Universidade de Brasília
Faculdade de Tecnologia

Estudo e desenvolvimento de interface de comando para braço robótico acoplado à cadeira de rodas

Luiza de Castro Monção

TRABALHO DE GRADUAÇÃO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2022

Universidade de Brasília
Faculdade de Tecnologia

Estudo e desenvolvimento de interface de comando para braço robótico acoplado à cadeira de rodas

Luiza de Castro Monção

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Walter de Britto Vidal Filho

Brasília
2022

Monção, Luiza de Castro.
M737e Estudo e desenvolvimento de interface de comando para braço robótico acoplado à cadeira de rodas / Luiza de Castro Monção; orientador Walter de Britto Vidal Filho. -- Brasília, 2022.
116 p.

Trabalho de Graduação (Engenharia de Controle e Automação) -- Universidade de Brasília, 2022.

1. Interface de comando. 2. braço robótico. 3. visão computacional. 4. WMRA. I. Vidal Filho, Walter de Britto, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Estudo e desenvolvimento de interface de comando para braço robótico acoplado à cadeira de rodas

Luiza de Castro Monção

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 23 de Setembro de 2022:

Prof. Dr. Walter de Britto Vidal Filho, UnB/FT/ENM
Orientador

Prof. Dr. José Maurício Santos Torres da Motta, UnB/FT/ENM
Examinador interno

Prof. Dra. Carla Cavalcante Koike, UnB/FT/CIC
Examinador interno

Brasília
2022

Este trabalho é dedicado àqueles que buscam, através da educação, ciência e tecnologia, melhorar o mundo e diminuir as malezas da realidade.

Agradecimentos

Agradeço aos meus pais pelo amor incondicional; aos meus irmãos Danilo e Laiza pelo companherismo e incentivo que eles dão como ninguém; aos meus amigos do curso que tornaram a jornada da graduação mais leve, especialmente ao Carlos que me ajudou a fazer infinitas medições para este projeto e ao Alexander por ter me emprestado seu braço robótico para que eu pudesse fazer testes com a interface desenvolvida. Agradeço ainda aos meus amigos Myllena, Eric, Gabriel, Stephanie, Tatiane e Rafaella, com quem dividi inúmeras aventuras e desventuras durante a graduação; à Empresa Júnior Mecajun por ter me acolhido e me ensinado tanto em tão pouco tempo, e aos amigos que lá fiz. Agradeço ao Prof. Britto por ter me orientado no desenvolvimento de um projeto voltado para tecnologias assistivas, que sempre foi meu sonho e objeto de interesse dentro das possíveis aplicações da engenharia.

“A fé na vitória tem que ser inabalável.”
(O Rappa)

Resumo

As tecnologias assistivas são tecnologias e metodologias empregadas para melhorar a qualidade de vida de pessoas com alguma deficiência. Essas tecnologias estão ganhando um espaço maior devido às políticas de inclusão social para pessoas portadoras de deficiência e a crescente democratização tecnológica. O presente projeto está no campo das tecnologias assistivas e visa desenvolver uma interface homem-máquina que permita a um usuário tetraplégico comandar um braço robótico acoplado à cadeira de rodas, de forma a realizar tarefas do dia-a-dia.

Após uma profunda pesquisa do estado da arte referente aos tipos de interface comumente utilizas em tecnologias assistivas, este projeto propõe a utilização de uma interface laser, acoplado a um suporte, com o qual o usuário poderá expressar ao robô o objeto de seu interesse. O laser é identificado através de visão computacional com a utilização de duas câmeras que permitem a visualização estéreo, viabilizando a estimação das coordenadas do objeto através da algoritmos de triangulação. Objetivando melhorar a estimação de coordenadas, foi realizada uma compensação empírica dos erros que quando comparada com as soluções disponíveis para retificação de distorções no MatLab e OpenCV possui um resultado superior e alcança um erro médio de menos de 2cm na posição 3D.

Este trabalho foi motivado por um projeto apoiado pela FAPDF.

Palavras-chave: Interface de comando. braço robótico. visão computacional. WMRA.

Abstract

Assistive technologies are technologies and methodologies used to improve the quality of life of people with disabilities. These technologies are gaining more space due to social inclusion policies for people with disabilities and the growing technological democratization. The current project is in the field of assistive technologies. It aims to develop a human-machine interface that allows a quadriplegic user to command a robotic arm attached to a wheelchair to perform day-to-day tasks.

After deep research of the state of art regarding the types of interface commonly used in assistive technologies, this project proposes the use of a laser interface, coupled with an eyeglass, with which the user can express the object of interest to the robot. The laser is identified through computer vision using two cameras that allow stereo visualization, enabling the estimation of the object coordinates through triangulation algorithms. In order to improve the estimation of coordinates, an empirical compensation of errors was performed, which, when compared to the solutions available for correction of distortions in MatLab and OpenCV, has a superior result and reaches an average error of less than 2 cm in the 3D position.

This work was motivated by a project supported by FAPDF.

Keywords: Command interface. robotic arm. computer vision. WMRA.

Lista de ilustrações

Figura 1 – Níveis de lesão vertebral e diagnósticos. 18
Figura 2 – Joysticks adaptados. Fontes (ANDITEC, s.d.) e (OLIVEIRA, 2019) . . . 22
Figura 3 – Princípio de funcionamento do sensor de pressão. Fonte (PINTO, 2016) 22
Figura 4 – Transdutor de ar. Fonte: (OLIVEIRA, 2019) 23
Figura 5 – Painel de LEDs. Fonte: (OLIVEIRA, 2019) 23
Figura 6 – Ilustração das regiões exploradas na eletro-oculogragia. Fonte (BAREA; LÓPEZ; MAZO, 2002) 24
Figura 7 – Touca com eletrodos utilizado na técnica de eletroencefalografia. Fonte: (JULIÃO, 2021) 24
Figura 8 – Ilustração de sensor posicionado no topo da cabeça para detecção de movimento. Fonte: (SILVA, 2013) 25
Figura 9 – Cadeira de rodas controlada pelo movimento dos olhos. (BATISTOTI, 2018) 26
Figura 10 – Cadeira de rodas controlada por reconhecimento de expressões faciais. (SOUSA, 2019) 27
Figura 11 – Esquemático do projeto desenvolvido em (ZHONG; ZHANG, Y.; YANG, 2019) 27
Figura 12 – Uso de *machine learning* na classificação de objetos em (ZHONG; ZHANG, Y.; YANG, 2019) 28
Figura 13 – Design do braço Jaco. (ROBOTICS, 2022) 29
Figura 14 – Exemplos de uso do braço robótico Jaco. (ROBOTICS, 2022) 29
Figura 15 – Usuário utilizando KARES (SONG et al., 1998). 30
Figura 16 – Relações de entrada e saída para cada subsistema de interface. Fonte: (BIEN; CHUNG; CHANG, 2004) 30
Figura 17 – Partição sequencial da vista em 4 quadrantes. Fonte: (TSUI; YANCO, 2007) 31
Figura 18 – Estrutura do braço robótico da Weston. Fonte: (HILLMAN et al., 2002) . 32
Figura 19 – Vista frontal do FRIEND-I. (MARTENS; PRENZEL; GRAESE, 2007) . . 33
Figura 20 – Esquemático do FRIEND-I. (MARTENS; PRENZEL; GRAESE, 2007) . . 33
Figura 21 – Design WMRA do FRIEND-II. Fonte: (MARTENS; PRENZEL; GRAESE, 2007) 34
Figura 22 – Design WMRA do Raptor. (BARROS DA SILVA NÉTO, 2019) 34
Figura 23 – Projeto conceitual. Fonte: (BARROS DA SILVA NÉTO, 2019) 35
Figura 24 – Configuração do manipulador com seus elos indicados. Fonte: (BARROS DA SILVA NÉTO, 2019) 36
Figura 25 – Relação entre cinemática direta e inversa. Fonte: (CARRARA, 2015) . . 36
Figura 26 – Desenho esquemático do manipulador com os parâmetros D-H. Fonte: (BARROS DA SILVA NÉTO, 2019) 37

Figura 27 – Configuração do manipulador *Pégaso*. Fonte: (PASCHOALETTO, 2022) 38
Figura 28 – Configuração do manipulador *Pégaso* e seus elos indicados. 38
Figura 29 – Interface de comando do manipulador *Pégaso*. Fonte: (PASCHOALETTO, 2022) 39
Figura 30 – Ilustração do método da triangulação para cinemática inversa. Fonte: (MÜLLER-CAJAR; MUKUNDAN, 2007) 40
Figura 31 – Representação de uma imagem digital. Adaptado de (DIAS; BOM; ALBUQUERQUE, 2017) 42
Figura 32 – Relação entre coordenadas da câmera e coordenadas do espaço. Adaptado de (HOSEINI; KABIRI, 2018) 45
Figura 33 – Triangulação. Fonte: (MORAIS MADALENA; MENOTTI, s.d.) 46
Figura 34 – Plano epipolar. Fonte: (FARIAS, s.d.) 49
Figura 35 – Esquemático 1 de determinação de profundidade de um objeto na imagem. Fonte: (PINHO, s.d.) 51
Figura 36 – Esquemático 2 de determinação da profundidade de um objeto na imagem. Fonte: (JESUS et al., s.d.) 52
Figura 37 – Análise de valores assumidos pelo ponto laser no espaço de cores HSV. Fonte:(ZHONG; ZHANG, Y.; YANG, 2019) 53
Figura 38 – Dimensões do joystick. (ELETRÔNICA, 2021) 58
Figura 39 – Esquemático elétrico do joystick. (ELETRÔNICA, 2021) 58
Figura 40 – Esquemático da solução proposta. Fonte: adaptado de (ZHONG; ZHANG, Y.; YANG, 2019). 59
Figura 41 – Detecção do ponto laser - objeto maçã. 64
Figura 42 – Detecção do ponto laser - objeto mamão. 65
Figura 43 – Sistema de visão estéreo implementado. 65
Figura 44 – Processo de medição do ângulo de abertura horizontal. 66
Figura 45 – Processo de medição do ângulo de abertura vertical. 67
Figura 46 – Processo de medição dos ângulos de abertura. 67
Figura 47 – Ilustração da distância focal horizontal. 68
Figura 48 – Ilustração da distância focal vertical. 69
Figura 49 – Ilustração da origem do sistema de coordenadas global estabelecido. . . 69
Figura 50 – Ilustração das relações geométricas da triangulação de câmeras. 70
Figura 51 – Ilustração das relações geométricas para obtenção do ângulo formado entre um ponto e o centro da imagem. 70
Figura 52 – Ilustração das relações geométricas para obtenção a distância a partir da disparidade. Fonte: (NIELSEN, 2022) 72
Figura 53 – Ambiente estruturado para medições com papel milimetrado. 73
Figura 54 – Medições para distância de 15 cm. 73
Figura 55 – Medições para distância de 30 cm 73

Figura 56 – Processo empírico das medições 90 cm. 73
Figura 57 – Programa implementado para obtenção das medições. 74
Figura 58 – Gráfico e regressão linear das medições realizadas. 75
Figura 59 – Gráfico do erro em função da distância real e regressões polinomiais associadas. 75
Figura 60 – Imagem original 76
Figura 61 – Imagem com remoção de distorções 76
Figura 62 – Limite esquerdo da área comum do sistema de visão estéreo. 76
Figura 63 – Limite direito da área comum do sistema de visão estéreo. 77
Figura 64 – Gráfico do erro para distância de 400 mm e regressão polinomial associada 78
Figura 65 – Erro para todas as distâncias. 79
Figura 66 – Análise do coeficiente a das regressões realizadas para cada distância. . 80
Figura 67 – Análise do coeficiente b das regressões realizadas para cada distância. . 81
Figura 68 – Análise do coeficiente c das regressões realizadas para cada distância. . 81
Figura 69 – Erro corrigido para todas as distâncias medidas variando X. 83
Figura 70 – Gráfico do erro para distância de 200 mm e regressão polinomial associada 84
Figura 71 – Erro para todas as distâncias. 85
Figura 72 – Erro corrigido para todas as distâncias medidas variando Y. 86
Figura 73 – Malha quadriculada utilizada para calibração do sistema estéreo. 87
Figura 74 – Layout do aplicativo de calibração do Matlab. 87
Figura 75 – Aplicação do processo de retificação. 88
Figura 76 – Reconstituição do processo de calibração a partir dos parâmetros extrínsecos encontrados. 88
Figura 77 – Layout do aplicativo de threshold. 89
Figura 78 – Aplicação do threshold para isolar somente o ponto laser. 90
Figura 79 – Processo de calibração pelo OpenCV. 91
Figura 80 – Indicação dos sistemas de coordenadas da câmera e do robô. 93
Figura 81 – Ilustração do circuito de acionamento do laser. 95
Figura 82 – Modelagem do suporte laser modelo óculos. Vista frontal. 96
Figura 83 – Modelagem do suporte laser modelo óculos. Vista isométrica. 96
Figura 84 – Modelagem do suporte laser modelo óculos. Detalhes internos. 97
Figura 85 – Cenário montado para testes. 98
Figura 86 – Cenário de teste pela câmera 1. 99
Figura 87 – Cenário de teste pela câmera 2. 99

Lista de tabelas

Tabela 1 – Dimensões dos elos. Fonte: (BARROS DA SILVA NÉTO, 2019) 35
Tabela 2 – Dimensões dos elos. Fonte: (BARROS DA SILVA NÉTO, 2019) 35
Tabela 3 – Parâmetros de D-H e seus valores. Fonte: (BARROS DA SILVA NÉTO, 2019) 37
Tabela 4 – Dimensões dos elos do robô *Pégaso*. 39
Tabela 5 – Comparativo entre interfaces - Matriz de decisão. 54
Tabela 6 – Especificações das câmeras. 65
Tabela 7 – Especificações das câmeras. 67
Tabela 8 – Resumo das medições realizadas. Dimensões em milímetro (mm). . . . 74
Tabela 9 – Especficações do diodo laser. 94
Tabela 10 – Especificações das baterias. 95
Tabela 11 – Coordenadas conhecidas dos cubos do cenário. 99
Tabela 12 – Coordenadas encontradas pelo método empírico. 100
Tabela 13 – Coordenadas encontradas pelo software Matlab. 100
Tabela 14 – Coordenadas encontradas pela biblioteca OpenCV. 100
Tabela 15 – Análise entre as abordagens utilizadas para obtenção das coordenadas 3D. As dimensões estão em milímetros. 101

Lista de abreviaturas e siglas

ABRAFIN	Associação Brasileira de Fisioterapia Neurofuncional	20
CAD	Computer Aided Design	63
CAE	Computer-Aided Engineering	63
CAM	Computer-Aided Manufacturing	63
CNN	Convolutional Neural Network	60
D-H	Denavit-Hartenberg	36
HSV	Hue-Saturation-Value	53
HTML	HyperText Markup Language	39
HTTP	Hypertext Transfer Protocol	39
IDE	Interface de Desenvolvimento	62
IHM	Interface Homem-Máquina	35
PCB	Printed Circuit Board	63
TA	Tecnologia Assistiva	20
UnB	Universidade de Brasília	20
WMRA	Wheelchair-Mounted Robotic Arms	28

Sumário

1 INTRODUÇÃO 17
1.1 Contextualização 17
1.2 A deficiência 18
1.3 Problematização 19
1.4 Objetivo geral 19
1.5 Objetivos específicos 19
1.6 Metodologia e Organização do trabalho 19

2 REVISÃO BIBLIOGRÁFICA 21
2.1 Interfaces de comando para cadeira de rodas 21
2.1.1 Joystick 21
2.1.2 Sensor de Sopro/sucção 22
2.1.3 Eletrodos 23
2.1.4 Movimentação da cabeça e uso de sensores 25
2.1.5 Comando de voz 25
2.1.6 Visão computacional 26
2.2 Estado da arte: Braços Robóticos Montados sobre Cadeiras de Rodas (WMRA) 28
2.2.1 Jaco 28
2.2.2 KARES 29
2.2.3 Manus Arm Robot 31
2.2.4 Weston Wheelchair 31
2.2.5 FRIEND 32
2.2.6 Raptor 33
2.3 Projetos de base 34
2.3.1 Manipulador robótico utilizado de base teórica 34
2.3.1.1 Especificações do Manipulador 35
2.3.1.2 Modelagem cinemática 36
2.3.2 Manipulador utilizado para testes 37
2.3.2.1 Especificações do Manipulador 38
2.3.2.2 Modelagem cinemática 39

3 FUNDAMENTAÇÃO TEÓRICA 42
3.1 Imagem digital 42
3.2 Processamento de Imagem 43
3.3 Visão computacional 44

3.3.1 Métodos para reconstrução de coordenadas 3D . . . 44
3.3.1.1 Visão Estéreo . . . 45
3.3.1.2 Calibração da câmera . . . 46
3.3.1.3 Geometria epipolar . . . 48
3.3.1.4 Retificação . . . 50
3.3.1.5 Cálculo de profundidade por trigonometria . . . 50
3.3.1.6 Cálculo de profundidade conhecendo informações sobre o objeto alvo . . . 51
3.3.2 Thresholding . . . 52
3.3.3 Detecção do ponto laser . . . 53

4 METODOLOGIA . . . 54
4.1 Análise das interfaces . . . 54
4.2 Interface baseada em Joystick . . . 58
4.3 Interface baseada em Visão Computacional . . . 58

5 IMPLEMENTAÇÃO . . . 61
5.1 Visão Computacional . . . 63
5.1.1 Detecção do ponto laser . . . 63
5.1.2 Visão estéreo - Abordagem com ajuste polinomial de erro . . . 65
5.1.2.1 Sistema de triangulação e cálculo das coordenadas 3D . . . 69
5.1.2.2 Resultados e Ajustes de erros . . . 72
5.1.2.2.1 Medições de distâncias (coordenada Z) . . . 72
5.1.2.2.2 Análise de distorção (medições horizontais) . . . 76
5.1.2.2.3 Análise de distorção (medições verticais) . . . 84
5.1.3 Visão estéreo - Abordagem com ferramentas computacionais . . . 86
5.1.3.1 Matlab . . . 86
5.1.3.2 OpenCV . . . 90
5.2 Comunicação com o robô . . . 92
5.3 Sistema de acionamento do ponto laser . . . 94
5.4 Modelagem do suporte laser . . . 95

6 RESULTADOS E DISCUSSÕES . . . 98

7 CONCLUSÃO . . . 102

REFERÊNCIAS . . . 104

APÊNDICES 109

APÊNDICE A – FIGURAS E GRÁFICOS . . . 110

APÊNDICE B – CÓDIGOS DE PROGRAMAÇÃO 113
B.1 Implementação da visão estéreo e triangulação 113

1 Introdução

1.1 Contextualização

O crescente desenvolvimento tecnológico testemunhado nos diversos âmbitos da sociedade vem trazendo melhorias na qualidade de vida, na prestação de serviços, na aquisição de novos conhecimentos, na geração de novos ofícios, dentre outros benefícios. Uma das áreas que se destaca nesse sentido é a robótica, que busca facilitar, otimizar e aprimorar uma série de atividades por meio de manipuladores eletro-mecânicos controlados por algoritmos computacionais, sendo esta uma área interdisciplinar que integra conhecimentos de engenharia mecânica, elétrica/eletrônica e da ciência da computação. O uso da robótica vem sendo integrado em inúmeras áreas da sociedade, como por exemplo, em indústrias, na exploração espacial, na medicina, na prestação de serviços gerais, etc.(ROSÁRIO, 2010)

O escopo deste trabalho debruça-se acerca da robótica aplicada ao campo da medicina e reabilitação. Dentro deste contexto, surgem as Tecnologias Assistivas (TA) que são "produtos, equipamentos, dispositivos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e à participação da pessoa com deficiência ou com mobilidade reduzida, visando à sua autonomia, independência, qualidade de vida e inclusão social" segundo (BRASIL, 2015). Assim sendo, o uso da robótica e de técnicas associadas com o propósito de melhorar a qualidade de vida de indivíduos portadores de deficiência configura-se uma TA. Especificamente, o presente trabalho abordará uma tecnologia assistiva direcionada à indivíduos com deficiências motoras causadas por lesão medular, na condição da tetraplegia.

Segundo o Censo IBGE de 2010, estima-se que cerca de 23,9% da população brasileira conviva com alguma deficiência/limitação, das quais cerca de 6,27% configura-se em deficiência motora (IBGE, 2010). No contexto global, é estimado que aproximadamente 15% da população mundial conviva com algum tipo de deficiência, e cerca de 250 milhões de idosos sofrem de deficiências motoras moderadas a graves (NATIONS, 2018), o que é um dado relevante levando em consideração a tendência global de envelhecimento populacional. Diante disto, é notória a pertinência da temática sobre Tecnologias Assistivas, visto que grande parte da população portadora de deficiência física convive com inúmeras limitações, não só de ordem física, mas também de ordem social, profissional e financeira, acentuando ainda mais as desigualdades.

1.2 A deficiência

A lesão medular espinhal é qualquer trauma às estruturas do canal vertebral que interrompe parcial ou totalmente o sinal neurológico através da medula, resultando em paralisia e ausência de sensibilidade do nível da lesão para baixo, (ABRAFIN, 2018) podendo levar a alterações motoras, sensitivas, autonômicas e psicoafetivas. (SAÚDE, 2013) Segundo a Associação Brasileira de Fisioterapia Neurofuncional (ABRAFIN), o grau de limitação do indivíduo dependerá do nível da lesão, que corresponde à altura em relação à extensão da coluna e ao tamanho da lesão, que pode ser completa ou incompleta. (ABRAFIN, 2018) Na Figura 1 é apresentado um esquema onde é possível verificar o grau da limitação motora de acordo com o nível (localização) da vértebra atingida.

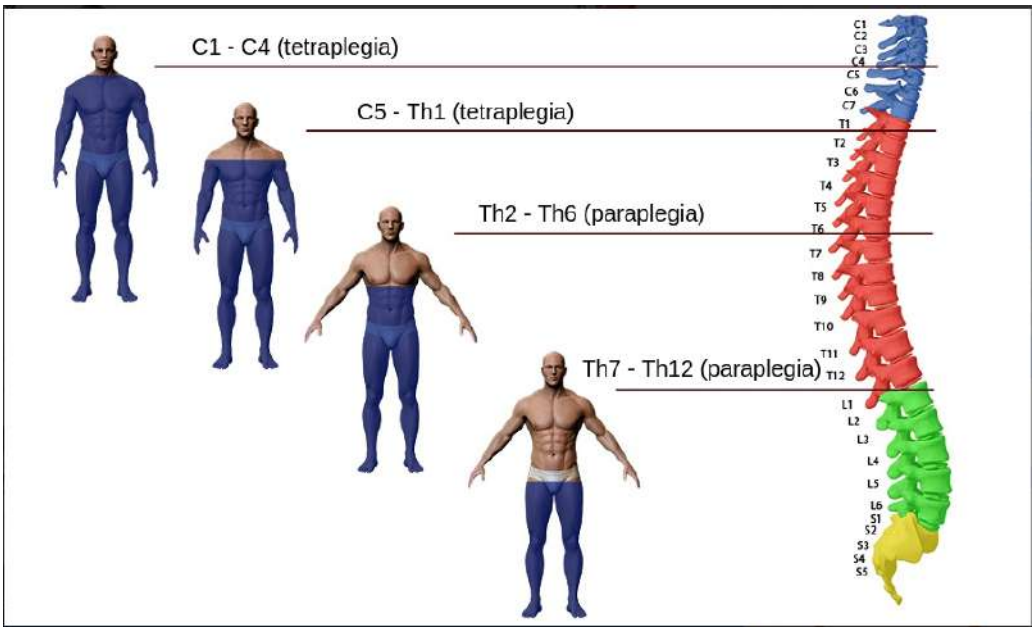


Figura 1 – Níveis de lesão vertebral e diagnósticos.

Fonte: Adaptado de (GARCIA, 2009)

A tetraplegia ou quadriplegia é uma lesão medular que compromete os movimentos dos membros superiores e inferiores e, em casos mais graves, limita a movimentação do pescoço, da respiração e da fala. Essa lesão é decorrente de fraturas localizadas nas regiões cervicais e torácicas, entre as vértebras C1 e T1. As principais causas da tetraplegia são de origem traumática, decorrente de acidentes em/com automóveis, quedas, mergulhos de cabeça ou ferimento por arma de fogo; mas também pode ser decorrente de causas não traumáticas, e sim de patologias como acidente vascular cerebral, isquemia, hérnia de disco, tumores intra ou extra medulares, entre outros. (SAÚDE, 2013)

1.3 Problematização

Como explicitado anteriormente, a tetraplegia é uma limitação motora severa, que compromete a autonomia e socialização do indivíduo. Estima-se que o número de indivíduos portadores de tetraplegia seja muito maior do que o documentado, pois muito destes vivem isolados da sociedade e frequentemente são mantidos acamados devido às dificuldades envolvidas no seu no cuidado diário, já que dependem de ajuda para executar atividades simples e básicas para sobrevivência, como se alimentar. Além disso, os problemas psicológicos decorrentes do grau de limitação e dependência podem tornar as condições de vida do portador de tetraplegia muito precárias.

1.4 Objetivo geral

Realizar um estudo sobre as principais abordagens e soluções de interfaces de comando homem-máquina (IHM), no âmbito de tecnologias assistivas, para comandar um braço robótico acoplado à cadeira de rodas, destinado a usuários portadores de tetraplegia, para então propor e implementar uma solução baseada nas análises do estudo gerado.

1.5 Objetivos específicos

Propor uma solução de interface de comando entre indivíduos tetraplégicos e um braço robótico acoplado à cadeira de rodas motorizada, com o intuito de fornecer maior autonomia e melhoria nas condições de vida destes indivíduos. Para isto, deseja-se:

1. realizar uma análise sobre as principais interfaces comumente utilizadas em projetos de tecnologias assistivas;
2. propor uma solução a nível de software baseada nas análises levantadas, que desempenhará o papel de comando para o braço robótico, processando as informações obtidas pelos sensores e calculando as coordenadas tridimensionais do alvo;
3. propor uma solução a nível de hardware que irá interagir com o software, complementando a interface.

Este projeto é uma extensão do Trabalho de Conclusão de Curso desenvolvido por (BARROS DA SILVA NÉTO, 2019), atendendo à sugestão de projetos futuros. Assim, a estrutura do manipulador desenvolvido será utilizada como base para o presente projeto.

1.6 Metodologia e Organização do trabalho

O presente documento está estruturado da seguinte forma:

- Introdução: capítulo atual contendo introdução, contextualização, exposição da problemática e definição dos objetivos do projeto;
- Revisão bibliográfica: capítulo que contempla as pesquisas e revisões bibliográficas acerca de interfaces de comando comumente utilizadas em TAs, bem como o estado da arte referente à braços robóticos montados sobre cadeiras de rodas, e também a apresentação de trabalhos anteriores diretamente relacionados;
- Fundamentação teórica: capítulo que aborda toda a teoria e conhecimentos necessários para a realização deste trabalho;
- Metodologia: capítulo onde se realiza uma análise detalhada das interfaces, apresenta propostas de soluções baseadas na análise gerada e expõe as implicações e problemáticas associadas às soluções;
- Implementação: capítulo que aborda as implementações realizadas da solução proposta;
- Resultados: capítulo contendo os resultados obtidos e análises associadas;
- Considerações Finais: capítulo contendo conclusão do trabalho e perspectivas de trabalhos futuros.

A metodologia utilizada no desenvolvimento deste trabalho para avaliação das propostas levantadas foi baseada no uso de matriz de decisão, levando em consideração os aspectos considerados mais relevantes para a solução do problema, como usabilidade e autonomia.

2 Revisão Bibliográfica

Neste capítulo serão abordadas as revisões bibliográficas de suporte para o desenvolvimento do presente trabalho: as interfaces de comando mais utilizadas em tecnologias assistivas, o estado da arte referente aos braços robóticos acoplados em cadeiras de rodas, comumente referenciados como WMRA (do inglês *Wheelchair-mounted Robotic Arms*), o projeto do manipulador utilizado de base para o desenvolvimento deste projeto e o projeto do manipulador utilizado para fins de teste da interface desenvolvida.

2.1 Interfaces de comando para cadeira de rodas

A revisão bibliográfica foi realizada levando em consideração as interfaces de comando comumente utilizadas por usuários com limitações motoras severas em projetos envolvendo cadeira de rodas. Assim, as pesquisas abrangeram diferentes escopos de projeto, de modo a conhecer o estado da arte e analisar as opções que melhores se encaixam dentro do escopo do presente projeto. A busca inicialmente restringiu-se ao âmbito da Universidade de Brasília, e posteriormente foi estendendo-se às esferas regionais, nacionais e internacionais.

2.1.1 Joystick

O joystick é uma das principais interfaces de comando utilizadas em equipamentos eletrônicos, visto que seu uso é simples e intuitivo. Este consiste de uma haste vertical fixada em uma base, que pode ser fletida comumente em dois eixos, e transmite o ângulo de inclinação em duas ou três dimensões. Existem diversos tipos de joystick, podendo variar também a quantidade de eixos de controle permitidos, e sua aplicação é vista em consoles de videogames, controle de máquinas pesadas como elevadores, guindastes, caminhões, braços robóticos, etc.

O uso do joystick também é muito comum em tecnologias assistivas, devido sua fácil adaptabilidade. Uma estratégia muito comum é adaptar o joystick conforme o grau de limitação do usuário, surgindo, por exemplo, os joystick de queixo, destinados à usuários com limitações de movimento nos membros superiores e inferiores; joystick de pé, destinados à usuários com limitações de movimento nos membros superiores; mini joystick, que é mais sensível ao toque e seu uso é recomendado para usuários com baixo tônus muscular; joystick adaptado com botões, entre outros. Estas soluções podem ser conferidas na Figura 2 a seguir.

Como no escopo deste projeto o público-alvo são indivíduos portadores de tetraplegia, a configuração mais adequada que atenderia aos requisitos e limitações destes seria o joystick de queixo. Como esta solução de joystick comumente apresenta controle em 2 graus de

Figura 2 – Joysticks adaptados. Fontes (ANDITEC, s.d.) e (OLIVEIRA, 2019)

liberdade, se faz necessário o uso de algum dispositivo auxiliar, como um botão, para tornar possível comandar mais graus de liberdade, como exigido pelo efetuador terminal do braço robótico estudado, que possui 3 graus de liberdade (movimentos de pitch, yaw e roll).

2.1.2 Sensor de Sopro/sucção

A interface de sopro e sucção é indicada para usuários com perda total do movimentos dos membros superiores e inferiores e que sofrem de espasmos, que impossibilita o uso do joystick. A interface é constituída de dutos (canudos) com sensores de fluxo de ar acoplados, capazes de detectar os movimentos de sopro e sucção do usuário e, a partir disso, associar os comandos a serem realizados pela máquina.

Essa interface é amplamente utilizada no controle de direção de cadeira de rodas, sendo comum o uso de um ou dois tubos. No projeto desenvolvido por (PINTO, 2016) foram utilizados dois tubos para controlar uma cadeira de rodas, um para enviar comandos para frente (sopro) e para trás (sucção), e outro para enviar comandos para esquerda (sopro) e para direita (sucção). A Figura 3 mostra o esquema construtivo do sensor desenvolvido, no qual cada tubo possui dois canais para ligação com o controlador, sendo que um canal é acionado pelo movimento de sopro, e o outro canal é acionado pelo movimento de sucção; resultando no total de 4 canais referentes às direções frente, trás, esquerda e direita.

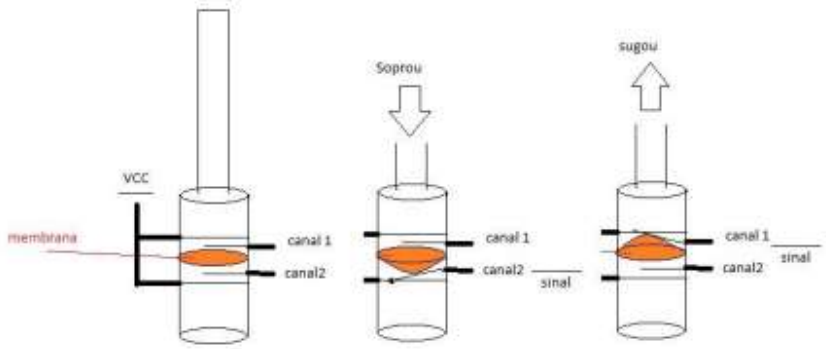


Figura 3 – Princípio de funcionamento do sensor de pressão. Fonte (PINTO, 2016)

Há também soluções que utilizam um único tubo de sopro e sucção associado a um

painel de controle de leds para determinar direção, sentido e velocidade do movimento da cadeira de rodas antes de ser efetuado, como pode ser conferido no trabalho desenvolvido em (FERREIRA, 2008). Nas figuras 4 e 5 são apresentados os esquemáticos dessa solução, onde pode-se conferir os tubos de sopro e sucção, uma chave alavanca liga/desliga para os sensores, e o painel de led de seleção de comandos, com 8 direções e 3 velocidades possíveis.

Figura 4 – Transdutor de ar. Fonte: (OLIVEIRA, 2019)

Figura 5 – Painel de LEDs. Fonte: (OLIVEIRA, 2019)

Esse tipo de interface é uma ótima alternativa para usuários com limitações severas controlarem o movimento de uma cadeira de rodas, porém apresenta algumas desvantagens, como ser possível acionar apenas um movimento por vez (não permite enviar comandos simultâneos), além de ser necessário um esforço pulmonar do usuário, podendo levá-lo rapidamente à fadiga. Como cada tubo é associado a um grau de liberdade, para a aplicação desta interface no projeto vigente seria necessário o uso de estratégias para aumentar o grau de liberdade, como o uso de mais tubos ou uso de um painel de LEDs indicativo de movimentos.

2.1.3 Eletrodos

As interfaces de eletromiografia, eletro-oculografia e eletroencefalografia consistem basicamente em monitorar as atividades elétricas de partes específicas do corpo humano através de eletrodos.

Na eletromiografia deseja-se obter a atividade elétrica das células musculares, e corresponde ao somatório algébrico de todos os sinais detectados sob a área de alcance dos eletrodos (BASMAJIAN; LUCA, 1985). Essa técnica é uma alternativa para pessoas com deficiência motora severa, visto que os eletrodos podem ser posicionados em músculos nos quais o paciente possui controle.

A eletro-oculografia, técnica ilustrada na Figura 6 consiste em obter atividades elétricas associadas a movimentação dos olhos, através de eletrodos posicionados em pontos

estratégicos, onde seja possível identificar os sinais nervosos deste local. Esta técnica, apesar de ser utilizada em um local sensível, é não-invasiva, e seu uso é indicado para pessoas com limitações motoras severas.

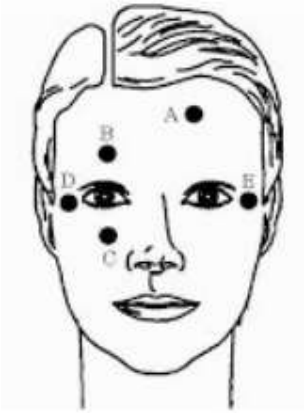


Figura 6 – Ilustração das regiões exploradas na eletro-oculogragia. Fonte (BAREA; LÓPEZ; MAZO, 2002)

Por fim, a eletroencefalografia consiste em obter atividades elétricas associadas ao cérebro, sendo um método não invasivo com eletrodos posicionados no couro cabeludo e, assim, é capaz medir as flutuações de tensão resultante da corrente iônica dentro dos neurônios do cérebro (JUNIOR, 1990). Este método é uma boa alternativa para pessoas com limitações de mobilidade, especialmente sem os movimentos dos membros superiores. O tempo de resposta também é muito rápido, comparado com outros sinais biomédicos. Essa técnica é ilustrada na Figura 7.

Figura 7 – Touca com eletrodos utilizado na técnica de eletroencefalografia. Fonte: (JULIÃO, 2021)

Tais técnicas apresentadas, apesar de serem ótimas opções para usuários portadores de limitações severas, apresentam alguns inconvenientes, como: os sinais resultantes possuem baixa ordem de grandeza, sendo necessário o uso de técnicas de filtragem e amplificação do sinal para manipulação dos dados, e isto faz com que a técnica seja muito suscetível

à interferências, ruídos e movimentos involuntários. Além disso, os algoritmos computacionais para processamento dos dados necessitam passar por treinamento para reconhecimento de padrões provenientes desses sinais *(machine learning)*, para posteriormente associá-los à movimentações desejadas, sendo necessário, portanto, recursos computacionais elevados. Outro fator a ser levado em consideração é que os sinais vitais obtidos destas interfaces variam de pessoa para pessoa, e necessitam de calibragem constante.

2.1.4 Movimentação da cabeça e uso de sensores

Interfaces baseadas pela movimentação da cabeça são associadas ao uso e interpretação de sensores que medem a sua inclinação, como acelerômetros localizados no topo da cabeça (Figura 8). No trabalho desenvolvido por (SILVA, 2013) esta interface é utilizada para controlar uma cadeira de rodas através da detecção do ângulo formado nos eixos X e Y em relação à posição de repouso.

Neste caso, é necessário realizar um refinamento dos sinais obtidos, criando uma zona de tolerância com tempo de resposta adequado para que a cadeira não se movimente perante movimentos involuntários do usuário, de forma que seja possível distinguir movimentos associados a comandos de movimentos naturais. Essa interface é uma boa alternativa para pessoas com limitações motoras severas dos membros superiores e inferiores e, além disso, os sensores são baratos e facilmente encontrados, sendo, portanto, um ótimo custo benefício.

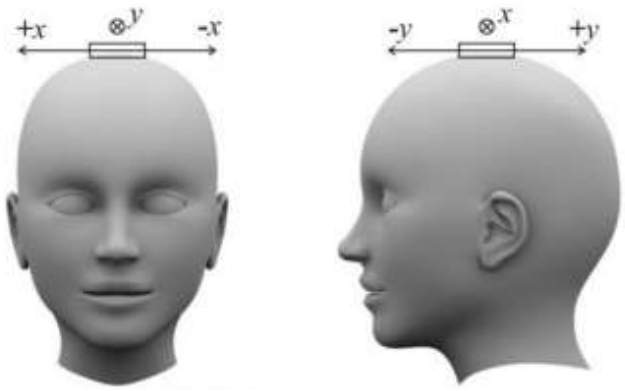


Figura 8 – Ilustração de sensor posicionado no topo da cabeça para detecção de movimento. Fonte: (SILVA, 2013)

2.1.5 Comando de voz

Interface por comando de voz é uma boa alternativa para pessoas com limitações motoras severas, e sua utilização é prática e intuitiva. Esta interface consiste em capturar os comandos informados pelo usuário através do microfone, processar os dados obtidos em um software de reconhecimento de voz, detectar e reconhecer os comandos (palavras), validar a instrução e, então, executar o comando.

Algumas das desvantagens deste método é a exigência de técnicas de filtragem de sinal e de softwares específicos para reconhecimento de voz, além de ser suscetível a ruídos sonoros do ambiente e depender da qualidade de vocalização do usuário.

2.1.6 Visão computacional

O campo da visão computacional é muito amplo, e sua aplicação como interface de comando em tecnologias assistivas apresenta inúmeras possibilidades e soluções. No contexto de sua aplicação para controle de direção de cadeira de rodas, é possível aplicar visão computacional baseado na detecção do movimento ocular, como desenvolvido pela Microsoft (BATISTOTI, 2018); em reconhecimento de expressões faciais associadas à comandos, como desenvolvido por (SOUSA, 2019), e ainda realizar um reconhecimento de ambientes.

No projeto desenvolvido pela Microsoft Brasil, exposto na Figura 9 uma tela contendo setas indicativas de direção é disponibilizada ao usuário, e uma câmera acoplada à cadeira de rodas faz o rastreio do movimento ocular, de maneira que funciona similarmente a um cursor mouse, associando o movimento às direções, que posteriormente são processados e transformados em comandos elétricos para acionar a cadeira. (BATISTOTI, 2018)

Figura 9 – Cadeira de rodas controlada pelo movimento dos olhos. (BATISTOTI, 2018)

O trabalho desenvolvido em (SOUSA, 2019) utiliza visão computacional para realizar reconhecimento de expressões faciais, no qual foram estabelecidas expressões simples de serem realizadas, e posteriormente foram associadas a cada um dos possíveis comandos para movimentar a cadeira. As expressões faciais e seus respectivos comandos foram determinados da seguinte forma: virar para a esquerda - inclinar a cabeça para esquerda e abrir a boca; virar para a direita - inclinar a cabeça para direita e abrir a boca; andar para trás - abrir a boca e piscar; e andar para frente - pressionar os lábios e piscar. Há ainda uma interface gráfica que apresenta ao usuário as direções selecionadas pelo seu comando (Figura 10).

(a) Frente (b) Trás

(c) Esquerda (d) Direita

Figura 10 – Cadeira de rodas controlada por reconhecimento de expressões faciais. (SOUSA, 2019)

Há ainda soluções mais complexas envolvendo o uso de processamento de imagem, visão computacional e técnicas de *machine learning* para comandar um braço robótico acoplado à cadeira de rodas, com o intuito de pegar objetos pré-determinados e treinados numa rede neural convolucional (CNN), como desenvolvido no artigo (ZHONG; ZHANG, Y.; YANG, 2019), em que utilizam um ponteiro laser para indicar o objeto requisitado pelo usuário, conforme ilustram as Figuras 11 e 12 . Porém, seu público alvo não são pessoas tetraplégicas, visto que o ponteiro laser é operado manualmente. Neste artigo (ZHONG; ZHANG, Y.; YANG, 2019) uma das técnicas utilizadas para determinação das coordenadas 3D do objeto é o uso de uma câmera RGB-D, que fornece informações sobre a profundidade (distância) do objetos para a câmera, o que auxilia na precisão dos cálculos de estimação de coordenadas.

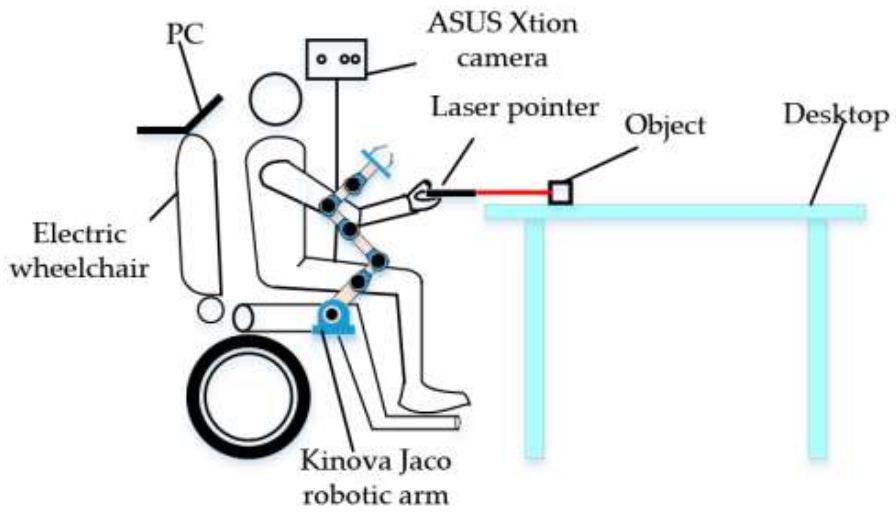


Figura 11 – Esquemático do projeto desenvolvido em (ZHONG; ZHANG, Y.; YANG, 2019)

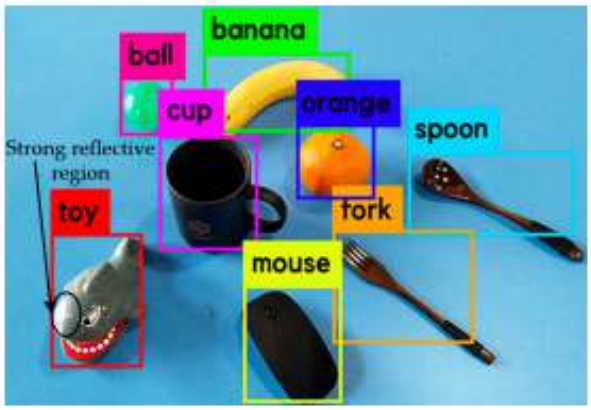


Figura 12 – Uso de *machine learning* na classificação de objetos em (ZHONG; ZHANG, Y.; YANG, 2019)

2.2 Estado da arte: Braços Robóticos Montados sobre Cadeiras de Rodas (WMRA)

Ao longo das últimas décadas, com o avanço da tecnologia em diversas vertentes, a aplicação da robótica em TA's tem recebido grande atenção, devido à crescente demanda oriunda do envelhecimento da população e da melhoria nas condições de vida de pessoas portadoras de deficiência ou limitações motoras. Uma das principais aplicações da robótica em TA é o braço robótico montado sobre cadeira de rodas, conhecido como WMRA (do inglês *Wheelchair-Mounted Robotic Arms*). A WMRA é um sistema mecatrônico que engloba diversos subsistemas, com o intuito de automatizar a cadeira de rodas comum para auxiliar usuários com limitações motoras dos membros superiores a realizar atividades da vida cotidiana de maneira independente, como se locomover, se alimentar, pegar objetos, escovar os dentes, etc.

Usualmente, as WRMAs possuem um conjunto de interfaces homem-máquina, com o objetivo de adaptar o sistema dependendo das necessidades do usuário, visto que as limitações de cada indivíduo podem ser diferentes, e portanto, uma interface pode se adequar melhor do que outra.

Esta seção busca avaliar o estado da arte das WMRAs, com foco nas soluções de interfaces de comando utilizadas.

2.2.1 Jaco

Jaco é um braço robótico independente, desenvolvido pela empresa Kinova Technology, para ser acoplado à cadeira de rodas motorizada. Possui 6 (seis) graus de liberdade, que permite realizar movimentos suaves e próximos à de um braço humano, uma garra de três dedos que permite pegar objetos de diferentes tamanhos e formatos, e possui um *payload* (capacidade de carga) de 1,6 kg. O sistema ainda possui uma aplicação software

onde é possível definir parâmetros e selecionar atividades pré-programadas, como *drinking mode*, abrir portas, usar chaves, microondas, entre outras. As Figuras 13 e 14 mostram o braço robótico e algumas de suas aplicações. [(ROBOTICS, 2022)

Este braço robótico apresenta as seguintes interfaces de comando: joystick de 3 graus de liberdade, controle por movimento da cabeça, sopro e sucção, joystick de queixo, e possibilita usar combinações dessas interfaces. O Jaco é um produto bem consolidado no mercado, seu preço é de aproximadamente R$160.000,00, sendo portanto uma solução de alto custo para a maioria dos usuários.

Figura 13 – Design do braço Jaco. (ROBOTICS, 2022)

Figura 14 – Exemplos de uso do braço robótico Jaco. (ROBOTICS, 2022)

2.2.2 KARES

KARES (KAIST Rehabilitation Engineering System) é uma WMRA com 6 (seis) graus de liberdade, desenvolvido por pesquisadores sul-coreanos, e que possui duas versões. Inicialmente foi desenvolvido para executar quatro atividades básicas: pegar um copo sobre a mesa, pegar um objeto do chão, aproximar o copo ao rosto do usuário, e ligar/desligar o interruptor de luz. O KARES I era dotado de interfaces de sistema de visão única, comando de voz, sensor de força-torque de 6 dimensões, e dispositivo de entrada 3D (SpaceBallTM). (SONG et al., 1998) A figura 15 apresenta um usuário sobre o KARES I.

Na segunda versão (KARES II) o sistema foi aprimorado para atender a diferentes níveis de deficiência motora e reduzir a vibração presente na base do KARES I que resultava em operações instáveis, prezando pela modularização do sistema para ser mais compacto ao usuário, dependendo de suas necessidades. As atividades alvo também foram expandidas para atender mais tarefas do cotidiano dos usuários, totalizando em 12, sendo algumas delas: servir refeição, abrir/fechar portas, fazer chá, jogar vídeo-game e barbear. Nessa

Figura 15 – Usuário utilizando KARES (SONG et al., 1998).

versão, o sistema apresenta as seguintes interfaces de comando: eletromiografia, detecção do movimento dos olhos, movimentação da cabeça, reconhecimento de movimentação dos ombros através da visão computacional, reconhecimento de expressões faciais através da visão computacional, além de contar com uma plataforma móvel separada que pode ser controlada remotamente. A Figura 16 apresenta um esquemático do sistema e interfaces presentes. (BIEN; CHUNG; CHANG, 2004)

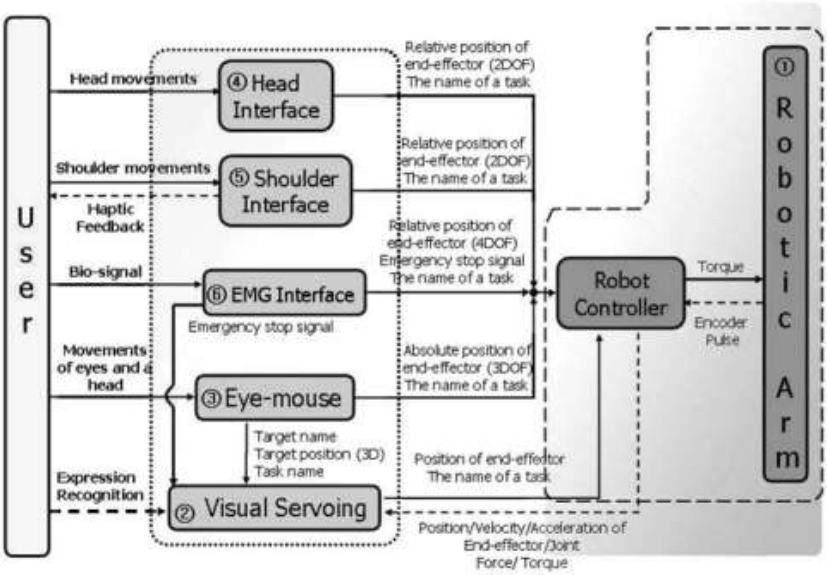


Figura 16 – Relações de entrada e saída para cada subsistema de interface. Fonte: (BIEN; CHUNG; CHANG, 2004)

2.2.3 Manus Arm Robot

O Manus Assistive Robotic Manipulator (ARM) é uma WMRA desenvolvido pela Exact Dynamics, empresa europeia do grupo Assistive Innovations, que possui 6+2 graus de liberdade, com encoders em cada junta, e uma garra de dois dedos. Esse manipulador apresenta diversas interfaces de comando, e o usuário pode controlá-lo manualmente acessando menus por meio de teclado, joystick ou botão switch, contendo diferentes modos presentes, como modo cartesiano, modo conjunto, etc. Além do controle manual, o Manus pode ser controlado por comando de voz e por visão computacional, sendo este último acrescentado recentemente. (TSUI; YANCO, 2007)

O sistema de visão computacional desenvolvido em (TSUI; YANCO, 2007) para o Manus é constituído por duas câmeras, uma na garra e outra no ombro do usuário, e para seleção/identificação do objeto a ser pegado pela garra, utiliza a abordagem de dividir o frame capturado pela câmera localizada no ombro em quatro quadrantes, e o usuário deve selecionar o quadrante cujo objeto desejado está contido. Em seguida, o quadrante escolhido é dividido novamente em quatro quadrantes, e esse processo é repetido até que o objeto ocupe consideravelmente o frame. (TSUI; YANCO, 2007) Esse processo é ilustrado na Figura 17.

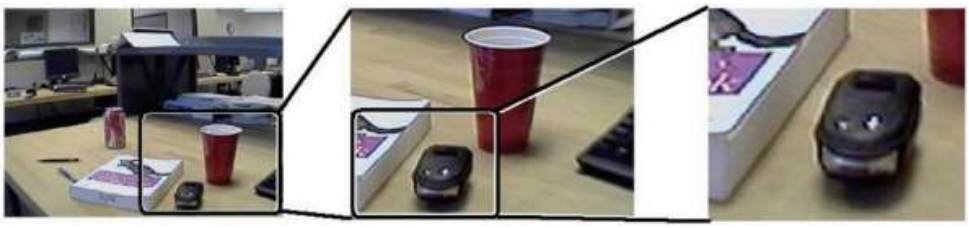


Figura 17 – Partição sequencial da vista em 4 quadrantes. Fonte: (TSUI; YANCO, 2007)

O braço robótico Manus então é movido no plano xy em direção às coordenadas do centro do quadrante final. A coordenada z (profundidade) foi fixada, pois ainda não havia sido desenvolvido um método para locomoção na terceira dimensão.

O Manus ARM então se move no plano xy em direção ao centro do quadrante selecionado emulando o movimento humano ao controle.

2.2.4 Weston Wheelchair

A Weston Wheelchair, ilustrada na Figura 18, é uma WMRA desenvolvida pela Bath Insitute of Medical Engineering que apresenta uma configuração "SCARA Modificado", ou seja, possui uma junta prismática, e a maioria das articulações (juntas rotativas) opera em um plano horizontal, enquanto que o movimento vertical é realizado por um único atuador vertical. Essa configuração apesar de ser um diferencial, apresenta algumas desvantagens, como por exemplo, o alcance frontal é ruim, a posição do braço robótico pode interferir no encosto reclinável da cadeira de roda, e não é possível pegar objetos no chão.

A interface de comando homem-máquina considerada nessa proposta foi uma entrada bidimensional, como um joystick, por exemplo, para acessar as funções ou rotinas pré-definidas. Portanto, podemos notar que esta solução não atende as necessidades de pessoas com limitações motoras dos membros superiores, pois utiliza-se de display e joystick manipulado pelas mãos. (HILLMAN et al., 2002)

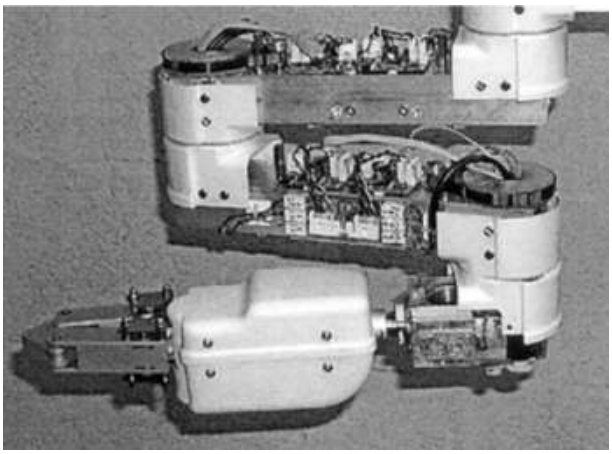


Figura 18 – Estrutura do braço robótico da Weston. Fonte: (HILLMAN et al., 2002)

2.2.5 FRIEND

O sistema robótico de reabilitação FRIEND-I e seu sucessor FRIEND-II foram desenvolvidos pelo Instituto de Automação da Universidade de Bremem, na Alemanha, em 1997 e 2003 respectivamente, com foco em atender usuários tetraplégicos ou portadores de deficiências motoras semelhantes que não conseguem controlar o manipulador por meio das interfaces de joystick ou teclado. Além disso, outro objetivo desse projeto era garantir autonomia e independência aos usuários por pelo menos 1,5h.

O FRIEND-I consiste de uma cadeira de rodas elétrica com um braço robótico Manus montado na parte traseira da cadeira, e sendo controlado por um PC. Um display LCD é usado para interação com o usuário. Para executar diferentes tarefas e de forma autônoma, o FRIEND-I é equipado com um sistema de câmera estéreo pan-tilt-zoom, posicionado na parte de trás da cadeira, e com uma bandeja inteligente, montada na parte frontal da cadeira. A principal interface de comando utilizada neste projeto foi o comando de voz, e para não tornar essa interação homem-máquina cansativa para o usuário, o sistema era pré-programado com as tarefas mais frequentes, como servir uma bebida ou colocar objetos sobre a bandeja. Além desta interface, o sistema também utiliza uma câmera acoplada ao efetuador terminal para pegar objetos arbitrários, onde o usuário move a garra para a vizinhança do objeto a ser agarrado, até que este possa ser visto pela câmera, e então um algoritmo servo visual interpreta continuamente as imagens e atribui marcadores naturais ou artificiais nos objetos. As Figuras 19 e 20 apresentam o sistema FRIEND-I. (MARTENS; PRENZEL; GRAESE, 2007)

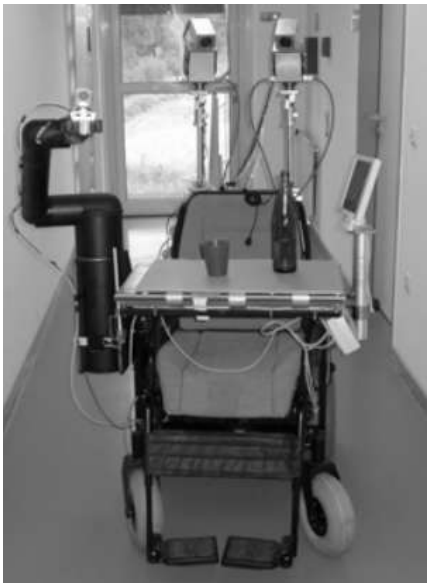


Figura 19 – Vista frontal do FRIEND-I. (MARTENS; PRENZEL; GRAESE, 2007)

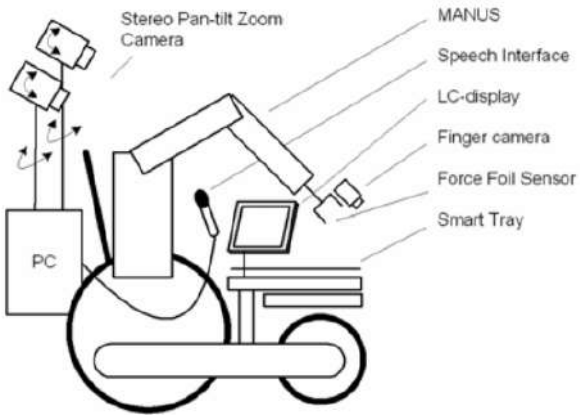


Figura 20 – Esquemático do FRIEND-I. (MARTENS; PRENZEL; GRAESE, 2007)

O projeto do FRIEND-II foi desenvolvido com o intuito de melhorar o projeto anterior, e uma das principais mudanças é no braço robótico. Desta vez foi utilizado um com 7 (sete) graus de liberdade desenvolvido pela Amtec Robotics, que apresenta uma estrutura cinemática semelhante à humana, constituído por uma série de juntas giratórias com eixos perpendiculares, e no seu punho é integrado um sensor multieixo de força/torque, modelo Gamma, da ATI-Industrial Automation. O braço do robô é equipado com um Otto Bock SensorHand5 como uma garra. O FRIEND-II, como seu antecessor, é equipado com uma bandeja inteligente, e uma câmera Sony EVI-D70P foi selecionada para o sistema de aquisição de imagem. Assim como o FRIEND-I, o FRIEND-II também foi pré-programado com as tarefas mais frequentes, e utiliza as mesmas interfaces na interação homem-máquina. Por ser um sistema mais robusto com ajustes mais finos, que atende a mais atividades comuns do cotidiano do usuário, há também maior complexidade associada.(MARTENS; PRENZEL; GRAESE, 2007) O projeto FRIEND-II pode ser visualizado na Figura 21 :

2.2.6 Raptor

O Raptor é uma WMRA fabricada pela Phybotics [Phybotics 2006], com o braço robótico de 4 graus de liberdade e uma garra com dois dedos para manipulação. O sistema se move por reconfiguração de junta, e não conta com encoders de junta, portanto o controle em coordenadas cartesianas não é possível, e não pode ser pré-programado, como braço robótico industrial. O manipulador é controlado por meio de joystick e um teclado.(BARROS DA SILVA NÉTO, 2019) A simplicidade desse modelo de WMRA, ilustrado na Figura 22, o torna mais acessível em comparação com os demais.

Figura 21 – Design WMRA do FRIEND-II. Fonte: (MARTENS; PRENZEL; GRAESE, 2007)

Figura 22 – Design WMRA do Raptor. (BARROS DA SILVA NÉTO, 2019)

2.3 Projetos de base

2.3.1 Manipulador robótico utilizado de base teórica

Este projeto tem como base de estudo o manipulador robótico acoplado à cadeira de rodas desenvolvido por (BARROS DA SILVA NÉTO, 2019) em seu Trabalho de Conclusão de Curso intitulado *Desenvolvimento de um manipulador robótico para aplicações assistivas*, pela Universidade de Brasília, que tinha como objetivos desenvolver o projeto mecânico e construção de um protótipo de baixo custo de manipulador para ser montado em uma cadeira de rodas, com a finalidade de auxiliar pessoas com deficiência motora dos membros superiores e inferiores.

Desta forma, o intuito do presente trabalho é contribuir através do desenvolvimento

de uma solução viável que possa ser utilizada como interface Homem x Máquina (IHM) para controlar o braço robótico projetado. Para isto, é necessário compreender os resultados alcançados por (BARROS DA SILVA NÉTO, 2019).

2.3.1.1 Especificações do Manipulador

O projeto do braço robótico possui uma configuração articulada, com 6 graus de liberdade, sendo 3 graus para posicionamento do braço no espaço, e 3 graus de orientação referentes ao punho para os movimentos de pitch (arfagem), yaw (guinada) e row (rolamento). O efetuador terminal é uma garra com 2 dedos. O projeto conceitual é ilustrado na Figura 23:

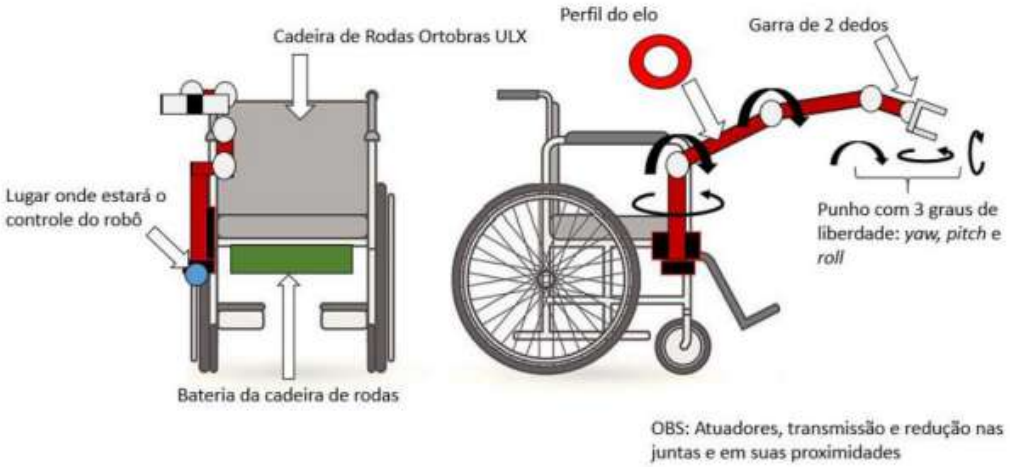


Figura 23 – Projeto conceitual. Fonte: (BARROS DA SILVA NÉTO, 2019)

Uma das premissas do projeto é o baixo custo, com o intuito de tornar a solução acessível. Com isto, os materiais escolhidos para a fabricação do braço robótico foi o PVC, plástico PLA/ABS, nylon e alumínio. O braço possui um payload de 1,2 kg (capacidade de carga). Os elos do manipulador, ilustrados na Figura 24 , possuem as dimensões indicadas na Tabela 1, resultando em um alcance de 1,1m.

Tabela 1 – Dimensões dos elos. Fonte: (BARROS DA SILVA NÉTO, 2019)

Elo	L_0	L_1	L_2	L_G
Comprimento	0,17 m	0,35 m	0,35 m	0,35 m

Foram determinados também os limites angulares de cada junta, indicados na Tabela 2, de maneira a se respeitar os limites construtivos e o alcance de 1,1m.

Tabela 2 – Dimensões dos elos. Fonte: (BARROS DA SILVA NÉTO, 2019)

Junta	θ_1 (base)	θ_2 (ombro)	θ_3 (cotovelo)	θ_4 (pitch)	θ_5 (yaw)	θ_6 (roll)
Limite angular	$\pm$ 90°	-45° a +135°	-135° a +160°	$\pm$ 160°	$\pm$ 90°	$\pm$ 360°

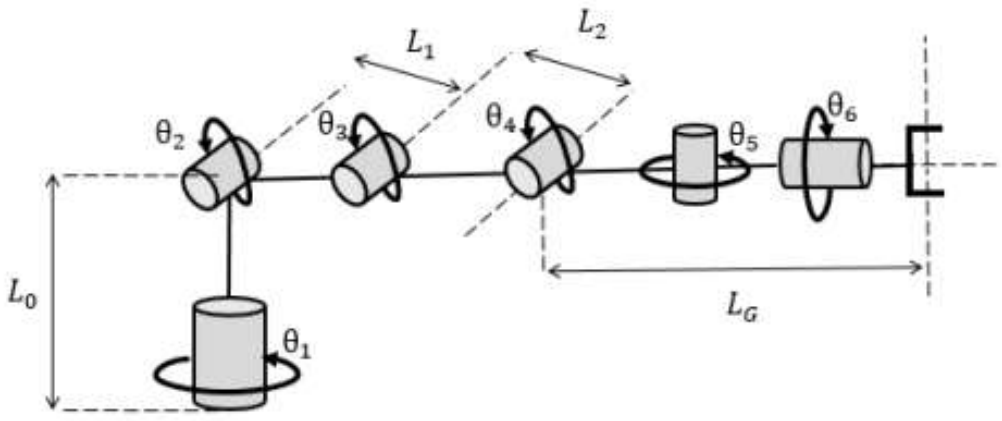


Figura 24 – Configuração do manipulador com seus elos indicados. Fonte: (BARROS DA SILVA NÉTO, 2019)

2.3.1.2 Modelagem cinemática

A cinemática de manipuladores se refere ao estudo e estabelecimento de relações da sua movimentação para definições de trajetórias a serem realizadas, e pode ser classificada como direta e inversa. A cinemática direta é o processo de obtenção das coordenadas do efetuador terminal a partir das variáveis de juntas. Já a cinemática indireta é o processo inverso, ou seja, de obtenção das variáveis de juntas a partir das coordenadas do efetuador terminal. Estes processos são bem ilustrados na Figura 25:

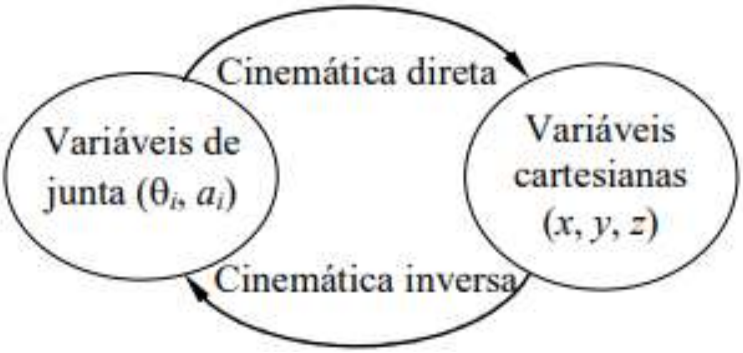


Figura 25 – Relação entre cinemática direta e inversa. Fonte: (CARRARA, 2015)

Existem diferentes métodos para realização da cinemática direta ou inversa, sendo categorizados principalmente entre métodos analíticos e métodos numéricos. Alguns exemplos de métodos são o geométrico, parâmetros de Denavit-Hartenberg (D-H) e análise matricial. No projeto desenvolvido por (BARROS DA SILVA NÉTO, 2019) foi empregado o método de D-H para análise da cinemática direta, que consiste em um método sistemático de descrição da posição e orientação entre dois elos consecutivos através dos seguintes parâmetros: ângulo de rotação da junta θ, o ângulo de torção da junta α, o comprimento do elo a e o deslocamento da junta d (CARRARA, 2015), e com base nestes parâmetros são montadas as matrizes de transformações homogêneas. Os parâmetros de D-H para o manipulador estudado estão explicitados na Tabela 3 e Figura 26 :

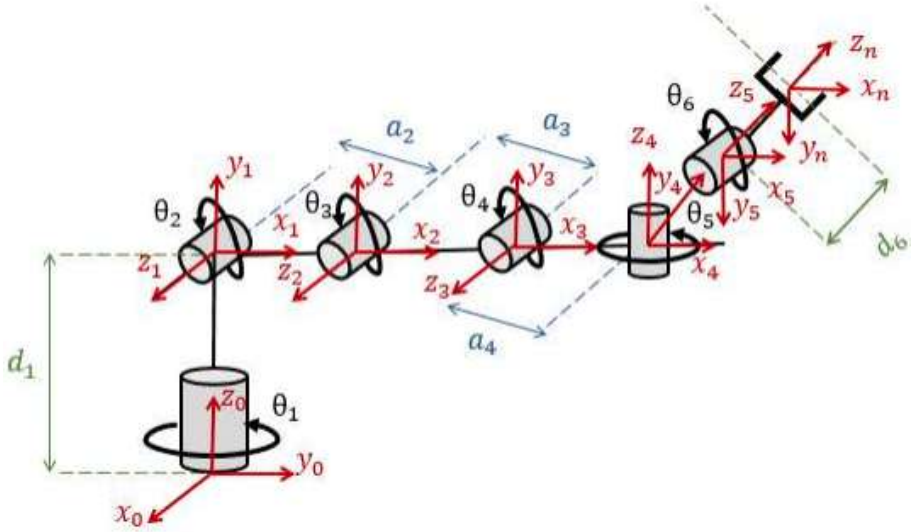


Figura 26 – Desenho esquemático do manipulador com os parâmetros D-H. Fonte: (BARROS DA SILVA NÉTO, 2019)

Tabela 3 – Parâmetros de D-H e seus valores. Fonte: (BARROS DA SILVA NÉTO, 2019)

Junta	θ_i*	d_i	a_i	α_i
1	θ_1*	$d_1 = 0{,}17m$	0	+90°
2	θ_2*	0	$a_2 = 0{,}35m$	0
3	θ_3*	0	$a_3 = 0{,}35m$	0
4	θ_4*	0	$a_4 = 0{,}15m$	-90°
5	θ_5*	0	0	-90°
6	θ_6*	$d_6 = 0{,}20m$	0	0

Em que o * significa que é uma variável que se altera dependendo da posição e orientação do robô.

2.3.2 Manipulador utilizado para testes

Como o projeto utilizado de referência teórica (BARROS DA SILVA NÉTO, 2019) no desenvolvimento do atual trabalho não estava disponível para testes físicos, foi utilizado o manipulador robótico *Pégaso* desenvolvido por (PASCHOALETTO, 2022) em seu Trabalho de Conclusão de Curso intitulado *Desenvolvimento de um Braço Robótico de 6 Graus de Liberdade para Fins Cinematográficos*, pela Universidade de Brasília, que tinha como objetivo desenvolver um manipulador acessível para fins cinematográficos na cultura *maker*. Desta forma, o robô possui um ajuste fino de deslocamento, contando com sistemas de redução planetária, três motores de passos, três servomotores e drivers TI DRV8825, que permitem dividir um passo cheio do motor em até 32 micropassos.

2.3.2.1 Especificações do Manipulador

O manipulador possui uma configuração articulada, com 6 graus de liberdade, sendo 3 graus de posicionamento e 3 graus de orientação do efetuador terminal. Porém, a configuração de suas juntas de orientação se diferem do manipulador teórico citado na subseção 2.3.1.1, como pode ser observado na Figura 27, pois a ordem de movimentos do punho do *Pégaso* é row, pitch e yaw, enquanto que no manipulador teórico (Figura 24) a ordem era pitch, yaw e row.

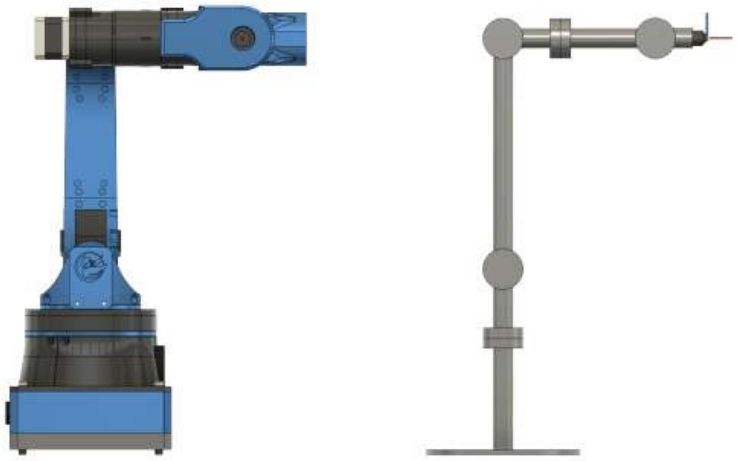


Figura 27 – Configuração do manipulador *Pégaso*. Fonte: (PASCHOALETTO, 2022)

Os elos do manipulador, ilustrados na Figura 28, possuem as dimensões indicadas na Tabela 4, resultando em um alcance de 42,83 cm.

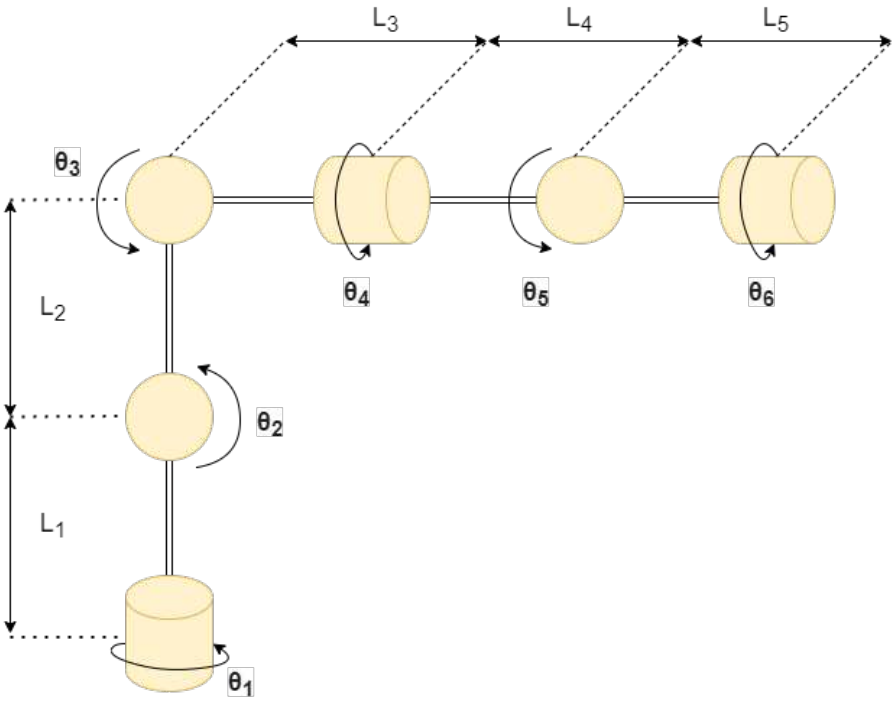


Figura 28 – Configuração do manipulador *Pégaso* e seus elos indicados.

Tabela 4 – Dimensões dos elos do robô *Pégaso*.

Elo	L_1	L_2	L_3	L_4	L_5
Comprimento	182,71 mm	226,7 mm	56.649 mm	93.835 mm	51.165 mm

O controle do manipulador *Pégaso* se dá através de um microcontrolador Raspberry Pi Pico e ele conta ainda com o módulo ESP-32 para conexão Wi-Fi. Sua interface com o usuário se dá através de uma requisição HTTP, aproveitando-se da conexão Wi-Fi disponível, e isto implica que o robô pode ser facilmente controlado de maneira remota através de uma página HTML implementada, que pode ser visualizada na Figura 29. Esta página conta com três modos de controle para o robô: modo referencial, modo de controle por eixo e modo de controle cartesiano.

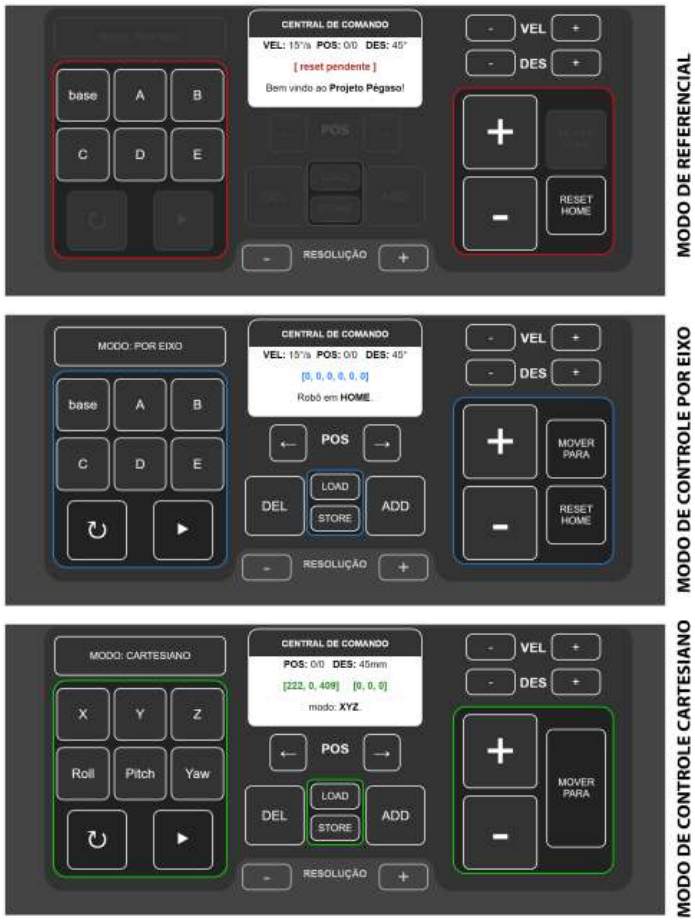


Figura 29 – Interface de comando do manipulador *Pégaso*. Fonte: (PASCHOALETTO, 2022)

2.3.2.2 Modelagem cinemática

A modelagem cinemática do manipulador *Pégaso* foi realizada de maneira diferente, se comparada com a modelagem do manipulador teórico abordada na subseção 2.3.1.2 . A cinemática inversa foi feita através de um método denominado *Triangulação*, que pode ser conferido no artigo (MÜLLER-CAJAR; MUKUNDAN, 2007), e trata-se de método cuja

"ideia é analisar a trajetória partindo da base da cadeia cinemática em direção ao efetuador, percorrendo cada junta intermediária e formando um triângulo que se conecta com o efetuador terminal e o alvo desejado"(PASCHOALETTO, 2022).

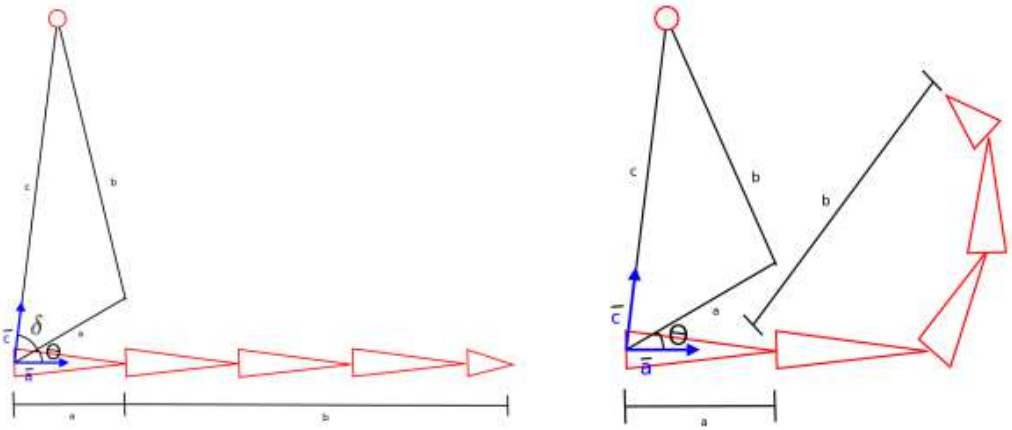


Figura 30 – Ilustração do método da triangulação para cinemática inversa. Fonte: (MÜLLER-CAJAR; MUKUNDAN, 2007)

O algoritmo funciona da seguinte forma: ao longo da cadeia cinemática, são formados triângulos entre a junta atual **a**, a junta seguinte **b** e o alvo **c**. Assim:

1. O lado **a** do triângulo é o comprimento do link entre a junta atual e a próxima;
2. o lado **b** é a distância em linha reta entre a próxima junta da cadeia e o efetuador terminal;
3. o lado **c** é a distância entre a junta atual e o alvo.

Como a distância **b** é sempre menor ou igual ao tamanho da cadeia cinemática restante, subentende-se que se o triângulo puder ser formado, então o alvo pode ser atingido. (PASCHOALETTO, 2022) Após a formação do triângulo, é aplicada a lei dos cossenos no Δ**abc** e determina-se o ângulo δ indicado na Figura 30. Em seguida, encontra-se também dois vetores:

1. o vetor $\vec{a}$ parte da junta atual e aponta para a próxima junta da cadeia; e
2. o vetor $\vec{c}$ parte da junta atual e aponta para o alvo que deseja-se atingir.

O produto escalar destes vetores resulta no ângulo que a junta atual se encontra em relação ao alvo. Subtraindo este ângulo de δ, calculado pela lei dos cossenos, resulta no ângulo θ correspondente ao valor que a junta atual deve girar em direção ao ponto desejado.

Em seguida, o próximo triângulo a ser analisado é formado por **a+b=c**, e θ se anula. "Na prática, isso indica que o cálculo do ângulo da próxima junta será o último a ser realizado, e pode-se iniciar o produto escalar entre dois novos vetores $\vec{a}$ e $\vec{c}$ formados.

O algoritmo da triangulação é um método iterativo que não exige que toda a cadeia cinemática seja percorrida, pois pode-se chegar antes ao alvo.

3 Fundamentação Teórica

Este capítulo abordará os alicerces teóricos necessários para o desenvolvimento do presente trabalho, expondo os principais conceitos e teorias envolvidas na elaboração do projeto conceitual.

3.1 Imagem digital

Uma imagem digital é uma função bidimensional $f(x,y)$, na qual x e y são coordenadas espaciais e o valor de f em qualquer ponto (x,y) é proporcional à intensidade luminosa (brilho ou nível de cinza) no ponto considerado. (QUEIROZ; GOMES, 2001) Como os computadores não são capazes de processar imagens contínuas, apenas arrays de números binários 0 ou 1, é necessário representar imagens como arranjos bidimensionais de pontos. Cada ponto na grade bidimensional (matriz) que representa a imagem digital é denominado elemento de imagem ou pixel, ilustrado na Figura 31 . (QUEIROZ; GOMES, 2001)

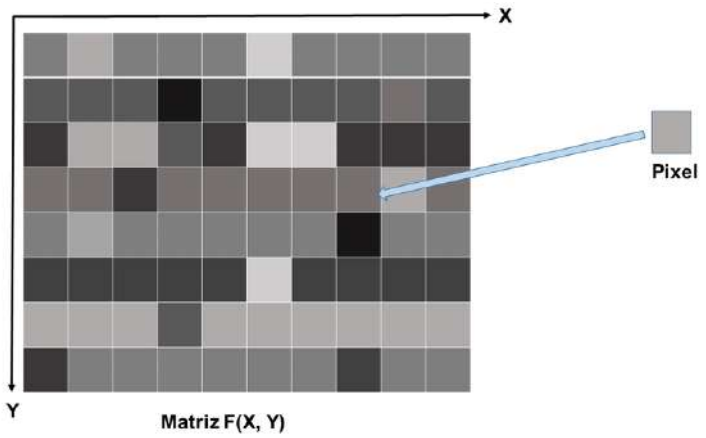


Figura 31 – Representação de uma imagem digital. Adaptado de (DIAS; BOM; ALBUQUERQUE, 2017)

A função $f(x,y)$ pode ser caracterizada por dois componentes: (1) a quantidade de iluminação da fonte que incide na cena que está sendo vista; e (2) a quantidade de iluminação refletida pelos objetos na cena. Esses elementos são chamados de componentes de iluminação e refletância e são expressos por $i(x,y)$ e $r(x,y)$, respectivamente.

Existem vários formatos de imagem, que podem também ser representadas em diferentes espaços de cores, por exemplo: imagens binárias, imagens de escala de cinza, imagens RGB, imagens HSV, imagens em ponto flutuante, etc. As imagens lógicas ou binárias é uma representação em matrizes de duas dimensões, onde pode ser atribuído valores de 0 ou 1 a cada pixel, correspondendo às cores preta e branca, respectivamente. As imagens

de escala de cinza são matrizes bidimensionais cujos valores de intensidade de cada pixel dependem da resolução de bits da imagem. As imagens em ponto flutuante, assim como as demais, também são representadas em matrizes, e utilizam o ponto flutuante para descrever a intensidade de cada pixel, ao invés de valores inteiros.

As imagens coloridas são comumente representadas no formato RGB. Em uma imagem digital colorida no sistema RGB, um pixel pode ser visto como um vetor cujas componentes representam as intensidades de vermelho, verde e azul de sua cor. A imagem colorida pode ser vista como a composição de três imagens monocromáticas, i.e.:

$$f(x,y) = f_R(x,y) + f_G(x,y) + f_B(x,y) \tag{3.1}$$

onde $f_R(x,y)$, $f_G(x,y)$ e $f_B(x,y)$ representam, respectivamente, as intensidades luminosas das componentes vermelha, verde e azul da imagem, no ponto (x,y).

3.2 Processamento de Imagem

Processamento de Imagem, como o próprio nome sugere, é a área responsável por realizar o processamento dos dados que compõem uma imagem, de forma a viabilizar sua aplicação em diversas finalidades. O processo se inicia com a captura de uma imagem, a qual corresponde à captura da iluminação que é refletida sobre a superfície dos objetos. Esta é realizada através de um sistema de aquisição em hardware (câmera e sensores). Após a captura, a imagem passa por um processo de digitalização, sendo representada de forma apropriada para o tratamento computacional. Imagens podem ser representadas em duas ou mais dimensões. (QUEIROZ; GOMES, 2001)

O primeiro passo do processamento é conhecido como pré-processamento, cujos passos envolvem o processo de filtragem de ruídos introduzidos pelos sensores, e correção de distorções geométricas causadas pelas lentes das câmeras. Outros processos podem ser implementados para análise e identificação de objetos, como por exemplo, identificação de bordas e contornos, texturas e vizinhanças com demais objetos da imagem.

É possível aplicar ainda processamentos mais complexos, como separação do plano de fundo através da técnica de segmentação que identifica características constantes e descontinuidades, extração de detalhes específicos através da técnica de *threshold*, aplicação de operadores morfológicos capazes de modificar formas geométricas identificadas, e extração de informações sobre particularidades de objetos que podem ser utilizadas como entrada para algoritmos de classificação. (QUEIROZ; GOMES, 2001)

A classificação é uma das tarefas de mais alto nível e que normalmente envolve conceitos de outras áreas da computação, como *machine learning*, e tem como objetivo reconhecer ou inferir a identidade dos objetos a partir das características colhidas no processamento da

imagem.

A área de Processamento de Imagens incorpora fundamentos de várias ciências, como Física, Computação e Matemática. Conceitos como Óptica, Álgebra Vetorial, Estatística, dentre outros conhecimentos, são comumente empregados em técnicas de processamento de imagens. Existe também uma intersecção forte entre Processamento de Imagem e outras áreas como Redes Neurais, Inteligência Artificial, Percepção Visual, Ciência Cognitiva, dando origem a outra área igualmente relevante, denominada Visão Computacional. (QUEIROZ; GOMES, 2001)

3.3 Visão computacional

A visão computacional é a área que explora formas de transmitir às máquinas a capacidade de interpretar dados visuais, ou seja, "enxergar". A visão computacional não se resume apenas à captura de imagens, mas também envolve o processo de extração e associação de informações entre imagens, para que seja possível realizar interpretações, reconhecimentos (de objetos, ações ou pessoas), movimentos e reconstrução de cenas. A visão computacional tem como objetivo reproduzir de maneira similar o processo biológico da visão humana, onde somos capazes de enxergar, realizar associações e interpretações de acordo com nossas experiências de vida.

Semelhante ao processamento de imagens, a Visão Computacional é um ramo que incorpora fundamentos de diversas áreas científicas, como Óptica, Física do Estado Sólido, Teoria dos Grafos, Álgebra Linear, Álgebra Matricial, Estatística, Processamento de Imagens, Trigonometria, Percepção Visual, Ciência Cognitiva, entre outras, sendo uma área relativamente nova e em grande expansão.

As principais aplicações da visão computacional são na área de segurança de ambientes, reconhecimentos de padrões em imagens médicas, monitoramento de áreas de preservação ambiental, reconhecimento facial, visão de robôs, entre outros. Atualmente as pesquisas debruçam-se na otimização e desempenho de algoritmos. (BACKES; MESQUITA SÁ JUNIOR, 2016)

3.3.1 Métodos para reconstrução de coordenadas 3D

Como mencionado anteriormente, a imagem digital é uma função bidimensional f(x,y), na qual x e y são coordenadas espaciais e o valor de f em qualquer ponto (x,y) é proporcional à intensidade luminosa (brilho ou nível de cinza) no ponto considerado. Desta forma, uma imagem digital é uma representação em 2D de uma cena real, como ilustra a Figura 32.

É possível relacionar o sistema de coordenadas de uma câmera com o sistema de

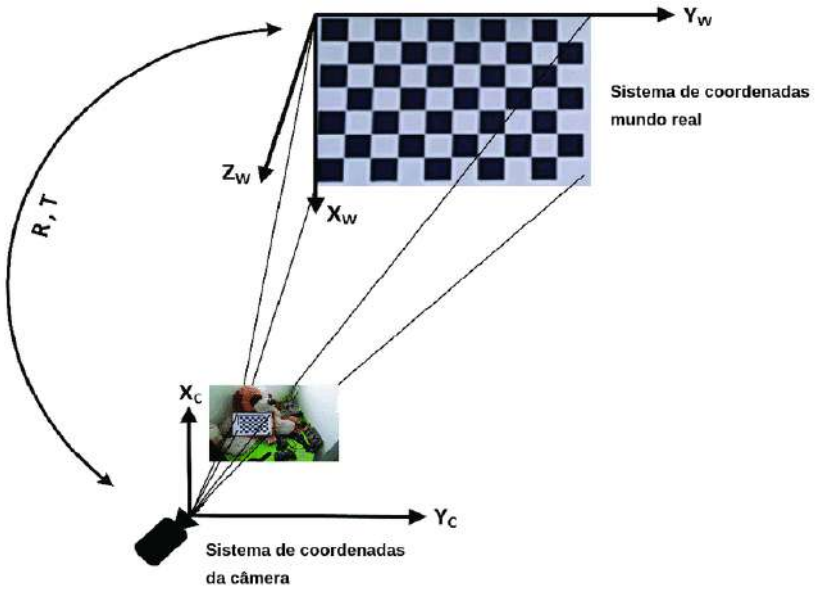


Figura 32 – Relação entre coordenadas da câmera e coordenadas do espaço. Adaptado de (HOSEINI; KABIRI, 2018)

coordenadas do mundo real, definindo o referencial de cada sistema (origem) e as matrizes de translação e rotação associadas. Porém, esta não é uma tarefa trivial, visto que um ponto (x,y,z) do sistema de coordenadas 3D do mundo real é convertido em um ponto f(x,y) 2D de acordo com sistema de coordenadas da câmera, ou seja, a informação da terceira componente 3D é perdida.

A relação entre as coordenadas X e Y do sistema de coordenadas do mundo real e da câmera podem ser estabelecidas a partir de um referência e dos parâmetros de calibração da câmera, que podem ser divididos em parâmetros intrínsecos e extrínsecos. Este assunto é abordado mais adiante neste capítulo.

Como a coordenada Z não consta no sistema de coordenadas da câmera, para obter esta informação a partir da imagem capturada é necessário o uso de alguma técnica. Algumas destas técnicas são: visão estéreo, triangulação, geometria epipolar, fator de escala, iluminação ativa, casamento de vistas, modelagem por pontos, profundidade por variação de foco, sombreamento, entre outras.

Iremos nos referir à coordenada Z como sendo a distância ou profundidade entre a câmera e o objeto de captura no mundo real, e esta subseção apresenta alguns conceitos e técnicas envolvidas no processo de obtenção desta distância.

3.3.1.1 Visão Estéreo

Visão estéreo é a área da visão computacional que estuda a reconstrução da informação tridimensional de objetos, a partir de um par de imagens capturadas simultaneamente a partir de duas ou mais câmeras deslocadas horizontalmente entre si, processo conhecido

também por triangulação de imagens. O primeiro passo da visão estéreo é extrair características das imagens observadas por cada câmera e então identificar características semelhantes observadas na cena. Após isso, utiliza-se dessas informações e aplica-se geometria no sistema estéreo montado entre as duas câmeras e a cena, para recuperar a informação da coordenada Z de profundidade perdida, ou seja, a reconstrução da imagem 3D. (MORAIS MADALENA; MENOTTI, s.d.)

A visão estéreo é o mesmo mecanismo utilizado pela visão humana para obter informações de profundidade. Assim, precisamos do uso dos olhos (sensores) para formar uma imagem tridimensional e formular estimativas de profundidade. O olho esquerdo e o olho direito sempre vêem imagens diferentes, apesar de muito parecidas. A partir dessas diferenças, a noção de profundidade é construída pelo cérebro. (MORAIS MADALENA; MENOTTI, s.d.)

Paralelamente, a partir de duas câmeras (sensores) posicionadas de forma correta, com suas posições e parâmetros de calibração conhecidos, é possível determinar as relações trigonométricas que resultam no encontro da posição 3D de qualquer ponto neste espaço, desde que este ponto possa ser localizado em ambas as imagens capturadas pelas câmaras, ou seja, na região comum entre elas. A determinação da posição de um ponto dentro deste espaço pode ser obtida triangulação, como ilustrado na Figura 33

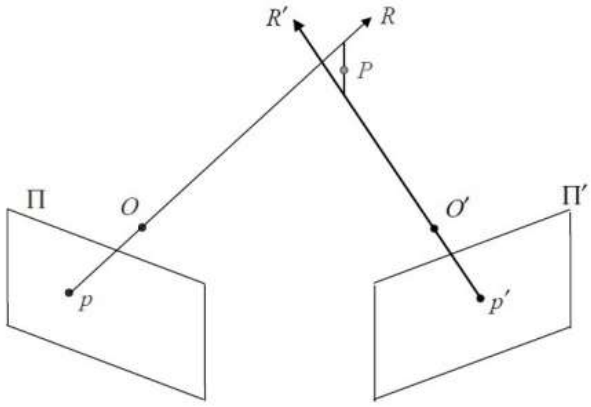


Figura 33 – Triangulação. Fonte: (MORAIS MADALENA; MENOTTI, s.d.)

3.3.1.2 Calibração da câmera

Como ilustrado na Figura 32, os sistemas de coordenadas da câmera e do mundo real são relacionados por uma translação, expressa pela vetor **T**, e uma rotação, representada pela matriz **R**. O vetor de translação **T** descreve uma mudança na posição dos centros de coordenadas **Oc** (origem do sistema de coordenadas da câmera) e **Ow** (origem do sistema de coordenadas do mundo). A rotação, por sua vez, altera os eixos de orientação correspondentes de cada sistema. Esta mudança é descrita por uma matriz ortogonal **R** de dimensões 3×3. (CYGANEK; SIEBERT, 2009) As matrizes **R** e **T** são especificadas nas Equações 3.2 e 3.3.

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3.2}$$

$$\mathbf{T} = \mathbf{O_w} - \mathbf{O_c} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{3.3}$$

Para determinar as relações e parâmetros entre os sistemas de coordenadas real e da câmera, é necessário realizar a calibração desta, processo pelo meio do qual é possível determinar as características geométricas e ópticas da câmera (parâmetros intrínsecos) e a posição e orientação da câmera em relação ao ambiente externo (parâmetros extrínsecos). Os parâmetros intrínsecos são estabelecidos pela distância focal, deslocamento do ponto principal, coeficiente de inclinação e coeficientes de distorções da imagem (radial e tangencial). Os parâmetros extrínsecos fornecem a posição da origem do sistema de coordenadas da câmera em relação à origem das coordenadas do mundo, estabelecidos pela matriz de rotação 3x3 e pelo vetor de translação 3x1. (ZHANG, Z., s.d.)

Definindo um ponto 2D na imagem como $m = [u, v]^T$ e um ponto 3D no mundo real como $M = [X, Y, Z]^T$. Representando o vetor com a inclusão da coordenada 1 como último elemento do vetor como $\tilde{x}$, sendo assim $\tilde{m} = [u, v, 1]^T$ e $\tilde{M} = [X, Y, Z, 1]^T$.[33] Seguindo uma modelagem *pinhole* para a câmera, a relação entre um ponto 3D e a sua projeção no plano da imagem é dada pela Equação 3.4:

$$s\tilde{m} = \mathbf{A}[\mathbf{R},\mathbf{T}]\tilde{M} \tag{3.4}$$

onde s é um fator escalar arbitrário, **[R, T]** é a matriz de parâmetros extrínsecos que descreve a translação e rotação que relaciona o sistema de coordenadas mundial ao sistema de coordenadas da câmera, e **A** é a matriz de parâmetros intrínsecos da câmera, representada na Equação 3.5:

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_o \\ 0 & \beta & v_o \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

em que (u_0, v_0) são as coordenadas do ponto principal(centros ópticos), α e β os fatores de escala nos eixos u e v da imagem (distâncias focais) e γ o parâmetro que descreve a assimetria dos dois eixos da imagem. (ZHANG, Z., s.d.)

A determinação da matriz de calibração é feita através da transformação projetiva entre os planos, denominada Homografia. A homografia é representada por uma matriz de transformação linear de dimensões 3×3, como definida na Equação 3.6 (ROCHA, s.d.)

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.6}$$

Sem perda de generalidade, pode-se assumir que o plano do modelo está em Z=0 do sistemas de coordenadas global **Ow**. Denotando a i-ésima coluna da matriz de rotação **R** por $\mathbf{r_i}$ (ZHANG, Z., s.d.). Da Equação 3.4, temos:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{r_3} & \mathbf{T} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{3.7}$$

$$= \boldsymbol{A} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{T} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.8}$$

Assim, considerando $Z = 0$, o ponto M será descrito como $M = [X,Y]^T$, e $\tilde{M}$ como $\tilde{M} = [X,Y,1]^T$. Portanto, para um ponto modelo M, sua representação na imagem m é relacionada por uma homografia **H**:

$$s\tilde{m} = \mathbf{H}\tilde{M} \tag{3.9}$$

sendo

$$\mathbf{H} = \mathbf{A} \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \tag{3.10}$$

sendo a matriz H definida por um fator de escala.

3.3.1.3 Geometria epipolar

A geometria epipolar é a geometria projetiva entre duas vistas, sendo independente da estrutura da cena e depende apenas dos parâmetros internos das câmeras e de sua posição relativa (HARTLEY; ZISSERMAN, 2004). Ela surgiu da necessidade de representar a relação entre imagens de diferentes pontos de vista sobre um mesmo conjunto de pontos 3D. (FARIAS, s.d.)

A partir de dois pontos de vistas diferentes de um ponto 3D, associados a duas câmeras C1 e C2, é possível definir a geometria epipolar. Como pode ser visto na Figura 34, os ponto x, x' e X, assim como os centros das câmeras $C1$ e $C2$ são coplanares. Daí surge a definição do plano epipolar π.

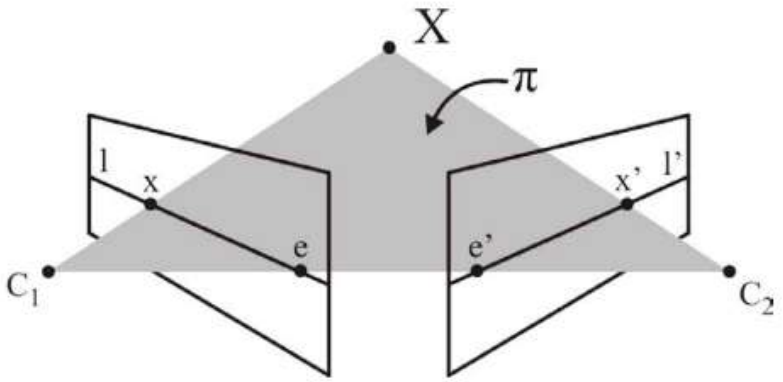


Figura 34 – Plano epipolar. Fonte: (FARIAS, s.d.)

Da intersecção do plano π com os planos de projeção das câmeras C1 e C2 surgem as linhas l e l' , denominadas linhas epipolares (HARTLEY; ZISSERMAN, 2004). Alguns dos conceitos importantes da geometria epipolar são:

- Epipolo: é o ponto de intersecção da linha que une os centros da câmera (a linha de base) com o plano da imagem. De forma equivalente, o epipolo é a imagem em uma visão do centro da câmera, na visão respectivamente oposta. É também o ponto de fuga da direção da linha de base (translação). Na Figura 34 os epipolos são os pontos e e e'.
- Um plano epipolar é um plano formado pela interseção dos centros das câmeras $C1$ e $C2$ e o ponto X, e que contém a linha de base. Existe um parâmetro família de planos epipolares.
- Uma linha epipolar é a intersecção de um plano epipolar com o plano da imagem. Todas as linhas epipolares se cruzam no epipolo. Um plano epipolar cruza os planos de imagem esquerdo e direito em linhas epipolares e define a correspondência entre as linhas. Na Figura 34 as linhas são denotadas por l e l'.
- Sobre as linhas epipolares, ainda se encontram as projeções 2D do ponto X nas câmeras C1 e C2, definidos por x e x'.

Assim, é possível estabelecer relação algébrica entre um ponto 2D qualquer e a linha epipolar gerada na imagem oposta, sendo denominada de Matriz Fundamental. Dado um ponto x', a linha epipolar l' passando por x' e o epipolo e' pode ser escrito como: (HARTLEY; ZISSERMAN, 2004)

$$l' = e' \times x' = [e'] \times x' \tag{3.11}$$

Reescrevendo $x' = H_\pi x$, tem-se então:

$$l' = [e'] \times H_\pi x = Fx \tag{3.12}$$

onde definimos $F = [e'] \times H_\pi$ como a matriz Fundamental. A matriz H_π é o mapeamento de transferência de uma imagem para outra por meio de qualquer plano epipolar. Além disso, a matriz fundamental satisfaz a seguinte relação: (HARTLEY; ZISSERMAN, 2004)

$$x'^T F x = 0 \tag{3.13}$$

Além da matriz fundamental, outro conceito importante é de matriz essencial, sendo esta um caso específico da matriz fundamental para coordenadas de imagem normalizadas. Desta forma, a matriz fundamental é uma generalização da matriz essencial, considerando a suposição de que as calibrações das câmeras não são conhecidas. Assim, a matriz essencial tem menos graus de liberdade e mais propriedades, se comparada com a matriz fundamental (HARTLEY; ZISSERMAN, 2004). Por definição, a matriz essencial é descrita como:

$$\tilde{x}'^T E \tilde{x} = 0 \tag{3.14}$$

em termos de coordenadas de imagem normalizadas para pontos correspondentes. Considerando uma matriz de calibração conhecida K, a relação entre a matriz fundamental e a matriz essencial pode ser descrita como:

$$E = K'^T F K \tag{3.15}$$

3.3.1.4 Retificação

Retificação é o processo de correspondência entre duas imagens estéreo através da transformação de ambas, de maneira que seus pares de linhas epipolares equivalentes tornam-se paralelos e colineares. Assim, a retificação alinhas duas imagens estéreo de modo que as informações de suas respectivas linhas sejam correspondentes. Quando um par estéreo é retificado, o sistema estéreo assume a configuração canônica, onde o problema de correspondência entre as imagens fica trivial, ajustando-se a uma análise unidimensional.

3.3.1.5 Cálculo de profundidade por trigonometria

De forma simplória, a distância Z pode pode ser determinada isoladamente através de trigonometria simples, conhecendo-se a distância entre as duas câmeras e os ângulos de inclinação formado pelo objeto sobre elas, como se segue na ilustração da Figura 35:

Por trigonometria, aplica-se a Lei dos Senos nos parâmetros conhecidos para estabeles as relações das Equações 3.16 e 3.17:

$$\gamma = 180° - \alpha - \beta \tag{3.16}$$

$$Z = \frac{b sin\alpha sin\beta}{sin\gamma} \tag{3.17}$$

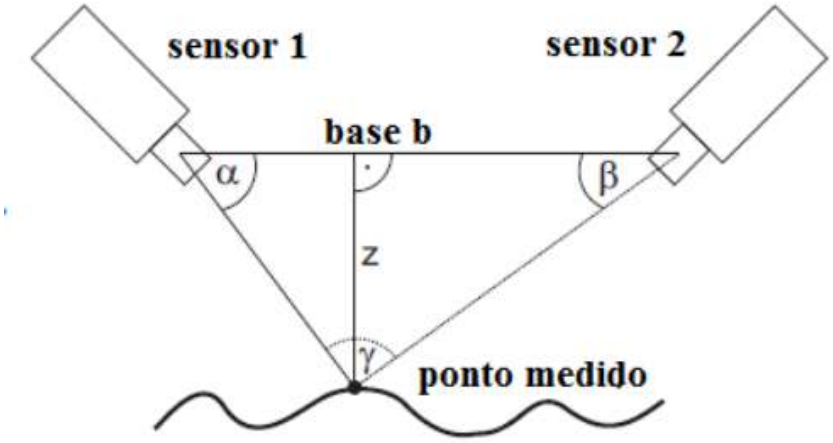


Figura 35 – Esquemático 1 de determinação de profundidade de um objeto na imagem. Fonte: (PINHO, s.d.)

Outro método possível para se obter a profundidade é descrito em (JESUS et al., s.d.) e ilustrado na Figura 36. O ponteiro laser é posicionado próximo à câmera, a uma distância h conhecida. O laser é projetado em um objeto a uma distância D, e este ponto vermelho é refletido e projetado no plano de imagem da câmera. A distância pfc (pixels do centro) entre o centro do plano da imagem (no eixo óptico) e o ponto vermelho no plano da imagem é proporcional à distância D. Assim, é possível calcular D pela Equação 3.18:

$$D = \frac{h}{tan(\theta)} \tag{3.18}$$

Essa expressão relaciona cada pixel da imagem em seu valor correspondente em centímetros. Pode-se expressar de acordo com os parâmetros da câmera e relação de pixels:

$$\theta = pfc \times rpp + ro \tag{3.19}$$

onde rpp é a expressão que relaciona a quantidade angular por pixel, chamada de radianos por pixel, e ro é um parâmetro de correção angular devido à deformação das lentes da câmera, chamado deslocamento radiano.

3.3.1.6 Cálculo de profundidade conhecendo informações sobre o objeto alvo

É possível também calcular a distância de um objeto pré-selecionado, mas de maneira menos precisa, conhecendo uma de suas dimensões (altura H ou largura W). Primeiramente é necessário realizar uma captura à uma distância D conhecida, e medir sua dimensão correspondente (altura ou largura) em pixels (P) na foto. Assim, pode-se calcular a distância focal da câmera F com a Equação 3.20:

$$\begin{aligned} F &= \frac{P \cdot D}{H} \quad \text{para altura;} \\ F &= \frac{P \cdot D}{W} \quad \text{para largura;} \end{aligned} \tag{3.20}$$

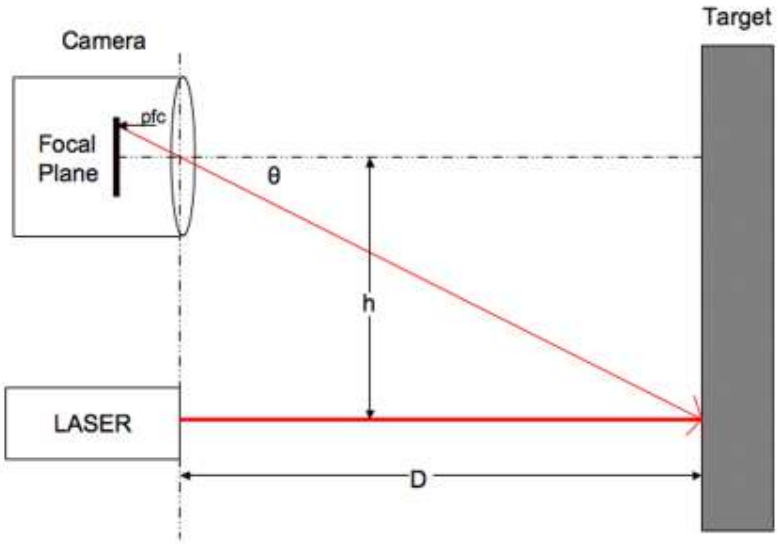


Figura 36 – Esquemático 2 de determinação da profundidade de um objeto na imagem. Fonte: (JESUS et al., s.d.)

Assim, sabendo a distância focal, pode-se determinar a distância do objeto para a câmera (Equação 3.21), usando do princípio de que quanto mais próximo da câmera, maior o objeto parecerá, e quanto mais distante, menor parecerá.

$$\begin{aligned} D &= \frac{F \cdot H}{P} \quad \text{para altura;} \\ D &= \frac{F \cdot W}{P} \quad \text{para largura;} \end{aligned} \tag{3.21}$$

3.3.2 Thresholding

O thresholding, em português limiarização, é uma técnica de processamento de imagem aplicada para segmentação de imagem que se baseia na diferença dos níveis de intensidade que compõem os diferentes objetos da imagem. Assim, de acordo com o limiar estabelecido baseado nas características dos objetos que se deseja isolar, a imagem é segmentada em dois grupos: o grupo de pixels com níveis abaixo do limiar e o grupo de pixels com níveis acima do limiar. Portanto, dada uma imagem $F(x,y)$ em tons de cinza, sua limiarização é dada por:

$$F(x_i,y_i) = \begin{cases} 255, & \text{se } F(x_i,y_i) \geq T \\ 0, & \text{se } F(x_i,y_i) < T \end{cases} \tag{3.22}$$

sendo $F(xi,yi)$ cada pixel da imagem. Logo, como resultado é obtido uma imagem binária, na qual os pixels com valores de intensidade maior ou igual ao critério T passam a apresentar um valor máximo de branco, e os demais pixels abaixo deste limiar passam a apresentar a cor preta. Desta forma a região ou objeto que se deseja isolar é destacado.Uma das dificuldades do processo reside na determinação do valor mais adequado de limiarização. (QUEIROZ; GOMES, 2001)

3.3.3 Detecção do ponto laser

A detecção de um ponto laser em uma imagem pode ser realizada através de conhecimentos de processamento de imagem e visão computacional. É necessário inicialmente conhecer as características de um ponto laser para que se adote as técnicas adequadas para detectá-lo, como por exemplo, conhecer a cor emitida (geralmente vermelha ou verde) e perceber que se trata de um "objeto"que emite brilho e exposição intensos. Desta forma, para detectar o ponto laser é preciso realizar uma busca na matriz de pixel que representa a imagem e identificar valores assumidos pelos seus canais de intensidade de cor, saturação e exposição.

Os principais problemas presentes na detecção do ponto laser pela câmera é a qualidade da imagem que pode variar dependendo das condições de luz do ambiente, a potência do laser, a qualidade da câmera (sensor de imagem, lentes, resolução, e FPS da câmera), e o chip da câmera, pois ele não é tão sensível quanto um olho humano. Por exemplo, um ponto de laser é visto pelo olho humano como um ponto vermelho, enquanto a câmera o vê como branco por conta da alta intensidade de luz emitida. Este problema pode ser parcialmente reduzido ajustando as configurações de exposição da câmera. (MATEJ MEŠKO, 2013)

A principal técnica utilizada para detecção de ponto laser é analisar a imagem no espaço de cores HSV (do inglês *Hue-Saturation-Value*) que correspondem aos parâmetros de matiz (tonalidade), saturação e intensidade de brilho respectivamente, e aplicar filtro passa-baixa para compensar os pixels dissonantes. A Figura 37 mostra a análise do ponto laser realizada no espaço de cores HSV no artigo (ZHONG; ZHANG, Y.; YANG, 2019).

Figura 37 – Análise de valores assumidos pelo ponto laser no espaço de cores HSV. Fonte:(ZHONG; ZHANG, Y.; YANG, 2019)

4 Metodologia

Neste capítulo são discutidas questões relevantes que direcionam o andamento do projeto, com base nas análises do que foi apresentado na revisão bibliográfica e fundamentação teórica, de forma que o objetivo seja propor uma solução à problemática exposta na introdução a fim de atingir os objetivos especificados, e gerar os requisitos para o projeto da interface de comando.

4.1 Análise das interfaces

Com base na revisão bibliográfica realizada no Capítulo 2 acerca das interfaces de comando comumente utilizadas em tecnologias assistivas, foi montada a Tabela 5 com os aspectos mais relevantes de cada interface analisada, enfatizando suas vantagens, desvantagens, usabilidade e conforto ao usuário alvo.

Tabela 5 – Comparativo entre interfaces - Matriz de decisão.

Interface	Vantagens	Desvantagens
Joystick comum usado em cadeira de rodas	Usabilidade intuitiva para o usuário; é possível associar controle de velocidade de acordo com o grau de inclinação da haste principal; relativamente barato.	Este modelo apresenta o inconveniente de ser operado pelas mãos, portanto não atende as necessidades do público-alvo do projeto (pessoas portadoras de tetraplegia). Além disso, comumente possui apenas 2 graus de liberdade, sendo necessárias outras estratégias para poder controlar mais graus, como uso de chaves comutadoras ou botões.
		Continua...

Tabela 5

Interface	Vantagens	Desvantagens
Joystick para robô industrial	Possui usabilidade intuitiva ao usuário; apresenta 3 graus de liberdade, excelente para manuseio de manipuladores robóticos; é possível associar controle de velocidade de acordo com o grau de inclinação da haste principal.	Este modelo apresenta o inconveniente de ser operado pelas mãos por conter um terceiro eixo giratório na ponta de sua haste, portanto não atende as necessidades do público-alvo do projeto (pessoa portadora de tetraplegia).
Joystick de queixo	Possui usabilidade intuitiva, boa alternativa para pessoas com limitações de movimento em membros superiores e inferiores; baixo custo.	Comumente apresentam apenas 2 eixos, sendo necessário o uso de botões ou de outra interface de auxílio para controlar os três graus de liberdade do efetuador terminal, ou mesmo a adaptação de um joystick de 3 eixos para ser operado pelo queixo.
Sopro e sucção	Boa alternativa para pessoas com limitações de movimentos de membros superiores. A interface pode ser facilmente fabricada com baixo custo associado.	Só é possível fazer o acionamento de um movimento por vez (não permite enviar comandos simultâneos). É necessário um esforço pulmonar contínuo do usuário, podendo levá-lo rapidamente à fadiga. Cada tubo é associado a um grau de liberdade, nesse caso é necessário o uso de estratégias para aumentar o grau de liberdade, como uso de mais tubos ou uso de um painel de LEDs indicativo de movimentos.

Continua...

Tabela 5

Interface	Vantagens	Desvantagens
Eletro Oculografia / Eletromiografia / Eletroencefalografia	Boa alternativa para pessoas com limitações de mobilidade, especialmente sem os movimentos dos membros superiores. O tempo de resposta é muito rápido, comparado com outros sinais biomédicos. Técnica não invasiva.	Sinais resultantes com baixo grau de ordem de grandeza, sendo necessário o uso de técnicas de filtragem para manipulação desses dados. Muito suscetível à interferências, ruídos e movimentos involuntários. O algoritmo necessita passar por treinamento para reconhecer padrões provenientes desses sinais cerebrais, para posteriormente associá-los com as movimentações desejadas. Os sinais do eletroencefalograma variam de pessoa para pessoa, e necessitam de calibragem constante. Possui um custo mais elevado.
Movimentação da cabeça (sensores)	Boa alternativa para pessoas com limitações de movimento em membros superiores. Os sensores são baratos e facilmente encontrados.	É necessário sempre se atentar às calibrações dos sensores para que movimentos involuntários do usuário não se confundam com comandos. A movimentação exigida do pescoço nos três eixos pode não ser trivial de ser executada pelo usuário. Possui um custo mais elevado pois necessita de sensores de qualidade e precisos.

Continua...

Tabela 5

Interface	Vantagens	Desvantagens
Comando de voz	Boa alternativa para pessoas com limitações severas de movimento nos membros superiores e inferiores. Utilização prática e intuitiva.	Exige técnicas de filtragem de sinal e uso de softwares específicos para reconhecimento de voz. Muito suscetível à ruídos sonoros do ambiente, e exige boa qualidade de vocalização do usuário. Exigiria também o uso de visão computacional para que o objeto indicado pelo usuário fosse reconhecido pela máquina. Custo muito elevado.
Visão computacional e processamento de imagem	Boa alternativa para pessoas com limitações severas de movimento nos membros superiores e inferiores, e que possuem dificuldade na fala. A solução confere maior autonomia à máquina, portanto é mais automatizada.	Interface com custo elevado, visto que necessita de, no mínimo, uma câmera com resolução considerável e um notebook ou tablet, pois requer um bom processamento computacional. É necessário ter uma boa iluminação no ambiente para identificação correta dos comandos e dos objetos.
Fonte: Própria autora.		

Diante das análises levantadas, pode-se inferir que as interfaces que melhor se adequam ao escopo do projeto são a de joystick de queixo e de visão computacional, visto que as demais ou possuem maiores complexidades associadas ao seu desenvolvimento ou a usabilidade não é ergonometricamente viável, como no caso da movimentação da cabeça em três eixos.

Assim, tendo como base as interfaces de joystick de queixo e de visão computacional, realizou-se análises detalhadas e propostas de soluções para a problemática central do projeto. Essas análises são apresentadas nas seções seguintes deste capítulo.

4.2 Interface baseada em Joystick

Inicialmente, tendo como foco uma solução teórica simples e de fácil implementação, foi avaliada a possibilidade de utilização de um joystick de 3 eixos adaptado para o queixo como interface para comandar o efetuador terminal do braço robótico.

Construtivamente, os joysticks de três eixos funcionam através do controle de 2 potenciômetros e um botão, pressionando o centro de sua haste para baixo. As duas entradas de potenciômetro são referentes aos eixos X e Y, e o botão é referente ao eixo Z.

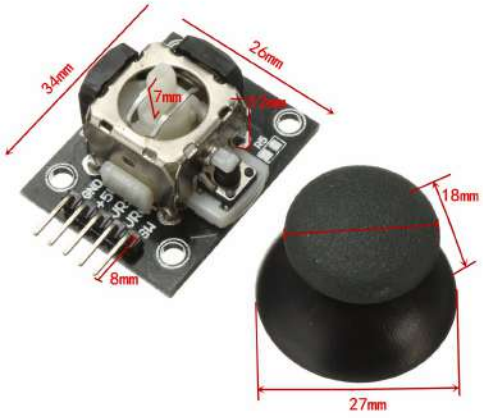


Figura 38 – Dimensões do joystick.
(ELETRÔNICA, 2021)

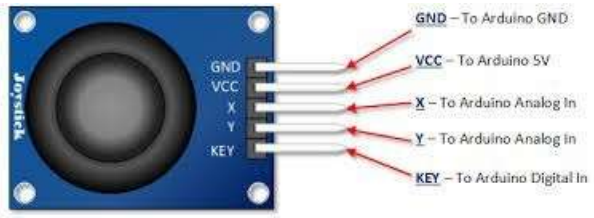


Figura 39 – Esquemático elétrico do joystick.
(ELETRÔNICA, 2021)

Nesta proposta, a ideia é controlar os eixos X e Y do efetuador terminal através dos potenciômetros, sendo possível controlar 2 graus de liberdade por vez; e utilizar a entrada de botão como um comutador para alterar o controle do plano X e Y pelo eixo Z do efetuador terminal, funcionando como uma espécie de chave. O joystick teria que ser posicionado em um suporte fixo próximo ao queixo do usuário, e seria necessário realizar também ajuste em seu design, para que o botão central da haste possa ser pressionado facilmente pela pressão do queixo.

Nesta solução, o usuário teria de controlar cada eixo do efetuador terminal até alcançar o objeto desejado (movimentos de pitch, yaw e roll), diminuindo a autonomia do sistema e exigindo mais esforço do usuário, além de ser ergonometricamente difícil pressionar uma haste com o queixo, visto que esta pode fletir para os lados. Tendo isso em vista, esta solução não foi levada a diante.

4.3 Interface baseada em Visão Computacional

A segunda solução idealizada foi a utilização da visão computacional como interface para identificar um objeto a ser capturado pelo braço robótico, tendo como referencial a

indicação de um ponteiro laser, operado pelo usuário, para este objeto. Neste sentido, os esforços desta solução são voltados para:

1. Adaptar um ponteiro laser para que um usuário tetraplégico possa utilizá-lo. Uma ideia inicial seria posicionar o ponteiro laser na lateral da cabeça do usuário, na altura dos olhos, e operar o seu disparo com um sensor de sopro;
2. Implementar um sistema de visão estéreo;
3. Aplicar conhecimentos de visão computacional e de processamento de imagem nos seguintes quesitos:
    a) detecção do ponto laser na imagem;
    b) calcular as coordenadas 3D do objeto com o sistema de visão estéreo;
4. Calcular trajetória a ser realizada pelo braço robótico para atingir as coordenadas 3D obtidas pela visão computacional, aplicando para isso a cinemática inversa;
5. Ativar o braço robótico para capturar o objeto e trazê-lo para próximo do rosto do usuário.

Como pode ser notado, apesar de mais complexa, esta é uma solução que apresenta grande grau de autonomia à máquina, sendo um processo automatizado onde o usuário apenas necessita indicar o objeto com um laser, e o controle de trajetória do braço robótico é feito pelo próprio sistema. Um esquemática dessa solução é apresentado na Figura 40

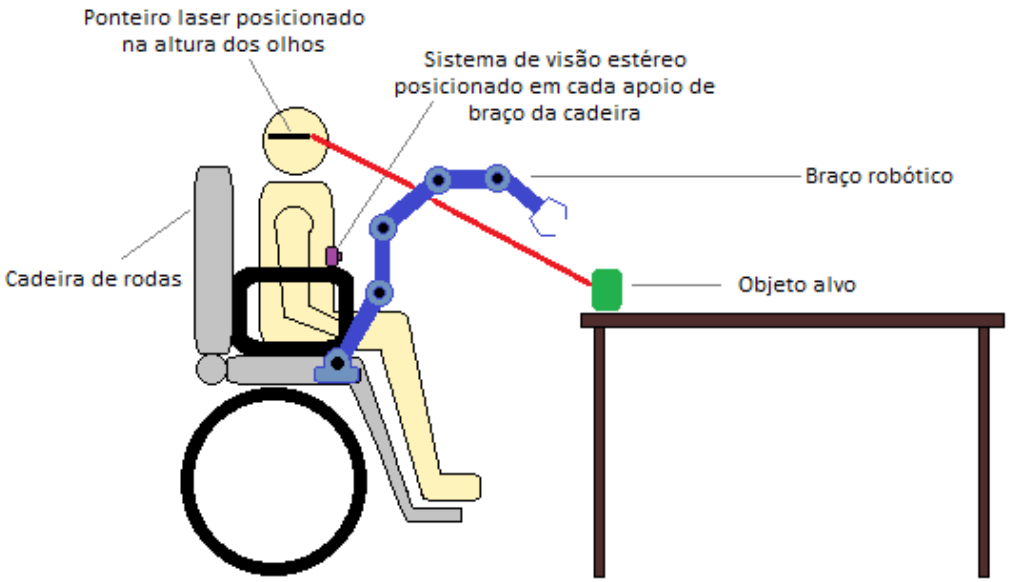


Figura 40 – Esquemático da solução proposta. Fonte: adaptado de (ZHONG; ZHANG, Y.; YANG, 2019).

Trabalhos similares envolvendo o uso de ponteiro laser em cadeira de rodas foram desenvolvidos em (ZHONG; ZHANG, Y.; YANG, 2019) e (BHARALI et al., s.d.). Em (ZHONG; ZHANG, Y.; YANG, 2019) foi desenvolvido um sistema de detecção de ponto laser com finalidade de também comandar um braço robótico acoplado a cadeira de rodas, através do uso de técnicas de visão computacional, processamento de imagem e uso de redes neurais convolucionais (CNN) *(machine learning)*. Porém, seu público alvo não é restrito a usuários portadores de tetraplegia, visto que o ponteiro laser é operado manualmente. Ainda sobre o trabalho (ZHONG; ZHANG, Y.; YANG, 2019), uma das técnicas aplicadas para determinação das coordenadas 3D do objeto é o uso de uma câmera RGB-D, que fornece informações sobre a profundidade (distância) dos objetos para a câmera, o que auxilia na precisão dos cálculos de estimação de coordenadas. Outro ponto a ser destacado é que esta solução é desenvolvida para identificar somente objetos pré-selecionados, no caso, uma banana, uma laranja, uma bola, um mouse, uma xícara, um brinquedo, uma colher e um garfo; tratando-se, portanto, de um projeto voltando para ambientes estruturados.

Em (BHARALI et al., s.d.) foi desenvolvido um sistema mais complexo com associação de interfaces de eletroencefalografia, eletro-oculografia e visão computacional para controle de direção de cadeira de rodas, tendo como público alvo usuários com limitações motoras severas como paraplegia, tetraplegia, doença do neurônio motor, entre outras deficiências. Nesta solução, sinais cerebrais dos pacientes são capturados através da técnica da eletroencefalografia, e esses sinais depois de processados e amplificados são utilizados para acionar um ponteiro laser na direção requerida, e então são utilizadas técnicas de visão computacional e processamento de imagem para identificação do laser, para que posteriormente seja realizado os cálculos de direção e ativação dos motores das rodas. Assim, a cadeira de rodas deve seguir o caminho indicado de forma autônoma na direção especificada. A interface de eletro-oculografia é utilizada como meio de aprimoramento de dados, para evitar que qualquer tipo de colisão aconteça. Esta solução é implementada para controlar a direção da cadeira de rodas em ambientes internos.

Buscando uma solução mais simplificada do que as mencionadas em (ZHONG; ZHANG, Y.; YANG, 2019) e (BHARALI et al., s.d.), de forma que o custo e complexidade associados não sejam tão elevados, esta proposta envolve a aplicação de visão computacional para detecção do ponto laser com técnicas clássicas de visão computacional, através da implementação de um sistema estéreo, e estimar as coordenadas do ponto laser com as técnicas descritas no Capítulo 3 de Fundamentação Teórica.

5 Implementação

Os objetivos técnicos almejados neste projeto são: possibilitar que o usuário indique um objeto que deseja que seja capturado, detectar a indicação/objeto, calcular as coordenadas no espaço tridimensional do objeto, acionar o braço robótico para que capture o objeto, e por fim, trazê-lo para perto do usuário.

Tendo em vista as análises realizadas no Capítulo 4, o projeto tem como propósito desenvolver uma interface de comando utilizando visão computacional para determinar as coordenadas espaciais de um objeto e comandar o braço robótico para pegá-lo. Portanto, as tarefas encarregadas a esta interface são: identificar o ponto laser acionado pelo usuário, que deverá estar apontando para o objeto desejado, calcular as coordenadas espaciais do ponto laser, realizar a cinemática inversa com os dados obtidos e comandar o braço robótico para buscar este objeto.

Neste sentido, os esforços empregados na implementação deste projeto consistem em:

1. criar um suporte para alojar o ponto laser a ser utilizado pelo usuário como indicador;
2. aplicar visão computacional e processamento de imagem nos seguintes quesitos:
    a) implementação da visão estereoscópica;
    b) detecção do ponto laser;
    c) cálculo das coordenadas do ponto laser no espaço tridimensional;
    d) calibração das câmeras;
3. aplicar cinemática inversa para calcular a trajetória a ser realizada pelo braço robótico com as coordenadas obtidas na visão computacional;
4. acionar o braço robótico para alcançar o objeto indicado;
5. pegar o objeto e levá-lo para uma posição específica, próxima ao usuário.

Este capítulo é dedicado a descrever as técnicas, implementações e recursos computacionais utilizados, bem como as justificativas de escolhas de projeto e procedimentos adotados.

O projeto foi desenvolvido em um notebook Acer com processador AMD Ryzen 7 5700U, sistema operacional Windows 11, e parcialmente em um notebook Dell com sistema operacional Linux Ubuntu 20.04.

Os algoritmos foram desenvolvidos na linguagem Python (versão 3.10.5), que é uma linguagem de alto nível e orientada a objetos, possuindo sintaxe clara, simples e concisa, o que torna a linguagem mais produtiva. Ainda conta com estruturas de alto nível, como listas e dicionários, possui uma coleção de módulos prontos para uso e a capacidade de importar frameworks desenvolvidos por terceiros. Desta maneira, a linguagem é multiplataforma, sendo suportada por diferentes sistemas operacionais. Outras vantagens importantes são:

1. É amplamente reconhecida e utilizada por grandes empresas da tecnologia da informação;
2. Documentação completa;
3. É uma das linguagens de programação mais utilizadas;
4. Possui suporte para bibliotecas de visão computacional, processamento de imagens, inteligência artificial, entre outros;
5. É possível integrar com outras linguagens, como Linguagem C. (SOUSA, 2019)

O desenvolvimento do código foi feito na IDE Visual Studio Code (VSCode), que é uma plataforma gratuita desenvolvida pela Microsoft, e reúne diversas ferramentas, configurações e extensões que auxiliam no desenvolvimento de programas, além de suportar as principais linguagens de programação e frameworks.

Para captura e processamento de imagens foi utilizado o OpenCV (Open Source Computer Vision Library), uma biblioteca de código aberto e multiplataforma, com ênfase no desenvolvimento de aplicações na área de visão computacional, processamento de imagens e aprendizagem de máquina. Foi desenvolvida pela Intel Corporation em 2000, e escrita originalmente na linguagem C/C++, mas fornece suporte para uso em outras linguagens como Python, Java, Matlab e Ruby. A biblioteca foi projetada com o intuito de tornar aplicações e estudos em visão computacional mais acessíveis. (ROCHA, s.d.)

A biblioteca OpenCV possui uma estrutura modular e recursos de processamento em tempo real de imagens, contando com módulos de processamento de imagens, processamento de vídeos, análise e estrutura de dados, álgebra, e ainda possui inúmeros algoritmos de Visão Computacional.

Ademais, foi utilizado o software Matlab, versão R2017b, que é uma ferramenta matemática de alta performance, com linguagem de programação de alto nível e interface de desenvolvimento própria, tendo como principais funções: cálculo numérico, análise e visualização de dados, manipulação de matrizes, construção de gráficos e compilação de funções. O MATLAB possui ainda uma grande quantidade de bibliotecas auxiliares denominadas Toolboxes, que otimizam o tempo despendido na realização de tarefas e fornecem interfaces gráficas.

Foi utilizado também o software Fusion 360°, um aplicativo para design CAD/CAM/-CAE/PCB desenvolvido pela Autodesk, que permite realizar modelagens livre em 2D e 3D, fornece ferramentas de análises, simulações e ensaios mecânicos e elétricos.

5.1 Visão Computacional

Nesta seção são retratados os experimentos e implementações de visão computacional realizados no projeto para alcance dos objetivos propostos. Na primeira subseção é abordado o processo de detecção do ponto laser. Na segunda subseção são apresentados os testes iniciais da aplicação teórica de visão estéreo, vista na subseção 3.3.1.1 do Capítulo 3, e são feitos experimentos manuais para obtenção de parâmetros das câmeras, cálculo de coordenadas 3D através da triangulação e ajustes de erros. Por fim, na terceira subseção são apresentadas implementações de visão estéreo utilizando ferramentas computacionais, mais especificamente o software Matlab e a biblioteca OpenCV, que proveem recursos de calibração estéreo, retificação de imagens, ajuste de distorção e algoritmos de triangulação.

5.1.1 Detecção do ponto laser

As câmeras são ligadas através da função cv2.VideoCapture() da biblioteca OpenCV. Para cada imagem captada pelas câmeras, são aplicadas operações de pré-processamento com a finalidade de obter imagens de maior qualidade, através da aplicação do filtro de média para redução de ruído. Em seguida, são aplicadas técnicas de threshold para isolar o ponto laser, objeto de interesse que buscamos identificar e localizar espacialmente.

Por padrão, a biblioteca OpenCV manipula as imagens no sistema de cores Blue, Green e Red (BGR). Assim, a primeira etapa da detecção do ponto laser consiste em transformar a representação da imagem para o espaço de cores HSV, formado pelas componentes hue (matiz), saturation (saturação) e value (valor), que separam a informação de luminância (brilho) da imagem das informações de cores, sendo portanto mais adequado para identificar a luz emitida pelo laser, em comparação com o sistema BGR que identifica apenas a intensidade das cores.

Em seguida, são identificadas as faixas de cada canal do sistema HSV referente ao espectro emitido pelo feixe laser, para realizar a técnica de threshold. Neste sistema, foi constatado experimentalmente que o comprimento de onda emitido pelo laser vermelho encontra-se na faixa de [150, 62, 200] para o limiar mais baixo à [179 , 255, 255] para o limiar mais alto, onde cada valor correspondente aos canais H, S e V, respectivamente. Com os valores dos limiares encontrados, foi utilizada a função cv2.inRange() da OpenCV, que verifica se existem pixels na imagem dentro dos intervalos dos limiares fornecidos, funcionando como uma máscara, e a função cv2.minMaxLoc() que retorna os valores mínimos e máximos observados na região segmentada, e sua localização na matriz de posição da imagem. O valor

máximo teoricamente indica o centro do ponto laser, com maior intensidade de brilho. Por fim, é desenhado um círculo verde ao redor do laser identificado.

Esse processo é descrito no Algoritmo 1 abaixo:

Algorithm 1 Detecção do ponto laser

Inicialização das câmeras img1, img2

$limiar\_baixo = ([150, 62, 200])$
$limiar\_alto = ([179, 255, 255])$

$img1 = filtroDeMedia(img1)$
$img2 = filtroDeMedia(img2)$

$hsv1 = convertToHSV(img1)$
$hsv2 = convertToHSV(img2)$

$mask1 = inRange(hsv1, limiar\_baixo, limiar\_alto)$
$(minVal1, maxVal1, minLoc1, maxLoc1) = cv2.minMaxLoc(mask1)$

$mask2 = inRange(hsv2, limiar\_baixo, limiar\_alto)$
$(minVal2, maxVal2, minLoc2, maxLoc2) = cv2.minMaxLoc(mask2)$

$cv1.circle(img1, maxLoc1, 20, (0, 255, 0), 2, cv2.LINE_AA)$
$cv2.circle(img2, maxLoc2, 20, (0, 255, 0), 2, cv2.LINE_AA)$

O resultado prático da implementadção pode ser observado nas Figuras 41 e 42

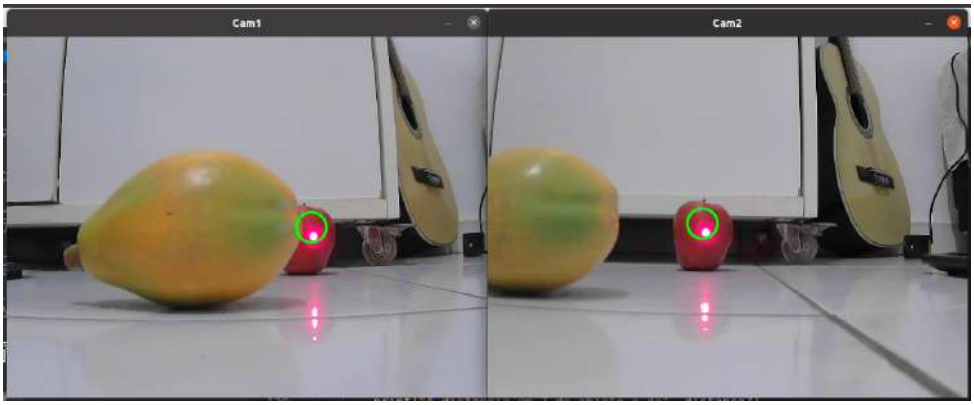


Figura 41 – Detecção do ponto laser - objeto maçã.

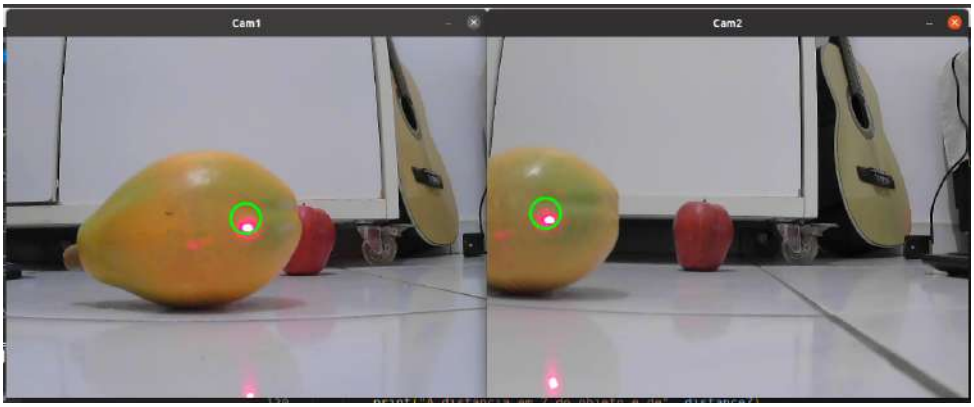


Figura 42 – Detecção do ponto laser - objeto mamão.

5.1.2 Visão estéreo - Abordagem com ajuste polinomial de erro

O sistema de visão estéreo foi implementado com o uso de 2 Webcams da Intelbras modelo CAM-720p, com as seguintes especificações, listadas na Tabela 6:

Tabela 6 – Especificações das câmeras.

Resolução	1280 x 720
Ângulo de abertura (diagonal x horizontal x vertical)	77° x 67° x 38°
Tipo de lente	3,6 mm
Tensão de alimentação	5Vdc
Dimensões	97 x 51 x 56 mm
Peso	147 g
Tipo de sensor	1/2 9"1.0 Mega Pixels - CMOS
Saída	USB 1.1, USB 2.0

As câmeras foram posicionadas com um distanciamento lateral de 97,5 mm entre suas lentes, conforme pode ser visto na Figura 43:

Figura 43 – Sistema de visão estéreo implementado.

As relações da visão estéreo foram estabelecidas através do método da triangulação entre duas câmeras e o objeto alvo, com uso de trigonometria e conceito de disparidade de imagens estéreo. Foram realizadas calibrações manuais para a determinação de alguns parâmetros importantes, como distância focal e ângulo de abertura.

Inicialmente foram feitas calibrações manuais para estimar a distância focal f (em pixels) do sensor da câmera para o seu frame, já que esta informação não é disponibilizada pelo fabricante. Esta etapa foi executada da seguinte maneira: foram tracejadas linhas centrais de orientação na tela de captura das câmeras, no sentido horizontal e vertical, com a função do OpenCV cv2.line. Em seguida, pegou-se uma folha em branco e traçou-se uma linha vertical em seu centro (folha no sentido paisagem). Por fim, a linha vertical da tela foi alinhada com a linha vertical da folha, e a linha horizontal da tela foi alinhada com o "horizonte" da folha.

Em seguida, foram feitas as demarcações dos limites do campo de visão das câmeras no papel, e então, determinados seus ângulos de abertura com o auxílio de um transferidor, conforme ilustram as Figuras 44 e 45:

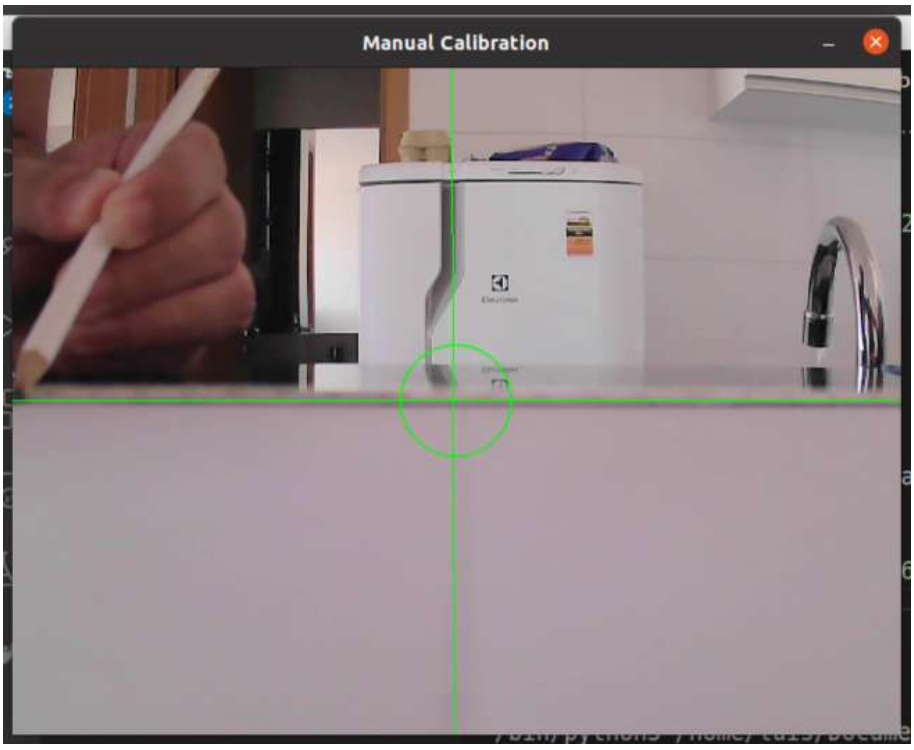


Figura 44 – Processo de medição do ângulo de abertura horizontal.

Este procedimento foi executado para as resoluções de 1280x720 e 640x480, pois percebeu-se que para a resolução de 1280x720 a biblioteca OpenCV não compilava adequadamente e seu processamento era interrompido, sendo necessário, portanto, diminuir a resolução padrão das câmeras para 640x480.

Os resultados obtidos para os ângulos de abertura, considerando a resolução de 640x480, estão na Tabela 7:

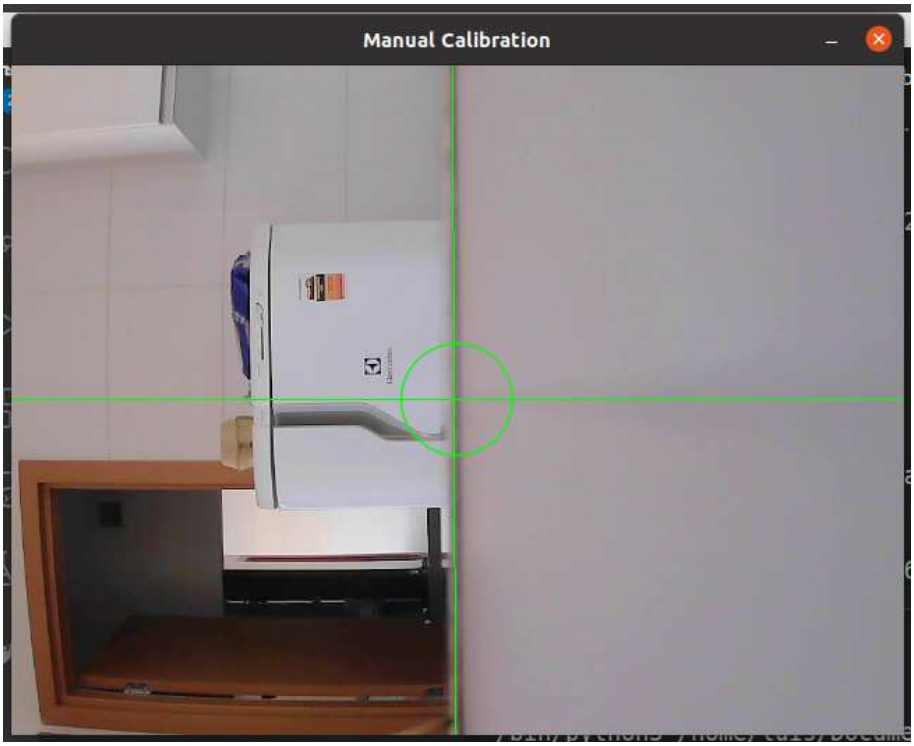


Figura 45 – Processo de medição do ângulo de abertura vertical.

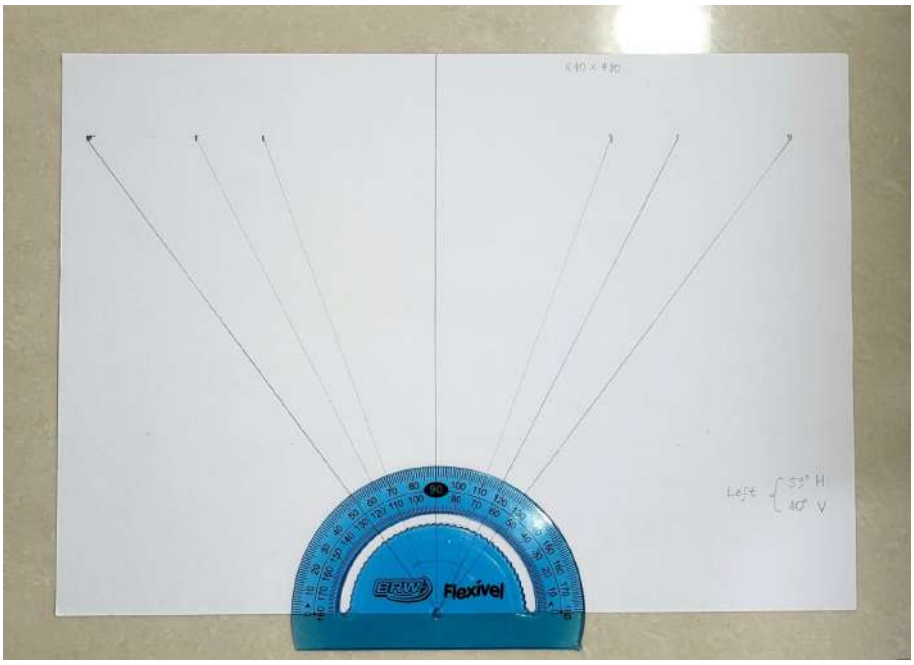


Figura 46 – Processo de medição dos ângulos de abertura.

Tabela 7 – Especificações das câmeras.

	Câmera 1 (esquerda)	Câmera 2 (direita)
Horizontal	53°	53,5°
Vertical	40°	41°

A diferença nos resultados obtidos entre as câmeras, apesar de pequena, pode ser devido a incertezas de medição durante o processo de calibração descrito acima, e também

devido a qualidade do processo de fabricação das câmeras, apesar deste último ser menor. De toda forma, em razão das pequenas discrepâncias presentes, pegou-se a média entre os valores observados para cinco medições consecutivas, para serem utilizados no cálculo da distância focal, correspondendo a 53,25º horizontal e 40,5º vertical.

Com os ângulos de abertura e a resolução em mãos, foi possível estimar a distância focal através da trigonometria. Sabendo que a resolução horizontal da tela de captura que estamos trabalhando é de 640 px, foi possível determinar a distância focal (Equação 5.1) aplicando a função tangente, considerando metade da resolução do frame e metade do ângulo horizontal encontrado, conforme ilustra a Figura 47, na qual a linha vermelha indica a distância focal f de interesse:

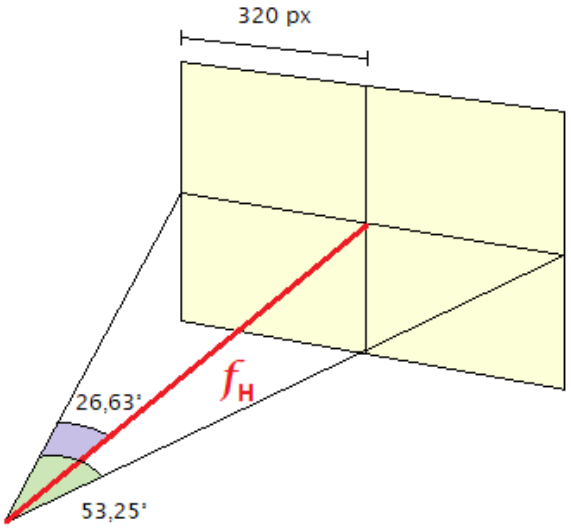


Figura 47 – Ilustração da distância focal horizontal.

$$f_H = \frac{\frac{640px}{2}}{tg(\frac{53,25°}{2})} \tag{5.1}$$

O mesmo processo foi feito para calcular a distância focal vertical, considerando metade da resolução vertical e metade do ângulo vertical encontrado (Equação 5.2), conforme ilustrado na Figura 48:

$$f_V = \frac{\frac{480px}{2}}{tg(\frac{40,5°}{2})} \tag{5.2}$$

Com isto, foram obtidas as distâncias focais de f_H=638,33px e f_V=650,55px. Teoricamente, ambas as distâncias focais, f_H e f_V, correspondem à mesma medida (cateto adjacente do triângulo), porém, devido aos aspectos construtivos das câmeras, essa proporção não é mantida.

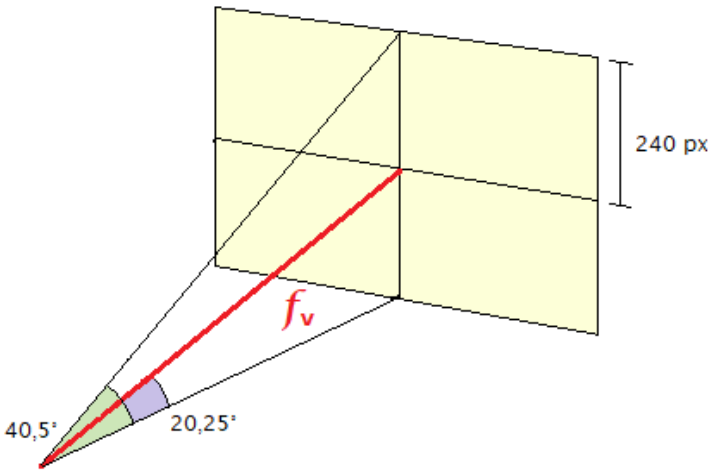


Figura 48 – Ilustração da distância focal vertical.

A origem do sistema de coordenadas global, usada para orientação dos cálculos das coordenadas 3D no mundo real, é considerada no centro óptico da Câmera 1 (esquerda), conforme ilustra a Figura 49:

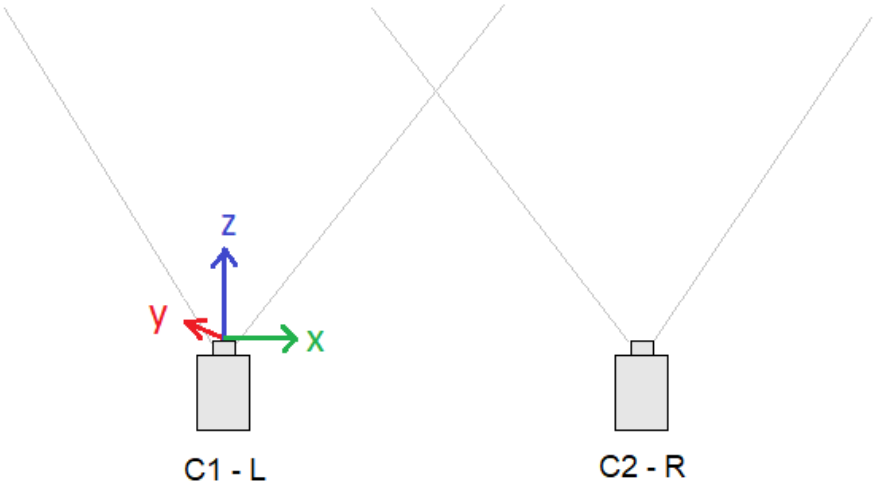


Figura 49 – Ilustração da origem do sistema de coordenadas global estabelecido.

5.1.2.1 Sistema de triangulação e cálculo das coordenadas 3D

Depois de realizada a identificação do ponto laser conforme descrito na seção 5.1.1, o próximo passo foi determinar suas coordenadas 3D com respeito ao sistema de coordenadas global. Para isto foi utilizado o método da triangulação, já descrito no Capítulo 3, seção 3.3.1.5, onde cada câmera e o alvo ocupam os vértices de um triângulo, conforme ilustrado na Figura 50:

O primeiro objetivo foi determinar a distância do objeto para as câmeras, informação

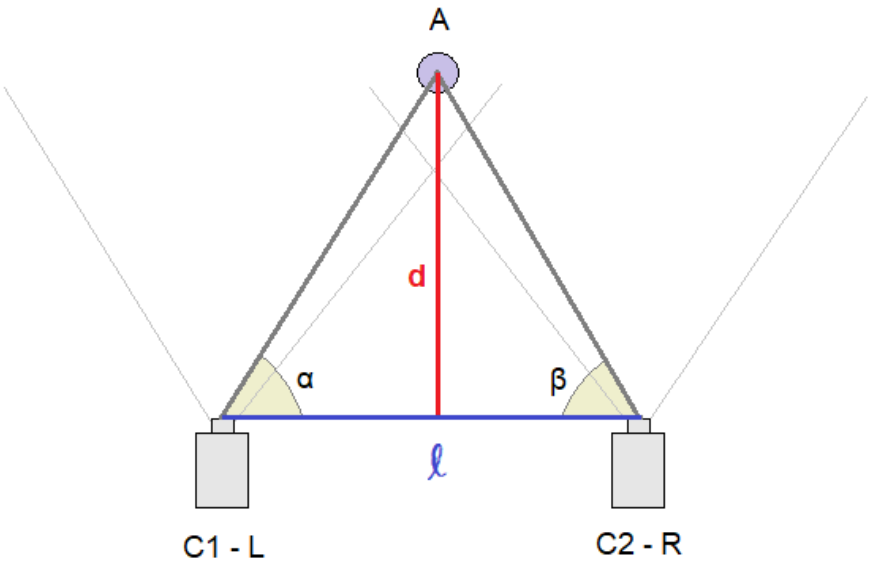


Figura 50 – Ilustração das relações geométricas da triangulação de câmeras.

que se perde na câmera, já que esta é uma representação 2D de um ambiente 3D. Para isto, foi necessário encontrar os ângulos identificados como α e β na Figura 50, que indicam os ângulos de orientação do ponto laser projetado no frame de cada câmera.

Para encontrar os ângulos de orientação do ponto laser, primeiramente foi determinado sua localização em pixel, em cada câmera, e posteriormente foi calculado o ângulo formado entre a posição do ponto laser e o centro óptico de cada câmera, conforme ilustra a Figura 51:

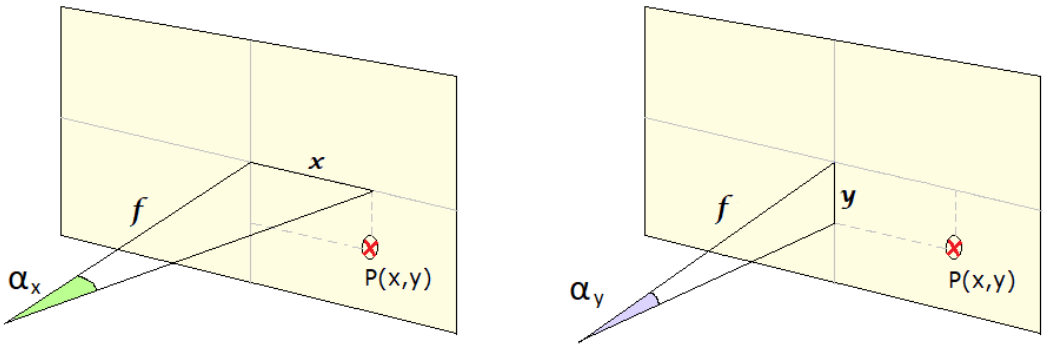


Figura 51 – Ilustração das relações geométricas para obtenção do ângulo formado entre um ponto e o centro da imagem.

Desta forma obteve-se as relações para os ângulos α_x e α_y através da função tangente:

$$\alpha_x = \arctan(\frac{x - 320}{f_x}) \tag{5.3}$$

$$\alpha_y = \arctan(\frac{x - 240}{f_y}) \tag{5.4}$$

Assim, a determinação dos ângulos de orientação do ponto laser é feita em relação ao centro do frame de cada câmera, considerando esta a origem do sistema de coordenadas interno das câmeras. É importante observar que os ângulos determinados aqui são, na verdade, os ângulos complementares daqueles indicados na Figura 50, e consequentemente, antes de desenvolver as relações que determinam a distância d, ajusta-se os ângulos da seguinte maneira:

$$\alpha = \begin{cases} 90° + \alpha_x, & \text{se } \alpha_x < 0 \\ 90° - \alpha_x, & \text{se } \alpha_x > 0 \end{cases} \tag{5.5}$$

Depois de realizar este processo para as duas câmeras, o restante da análise leva em consideração inicialmente os ângulos horizontais de cada frame, ou seja, α_x e β_x. Adotando α_x como α e βx como β, estabelece-se as relações trigonométricas que permitem determinar a distância d indicada na Figura 50, aplicando a função tangente da seguinte maneira:

$$l = \frac{d}{tan(\alpha)} + \frac{d}{tan(\beta)} \tag{5.6}$$

$$d = \frac{l * tan(\alpha) * tan(\beta)}{tan(\alpha) + tan(\beta)} \tag{5.7}$$

Ou, alternativamente, aplica-se a Lei dos Senos da forma explicitada nas Equações 3.16 e 3.17. Com a distância determinada e os ângulos de orientação do ponto laser em mãos, é possível determinar as coordenadas X, Y e Z do mundo real da seguinte forma:

$$\begin{cases} X = dtan(\alpha_x) \\ Y = dtan(\alpha_y) \\ Z = d \end{cases} \tag{5.8}$$

Outra forma simplificada de calcular a distância do objeto para a câmera (coordenada Z) através da geometria seria aplicando o Teorema de Tales e usando o conceito de disparidade entre imagens estéreo, conforme indica as proporções estabelecidas na Figura 52:

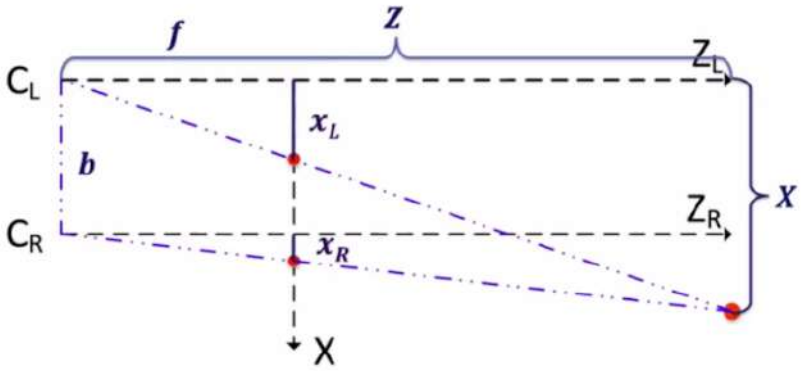


Figura 52 – Ilustração das relações geométricas para obtenção a distância a partir da disparidade. Fonte: (NIELSEN, 2022)

onde x_L é a projeção do ponto vermelho vermelho X na câmera da esquerda e x_R é a projeção do ponto X na câmera da direita, f é a distância focal e b é a distância entre o posicionamento das câmeras, resultando portanto na seguinte relação:

$$Z = \frac{fb}{x_L - x_R} \tag{5.9}$$

Utilizando esta abordagem ainda se faz necessário encontrar os ângulos de orientação do ponto laser no frame da câmera 1 para determinar as coordenadas X e Y.

5.1.2.2 Resultados e Ajustes de erros

Esta seção dedica-se a apresentar os resultados obtidos pelo método da triangulação descrito anteriormente, bem como descrever os ajustes de erros implementados.

5.1.2.2.1 Medições de distâncias (coordenada Z)

As primeiras medições realizadas tiveram o intuito de verificar se a distância calculada pelo método da triangulação atinge valores aceitáveis e próximos dos reais, para que posteriormente seja possível analisar sua aplicabilidade no projeto. De acordo com o método implementado, a distância Z é calculada entre a lente da câmera e o objeto indicado pelo ponto laser. Neste teste, o ponto laser foi mantido em uma base fixa (para diminuir a ocorrência de flutuações) e apontando para objeto fixo (parede), e foi feita a variação da distância da câmera para a parede com o auxílio de uma fita métrica, cuja incerteza de medição é de 0,5mm. O processo do experimento pode ser conferido nas Figuras a seguir:

Observou-se que a menor distância possível de ser calculada é de 100 mm, o que faz sentido, visto que o deslocamento lateral entre as duas câmeras é de 97,5 mm e o ponto laser precisa estar visível em ambos os frames para ser possível aplicar o método da triangulação.

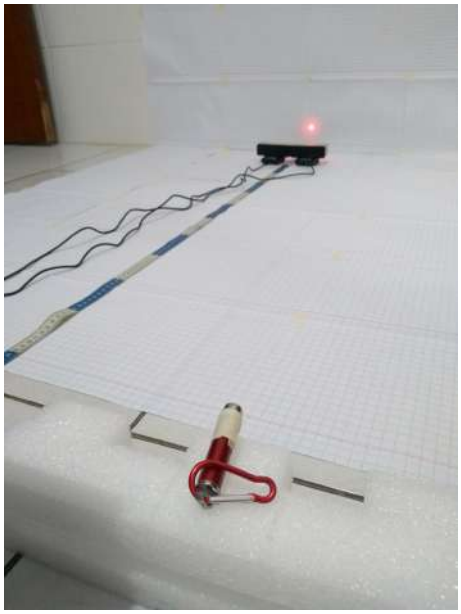


Figura 53 – Ambiente estruturado para medições com papel milimetrado.

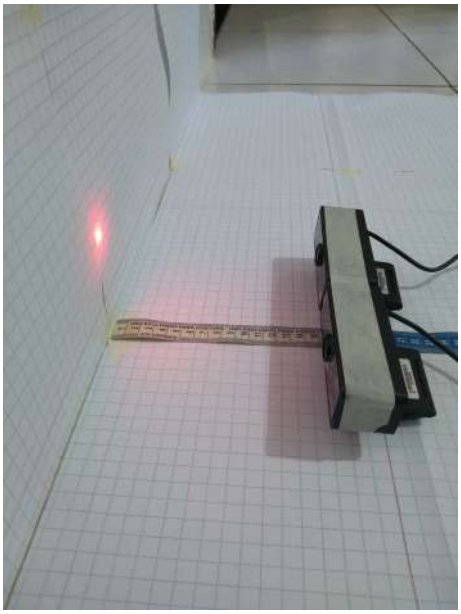


Figura 54 – Medições para distância de 15 cm.

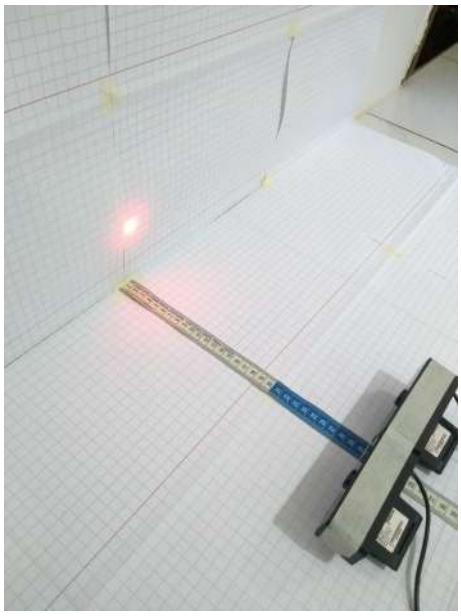


Figura 55 – Medições para distância de 30 cm

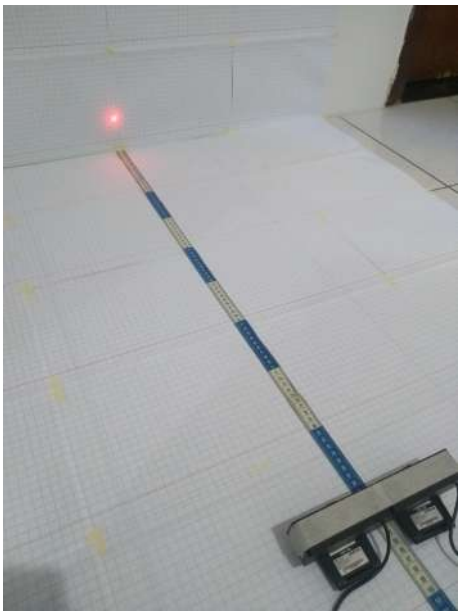


Figura 56 – Processo empírico das medições 90 cm.

Os cálculos e medições foram realizados para as distâncias de 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000 e 1100 mm, e para cada distância, foram adquiridas trinta amostras para obtenção de valores médios e desvios de cada posição. Os resultados são apresentados na Tabela 8:

Como é possível notar, à medida que a distância aumenta, também aumenta a imprecisão dos resultados. Para distâncias entre 150 e 600 mm temos uma boa precisão. Para distâncias entre 700 e 800 mm temos uma precisão razoável, enquanto que para distâncias entre 900 e 1100 mm temos precisão bem baixa. Como o braço robótico utilizado como base

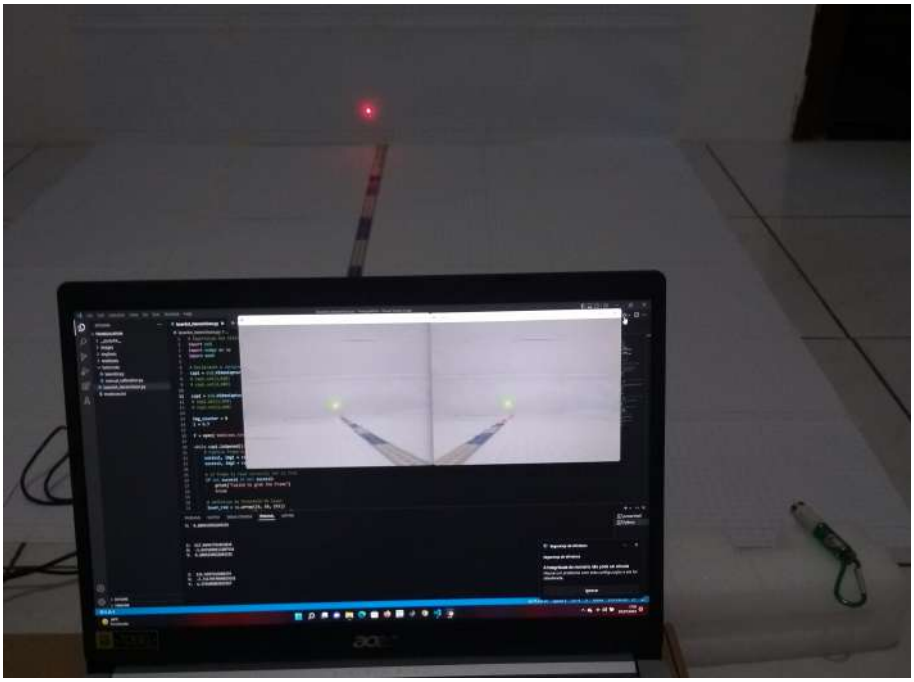


Figura 57 – Programa implementado para obtenção das medições.

Tabela 8 – Resumo das medições realizadas. Dimensões em milímetro (mm).

Distância real	Média da distância obtida	Erro absoluto	Erro percentual	Desvio padrão
$150 \pm 0{,}5$	152,6512	2,6512	1,7675	1,6392
$200 \pm 0{,}5$	201,6470	1,6470	0,8235	1,7626
$300 \pm 0{,}5$	296,2586	3,7413	1,2471	5,2100
$400 \pm 0{,}5$	400,7088	0,7088	0,1772	8,6139
$500 \pm 0{,}5$	508,2160	8,2160	1,6432	10,8002
$600 \pm 0{,}5$	595,3387	4,6612	0,7769	16,8062
$700 \pm 0{,}5$	710,3474	10,3474	1,4782	19,4259
$800 \pm 0{,}5$	811,8256	11,8256	1,4782	19,6159
$900 \pm 0{,}5$	932,9936	32,9936	3,6660	26,1367
$1000 \pm 0{,}5$	1041,8428	41,8429	4,1843	28,1017
$1100 \pm 0{,}5$	1179,4447	79,4448	7,2223	35,9135

para este projeto possui um alcance máximo de 1 m, podemos considerar que o erro máximo alcançado é de 4,18%.

Com os resultados em mãos, foi realizada uma regressão linear (Figura 58) para encontrar a descrição de seu comportamento.

A equação da regressão linear obtida foi $y = 1{,}064x - 22{,}15$, e pode-se observar que segue bem o comportamento dos resultados obtidos.

Foi feita ainda uma análise do erro absoluto para cada distância medida (Figura 59), aplicando-se regressões para verificar qual melhor adequa ao seu comportamento.

Comparando os valores do erro quadrático médio obtido em cada regressão, constatou-se que a polinomial de grau 3 se ajustou melhor no comportamento observado, descrita por

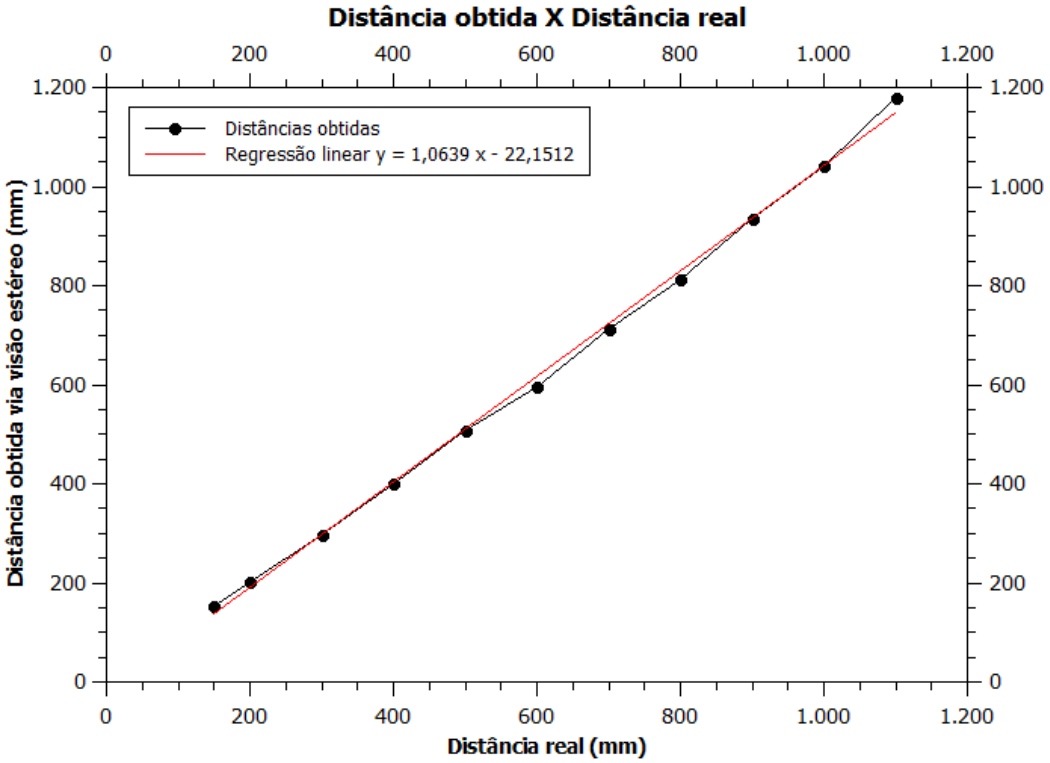


Figura 58 – Gráfico e regressão linear das medições realizadas.

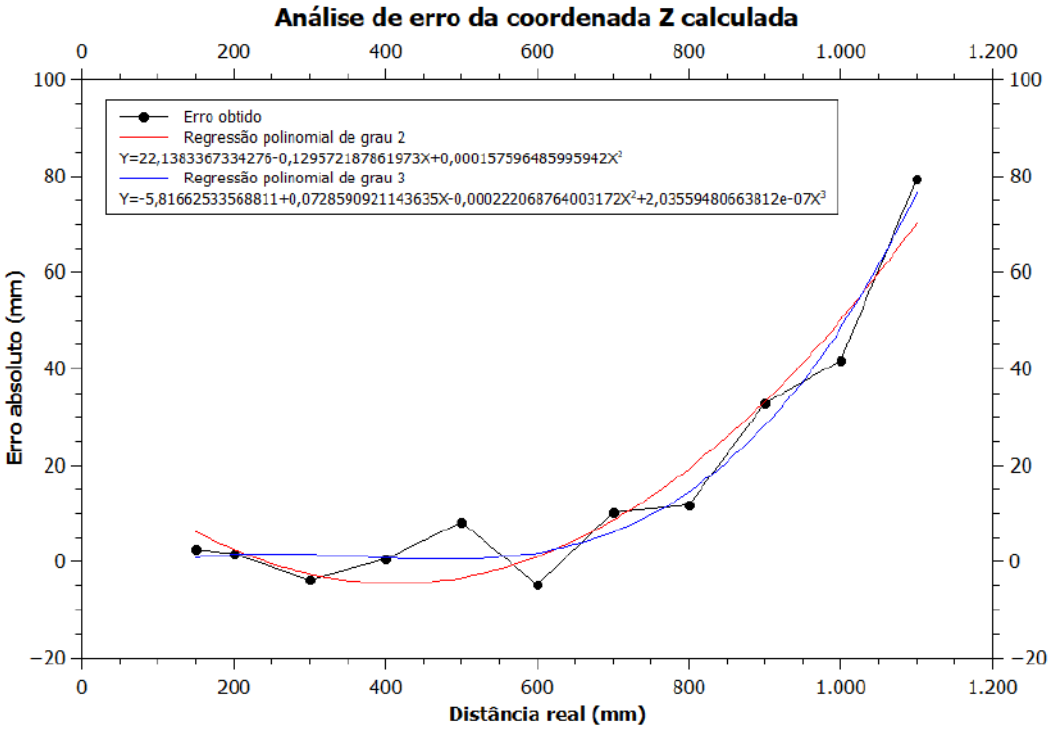


Figura 59 – Gráfico do erro em função da distância real e regressões polinomiais associadas.

$y = 5{,}8166 + 0{,}072859x - 0{,}000222x^2 + 2{,}0355910 - 7x^3$. Uma primeira ideia para tentar minimizar os erros obtidos seria somar a distância obtida com o inverso do comportamento descrito pela a regressão polinomial, assim, os erros para distâncias acima de 700 mm seriam

atenuados. Mas antes, foram feitos outros testes de medições e análises.

5.1.2.2.2 Análise de distorção (medições horizontais)

Posteriormente foram feitas medições longitudinais para verificar a interferência das distorções presentes nas lentes das câmeras no resultado da triangulação, pois foi observado que, à medida que o ponto laser se aproximava das bordas laterais do frame, as distâncias obtidas sofriam variações e erros grotescos da ordem de 30 cm.

Um exemplo do efeito da distorção das lentes das câmeras pode ser visualizada nas Figuras 60 e 61:

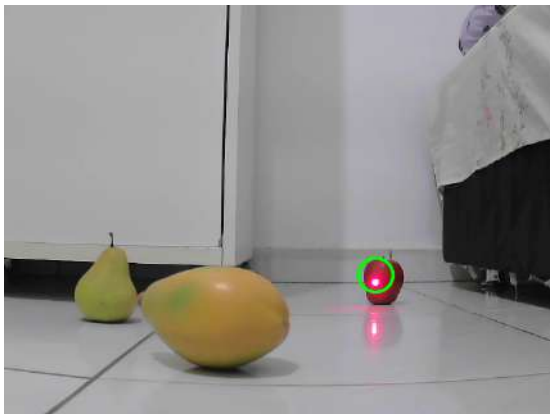


Figura 60 – Imagem original

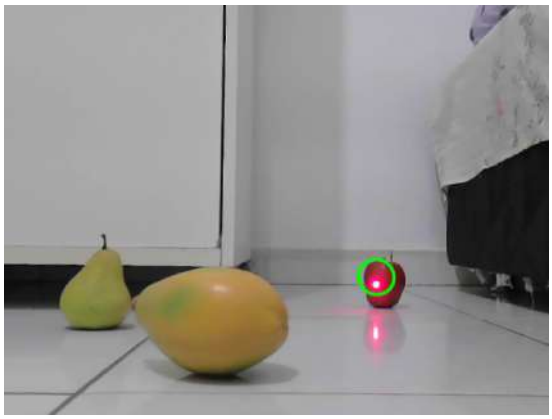


Figura 61 – Imagem com remoção de distorções

Portanto, o intuito deste experimento foi de verificar a variação do resultado da distância Z obtida em função da coordenada X no plano da imagem. Para realizar esta análise, foi aplicado o método da triangulação descrito na subseção 5.1.2.1, variando a posição X do laser no frame da tela, do limite esquerdo da câmera 2 ao limite direito da câmera 1, ou seja, no intervalo da área comum a ambas as câmeras, como pode ser visualizado nas Figuras 62 e 63:

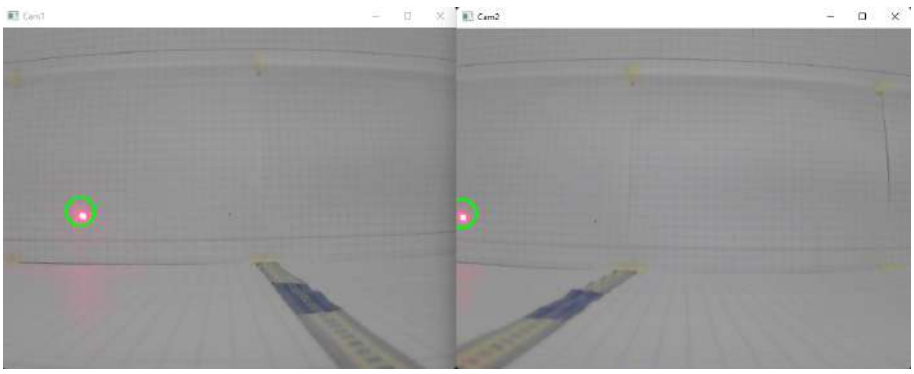


Figura 62 – Limite esquerdo da área comum do sistema de visão estéreo.

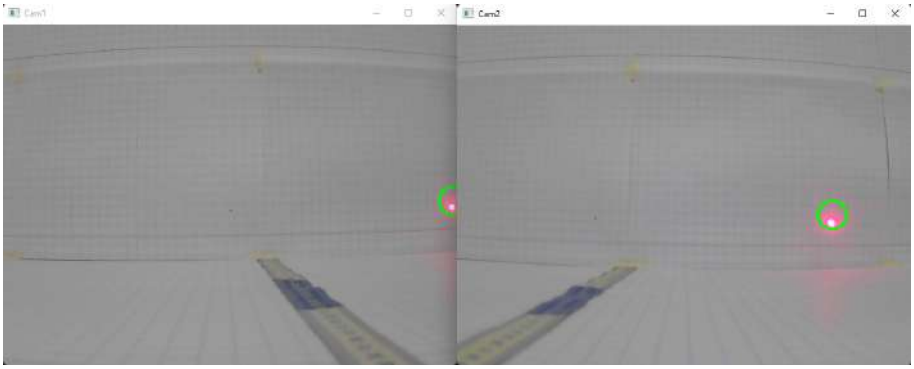


Figura 63 – Limite direito da área comum do sistema de visão estéreo.

A área útil foi subdividida em 30 pontos, e foi realizada a medição para cada ponto, de um extremo ao outro. Os resultados das coordenadas X e Z (distância calculada) foram salvos, e com isto foi possível analisar a interferência das distorções nos resultados do cálculo pela triangulação e encontrar padrões de comportamento.

O teste foi feito para diferentes distâncias, para que fosse possível verificar também se os parâmetros de distorção variavam com a distância, gerando assim uma análise mais complexa. Primeiramente foram feitas análises isoladas para cada distâncias de 200, 300, 400, 500, 600, 700, 800, 900 e 1000 mm, variando somente o valor da coordenada X do ponto laser no frame da imagem.

As análises foram feitas em função do erro absoluto da distância obtida (distância obtida menos distância real) e da coordenada X em pixel do ponto laser. Os resultados podem ser verificados nos gráficos em anexo na Seção Apêndice A.

Exemplificando, para a distância de 400 mm, foi obtida a seguinte análise de comportamento (Figura 64):

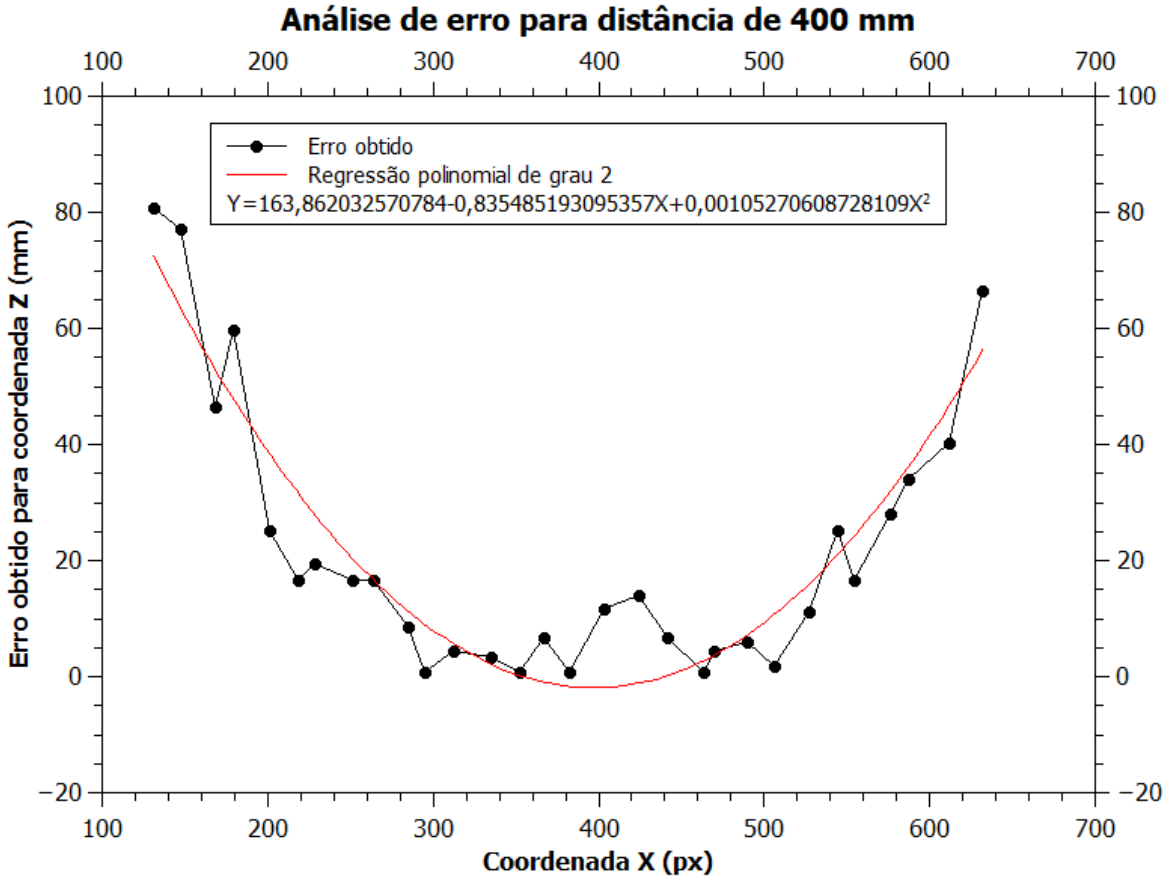


Figura 64 – Gráfico do erro para distância de 400 mm e regressão polinomial associada

Como pode ser observado, as medições apresentam dispersões pontuais, mas descrevem majoritariamente um comportamento polinomial quadrático, onde as laterais apresentam erros consideravelmente mais elevados do que os pontos centrais. Devido a isso, foi aplicada uma regressão polinomial de grau 2, cuja equação é descrita por $y = 163{,}862032 - 0{,}835452x + 0{,}001052x^2$.

As análises das demais distâncias foram realizadas de maneira similar. Os resultados condensados podem ser verificados na Figura 65:

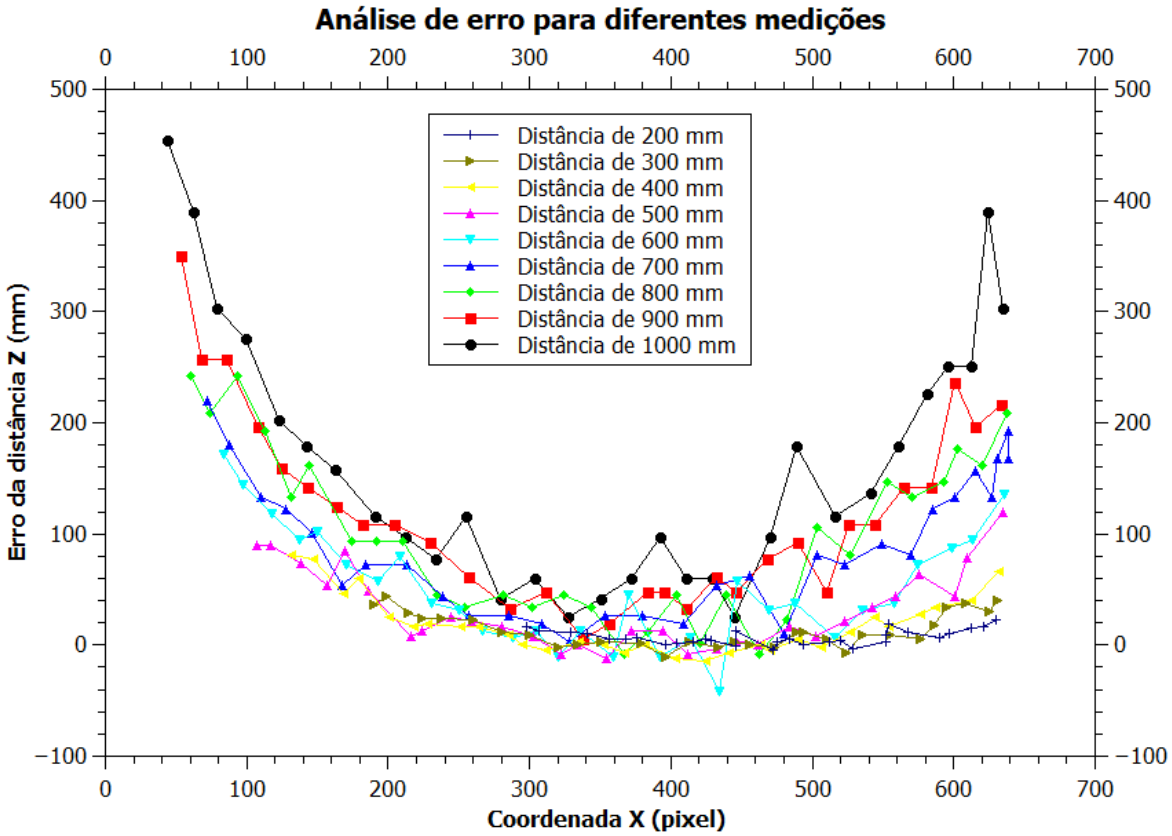


Figura 65 – Erro para todas as distâncias.

Como pode ser notado em todas as análises, o erro aumenta consideravelmente nas bordas do frame, e este aumento fica ainda mais evidente para distâncias maiores que 50 cm. É possível notar também que a lateral esquerda apresenta erros mais acentuados se comparados com a lateral direita. Isso significa que o lado esquerdo do sistema estéreo possui maior distorção. O erro médio de todas as medições do primeiro teste foi de 67,47, que corresponde a um percentual médio de 9,48%.

Para contornar o problema de distorção das lentes, a solução explorada foi de aplicar o inverso do comportamento descrito pelas regressões polinomiais na distância calculada pela triangulação, ou seja, somar a distância obtida com o negativo da regressão que descreve seu comportamento, para tentar reduzi-lo à zero:

$$Z_{obtido} = Z_{real} + Erro \tag{5.10}$$

considerando que o erro possa ser descrito por uma função polinomial de segunda grau, portanto:

$$Erro = ax^2 + bx + c \tag{5.11}$$

Como os coeficientes das regressões variam também em relação à distância Z, torna-se necessária a realização de uma análise que leve em consideração a variação do Z, para que os valores dos coeficientes a, b e c sejam escolhidos adequadamente na equação do erro. Desta forma, a grande questão passa a ser então encontrar uma equação bidimensional única que corrija o valor da distância Z obtida, considerando como variáveis independentes de entrada a posição X em pixel de onde o laser se encontra e a distância obtida pela triangulação.

Para isto, foi realizada uma análise da variação dos coeficientes das regressões polinomiais obtidas em função da variação da distância. Assim, foram encontrados os comportamentos descritos nas Figuras 66, 67 e 68 para cada coeficiente:

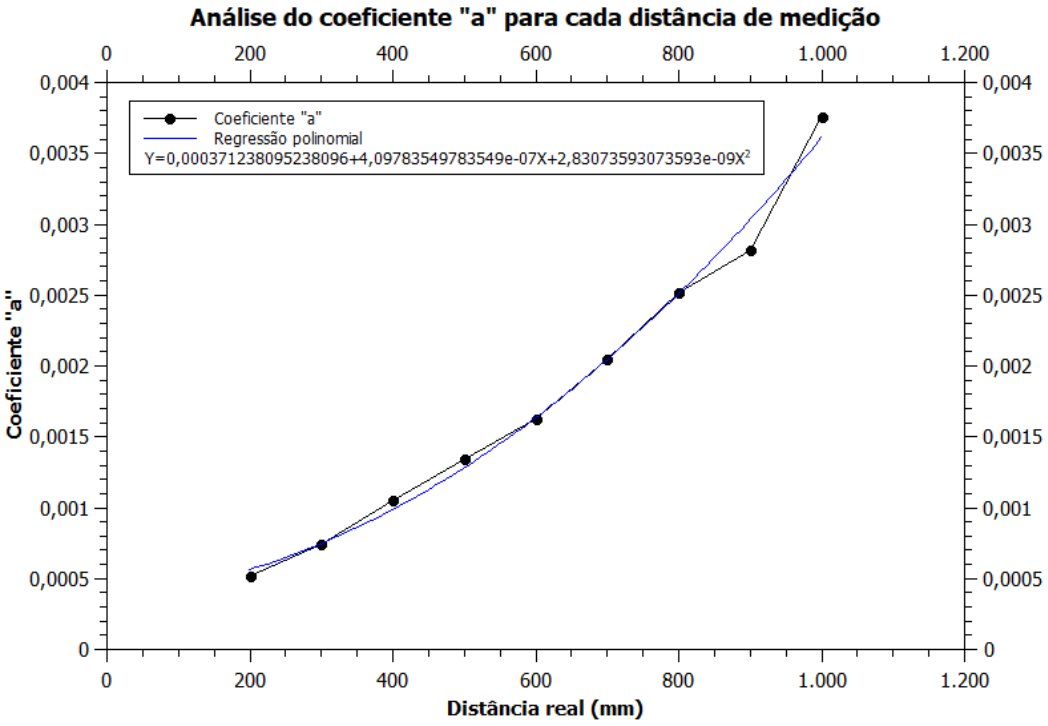


Figura 66 – Análise do coeficiente a das regressões realizadas para cada distância.

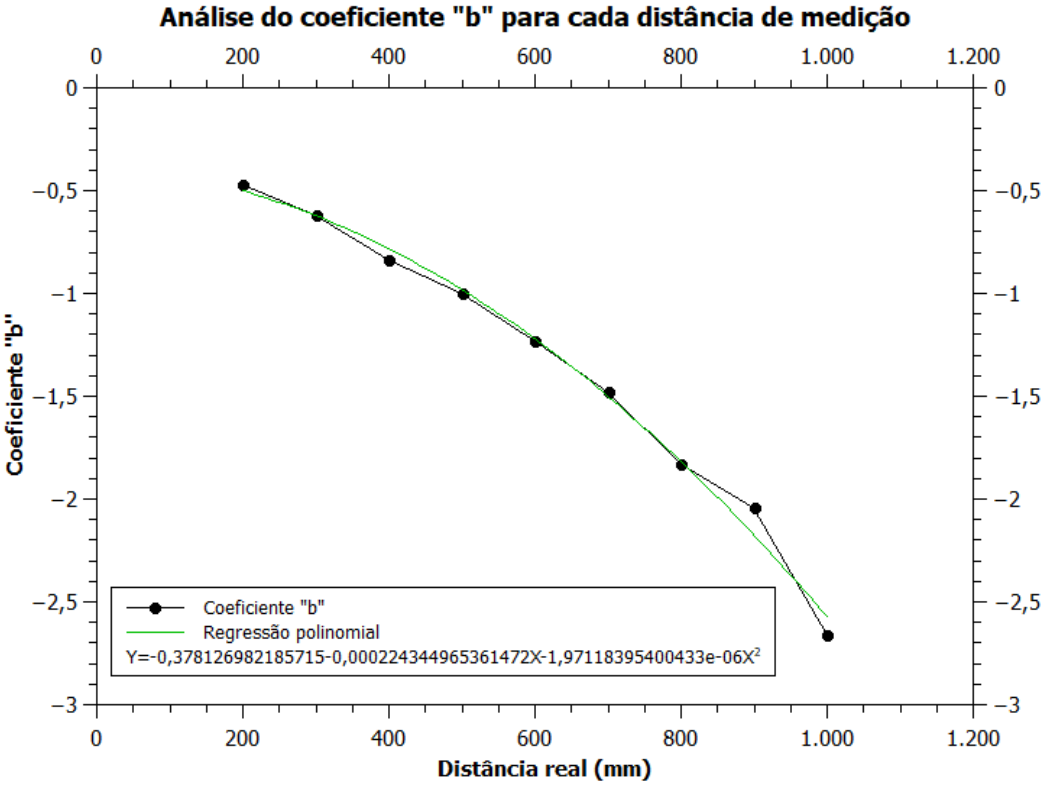


Figura 67 – Análise do coeficiente *b* das regressões realizadas para cada distância.

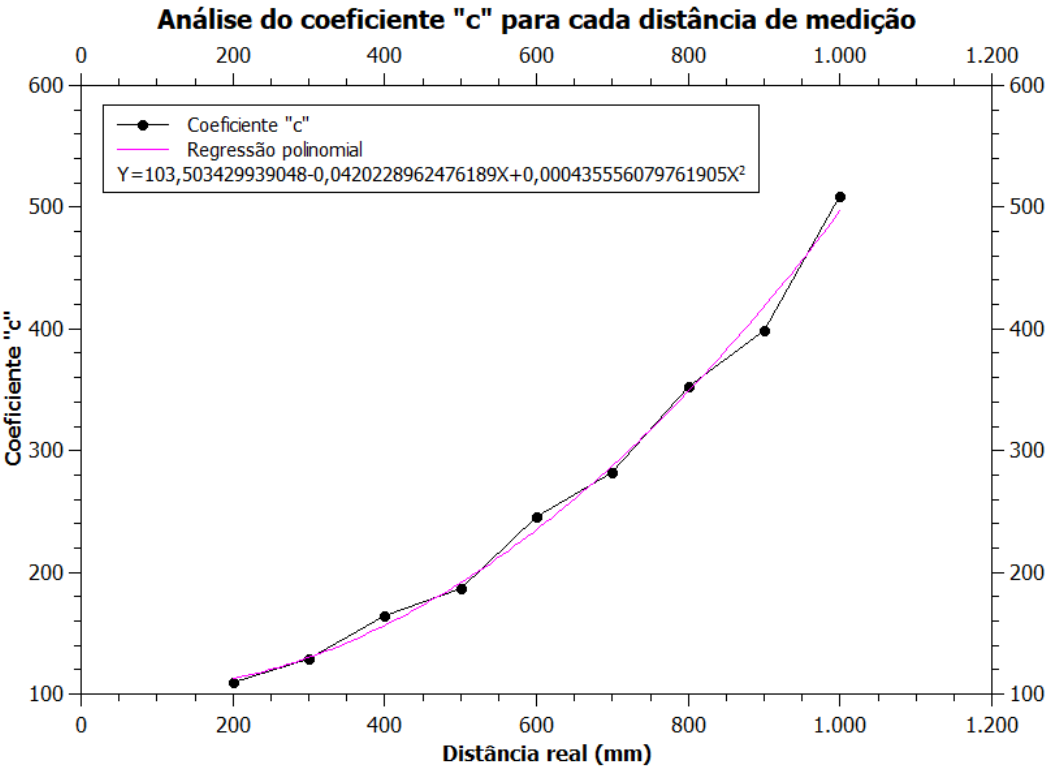


Figura 68 – Análise do coeficiente *c* das regressões realizadas para cada distância.

Com posse dos comportamentos de cada coeficiente, torna-se possível parametrizá-

los dentro da equação do erro, ou seja, cada coeficiente passa a ser definido também pelo seu comportamento de polinômio de segundo grau:

$$Erro(x, z_{real}) = (a_1 z_{real}^2 + a_2 z_{real} + a_3)x^2 + (b_1 z_{real}^2 + b_2 z_{real} + b_3)x + (c_1 z_{real}^2 + c_2 z_{real} + c_3) \quad (5.12)$$

Portanto, substituindo 5.12 na Equação 5.10, temos que:

$$Z_{obtido}(x, z_{real}) = z_{real} + (a_1 z_{real}^2 + a_2 z_{real} + a_3)x^2 + (b_1 z_{real}^2 + b_2 z_{real} + b_3)x + (c_1 z_{real}^2 + c_2 z_{real} + c_3) \quad (5.13)$$

Resolvendo para a variável Z_{real}, o valor de interesse, temos que o valor da nova distância corrigida é:

$$Z_{real} = \frac{-1 - a_2 x^2 + b_2 x - c_2 \pm \sqrt{(1 + a_2 x^2 + b_2 x + c_2)^2 - 4(a_1 x^2 + b_1 x + c_1)(a_3 x^2 + b_3 x + c_3 - Z_{obtido})}}{2(a_1 x^2 + b_1 x + c_1)} \quad (5.14)$$

Substituindo os coeficientes a_1, a_2 e a_3 pelos coeficientes encontrados na regressão polinomial do coeficiente a, b_1, b_2 e b_3 pelos coeficientes encontrados na regressão polinomial do coeficiente b, e c_1, c_2 e c_3 pelos coeficientes encontrados na regressão polinomial do coeficiente c, é possível determinar o valor corrigido de Z através do valor obtido pela triangulação.

Assim, foi analisado novamente o comportamento do erro aplicando a correção 5.14 para cada valor obtido ao longo das distâncias, gerando os comportamentos descritos na Figura 69, em paralelo ao comportamento do erro original contido na Figura 65:

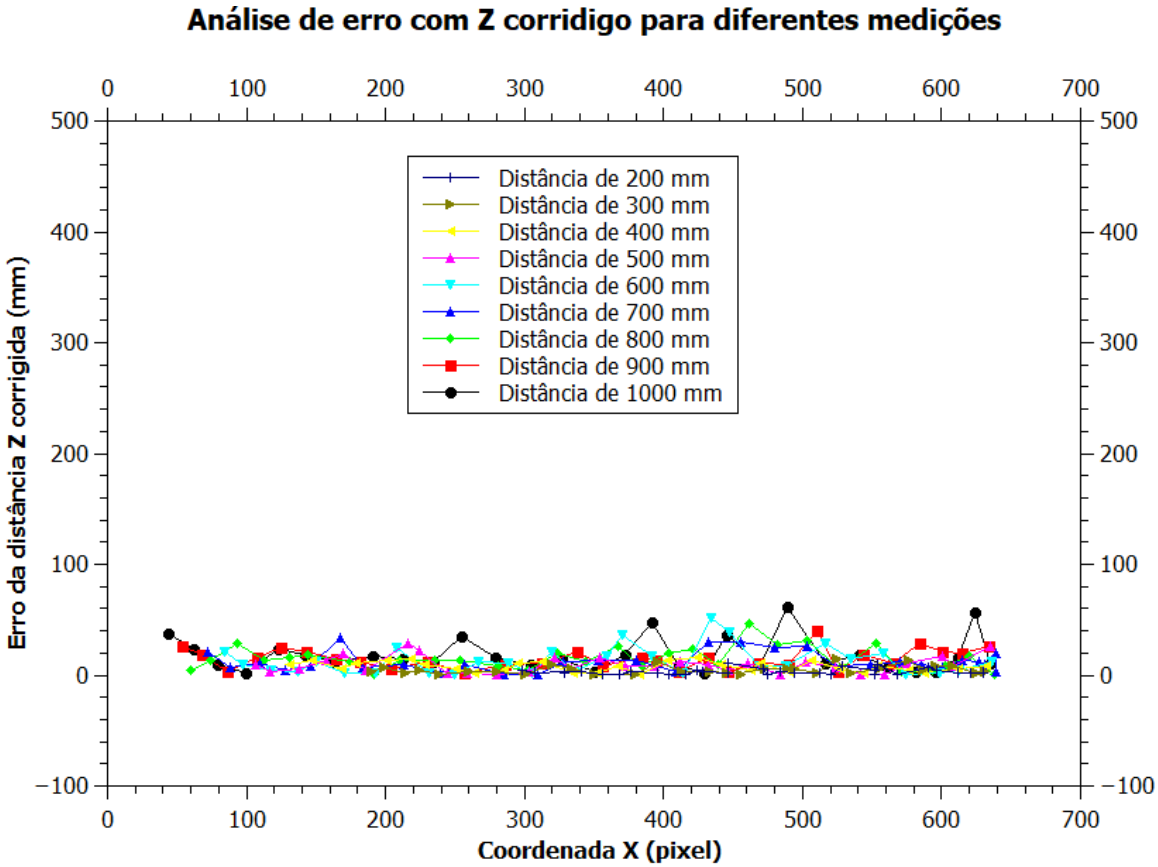


Figura 69 – Erro corrigido para todas as distâncias medidas variando X.

Como pode ser notado na Figura 69, o erro foi consideravelmente reduzido, principalmente se comparado aos valores nas bordas laterais da Figura 65. O erro médio de Z corrigido foi de 11,04mm, o que corresponde a uma redução de 9,48% para 1,82% de erro percentual médio.

5.1.2.2.3 Análise de distorção (medições verticais)

O mesmo processo adotado na seção anterior foi aplicado para analisar as distorções latitudinais presentes na visão estéreo, pois foi observado que, à medida que o ponto laser se aproximava das bordas superior e inferior do frame, os resultados obtidos pela triangulação sofreram consideráveis variações.

Portanto, o intuito deste experimento foi de verificar a variação do resultado da distância Z obtida em função da coordenada Y no plano da imagem. Para realizar esta análise, foi aplicado o método da triangulação descrito anteriormente, variando a posição Y do laser no frame da tela, de 0 a 480 px. Como a resolução vertical é menor do que a horizontal, percebeu-se que a variação vertical foi menor, o que resultou em menos amostras por distância e maiores discrepâncias entre cada medida.

O teste foi feito para diferentes distâncias, para que fosse possível verificar também se os parâmetros de distorção variam com a distância, gerando assim uma análise mais complexa. Primeiramente foram feitas análises isoladas para cada distâncias de 20, 30, 40, 50, 60, 70, 80, 90 e 100 cm, variando somente o valor da coordenada Y do ponto laser no frame da imagem.

As análises foram feitas em função do erro da distância obtida (distância obtida menos distância real) e da coordenada Y em pixel do ponto laser. Os resultados podem ser verificados nos gráficos em anexo na Seção Apêndice A. Como exemplo, é apresentado o comportamento obtido para a distância de 200 mm.

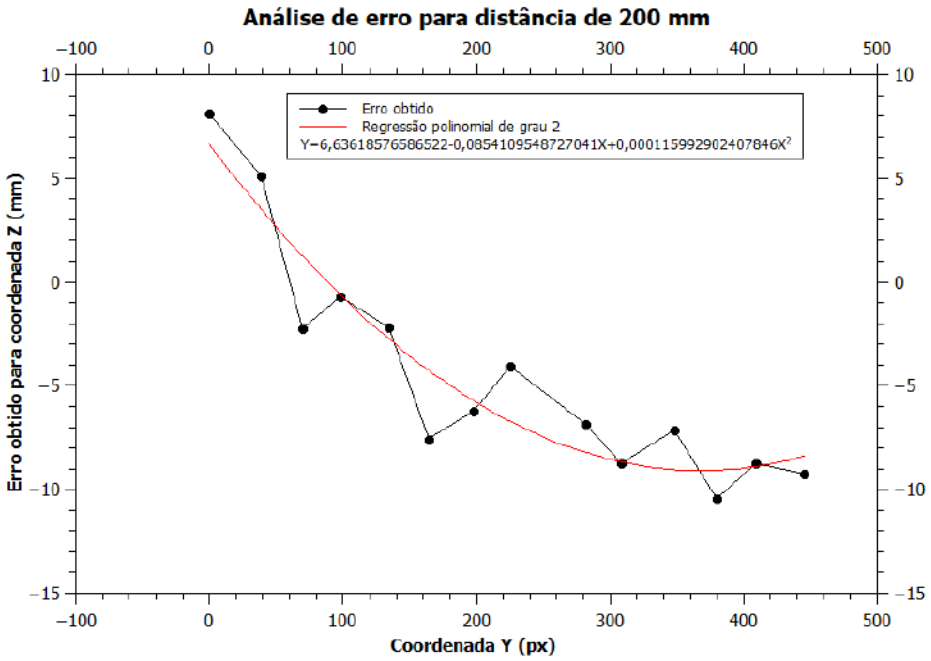


Figura 70 – Gráfico do erro para distância de 200 mm e regressão polinomial associada

O gráfico com resultados condensados é apresentado a seguir:

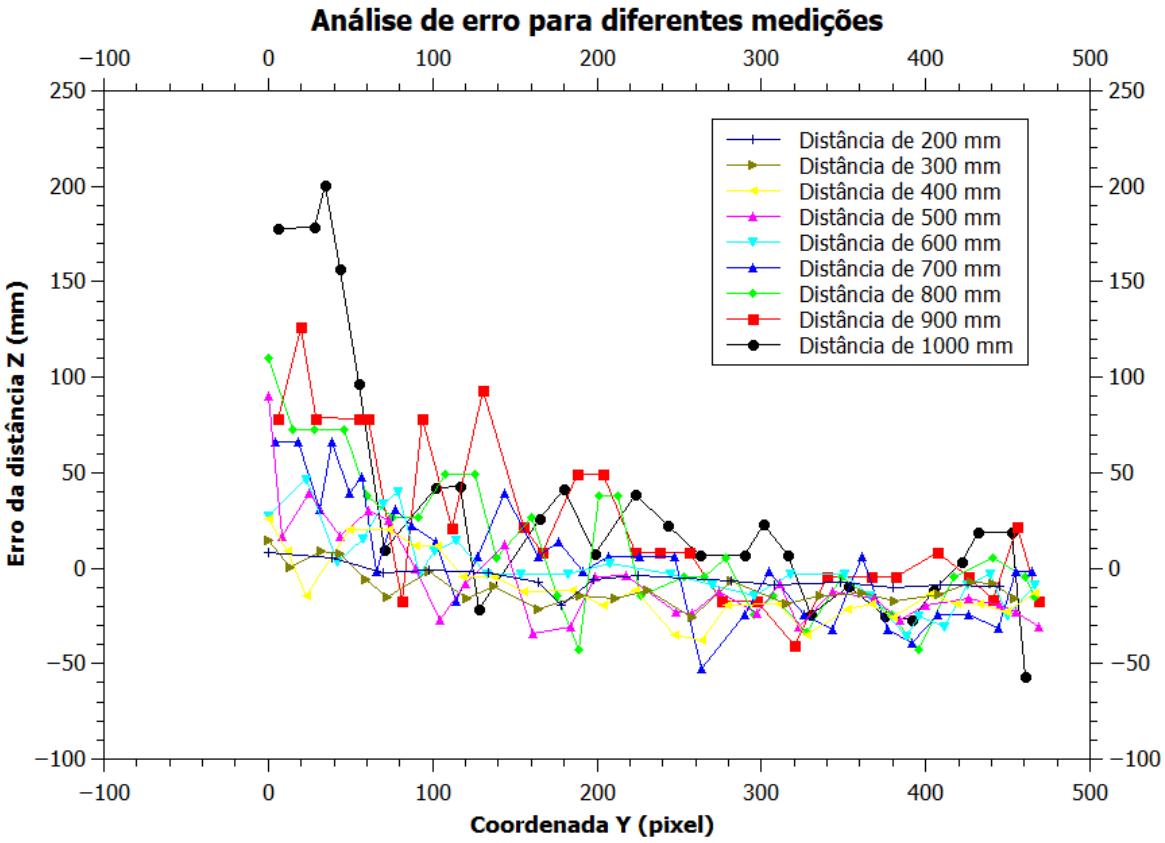


Figura 71 – Erro para todas as distâncias.

Como é possível notar, o erro apresenta uma tendência de ser mais acentuado na região superior da imagem (entre 0 e 70 pixels), e vai decaindo até atingir valores majoritariamente negativos. Além disso, é notável a alta discrepância presente entre medições consecutivas. O erro médio de todas as medições deste teste foi de 16,95mm, que corresponde a um percentual médio de 3,34%.

Foi feito o mesmo processo descrito na subseção 5.1.2.2.2 de aplicar o inverso do comportamento descrito pelas regressões polinomiais do erro na distância calculada pela triangulação, para contornar o problema de distorção das lentes, de forma que as suposições e deduções realizadas em 5.10 e 5.11 se mantenham, alterando apenas a variável de entrada de X para Y, resultando na Equação 5.15:

$$Z_{real} = \frac{-1 - a_2y^2 + b_2y - c_2 \pm \sqrt{(1 + a_2y^2 + b_2y + c_2)^2 - 4(a_1y^2 + b_1y + c_1)(a_3y^2 + b_3y + c_3 - Z_{obtido})}}{2(a_1y^2 + b_1y + c_1)} \tag{5.15}$$

Aplicando a correção 5.15, foi analisado novamente o comportamento do erro para todas as distância, resultando no gráfico da Figura 72, em paralelo ao gráfico da Figura 71:

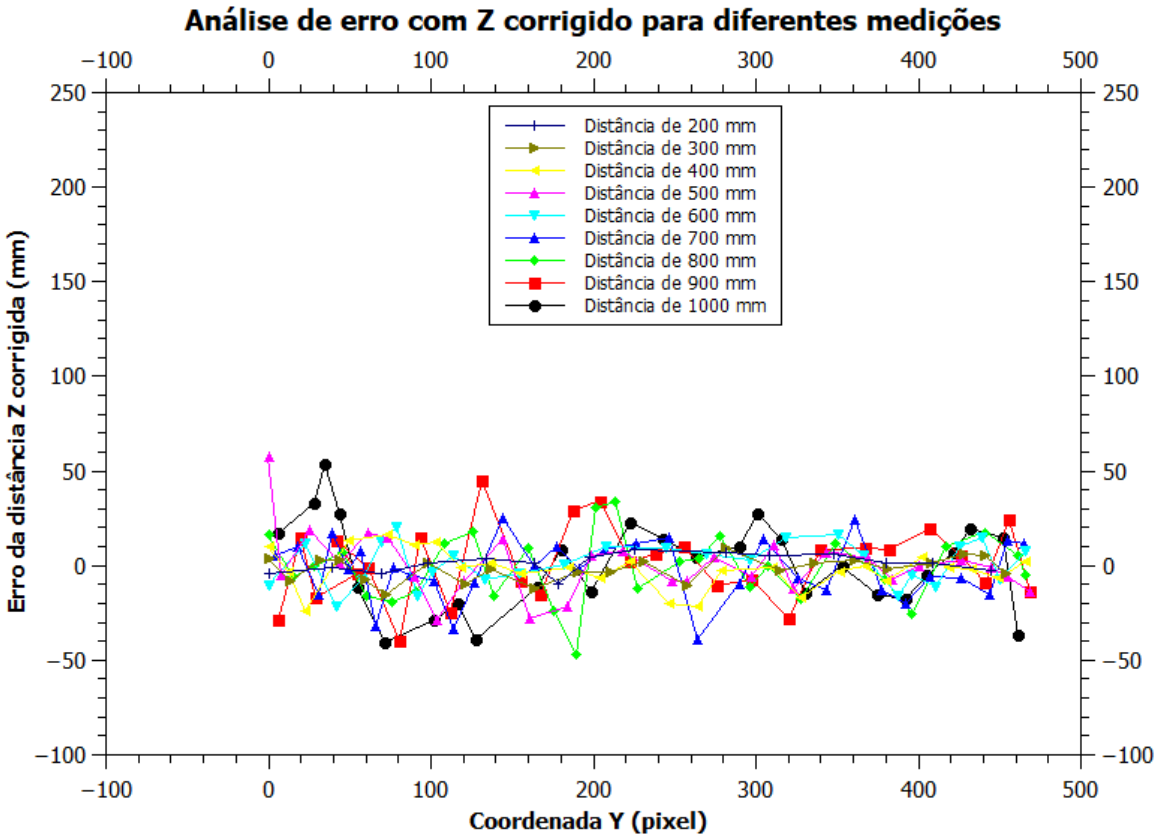


Figura 72 – Erro corrigido para todas as distâncias medidas variando Y.

Pode-se observar que o erro foi reduzido, se comparado com a Figura 71, principalmente para as distâncias entre 200 a 700 mm, porém com resultados não satisfatórios como obtidos na correção do eixo X (Figura 69). O erro médio das medições para Z corrigido foi de 9,15mm, que corresponde a uma redução de 3,34% para 1,54% de erro percentual médio. Apesar do erro médio ter sido reduzido consideravelmente, ainda é perceptível a presença de grandes discrepâncias entre as medições nos pixels iniciais da coordenada Y, sobretudo para distâncias maiores, como 900 e 1000 mm.

5.1.3 Visão estéreo - Abordagem com ferramentas computacionais

Em paralelo ao desenvolvimento empírico da visão estéreo e cálculo das coordenadas 3D, foram feitas também análises da visão estéreo com recursos computacionais e uso de bibliotecas. A vantagem de utilizar estas ferramentas é que elas proveem recursos para calibração de sistema estéreo, retificação de par estéreo, remoção de distorção das lentes, entre outros.

5.1.3.1 Matlab

A primeira ferramenta computacional explorada foi a *ToolBox* do Matlab destinada a visão computacional, denominada *Image Processing & Computer Vision*, no qual foram utilizados os aplicativos *Stereo Camera Calibrator* e *Color Thresholder*.

O aplicativo *Stereo Camera Calibrator* foi utilizado para calibração do sistema estéreo e obtenção dos parâmetros intrínsecos e extrínsecos das câmeras. A calibração é feita a partir da análise

de vários pares de imagens, capturadas simultaneamente por cada câmera, de um mesmo padrão previamente conhecido. Para isto, foi utilizada uma malha quadriculada padrão xadrez, na proporção 11x8, com cada quadrante possuindo dimensões de 25x25 mm, como pode ser visualizado na Figura 73:

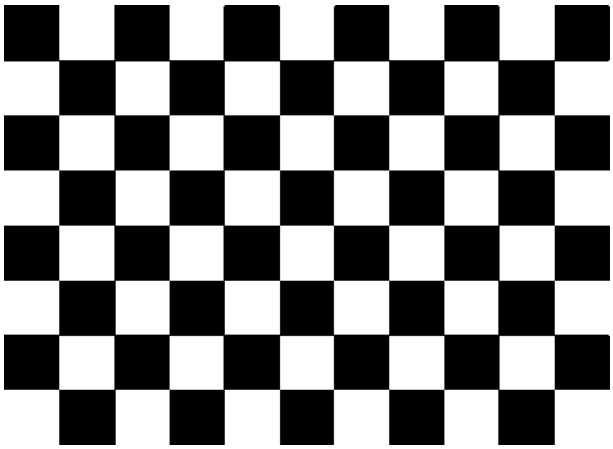


Figura 73 – Malha quadriculada utilizada para calibração do sistema estéreo.

Foram feitas capturas da malha quadriculada a partir de diferentes ângulos e direções, para que o programa consiga estimar corretamente os parâmetros e distorções presentes nas lentes, analisando a deformação sofrida pelo padrão original. Em seguida, as imagens foram processadas no aplicativo (Figura 74).

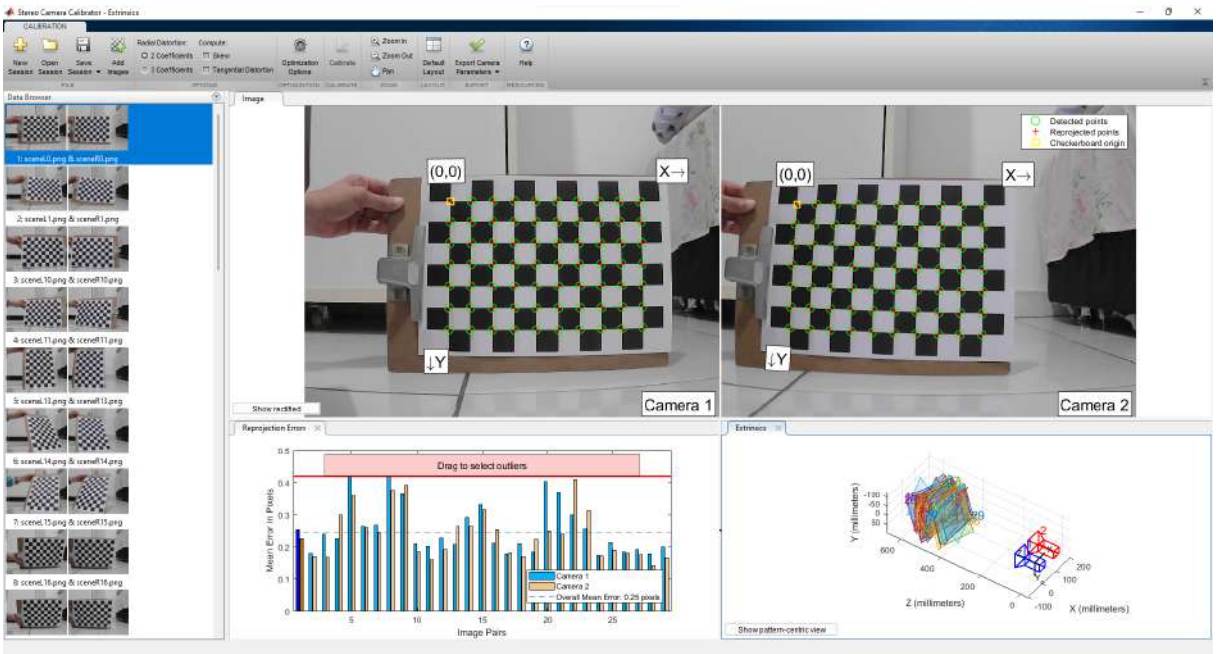


Figura 74 – Layout do aplicativo de calibração do Matlab.

Há ainda as funcionalidades de analisar os erros associados às estimativas, reconstituição do processo de captura com os parâmetros extrínsecos obtidos, e aplicação de retificação dos pares de imagens estéreo, como mostrado nas Figuras 75 e 76:

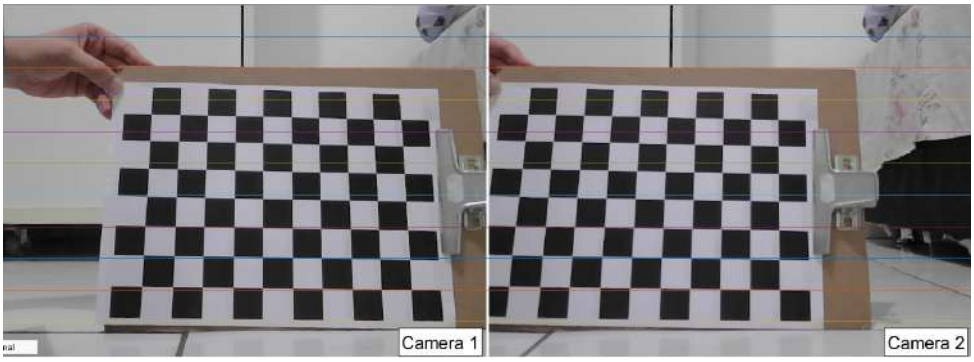


Figura 75 – Aplicação do processo de retificação.

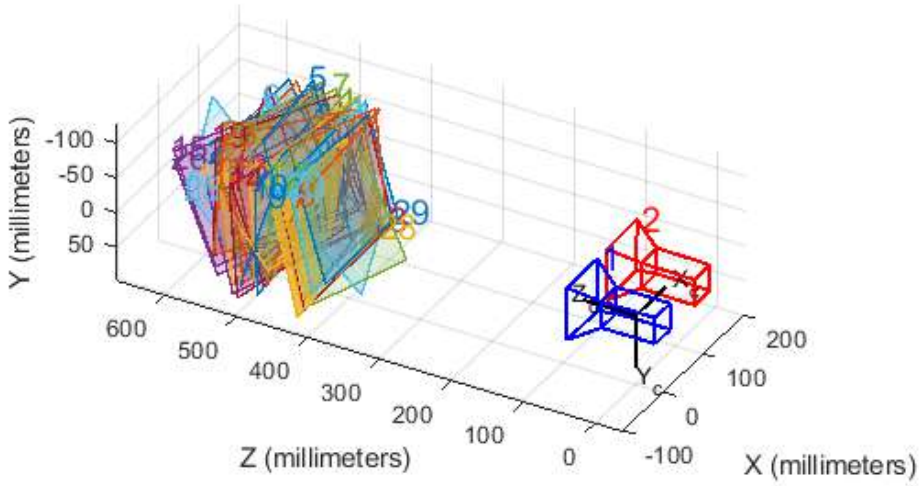


Figura 76 – Reconstituição do processo de calibração a partir dos parâmetros extrínsecos encontrados.

Os principais parâmetros obtidos pelo Matlab foram:

1. Matriz de rotação da câmera 2 em relação à câmera 1 (parâmetro extrínseco):

$$R = \begin{bmatrix} 0{,}99992 & 0{,}01082 & 0{,}00505 \\ -0{,}01071 & 0{,}99973 & -0{,}02028 \\ -0{,}00528 & 0{,}02023 & 0{,}99978 \end{bmatrix} \tag{5.16}$$

2. Vetor de translação da câmera 2 em relação à câmera 1 (parâmetro extrínseco):

$$T = \begin{bmatrix} 98{,}42315 \\ 1{,}48933 \\ 0{,}84084 \end{bmatrix} \tag{5.17}$$

3. Matriz Fundamental do sistema (parâmetro extrínseco):

$$F = \begin{bmatrix} -3{,}05148e^{-09} & -1{,}70695e^{-06} & 0{,}00248 \\ 6{,}71125^{-07} & 3{,}90419e^{-06} & -0{,}13940 \\ -0{,}00074 & 0{,}13723 & 0{,}97301 \end{bmatrix} \tag{5.18}$$

4. Matriz Essencial do sistema (parâmetro extrínseco):

$$E = \begin{bmatrix} -0{,}00156 & -0{,}87085 & 1{,}47199 \\ 0{,}34295 & 1{,}98767 & -98{,}406079 \\ -0{,}42454 & 98{,}41320 & 1{,}99906 \end{bmatrix} \tag{5.19}$$

5. Matriz de parâmetros intrínsecos da câmera 1:

$$A_1 = \begin{bmatrix} 717{,}91424 & 0 & 311{,}98989 \\ 0 & 715{,}23339 & 242{,}195283 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

6. Matriz de parâmetros intrínsecos da câmera 2:

$$A_2 = \begin{bmatrix} 713{,}29522 & 0 & 328{,}94801 \\ 0 & 711{,}81130 & 236{,}22029 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.21}$$

Com posse dos parâmetros de calibração das câmeras e do sistema estéreo, implementou-se o código para detecção do ponto laser e triangulação dos pontos para obtenção das coordenadas 3D. É interessante observar as diferenças obtidas nos parâmetro de distância focal contidos nas matrizes 5.20 e 5.21, que foi de 717,91 px e 713,30 px aproximadamente, em comparação com o valor obtido experimentalmente de 644,45 px.

A detecção do ponto laser foi feita com o auxílio da ferramenta *Color Thresholder*, onde foram analisadas imagens capturadas do ponto laser em diferentes ambientes, para melhor identificação de seu intervalo no sistema HSV. Exemplo do uso desta ferramenta pode ser observado nas Figuras 77 e 78:

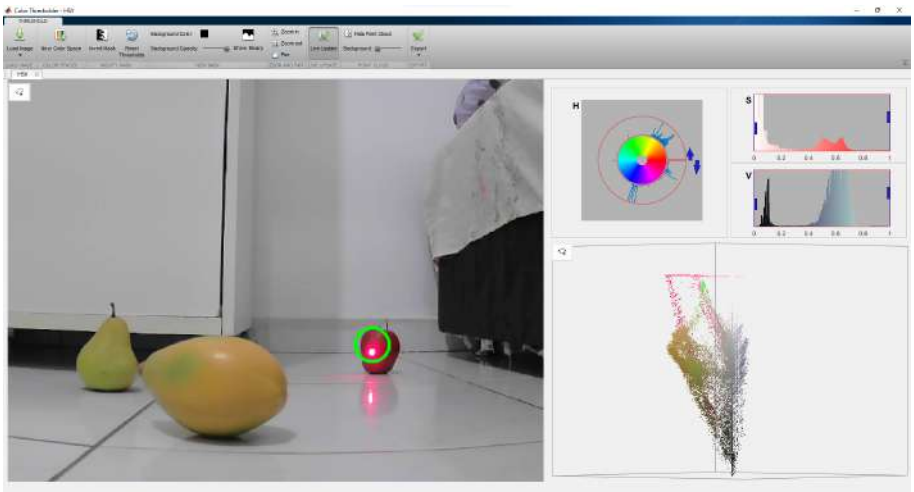


Figura 77 – Layout do aplicativo de threshold.

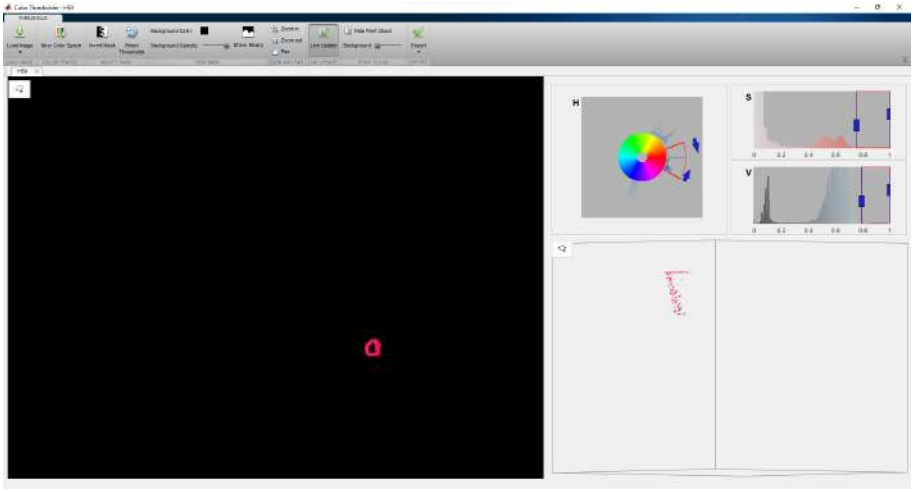


Figura 78 – Aplicação do threshold para isolar somente o ponto laser.

A partir do threshold encontrado com o auxílio da ferramenta, foi implementado um código para calcular a posição central do ponto laser em cada câmera e em seguida, estes pontos foram usados como parâmetros de entrada na função triangulate(), que retorna as coordenadas 3D correspondentes ao sistema de coordenadas mundiais, em milímetros. O Matlab também considera a origem do sistema de coordenadas global no centro óptico da câmera 1.

Os resultados alcançados no cálculo das coordenadas 3D pelo software foram ótimos, pois o Matlab conta com recursos de estimativa de erros, análise numérica detalhada dos parâmetros de distorção e funções próprias de triangulação com maior acurácia, porém ainda apresentou erro de $\pm 50mm$ para testes realizados com distância previamente conhecidas, como o próprio objeto da Figura 77 que estava a uma distância de 93 cm da câmera e o programa calculou uma distância de 88,72 cm.

5.1.3.2 OpenCV

Além do Matlab, a própria biblioteca do OpenCV também foi testada nas suas funções de visão estéreo. O mesmo padrão de malha quadriculada da Figura 73 foi utilizado no processo de calibração das câmeras, que funciona de maneira diferente, pois primeiro é feita a calibração individual de cada câmera através da função cv2.calibrateCamera(), que retorna os parâmetros intrínsecos como distância focal e centros ópticos, e em seguida aplica-se a função cv2.getOptimalNewCameraMatrix() que retorna uma nova matriz intrínseca da câmera com base em uma parâmetro de escala livre, garantindo menos perda de informação durante o processo de relacionamento entre os pixels das imagens. O sistema de visão estéreo é calibrado pela função cv2.stereoCalibrate(), que retorna os parâmetros extrínsecos como matrizes de rotação e translação, matriz fundamental e essencial. Com os parâmetros de calibração obtidos, é possível fazer a retificação dos pares estéreos através da função cv.stereoRectify(), para em seguida remover as distorções das lentes das câmeras, através da função cv2.undistort() [1].

[1] É possível ainda utilizar a função cv2.initUndistortRectifyMap() que realiza um mapeamento da imagem distorcida para a imagem não distorcida, embora este método seja mais complexo

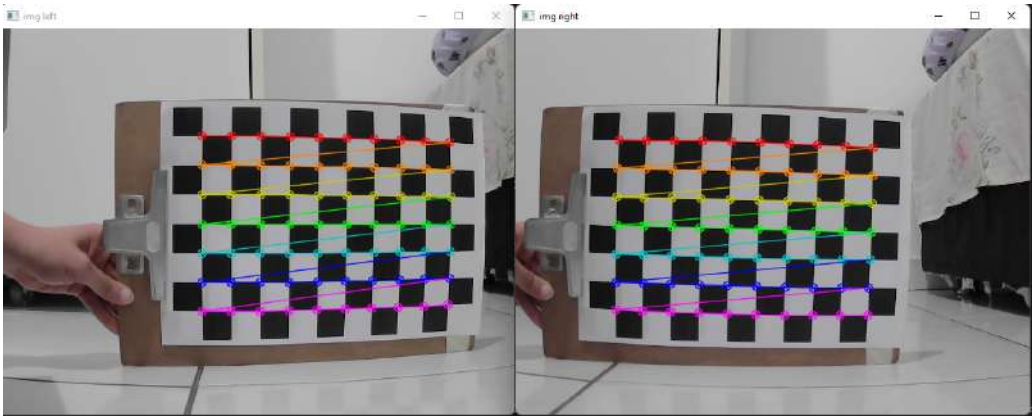


Figura 79 – Processo de calibração pelo OpenCV.

Os principais resultados obtidos pela biblioteca OpenCV foram:

1. Matriz de rotação da câmera 2 em relação à câmera 1 (parâmetro extrínseco):

$$R = \begin{bmatrix} 0.99987662 & -0.01102189 & 0.01119174 \\ 0.01077042 & 0.99969363 & 0.02228569 \\ -0.01143395 & -0.0221624 & 0.999689 \end{bmatrix} \tag{5.22}$$

2. Vetor de translação da câmera 2 em relação à câmera 1 (parâmetro extrínseco):

$$T = \begin{bmatrix} 88.34723445 \\ 2.22646369 \\ -15.00391986 \end{bmatrix} \tag{5.23}$$

3. Matriz Fundamental do sistema (parâmetro extrínseco):

$$F = \begin{bmatrix} 1.45948193e^{-07} & 1.60927434e^{-05} & -2.15575355e^{-03} \\ -1.50575695e^{-05} & 2.29446710e^{-06} & -5.71344127e^{-02} \\ 2.63016639e^{-03} & 5.30071587e^{-02} & 1.00000000 \end{bmatrix} \tag{5.24}$$

4. Matriz Essencial do sistema (parâmetro extrínseco):

$$E = \begin{bmatrix} 0.13614129 & 14.94997926 & 2.56014402 \\ -13.99191126 & 2.12335867 & -88.48767832 \\ -1.27465196 & 88.34470699 & 1.94396143 \end{bmatrix} \tag{5.25}$$

5. Matriz de parâmetros intrínsecos da câmera 1:

$$A_1 = \begin{bmatrix} 715.4795672 & 0 & 307.22497556 \\ 0 & 712.70393467 & 241.61549085 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.26}$$

6. Matriz de parâmetros intrínsecos da câmera 2:

$$A_2 = \begin{bmatrix} 715.74691553 & 0 & 327.60719381 \\ 0 & 714.16478857 & 235.35977574 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.27}$$

7. Matriz de parâmetros intrínsecos da função ótima[2] da câmera 1:

$$A_1 = \begin{bmatrix} 642.72399902 & 0 & 307.00979 \\ 0 & 640.09417725 & 240.78820052 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.28}$$

8. Matriz de parâmetros intrínsecos da função ótima da câmera 2:

$$A_2 = \begin{bmatrix} 618.59643555 & 0 & 328.28781827 \\ 0 & 616.22375488 & 233.99351413 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.29}$$

Como pode ser observado, os resultados das matrizes 5.22, 5.23, 5.26 e 5.27 foram próximos aos calculados pelo Matlab, correspondendo às matrizes 5.16, 5.17, 5.20 e 5.21, respectivamente. As matrizes fundamental e essencial obtidas pelo OpenCV se diferiram um pouco das obtidas pelo Matlab.

É interessante notar que as matrizes de parâmetros intrínsecos ótimos 5.28 e 5.29, calculadas pela função cv2.getOptimalNewCameraMatrix() que garante uma menor perda de informação no relacionamento entre as imagens, retornaram valores de distâncias focais mais próximas dos valores obtidos experimentalmente para f_x f_y.

Apesar de ter gerado resultados satisfatórios, as funções de visão estéreo do OpenCV não foram triviais de serem utilizadas e ainda contavam com pouquíssima documentação no site oficial da plataforma. Isso dificultou o uso da ferramenta, e não foi possível utilizar adequadamente a função cv2.triangulatePoints() pois retornavam resultados inconsistentes. Ao invés desta função, foi implementado o código da tringulação usando a disparidade entre as imagens, como descrito em 5.1.2.1.

5.2 Comunicação com o robô

Com posse das coordenadas tridimensionais do objeto, é possível enviá-las para o braço robótico, para que este implemente sua cinemática inversa e alcance o alvo. Porém, antes de enviar os comandos finais para o robô, é necessário fazer uma transformação do sistema de coordenadas da câmera para o sistema de coordenadas do robô. Para isto, é feito medições no sistema estéreo para encontrar as coordenadas de um ponto de referência conhecido pelo sistema de coordenadas do robô.

[2] Aplicação da função cv2.getOptimalNewCameraMatrix().

Assim, é possível fazer a relação entre os dois sistemas através de matrizes de transformação, com as operações de rotação e translação, conforme indica a Figura 80:

Figura 80 – Indicação dos sistemas de coordenadas da câmera e do robô.

No caso implementado, o eixo X da câmera foi alinhado com o eixo X do robô. Para garantir isto, foram feitas aquisições pelo sistema estéreo da coordenada Z de dois pontos conhecidos na base do robô, e o seu alinhamento foi feito de forma que as duas coordenadas Z se coincidam, garantindo que não haja deslocamento entre as direções do eixo X da câmera e o eixo X do robô.

Pode-se notar que a relação entre os dois sistemas de coordenadas ocorre por meio de uma rotação em torno do eixo X de 90°, trocando as direções dos eixos Y e Z da câmera para ficarem iguais as direções do sistema do robô. Em seguida ajusta-se os deslocamentos ocorridos em cada eixo baseado no ponto de referência utilizado. A matriz de transformação homogênea relativa a estas transformações é:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & cos\theta & -sin\theta & b \\ 0 & sin\theta & cos\theta & c \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

onde a, b e c são os valores correspondentes às translações em cada eixo e θ é o ângulo de rotação em torno do eixo X, no caso de 90°.

Como mencionado anteriormente, o braço robótico usado para testes foi o *Pégaso*. O robô está preparado para receber os comandos de movimento através de requisições HTTP do tipo GET, pois ele conta com um módulo Wi-Fi. Assim, os parâmetros X, Y e Z podem ser inseridos dentro da requisição. Uma vez ajustada as coordenadas do alvo para o sistema de coordenadas do robô, basta realizar a requisição passando-as no formato:

```
http://IP do Robô/Luiza?x&y&z
```

Para os ensaios de teste com o braço robótico, foi utilizado um imã como efetuador terminal objetivando atrair um objeto metálico localizado na posição destino. Neste cenário, duas compensações necessitaram ser feitas em código, uma para considerar as dimensões do efetuador e outra para posicionar o robô no centro do objeto, já que a visão estereo estima as coordenadas à partir da face frontal do objeto, mais próximo à câmera.

5.3 Sistema de acionamento do ponto laser

O circuito de acionamento do laser é simples, constituído por um diodo laser, 2 baterias, resistores e uma chave. As especificações do diodo laser e baterias utilizadas estão listadas nas Tabelas 9 e 10:

Tabela 9 – Especficações do diodo laser.

Cor	Vermelho
Comprimento de Onda	650nm
Formato de feixe de laser	Ponto
Potência	2-5mW
Tensão de Operação	5V
Temperatura de Operação	-10 á 40ºC
Dimensões	6x10mm
Comprimento dos fios (aprox.)	8cm

São usados ainda 3 resistores de 470 Ohm em paralelo.

A chave responsável pelo acionamento do sistema foi planejada para ser originalmente um sensor de sopro e sucção, cujo funcionamento se dá da seguinte forma: entre dois canais dentro de um tubo, é posicionada uma membrana constituída de material com propriedades de deformação

Tabela 10 – Especificações das baterias.

Tensão	3V
Dimensões	1,6x20,0mm
Peso	3,16g
Validade	5 Anos

elástica. À este tubo fica conectado um duto de ar (canudo) por onde o usuário pode soprar, fechando o circuito do canal 2, ou succionar, fechando o circuito do canal 1, conforme ilustra a Figura 3.

Assim, a ideia era que o usuário acionasse o ponto laser soprando o duto. Porém, para fins de testes e simplicidade, o circuito foi implementado com o uso de uma mini chave gangorra. Desta forma, o circuito assume a configuração ilustrada na Figura 81:

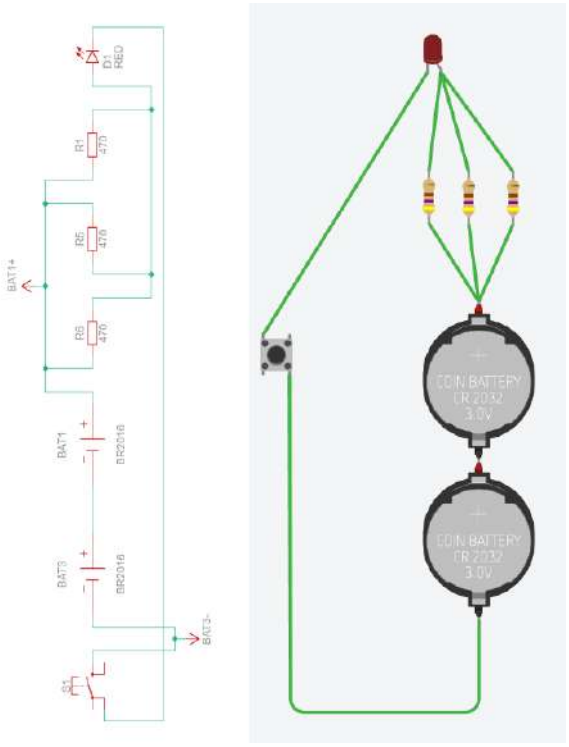


Figura 81 – Ilustração do circuito de acionamento do laser.

5.4 Modelagem do suporte laser

Como mencionado na introdução deste capítulo, a modelagem do suporte foi realizada no software Fusion 360°, e sua concepção atentou-se ao perfil do usuário e suas limitações. Uma vez que o público alvo deste projeto são pessoas tetraplégicas, foi considerado para esta solução o grau de lesão moderado, em que os usuários possuem movimentação do pescoço. Desta forma, é possível trabalhar com a ideia de que o usuário pode indicar o ponto laser para um objeto variando a posição de sua cabeça. Dadas as limitações, notou-se portanto que o suporte laser deve ficar posicionado na cabeça, de preferência próximo do campo de visão do usuário.

Foram feitas pesquisas de referências de produtos e suportes já existentes no mercado que

atendam aos requisitos. Os modelos de suporte encontrados, mais plausíveis para o projeto, foram o óculos Google Glass, fones de ouvido que passam por trás da cabeça e suporte de lanternas para cabeça com faixa.

Por maior simplicidade e possibilidade de realização de testes físico adaptados, foi escolhido como referência o modelo de suporte similar ao óculos Google Glass. Desta forma, foi modelado um suporte com estrutura básica de óculos, e acrescido em sua lateral uma caixa para comportar o circuito de acionamento do laser.

A modelagem foi feita baseada nas dimensões de um óculos padrão e dos componentes citados na subseção 5.3, e seu resultado pode ser conferido nas Figuras 82, 83 e 84.

Figura 82 – Modelagem do suporte laser modelo óculos. Vista frontal.

Figura 83 – Modelagem do suporte laser modelo óculos. Vista isométrica.

Figura 84 – Modelagem do suporte laser modelo óculos. Detalhes internos.

6 Resultados e discussões

Este capítulo traz os resultados alcançados das implementações realizadas e documentadas no Capítulo 5, acerca do sistema de visão estéreo. Na seção 5.1 foi apresentada três abordagens executadas para a implementação da visão estéreo: ajuste polinomial de erros, retratada na subseção 5.1.2; outra utilizando a ferramenta computacional Matlab e aplicando correções de distorções, descritas na subseção 5.1.3.1; e por fim, a última utilizando a biblioteca OpenCV e também aplicando correções de distorções e retificação.

Para fins de testes de validação dos sistemas estéreo, foi montado um cenário controlado que replica a utilização do sistema e do braço robótico *Pégaso*, com objetos estrategicamente posicionados dentro de seu volume de trabalho e com suas coordenadas 3D previamente conhecidas. O objetivo deste teste foi de verificar qual abordagem melhor atendeu às especificações de encontrar as coordenadas 3D dos objetos. As Figuras 85, 87 e 87 retratam o cenário de teste montado.

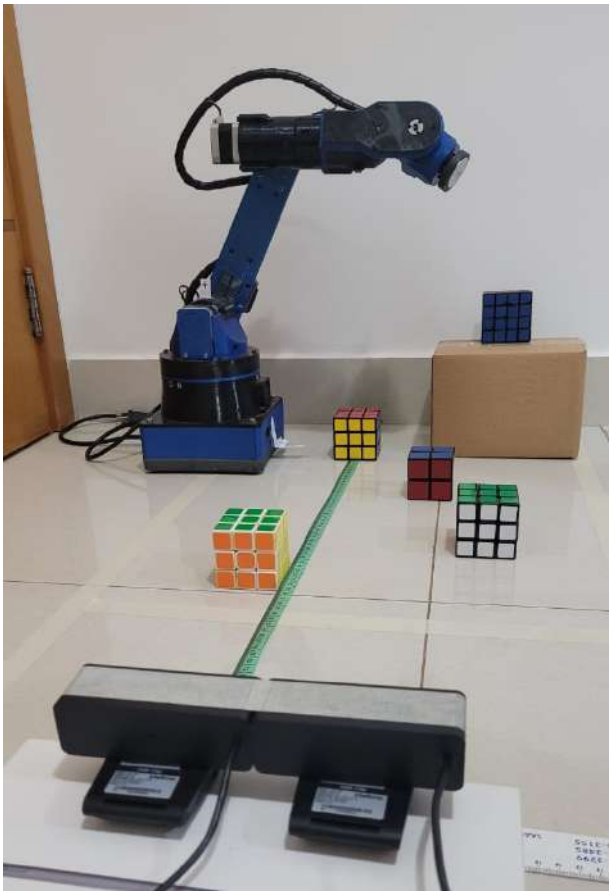


Figura 85 – Cenário montado para testes.

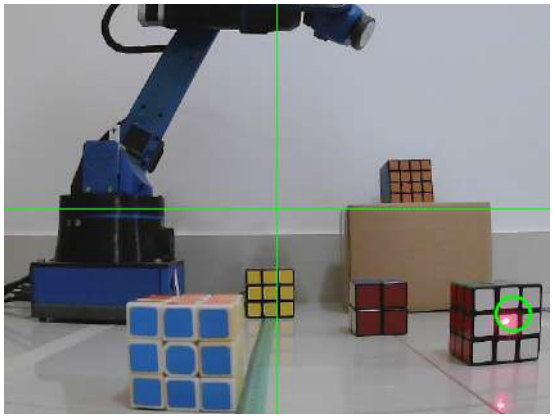


Figura 86 – Cenário de teste pela câmera 1.

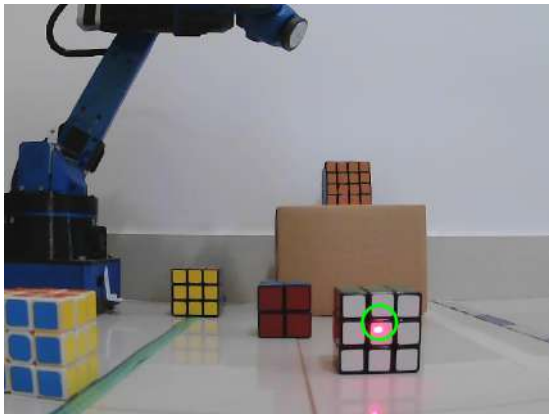


Figura 87 – Cenário de teste pela câmera 2.

O cenário montado era constituído por 5 cubos mágicos dispostos em regiões diferentes. As coordenadas reais de cada cubo foram medidas em relação ao centro da câmera esquerda, considerada esta a origem do sistema de coordenadas global, e estas medidas foram realizadas com o auxílio de uma fita métrica, com incerteza de 0,5mm, até o centro de cada cubo. Foram tracejadas linhas centrais de orientação no frame da câmera esquerda, para que pudesse ser medido o valor de sua coordenada Y em coordenada global, para usar de referência nas medições das coordenadas Y dos cubos. Assim, a linha horizontal central da câmera foi ajustada de forma que o valor de Y real se manteve tanto para objetos próximos, quanto para objetos distantes, eliminando assim, a interferência de inclinações no eixo X no sistema estéreo. O cubo de face amarela foi posicionado de tal forma que sua coordenada X em 0 ficasse localizada cima da linha preta que delimitam os pequenos quadrantes da direita, por ser um local de fácil medição.

As medições das coordenadas X para os demais cubos foram feitas a partir do cubo amarelo. Os valores encontram-se na Tabela 11, onde cada cubo é identificado pela cor de sua face frontal, conforme se apresentam nas Figuras 86 e 87.

Tabela 11 – Coordenadas conhecidas dos cubos do cenário.

Cor da face	Coordenada X (mm)	Coordenada Y (mm)	Coordenada Z (mm)
Azul	-30	-40	300
Branco	160	-40	400
Vermelho	95	-40	550
Amarelo	0	-40	700
Laranja	230	110	800

Para a abordagem empírica implementada, em que o cálculo das coordenadas 3D foram baseados no método da triangulação geométrica e os erros associados foram analisados e corrigidos experimentalmente, os resultados obtidos para o cálculo das coordenadas 3D dos cubos mágicos são apresentados na Tabela 12.

Para a abordagem empregando a ferramenta Matlab, em que utilizou-se sua *Toolbox* de visão estéreo para realizar calibração das câmeras, retificação, remoção de distorção e triangulação, os resultados alcançados no experimento estão na Tabela 13. Vale salientar que os pares de imagens estéreos utilizados para todos os testes deste capítulo foram exatamente os mesmos, ou seja, as

Tabela 12 – Coordenadas encontradas pelo método empírico.

Cor da face	Coordenada X (mm)	Coordenada Y (mm)	Coordenada Z (mm)
Azul	-34,2929	-55,0101	296,7643
Branco	173,8187	-55,3355	419,5340
Vermelho	109,4358	-45,5982	548,3384
Amarelo	4,42637	-55,42857	686,9293
Laranja	244,9871	110,6794	801,4176

coordenadas em pixel do ponto laser sobre cada cubo são exatamente as mesmas para todas as abordagens implementadas.

Tabela 13 – Coordenadas encontradas pelo software Matlab.

Cor da face	Coordenada X (mm)	Coordenada Y (mm)	Coordenada Z (mm)
Azul	-36,6029	-34,8896	309,2011
Branco	164,4938	-55,08072	404,4101
Vermelho	108,6501	-42,2148	545,4006
Amarelo	1,9483	-49,7585	692,5058
Laranja	217,8382	95,7672	727,3158

Por fim, para a abordagem empregando a biblioteca OpenCV, em que forma feitos os processos de calibração das câmeras, calibração do sistema estéreo, remoção de distorções e retificação, os resultados obtidos pela triangulação estão tabelados em 14:

Tabela 14 – Coordenadas encontradas pela biblioteca OpenCV.

Cor da face	Coordenada X (mm)	Coordenada Y (mm)	Coordenada Z (mm)
Azul	-43,2804	-38,5244	305,6839
Branco	202,6171	-60,9375	489,5718
Vermelho	155,1136	-64,2614	712,1045
Amarelo	9,2857	-85,1190	994,6857
Laranja	371,6038	171,0849	1182,3622

Como pode-se notar, os resultados alcançados pela solução empírica (manual) e pelo Matlab foram muito satisfatórios, sendo os dois possíveis de serem adotados para o projeto. Embora haja pequenas discrepâncias entre os dois resultados, ambos são praticáveis e levam o robô com sucesso ao objeto indicado, pois as dimensões reais do objeto compensam suas discrepâncias, visto que o objeto é bem maior do que o ponto laser usado como base de cálculo. Vale ressaltar, porém, que a abordagem empírica utiliza menos recursos computacionais, pois não realiza o processo de calcular as matrizes fundamental e essencial do sistema estéreo e os parâmetros intrínsecos e extrínsecos das câmeras, embora os parâmetros intrínsecos tenham sido obtidos experimentalmente.

Em contrapartida, os resultados alcançados pelo OpenCV foram inconsistentes com o esperado, de tal forma que não vale a comparação com as demais. Apenas o resultado da primeira medição foi consistente e conseguiria levar o braço robótico ao objeto. Razões disto pode ser devido a erros durante a implementação e uso das funções de visão estéreo da biblioteca, sobretudo porque não há documentação ou exemplos das funções de visão estéreo.

Na Tabela 15 é apresentada uma comparação entre as três abordagens implementadas, baseada no conceito de distância Euclidiana no espaço, apresentada na Equação 6.1, e no erro percentual, para que seja possível verificar qual abordagem atingiu a menor distância em relação ao ponto alvo. Ou seja, esta medida representa o erro entre as coordenadas medidas e a posição real dos objetos.

$$d = \sqrt{(x_{obtido} - x_{real})^2 + (y_{obtido} - y_{real})^2 + (z_{obtido} - z_{real})^2} \tag{6.1}$$

Tabela 15 – Análise entre as abordagens utilizadas para obtenção das coordenadas 3D. As dimensões estão em milímetros.

Cor da face	**Abordagem empírica**		**Matlab**		**OpenCV**	
	Distância	Errro per. %	Distância	Erro per. %	Distância	Erro per. %
Azul	15,9437	0,1238	12,4247	3,0153	14,5207	2,2979
Branco	28,4202	5,7340	16,3423	1,7054	101,3790	23,2674
Vermelho	15,5721	0,2560	14,5734	0,3321	174,5855	30,7473
Amarelo	20,6996	1,7066	12,4574	0,9767	298,2643	42,3912
Laranja	15,0693	0,6742	75,0564	8,8596	412,2909	49,0081
Média	**19,1410**	1,6989	**26,1708**	2,9778	**200,2081**	29,5424

Os dados da tabela corroboram com as análises realizadas, e é possível constatar ainda que a abordagem empírica manteve seus resultados mais próximos, em média, das coordenadas reais. Vale notar ainda que o Matlab forneceu ótimos resultados para pequenas distâncias, e divergiu somente no teste de maior distância. Como as dimensões dos cubos são de 55x55 mm, então os erros atingindos pela abordagem empírica ainda estão dentro das dimensões do objeto.

O sistema de visão computacional implementado apresenta algumas limitações. Uma delas é sua vulnerabilidade à variação de luz, visto que a luz do ambiente e possíveis objetos reflexivos na cena podem interferir na detecção do ponto laser. Para tratar estes problemas pode ser aplicado subtração de imagens para analisar a variação da luz. Outra limitação é que as medições são feitas a partir da face frontal do objeto, necessitando, portanto, de uma compensação relativa as suas dimensões antes de enviar o comando para o robô.

7 Conclusão

Este projeto debruçou-se no estudo e desenvolvimento de uma interface de comando para braço robótico acoplado à cadeira de rodas, motivado pelo interesse de melhorar a qualidade de vida de pessoas tetraplégicas através da engenharia. Na pesquisa do estado da arte referente aos tipos de interface comumente empregadas em tecnologias assistivas para pessoas tetraplégicas, público alvo do trabalho, constatou-se que a visão computacional seria a interface que melhor se adequaria aos objetivos e escopo do projeto, pois esta confere grande autonomia ao sistema, demandando pouco esforço do usuário.

Em virtude disto, foi implementado um sistema de visão estéreo para obtenção das coordenadas espaciais de objetos, de forma que o resultado possa ser enviado para um braço robótico e este, por sua vez, possa buscá-los para o usuário.

Três abordagens para o sistema de visão estéreo foram exploradas, sendo a primeira totalmente empírica, com análise de erros e ajuste de correção, a segunda utilizando a ferramenta computacional Matlab e a terceira utilizando as funções de visão estéreo da biblioteca OpenCV.

A primeira abordagem objetivava corrigir os erros associados à distorção através da análise empírica dos resultados obtidos. A partir dela foi possível modelar o erro de distorção em funções bidimensionais que foram aplicadas de maneira compensatória para a correção dos valores medidos.

A segunda abordagem fez uso da *Toolbox Stereo Vision* na ferramenta computacional Matlab para realizar o processo de calibração das câmeras, obtenção dos parâmetros intrínsecos e extrínsecos e das matrizes fundamental e essencial do sistema, seguindo a teoria de geometria epipolar. Em seguida foram aplicadas a retificação, remoção de distorção dos pares de imagens estéreo e função de triangulação de pontos tridimensionais.

A terceira abordagem seguiu os mesmos processos da segunda, porém utilizando as funções da biblioteca OpenCV para visão estéreo, que se mostrou mais difícil de ser implementar, tendo em vista a pouca documentação presente no site da plataforma para as funções estéreo disponíveis.

A abordagem empírica de correção dos erros se destacou quando comparada com as técnicas de correção de distorção do Matlab e OpenCV por obter um erro de aquisição mais baixo, com magnitude inferior à 2cm. Além disto, ela conta com menos recursos computacionais.

Na solução de interface proposta, a ideia era utilizar um ponto laser como indicador de objeto requisitado, e para isto, foi desenvolvido também um suporte laser para que o usuário tetraplégico possa operá-lo, indicando ao robô o objeto de seu interesse. Desta forma, o terceiro objetivo específico foi alcançado.

Diante dos resultados alcançados no desenvolvimento do presente trabalho, pode-se concluir que todos os objetivos propostos inicialmente foram alcançados, embora ainda haja espaço significativo para melhorias no projeto. Como sugestão para trabalhos futuros, sugere-se estudos sobre sistema de controle do efetuador terminal; aplicação de machine learning para identificação prévia

de objetos, assim o sistema pode verificar se o braço robótico suporta determinado objeto e fazer correções relativas às dimensões do objeto; implementação de funcionalidades padrão para WMRA, como beber água, abrir porta, pegar um objeto no chão, pentear o cabelo, etc.

Referências

ABRAFIN. Desordens da Função Neurológica e Fisioterapia. Brasília, DF, 2018. Disponível em: <http://abrafin.org.br/desordens-da-funcao-neurologica-e-fisioterapia/>. Acesso em: 16 set. 2021. Citado na p. 18.

ANDITEC. Joystick BJOY para Queixo. Lisboa, Portugal. Disponível em: <https://anditec.pt/produto/joystick-queixo/>. Citado na p. 22.

BACKES, A. R.; MESQUITA SÁ JUNIOR, J. J. de. **Introdução À Visão Computacional Usando MATLAB**. 1. ed. São Paulo, SP: Alta Books, 2016. Citado na p. 44.

BAREA, R.; LÓPEZ, E.; MAZO, M. System for assisted mobility using eye movements based on electrooculography. Espanha, 2002. Disponível em: <https://ieeexplore.ieee.org/abstract/document/1178091>. Citado na p. 24.

BARROS DA SILVA NÉTO, L. F. M. de. Desenvolvimento de um manipulador robótico para aplicações assistivas. Tese (Graduação) - Faculdade de Tecnologia, Universidade de Brasília. Brasília, DF, 2019. Disponível em: <https://bdm.unb.br/handle/10483/24931>. Citado nas pp. 19, 33–37.

BASMAJIAN, J. V.; LUCA, C. J. de. **Muscles alive: their functions revealed by electromyography**. Baltimore, EUA: Williams & Wilkins, 1985. Citado na p. 23.

BATISTOTI, V. Microsoft desenvolve cadeira de rodas controlada por movimento ocular. São Paulo, SP, 2018. Disponível em: <https://revistagalileu.globo.com/Tecnologia/noticia/2018/03/microsoft-desenvolve-cadeira-de-rodas-controlada-por-movimento-ocular.html>. Acesso em: 3 set. 2022. Citado na p. 26.

BHARALI, S.; MATHI, S.; CHANDRA, S.; DASH, P. A Self-Governing Wheelchair for Severely Disabled Patients in an Indoor Environment Utilizing EEG and Laser Technologies. Ernakulam, India. Disponível em: <https://ieeexplore.ieee.org/document/8529089>. Citado na p. 60.

BIEN, Z.; CHUNG, M.-J.; CHANG, P. H. Integration of a Rehabilitation Robotic System (KARES II) with Human-Friendly Man-Machine Interaction Units. Korea advanced Institute of Science e Technology (KAIST), Korea, 2004. Disponível em: <https://www.researchgate.net/publication/236617050_Integration_of_a_Rehabilitation_Robotic_System_KARES_II_with_Human-Friendly_Man-Machine_Interaction_Units>. Citado na p. 30.

BRASIL. Lei nº 13.146, de 06 de Julho de 2015. **Diário Oficial da República Federativa do Brasil**, Brasília, DF, 6 jul. 2015. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm>. Acesso em: 24 jul. 2022. Citado na p. 17.

CARRARA, V. **Introdução à Robótica Industrial**. 1. ed. São José dos Campos, SP: Instituto Nacional de Pesquisas Espaciais - INPE, 2015. Citado na p. 36.

CYGANEK, B.; SIEBERT, J. P. **An Introduction to 3D Computer Vision Techniques and Algorithms**. 1. ed. Reino Unido: John Wiley & Sons, Ltd, 2009. Citado na p. 46.

DIAS, L.; BOM, C.; ALBUQUERQUE, M. P. de. Estimativa de Permeabilidade Absoluta com Processamento de Imagens Utilizando Distribuição de Tamanho de Grãos. Rio de Janeiro, RJ, 2017. Disponível em: <https://www.researchgate.net/publication/321924427_Estimativa_de_Permeabilidade_Absoluta_com_Processamento_de_Imagens_Utilizando_Distribuicao_de_Tamanho_de_Graos>. Citado na p. 42.

ELETRÔNICA, A.
bibinitperiod. Joystick 3 Eixos (A.022). Disponível em: <https://www.arduinoeeletronica.com.br/produto/joystick-3-eixos/>. Acesso em: 15 set. 2021. Citado na p. 58.

FARIAS, T. S. M. C. de. Metodologia para reconstrução 3D baseada em imagens. Recife, PE. Disponível em: <https://www.gprt.ufpe.br/grvm/wp-content/uploads/Publication/Thesis/Farias_TeseDoutorado_2012.pdf>. Citado nas pp. 48, 49.

FERREIRA, C. L. L. Interface de sopro e sucção para controle de cadeira de rodas. Londrina, PR, 2008. Disponível em: <http://www.bibliotecadigital.uel.br/document/?code=vtls000150171>. Citado na p. 23.

GARCIA, V. O que é paraplegia e tetraplegia? **Deficiente Ciente**, Brasília, DF, 2009. Disponível em: <https://www.deficienteciente.com.br/o-que-e-paraplegia-e-tetraplegia.html>. Acesso em: 16 set. 2021. Citado na p. 18.

HARTLEY, R.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. 2. ed. Published in the United States of America by Cambridge University Press, New York: Cambridge University Press, 2004. Citado nas pp. 48–50.

HILLMAN, M.; HAGAN, K.; JEPSON, J.; ORPWOOD, R. The Weston wheelchair mounted assistive robot - the design story. Bath Institute of Medical Engineering, Bath, UK, 2002. Disponível em: <https://www.researchgate.net/publication/220103759_The_Weston_wheelchair_mounted_assistive_robot_-_the_design_story>. Citado na p. 32.

HOSEINI, S. A.; KABIRI, P. A Novel Feature-Based Approach for Indoor Monocular SLAM. Iran University of Science e Technology, Irã, 2018. Disponível em: <https://www.mdpi.com/2079-9292/7/11/305>. Citado na p. 45.

IBGE. **Censo Demográfico**. Rio de Janeiro: Centro de Documentação e Disseminação de Informações. Fundação Intituto Brasileiro de Geografia e Estatística, 2010. Citado na p. 17.

JESUS, F. B.; FILHO, J. P. C.; BITTENCOURT, J. C. N.; JESUS, T. C. Modeling and Simulation of Laser Rangefinder Architecture. Feira de Santana, Bahia. Disponível em: <https://sbmicro.org.br/sforum-eventos/sforum2015/07.pdf>. Citado nas pp. 51, 52.

JULIÃO, A. Algoritmo computacional associado a eletroencefalografia se mostra eficaz no diagnóstico de Alzheimer. São Paulo, SP, 2021. Disponível em: <https://agencia.fapesp.br/algoritmo-computacional-associado-a-eletroencefalografia-se-mostra-eficaz-no-diagnostico-de-alzheimer/35108/>. Citado na p. 24.

JUNIOR, L. R. P. **Eletroencefalogramas básicos**. São Paulo, SP: Roca, 1990. Citado na p. 24.

MARTENS, C.; PRENZEL, O.; GRAESE, A. The Rehabilitation Robots FRIEND-I II: Daily Life Independency through Semi-Autonomous Task-Execution, Rehabilitation Robotics. DOI: 10.5772/5159, 2007. Disponível em: <https://www.researchgate.net/publication/221786214_The_Rehabilitation_Robots_FRIEND-I_II_Daily_Life_Independency_through_Semi-Autonomous_Task-Execution>. Citado nas pp. 32–34.

MATEJ MEŠKO, Š. T. Laser Spot Detection. Eslováquia, 2013. Disponível em: <https://www.researchgate.net/publication/350124875_Laser_spot_detection>. Citado na p. 53.

MORAIS MADALENA, I. de; MENOTTI, D. Auto-Calibração de Câmeras em Visão Estéreo (Artigo: PPGCC - Programa de Pós-Graduação em Ciência da Computação). Ouro Preto, MG. Disponível em: <http://www.decom.ufop.br/menotti/paa111/files/PCC104-111-ars-11.1-IsraelDeMoraisMadalena.pdf>. Citado na p. 46.

MÜLLER-CAJAR, R.; MUKUNDAN, R. Triangulation: A new algorithm for Inverse Kinematics. Nova Zelândia, 2007. Disponível em: <https://www.researchgate.net/publication/251743615_Triangulation_A_new_algorithm_for_Inverse_Kinematics>. Citado nas pp. 39, 40.

NATIONS, U. Ageing and disability. **United Nations**, Brasília, DF, 2018. Disponível em: <https://www.un.org/development/desa/disabilities/disability-and-ageing.html>. Acesso em: 15 set. 2021. Citado na p. 17.

NIELSEN, N. Stereo Vision and Depth Estimation - Computer Vision and OpenCV C++ and Python. Disponível em: <https://www.youtube.com/watch?v=KOSS24P3_fY>. Acesso em: 25 abr. 2022. Citado na p. 72.

OLIVEIRA, F. M. P. de. Estudo e Desenvolvimento de uma interface de comando de cadeira de rodas motorizada para pessoas tetraplégica. Tese (Graduação) - Faculdade de Tecnologia, Universidade de Brasília. Brasília, DF, 2019. Disponível em: <https://bdm.unb.br/handle/10483/24741>. Citado nas pp. 22, 23.

PASCHOALETTO, A. Desenvolvimento de um Braço Robótico de 6 Graus de Liberdade para Fins Cinematográficos. Tese (Graduação) - Faculdade de Tecnologia, Universidade de Brasília. Brasília, DF, 2022. Citado nas pp. 37–40.

PINHO, M. S. Reconstrução 3D. Porto Alegre, RS. Disponível em: <https://www.inf.pucrs.br/pinho/CGII/PDFs/Reconstrucao3D.ppt.pdf>. Citado na p. 51.

PINTO, F. F. Interfaces de controle de cadeira de rodas para pessoas com tetraplegia. Tese (Graduação) - Faculdade de Tecnologia, Universidade de Brasília. Brasília, DF, 2016. Disponível em: <https://bdm.unb.br/handle/10483/17086>. Citado na p. 22.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao Processamento Digital de Imagens. Campina Grande, PB, 2001. Disponível em: <http://www.dsc.ufcg.edu.br/~hmg/disciplinas/graduacao/vc-2016.2/Rita-Tutorial-PDI.pdf>. Citado nas pp. 42–44, 52.

ROBOTICS, K. Robotic Arm Jaco, 2022. Disponível em: <https://assistive.kinovarobotics.com/product/jaco-robotic-arm>. Citado na p. 29.

ROCHA, W. C. J. Desenvolvimento de um Sistema de medição de distância baseado em Visão Computacional utilizando laser de linha. Cruz das Almas, BA. Disponível em: <https://www2.ufrb.edu.br/bcet/components/com_chronoforms5/chronoforms/uploads/tcc/20190604191229_2018.2_TCC_Walber_Conceio_de_Jesus_Rocha_-_Desenvolvimento_de_um_sistema_de_medio_de_distncia_baseado_em_viso_computacional_utilizando_laser_de_linha.pdf>. Citado nas pp. 48, 62.

ROSÁRIO, J. M. **Robótica Industrial I: Modelagem, Utilização e Programação**. 1. ed. São Paulo: Editora Baraúna SE Ltda, 2010. Citado na p. 17.

SAÚDE, M. da. Diretrizes de atenção às pessoas com lesão medula. Brasília, DF, 2013. Disponível em: <https://bvsms.saude.gov.br/bvs/publicacoes/diretrizes_atencao_pessoa_lesao_medular.pdf>. Acesso em: 16 set. 2021. Citado na p. 18.

SILVA, K. L. da. Protótipo De Uma Cadeira De Rodas Controlada Por Movimentos Da Cabeça. Uberlândia, MG, 2013. Citado na p. 25.

SONG, W.-K.; LEE, H.-Y.; KIM, J.-S.; YOON, Y.-S.; BIEN, Z. KARES: intelligent rehabilitation robotic system for the disabled and the elderly. Hong Kong, China, 1998. Disponível em: <https://ieeexplore.ieee.org/abstract/document/745226>. Citado nas pp. 29, 30.

SOUSA, I. M. de. Estudo e desenvolvimento de um sistema de reconhecimento de expressões faciais para controle de cadeira de roda. Tese (Graduação) - Universidade de Brasília. Brasília, DF, 2019. Disponível em: <https://bdm.unb.br/handle/10483/25070>. Citado nas pp. 26, 27, 62.

TSUI, K. M.; YANCO, H. A. Simplifying Wheelchair Mounted Robotic Arm Control with a Visual Interface. Lowell, MA, 2007. Disponível em: <https://www.aaai.org/Papers/Symposia/Spring/2007/SS-07-07/SS07-07-021.pdf>. Citado na p. 31.

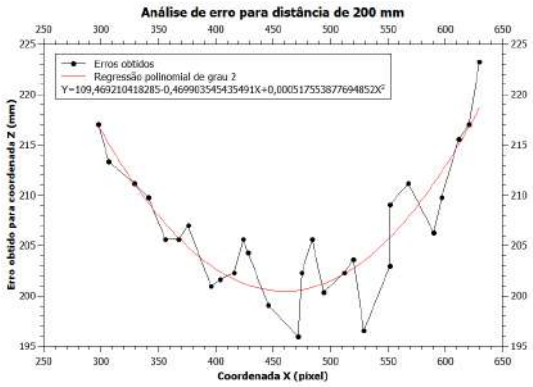
ZHANG, Z. "A flexible new technique for camera calibration"in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718. Washington, EUA. Disponível em: <https://ieeexplore.ieee.org/document/888718>. Citado nas pp. 47, 48.

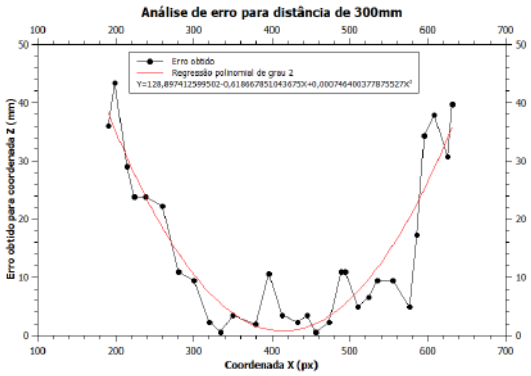
ZHONG, M.; ZHANG, Y.; YANG, X. Assistive Grasping Based on Laser-point Detection with Application to Wheelchair-mounted Robotic Arms. Weihai, China, 2019. Disponível em: <https://www.mdpi.com/1424-8220/19/2/303>. Citado nas pp. 27, 28, 53, 59, 60.

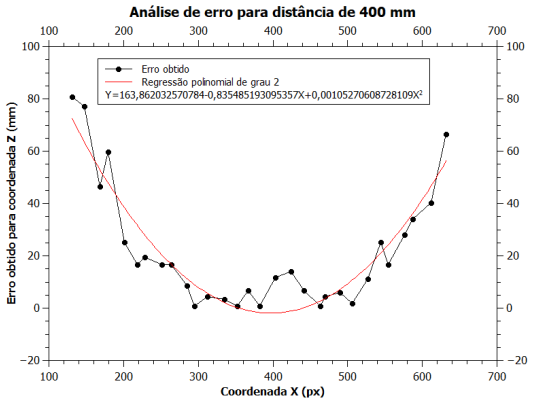
Apêndices

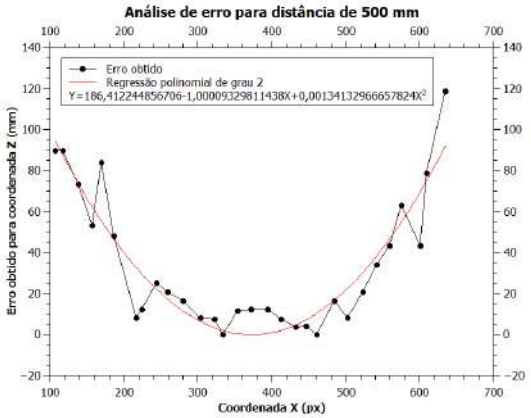
APÊNDICE A – Figuras e Gráficos

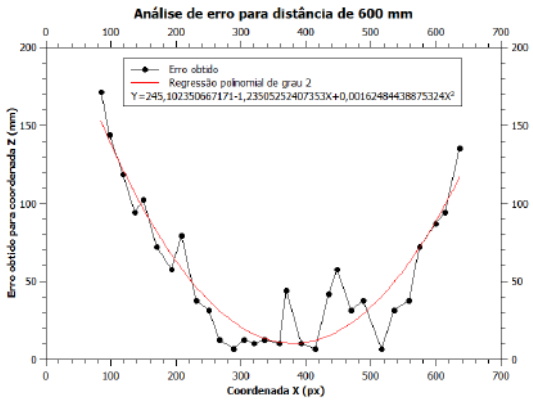
Regressão polinomiais aplicadas na análise de erros em relação à X na abordagem empírica.

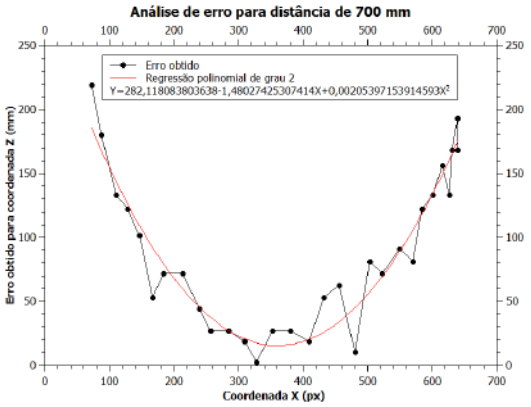


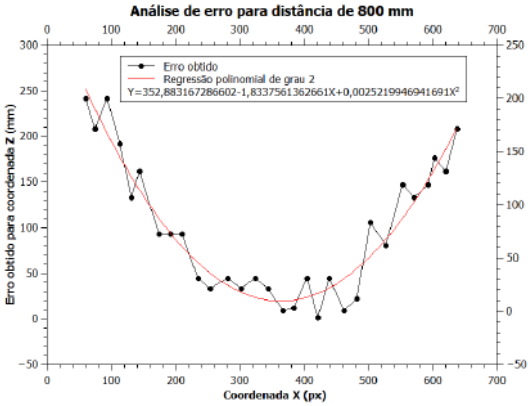


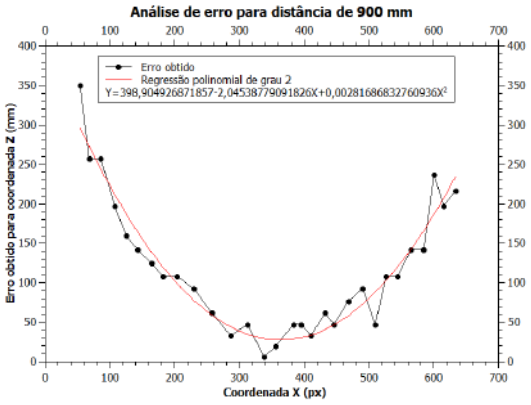


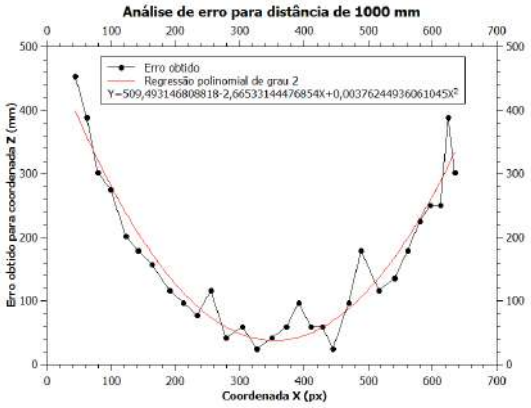




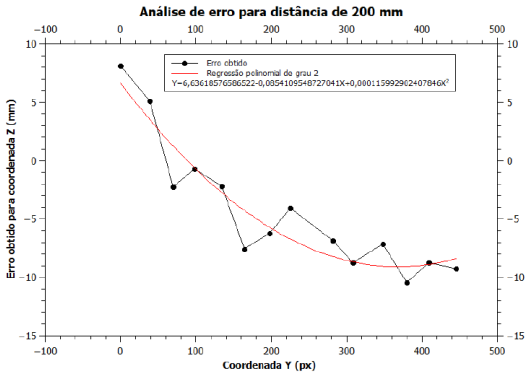


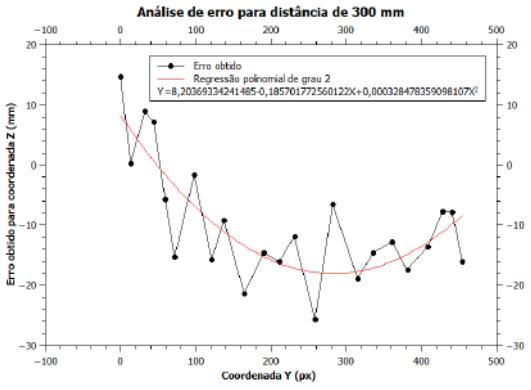


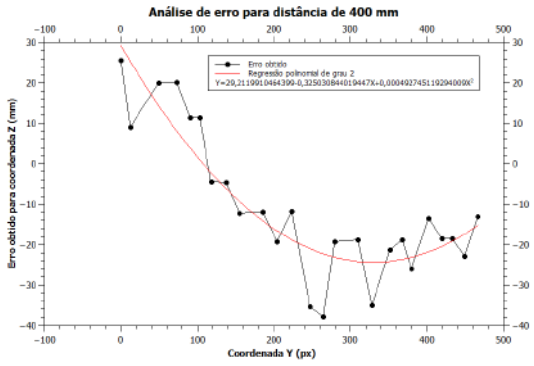


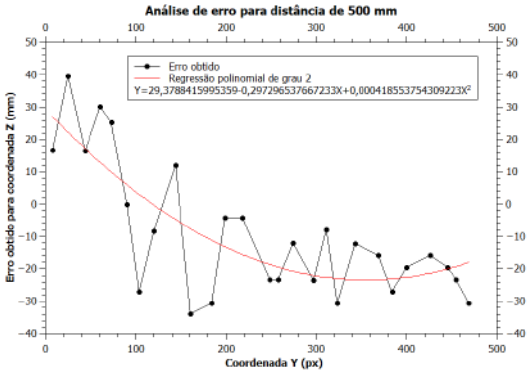


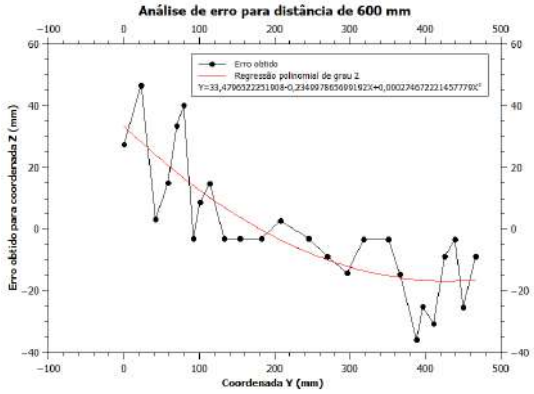
Regressão polinomiais aplicadas na análise de erros em relação à Y na abordagem empírica.

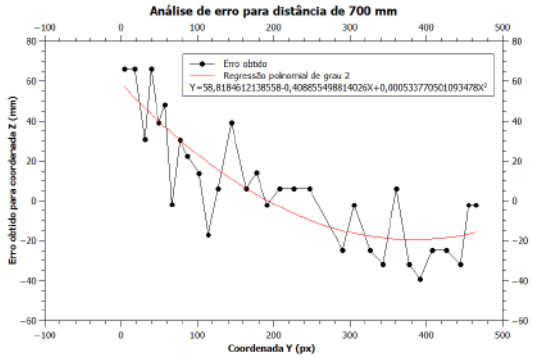


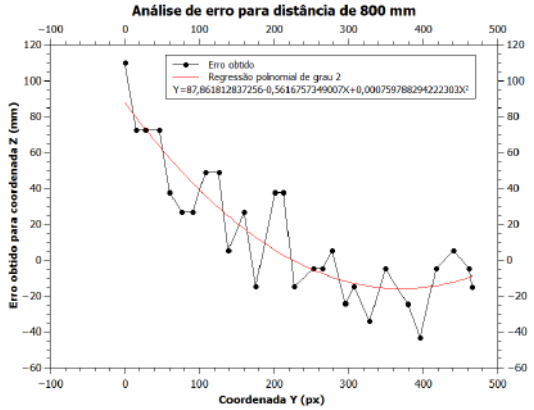


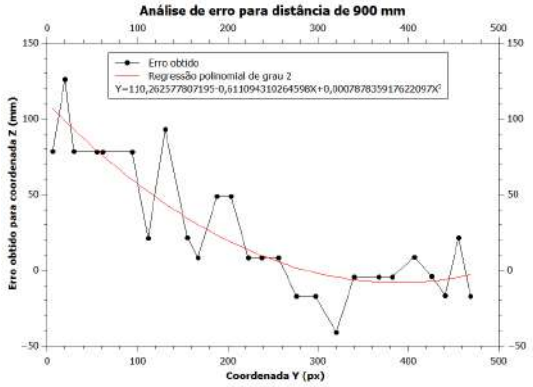


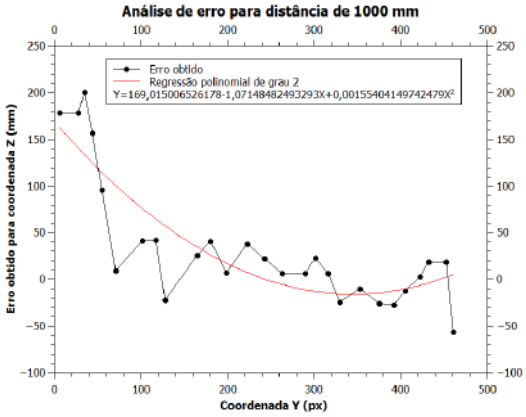



Análise de erro para distância de 600 mm
Erro obtido
Regressão polinomial de grau 2
Y=33,47965222251008+0,2349978656991923X+0,00027467222145777 9X²
Erro obtido para coordenada Z (mm)
Coordenada Y (mm)


Análise de erro para distância de 700 mm
Erro obtido
Regressão polinomial de grau 2
Y=58,81846121385 58-0,4088554988140 26X+0,0005337705010934 78X²
Erro obtido para coordenada Z (mm)
Coordenada Y (px)


Análise de erro para distância de 800 mm
Erro obtido
Regressão polinomial de grau 2
Y=87,861812837256-0,5616757349007X+0,0007597882942220 3X²
Erro obtido para coordenada Z (mm)
Coordenada Y (px)


Análise de erro para distância de 900 mm
Erro obtido
Regressão polinomial de grau 2
Y=110,26257780719 5-0,6110943102645 98X+0,0007878359176220 97X²
Erro obtido para coordenada Z (mm)
Coordenada Y (px)


Análise de erro para distância de 1000 mm
Erro obtido
Regressão polinomial de grau 2
Y=169,015006526178-1,071484824932 93X+0,0015540414974247 9X²
Erro obtido para coordenada Z (mm)
Coordenada Y (px)

APÊNDICE B – Códigos de programação

B.1 Implementação da visão estéreo e triangulação

Código B.1 – Código de Python

```python
# Importacao das bibliotecas
from binascii import a2b_base64
import cv2
import numpy as np
import math

# Declarando a variavel de captura
cap1 = cv2.VideoCapture(2)
cap2 = cv2.VideoCapture(1)


# deslocamento entre as cameras
l = 97.5

# coeficientes de ajuste de erro de distorcao da coordenada X
a = 0.0000000283073593073593
b = 0.000000409783549783549
c = 0.0000371238095238096
d = -0.0000197118395400433
e = -0.000224344965361472
f = -0.0378126982185715
g = 0.00435556079761905
h = -0.0420228962476189
i = 10.3503429939048

# coeficientes de ajuste de erro de distorcao da coordenada Y
a2 = 0.00000002475318077954 55
b2 = 0.000000167378666846212
c2 = 0.0000533233906316667
d2 = -0.0000153381600118615
e2 = 0.000873025562003377
f2 = -0.0289559214903714
g2 = 0.00285926369908225
h2 = -0.15937311063987
i2 = 3.28361954828571

img_counter = 0

while cap1.isOpened():
```

```
40      sucess1, img1 = cap1.read()
41      sucess2, img2 = cap2.read()
42
43      if not sucess1 or not sucess2:
44          print("Failed to grab the frame")
45          break
46
47      # definicao de threshold do laser
48      lower_red = np.array([150, 62, 200])
49      upper_red = np.array([179 , 255, 255])
50
51      # identificacao do ponto laser na camera 1
52      hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
53      mask1 = cv2.inRange(hsv, lower_red, upper_red)
54      (minVal, maxVal, minLoc, maxLoc1) = cv2.minMaxLoc(mask1)
55      cv2.circle(img1, maxLoc1, 20, (0, 255, 0), 2, cv2.LINE_AA)
56      cv2.line(img1, (319, 0), (319,720), (0, 255, 0), 1)
57      cv2.line(img1, (0, 239), (1280,239), (0, 255, 0), 1)
58      cv2.imshow("Cam1", img1)
59
60      # identificacao do ponto laser na camera 2
61      hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
62      mask2 = cv2.inRange(hsv, lower_red, upper_red)
63      (minVal, maxVal, minLoc, maxLoc2) = cv2.minMaxLoc(mask2)
64      cv2.circle(img2, maxLoc2, 20, (0, 255, 0), 2, cv2.LINE_AA)
65      cv2.imshow("Cam2", img2)
66
67
68      # Declaracao de variavel de identificacao de tecla
69      k = cv2.waitKey(1)
70
71      # Se a tecla pressionada for Esc
72      if k%256 == 27:
73          print("Escape hit, closing the application")
74          break
75
76      elif k%256 == 32:
77          x1, y1 = maxLoc1 # localizacao do ponto laser identificado
                na camera 1
78          x2, y2 = maxLoc2 # localizacao do ponto laser identificado
                na camera 2
79
80          ## CAM 1 - LEFT
81
82          cv2.imwrite('img/stereoLeft/sceneL' + str(img_counter) +
                '.png', img1)
83          cv2.imwrite('img/stereoRight/sceneR' + str(img_counter) +
                '.png', img2)
84          print("Screenshots taken")
85          img_counter+=1
86
87
```

```
88          distanceX1 = x1 - 320.0
89          tanX1 = distanceX1/644.4389
90          xrad1 = math.atan(tanX1)
91
92          distanceY1 = y1 - 240
93          tanY1 = distanceY1/644.4389
94          yrad1 = math.atan(tanY1)
95
96          # para o calculo da triangulacao entre as cameras,
               utilizamos o angulo complementar
97          if (xrad1 < 0):
98              alfaX = math.pi/2 + abs(xrad1)
99          else:
100             alfaX = math.pi/2 - abs(xrad1)
101
102
103
104         ## CAM 2 - RIGHT
105
106         distanceX2 = x2 - 320.0
107         tanX2 = distanceX2/644.4389
108         xrad2 = math.atan(tanX2)
109
110         distanceY2 = y2 - 240
111         tanY2 = distanceY2/644.4389
112         yrad2 = math.atan(tanY2)
113
114
115         # para o calculo da triangulacao entre as cameras,
               utilizamos o angulo complementar
116         if (xrad2 < 0):
117             betaX = math.pi/2 - abs(xrad2)
118         else:
119             betaX = math.pi/2 + abs(xrad2)
120
121
122         ## TRIANGULACAO PARA CALCULAR Z ##
123         # calculo da coordenada Z
124         # distancia entre as cameras eh de 9.7cm
125
126         tanAlfa = math.tan(alfaX)
127         tanBeta = math.tan(betaX)
128         num = l * tanAlfa * tanBeta
129         dem = tanAlfa + tanBeta
130         distanceZ = num/dem
131
132         # quando a relacao entre as cameras nao formar um
               triangulo retangulo
133         # aplicar-se lei dos senos para encontrar a distancia
134         if(distanceX1 < 0):
135             gammaX = math.pi - alfaX - betaX
136             gammaX = abs(gammaX)
```

```
137             leiSeno = l/(math.sin(gammaX)) if math.sin(gammaX) !=
                    0 else 999999999
138             x = leiSeno * math.sin(betaX)
139             complementar = math.pi - alfaX
140             distanceZ = (math.sin(complementar)) * x
141
142
143         ## TRIANGULACAO SIMPLIFICADA ##
144         z = (644.4389*l)/( distanceX1 - distanceX2 )
145         print("Formula simples: ",z)
146
147
148         ## CORRECAO DA DISTORCAO COORDENADA X ##
149         ZcorrigidoX = (-1 - b*(x1**2) - e*x1 - h +
                (math.sqrt((1+b*(x1**2)+e*x1+h)**2 - 4*(a*(x1**2) +
                d*x1 + g )*(c*(x1**2) + f*x1 + i -
                distanceZ))))/(2*(a*(x1**2) + d*x1 + g))
150
151
152         ## CORRECAO DA DISTORCAO COORDENADA Y ##
153         ZcorrigidoY = (-1 - b2*(y1**2) - e2*y1 - h2 +
                (math.sqrt((1+b2*(y1**2)+e2*y1+h2)**2 - 4*(a2*(y1**2) +
                d2*y1 + g2 )*(c2*(y1**2) + f2*y1 + i2 -
                ZcorrigidoX))))/(2*(a2*(y1**2) + d2*y1 + g2))
154
155         # calculo da coordenada X
156         x = ZcorrigidoX * math.tan(xrad1)
157         distanceX = distanceZ * math.tan(xrad1)
158
159
160         # calculo da coordenada Y
161         y = ZcorrigidoX * math.tan(yrad1)
162         distanceY = distanceZ * math.tan(yrad1)
163
164         print("Coordenadas cruas", distanceZ*math.tan(xrad1),
                "\t", distanceZ*math.tan(yrad1), "\t", distanceZ)
165         print("\nCorrecao de Z em X: ", ZcorrigidoX, "\n")
166         print("Correcao de Z em Y: ", ZcorrigidoY, "\n\n")
```