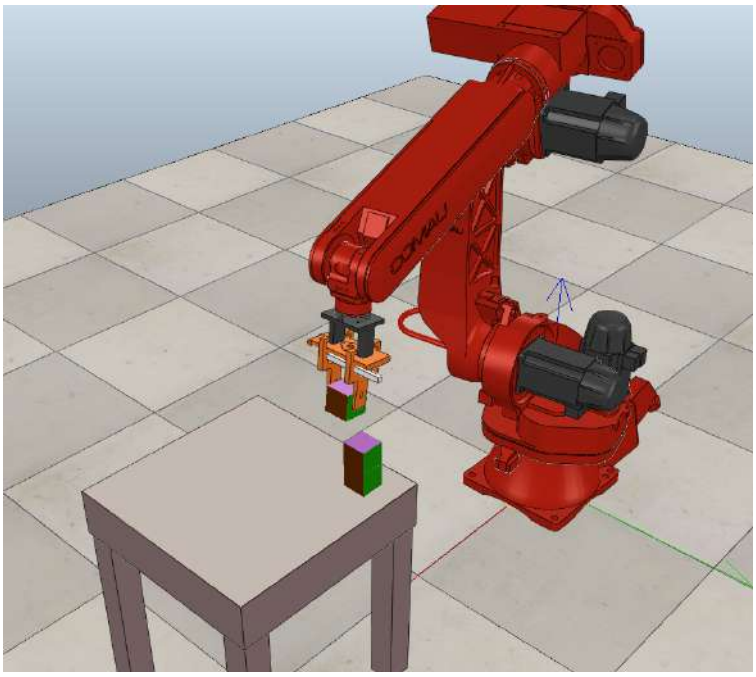


Figura 4.4: Robô simulado realizando o empilhamento

simulação este parâmetro não se tornou suficiente para que a ferramenta pudesse realizar a correta manipulação dos objetos. Utilizando apenas o contato físico, em alguns momentos os componentes se comportavam de forma inesperada, causando a imprecisão da manipulação.

Este problema foi contornado utilizando o método vrep.simxSetObjectParent(). Desta forma durante o agarre, o cubo se comportava como componente fixo dinamicamente à ferramenta. Ao final da movimentação, a mesma função foi utilizada para que os cubos novamente pertencessem à mesa mantendo suas características dinâmicas e de resposta ao contato. O vídeo desta simulação pode ser visto através do link: https://youtu.be/rd0dFAUPpMc.

4.6 Movimentação integrada com visão computacional

Para a integração dos desenvolvimentos segmentados, foi desenvolvido um algoritmo único estruturado, responsável pelo sistema desde a leitura da imagem até a movimentação do robô, conforme mostrado no fluxograma da Fig. 3.26.

Neste caso, por se tratar de uma integração entre o ambiente simulado e o real, a informação da captura de imagem foi utilizada para o posicionamento automático dos cubos utilizando a API remota em Python. Após a leitura da figura e identificação dos cubos, os mesmos foram posicionados na mesa da simulação através das funções vrep.simxSetObjectPosition() e vrep.simxSetObjectOrientation(). Conforme mostrado na Fig. 4.6, foi possível realizar o posicionamento automático dos cubos a partir da imagem externa.

Cabe-se destacar que, neste caso, a orientação das faces foi realizada anteriormente de forma manual para que a imagem resultante estivesse condizente com a fotografia real. Isto se deve ao fato de que apenas com uma câmera posicionada superiormente não é possível

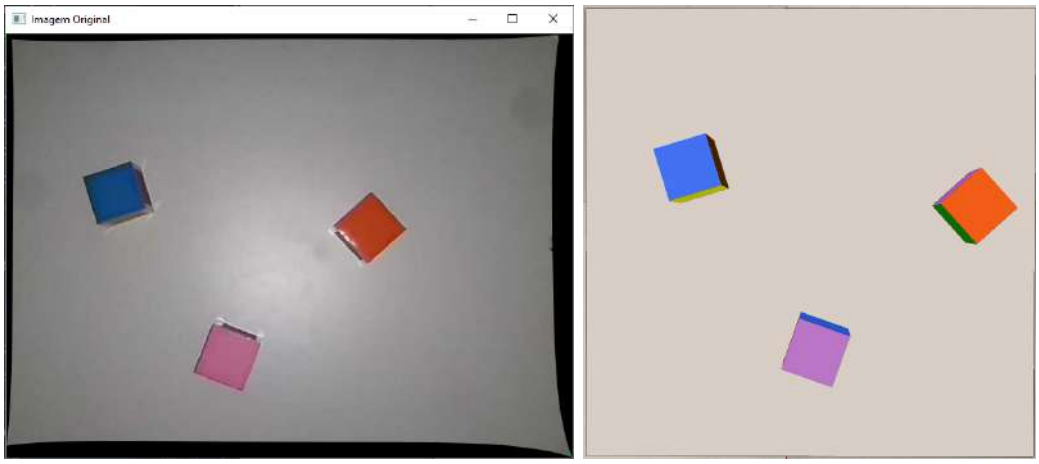


Figura 4.6: Comparação entre o posicionamento dos cubos na imagem da câmera e da simulação.

tratar este tipo de informação, sendo necessário uma segunda imagem para tal. Da mesma forma, não foi possível corrigir de forma autônoma a orientação das faces dos cubos ao longo do empilhamento. Com o código desenvolvido o robô manteve as faces azul, rosa e laranja para cima.

Deste modo, as etapas seguintes de movimentação foram realizadas conforme mostrado anteriormente via pontos fixos no espaço. Foram utilizados ao menos um ponto auxiliar superior a cada posição de manipulação. Assim, foi possível suavizar o movimento, aproximando de uma reta no espaço. Para este segundo caso, também foi gravado um vídeo que pode ser visto através do link: https://youtu.be/06JDNUfuIlM.

Capítulo 5

Considerações Finais

Neste capítulo são descritas as principais observações concluídas ao longo do desenvolvimento do Trabalho de Conclusão de Curso.

5.1 Conclusões

O presente trabalho teve por objetivo a elaboração de uma célula robótica simulada em conjunto com visão computacional. Um robô industrial Comau Smart5 SiX 6 1.4 através de coordenadas espaciais obtidas por visão computacional realizará o empilhamento de três cubos levando em consideração também a orientação de acordo com a cor de cada uma das faces. Além disso, também foi proposto a finalização de uma garra robótica para eventual aplicação da técnica no sistema real em trabalhos futuros.

O grande tempo investido em fundamentação teórica na fase inicial do trabalho abordando também possíveis aplicações da plataforma Open fez com que o desenvolvimento da primeira etapa do mesmo fosse comprometido. Entretanto, este mesmo fator fez com que o desdobramento das atividades na disciplina de Trabalho de Conclusão de Curso II ocorresse de forma equilibrada.

Como não foram encontrados os componentes necessários do robô Comau disponível no laboratório compatíveis com o *software* V-REP, foi desenvolvido um modelo funcional para permitir as simulações propostas. Também foi necessário elaborar o modelo referente à ferramenta a ser utilizada visando aproximar a simulação mais ainda da atuação real.

Durante o desenvolvimento, foram identificados mais problemas que os previstos na correção da garra. Assim, a etapa de torná-la funcional foi comprometida sendo necessário um novo projeto de ferramenta para tal. Já a placa de comunicação mostrou-se operacional nos testes realizados, sendo possível ser utilizada para todos os trabalhos que necessitarem de comunicação entre a controladora do robô e algum dispositivo acionado externamente. Assim, a mesma foi doada para o laboratório de robótica da instituição.

A movimentação do robô simulado foi realizada através de algoritmos de cinemática direta e inversa combinados. Devido às limitações possíveis da tarefa proposta, o método

geométrico foi eficaz para a movimentação ponto a ponto, conforme foi mostrado no capítulo anterior. Caso seja ampliado o espaço de trabalho referente ao posicionamento dos objetos, faz-se necessário a utilização da matriz de ângulos de Euler para identificação dos três últimos ângulos, pois a simplificação adotada não considera uma inclinação vertical resultante do distanciamento do centro da ferramenta do eixo X.

A utilização da API remota em linguagem de programação Python apresentou algumas limitações importantes em se tratando de manipuladores robóticos. Não há implementada uma função que aguarde o fim da movimentação dos motores para os passos seguintes. Então foi necessário adotar um período de *delay* após cada ponto alcançado, visando minimizar problemas na trajetória. Outra dificuldade encontrada foi em relação às juntas rotacionais. Com as mesmas dinamicamente habilitadas, caso ocorra algum movimento em sentido perpendicular ao seu giro, a mesma se comporta semelhante a uma mola, o que produz um efeito negativo tanto na parte visual quanto nos resultados das simulações de movimento.

A utilização do programa de simulação possibilitou com que fossem estudadas técnicas de movimentação de robôs de forma simplificada e sem custos, o que agrega em conhecimento e experiência para o estudante. Além disso, a técnica de visão computacional mostrou-se eficiente na identificação de posição e orientação dos cubos a partir da definição de cores fixas para cada face.

5.2 Propostas de Continuidade

Uma vez que não foi possível atingir todos os objetivos iniciais do trabalho ao longo deste desenvolvimento, tem-se como proposta de continuidade:

- Aplicação da técnica em conjunto com ROS e o robô industrial Smart5 SiX disponível no laboratório de robótica da instituição utilizando a plataforma OPEN;
- Acoplamento da segunda câmera para identificação da posição e orientação de forma independente e autônoma;
- Aperfeiçoamento do algoritmo de geração de trajetória utilizado buscando estudar o menor custo para conclusão da tarefa;
- Desenvolvimento de um novo projeto de ferramenta para a manipulação de objetos leves com precisão;
- Substituição da técnica pelo servomecanismo para que seja possível operar com objetos em movimento.

5.3 Custos Finais

Uma vez que grande parte do trabalho foi realizada em computador, os custos de desenvolvimento se tornaram bastante reduzidos. Os componentes adquiridos serão utilizados apenas para a finalização da ferramenta robótica. Os custos integrais do desenvolvimento da placa de comunicação são listados na Tab. 5.1

Tabela 5.1: Custo dos componentes da placa

Componente	**Qtd.**	**Preço (R$)**	**Subtotal (R$)**
Resistor 1,2kΩ 1W	12	0,32	3,84
Resistor 220Ω 0,25W	6	0,08	0,48
Resistor 510Ω 0,25W	6	0,08	0,48
EL817	12	0,49	5,88
Soquete 24 pinos	2	0,25	0,50
Borne 3 conexões	10	1,18	11,80
Placa fenolite	1	12,96	12,96
Broca 1mm	1	4,80	4,80
Frete	-	-	20,92
Total			61,66

Uma vez que a placa de comunicação foi desenvolvida em uma parceria com outra aluna do curso, seus custos foram divididos igualmente entre as partes e o valor total investido no trabalho é detalhado na Tab 5.2.

Tabela 5.2: Custo final do trabalho

Componente	**Modelo**	**Qtd.**	**Preço (R$)**
Arduino	UNO R3	1	-
Motor CC	12V 200 RPM	1	40,00
Ponte-H	L298N	1	-
Chave Fim de Curso	-	3	-
Placa de Comunicação	-	1	30,83
Total			**70,83**

Os itens listados sem indicação de preço encontravam-se disponíveis para utilização e foram fornecidos pelo orientador ou obtidos a partir de sucata eletrônica.

Apêndice A

Desenho das peças

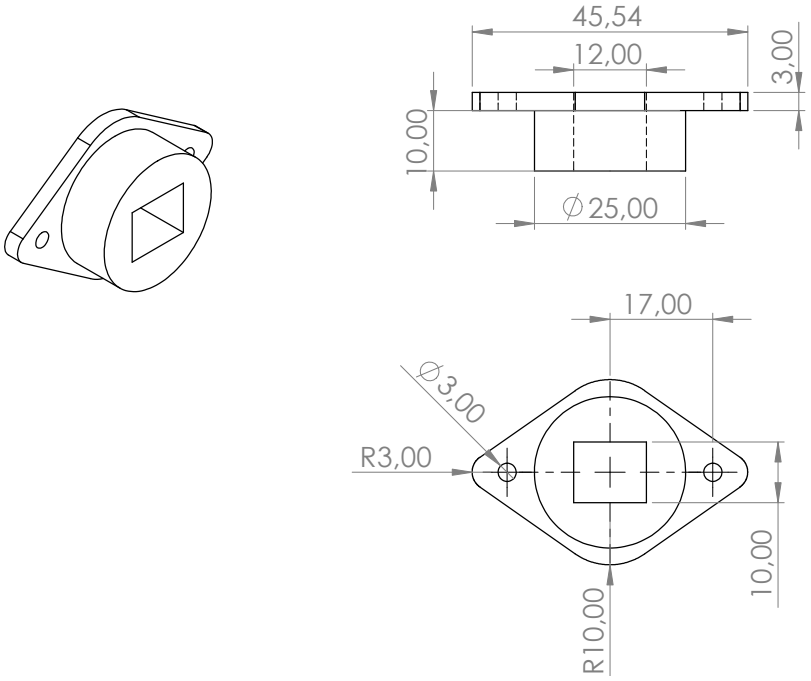


Figura A.1: Suporte do motor

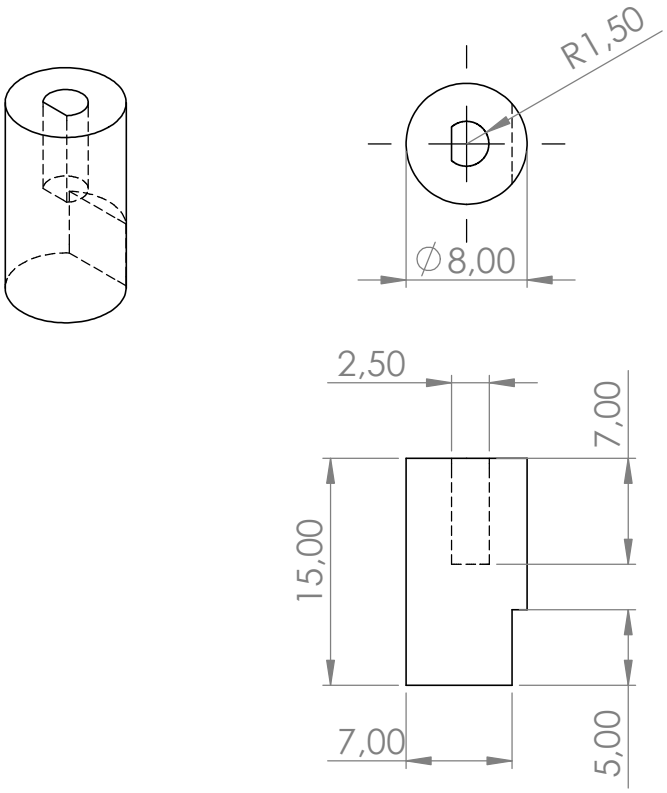


Figura A.2: Acoplamento para o eixo do motor

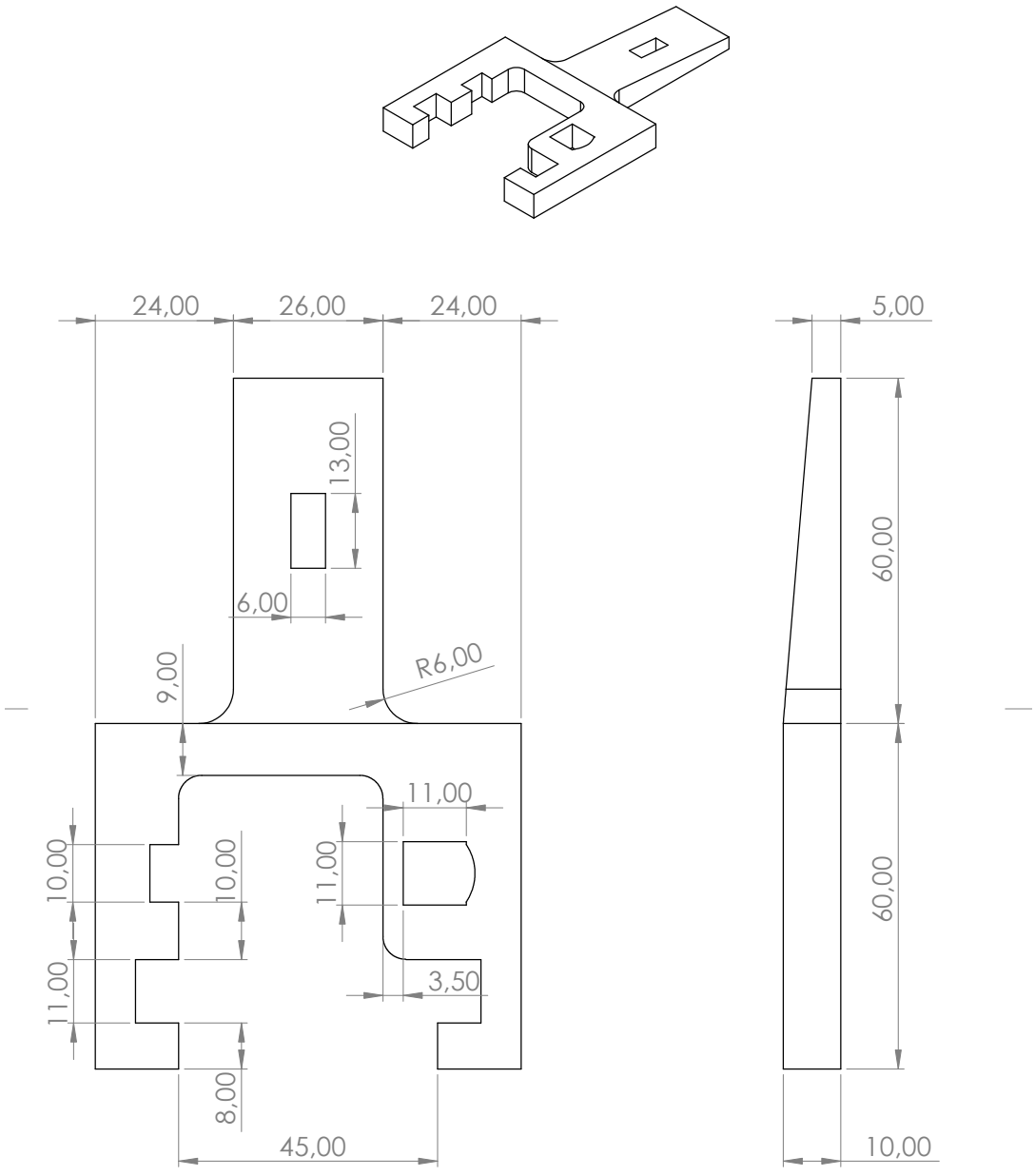


Figura A.3: Parte móvel da garra

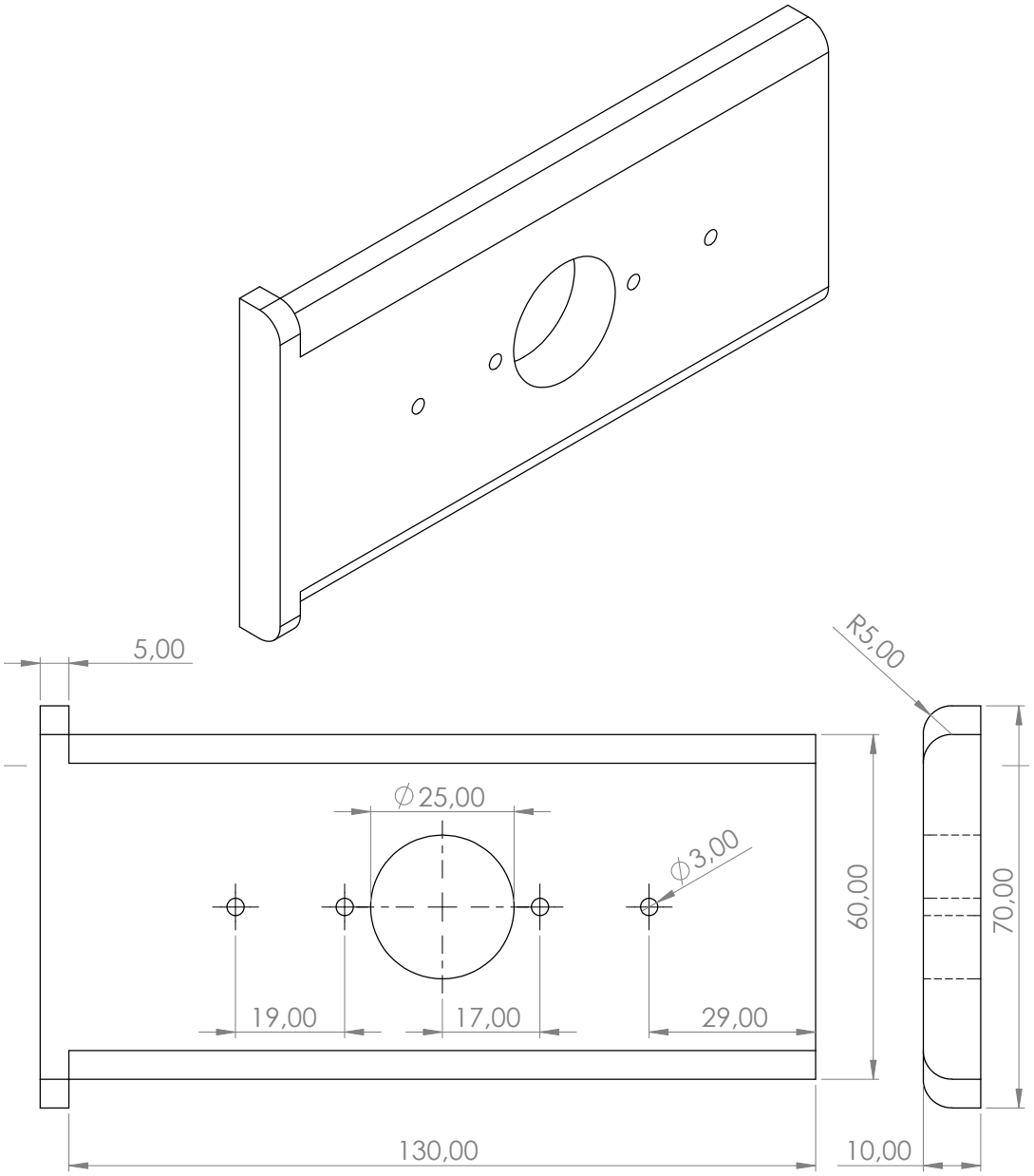


Figura A.4: Base para parte móvel

Referências

B & R, I. A. *Datasheet X20(c)DI9371*. 2018.

BALADEZ, F. O passado, o presente e o futuro dos simuladores. *Fasci-Tech Periódico Eletrônico da FATEC-São Caetano do Sul, São Caetano do Sul*, São Paulo - SP, Brasil, 2009.

BERRI, R.; GRASSI JR., V.; OSORIO, F. Simulação de Robôs Móveis e Articulados: aplicações e prática. In: SALES, C. L. S.; REBÊLO, H. (Ed.). *34ª Jornada de Atualização de Informática do XXXV CSBC*. Recife: SBC, 2015. p.274–322.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV*: computer vision with the opencv library. 1.ed. Sebastopol-CA, USA: O'Reilly, 2008.

COMAU. *SMART SiX*: technical specifications. Italy: Comau Robotics, 2005.

COMAU. *SMART SiX*: c5g controller unit. Italy: Comau Robotics, 2008.

COMAU. *PDL2*: programming language manual. Italy: Comau Robotics, 2014.

COMAU. *Six-6-1.4*: characteristics and technical specifics. Disponível em: <https://www.comau.com/EN/our-competences/robotics/robot-team/six_6-14>. Acesso em 31 mai. 2019.

COPPELIAROBOTICS. *V-REP*: virtual robot experimentation platform. Disponível em: <http://www.coppeliarobotics.com/>. Acesso em 26 mai. 2019.

COPPELIAROBOTICS. *Virtual Robot Experimentation Platform USER MANUALs*. Disponível em: <http://www.coppeliarobotics.com/helpFiles/index.html>. Acesso em 26 mai. 2019.

DÂMASO, R. S. *Implementação e controle servo visual e coordenação visuo-motora em robôs manipuladores*. 2006. Tese (Doutorado em Engenharia Elétrica) — Universidade Federal do Espírito Santo, Vitória, Brasil.

EVERLIGHT. *4 PIN DIP PHOTOTRANSISTOR PHOTOCOUPLER EL817 Series*. 2010.

FAIRCHILD, C.; HARMAN, T. L. *ROS Robotics By Example*: bring life to your robots usig ros applications. 1.ed. Birmingham, UK: Packt Publishing, 2016.

FARIAS, G.; TORRES, E.; FABREGAS, E.; VARGAS, H.; DORMIDO-CANTO, S.; DORMIDO, S. Navigation control of the Khepera IV model with OpenCV in V-REP simulator. *2018 IEEE International Conference on Automation / XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, Chile, p.1–6, Out 2018.

FERRARA, V. *C5GOpen*: the latest generation of the industrial robots open control system for university and smes. 2013. 2nd Level Master in Industrial Automation — Comau Academy, Italy.

GONZALEZ, R. C. *Processamento de Imagens Digitais*. 1.ed. São Paulo: Editora Blucher, 2000. (Advances in Industrial Control).

HARRIS, A.; CONRAD, J. M. Survey of popular robotics simulators, frameworks, and toolkits. *2011 Proceedings of IEEE Southeastcon*, Nashville, USA, p.243–249, Mar 2011.

JESUS, E. O.; COSTA JR., R. A Utilização de Filtros Gaussianos na Análise de Imagens Digitais. *XXXV CNMAC*, Natal-RN, Brasil, 2014.

MEDINA, B. V. O. *Sistema de Visão Computacional Aplicado a um Robô Cilíndrico Acionado Pneumaticamente*. 2015. (Mestrado em Engenharia) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.

OLIVARES-MENDEZ, M. A.; KANNAN, S.; VOOS, H. Vision based fuzzy control autonomous landing with UAVs: from v-rep to real experiments. *2015 23rd Mediterranean Conference on Control and Automation (MED)*, Torremolinos, Espanha, p.14–21, Jun 2015.

OPENCV. *OpenCV-Python Tutorials - Smoothing Images*. Disponível em: <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html>. Acesso em 24 mai. 2019.

QUIGLEY, M. *et al*. *ROS*: an open-source robot operating system. Kobe, Japan, 2009. 5p. v.3, n.3.2.

ROHMER, E.; SINGH, S. P. N.; FREESE, M. V-REP: a versatile and scalable robot simulation framework. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, p.1321–1326, 2013.

RUDEK, M.; COELHO, L. S.; JUNIOR, O. C. Visão Computacional Aplicada a Sistemas Produtivos: fundamentos e estudo de caso. *XXI Encontro Nacional de Engenharia de Produção*, Salvador, 2001.

SCHITCOSKI, R. *Uma aquitetura modular para sistemas de treinamento militar em operações táticas*. 2009. (Mestrado em Sistemas e Computação) — Instituto Militar de Engenharia, Rio de Janeiro - RJ, Brasil.

SCHROEDER, B. *Robot Vision*: letting robots see. UMass Lowell.

SCIAVICCO, L.; SICILIANO, B. *Modeling and Control of Robot Manipulators*. 2.ed. London: Springer, 2000. (Advanced textbooks in control and signal processing).

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot Modeling and Control*. 1.ed. USA: John Wiley & Sons, 2006.