



Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-graduação em Engenharia Elétrica

Controle cinemático de robôs cooperativos usados na manipulação de objetos

Felipe Henrique Neiva do Nascimento

Brasil

2019, Julho

Felipe Henrique Neiva do Nascimento

Controle cinemático de robôs cooperativos usados na manipulação de objetos

Dissertação de Mestrado apresentada à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Universidade Federal de Campina Grande – UFCG

Departamento de Engenharia Elétrica

Programa de Pós-Graduação

Orientador: Antonio Marcus Nogueira Lima

Coorientador: Marcos Ricardo Alcântara de Moraes


Brasil

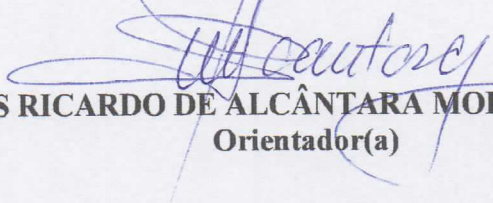
2019, Julho

**"CONTROLE CINEMÁTICO DE ROBÔS COOPERATIVOS USADOS NA MANIPULAÇÃO
DE OBJETOS"**


FELIPE HENRIQUE NEIVA DO NASCIMENTO

DISSERTAÇÃO APROVADA EM 25/07/2019


ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador(a)


MARCOS RICARDO DE ALCÂNTARA MORAIS, D.Sc., UFCG
Orientador(a)


ALEXANDRE CUNHA OLIVEIRA, D.Sc., UFCG
Examinador(a)


LUCIANA RIBEIRO VELOSO, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

N244c Nascimento, Felipe Henrique Neiva do.
Controle cinemático de robôs cooperativos usados na
manipulação de objetos / Felipe Henrique Neiva do Nascimento. –
Campina Grande, 2019.
132 f. : il. color.

Dissertação (Mestrado em Engenharia Elétrica) –
Universidade Federal de Campina Grande, Centro de Engenharia
Elétrica e Informática, 2019.
"Orientação: Prof. Dr. Antonio Marcos Nogueira Lima, Prof.
Dr. Marcos Ricardo Alcântara de Moraes".
Referências.

1. Manipuladores Colaborativos. 2. Controle Cinemático. 3.
Filtro de Partículas. 4. Localização Baseada em Objetos. 5. ROS.
6. Visão Computacional. I. Lima, Antonio Marcos Nogueira. II.
Moraes, Marcos Ricardo Alcântara de. III. Título.

CDU 621.865.8(043)

Este trabalho é dedicado a seu Fernando. Que nos conforte com sua presença por muitos anos.

Agradecimentos

Agradeço meu pai, Fernando, pelo companheirismo e apoio em dias difíceis, e a educação e caráter que recebi.

Aos professores Antônio Marcus e Marcos Morais, pelo voto de confiança, pela paciência e pela orientação ao longo dos últimos cinco anos.

A Arthur, que coletou as pedras para construção dessa estrada.

Ao Virtus, a Igor e ao professor Alexandre, pela oportunidade, pelos sacrifícios, e pela confiança nos últimos anos.

A meus irmãos, Pedro e Athina, por se manterem próximos e me lembrarem que estamos juntos.

Por fim, agradeço a meus companheiros de laboratório Gabriel, Davi e Yuri, por compartilharem dos cafés e das angústias.

Resumo

O trabalho aborda uma solução de controle cinemático de múltiplos manipuladores cooperativos ao longo de uma trajetória pré definida com manipulação de objetos, enquanto um sistema de visão em profundidade localiza e realiza o rastreo do objeto utilizando Filtro de Partículas. O objetivo do trabalho é apresentar uma solução para o problema do controle cinemático em espaço de trabalho para múltiplos manipuladores, com a capacidade de rastrear a posição do objeto manipulado. Para isso, são utilizadas técnicas de controle cinemático de manipuladores em espaço de trabalho com uma abordagem para múltiplos robôs cooperativos. Três técnicas de inversão de jacobiano são utilizadas, sendo elas inversa do jacobiano, pseudo inversa e inversa amortecida. A solução de controle dos manipuladores é feita utilizando técnicas de controle proporcional, controle proporcional e integral e controle proporcional e feedforward. As técnicas são implementadas em ambiente de simulação utilizando V-REP e em ambiente experimental com dois manipuladores UR5. São feitos testes de resposta ao degrau e de controle em uma trajetória pré definida. Para localização e rastreo do objeto, é utilizado um Kinect em conjunto com ambiente ROS e PCL. Uma técnica de localização baseada em modelos com filtro de partículas é utilizada. São realizados experimentos com os dois manipuladores manipulando um objeto, utilizando princípios de manipulação cooperativa e as técnicas de controle cinemático desenvolvidas. Ainda é feita a integração do controle cinemático dos dois manipuladores com o sistema de localização e rastreo. As técnicas de inversão do jacobiano são aplicadas com sucesso e uma comparação entre os resultados é realizada, sendo optado a técnica de inversa amortecida para o resto do trabalho. O controle da trajetória é realizado com sucesso nos três tipos de controle, mas apenas no controle proporcional feedforward o erro de regime permanente é zero. A manipulação cooperativa é feita com sucesso, mas devido a um atraso na comunicação, existe um erro de regime permanente. A localização baseada em modelos desenvolvida consegue localizar e rastrear o objeto durante a trajetória, porém, devido a diversas incertezas no ambiente de experimento e imprecisão do sensor, a qualidade do rastreo é menor do que a esperada. Mesmo assim, o rastreo do objeto é feito com sucesso.

Palavras-chave: Controle Cinemático; Filtro de Partículas; Localização baseada em objetos; Manipuladores Colaborativos; Múltiplos Manipuladores; ROS; Visão Computacional.

Abstract

This work addresses a kinematic control solution for multiple cooperative manipulators in a pre defined trajectory with object manipulation, while a depth vision system locates and tracks an object using Particle Filter. The general objective is to present a solution for task space control with kinematic control for multiple manipulators, with the ability to locate and track the position of the manipulated object. To this end, kinematic control techniques are used with an approach for multiple cooperative robots. Three techniques for jacobian inverse are used, being them jacobian inverse, pseudo inverse and damped least squares. The control solutions used for the manipulators are proportional control, proportional and integral control and proportional feedforward control. These techniques are implemented in the V-REP simulator and in an experimental environment with two UR5 manipulators. Step response and pre defined trajectory control tests are realized. For object location and tracking, a Kinect is used in the ROS ambient with PCL. A technique for localization based on models with particle filter is used. Experiments with both arms manipulating an object are done, using cooperative manipulators principles and the kinematic control techniques developed. Both systems are integrated together for dual arm kinematic control with object localization and tracking. The jacobian inverse techniques developed are applied with success and a comparison between the performance of the three is done, where it was found that the damped least squares method is the best method for the rest of the work. The trajectory control is successfully done with the three controllers, but the steady state error goes to zero only with the proportional feedforward control. The dual arm manipulation is done with success, but there is a delay in the communication system, resulting on a steady state error. The model based localization method developed can locate and track the object during the trajectory but, because a uncertainty in the experimental environment and sensor inaccuracy, the tracking quality is smaller than expected. Nevertheless, the tracking is done successfully.

Keywords: Kinematic Control; Particle Filter; Model-Based Localization; Cooperative Robots; Dual Arm; ROS; Computer Vision.

Sumário

	Introdução	15
	Revisão bibliográfica	19
1	REPRESENTAÇÃO DE POSIÇÃO E ORIENTAÇÃO	23
1.1	Pose e pose relativa	23
1.2	Posição	25
1.3	Rotação	25
1.3.1	Matriz de rotação ortonormal	26
1.4	Matriz de transformação homogênea	28
1.5	Singularidades	29
1.6	Representação vetor-ângulo	29
1.7	Quatérnions	31
1.8	Transformações de sistemas de orientação	33
1.8.1	Matrizes de rotação para ângulos de Euler ZYX	33
1.8.2	Matrizes de rotação para vetor-ângulo	34
1.8.3	Matrizes de rotação para quatérnions unitários	34
1.9	Conclusões	35
2	MANIPULADOR SÉRIE	36
2.1	Descrição geométrica	36
2.2	Cinemática	38
2.2.1	Cinemática direta	38
2.2.2	Cinemática inversa	40
2.3	Velocidade cinemática	40
2.3.1	Jacobiano geométrico	41
2.3.2	Jacobiano analítico	41
2.4	Dinâmica	42
2.4.1	Modelo e controle de junta independente	43
2.4.2	Dinâmica de corpo rígido	44
2.5	Conclusões	45
3	VISÃO COMPUTACIONAL	46
3.1	Imagens	46
3.2	Visão estéreo	48
3.2.1	Geometria epipolar	48

3.2.2	Nuvem de pontos	50
3.3	Localização de objetos baseada em modelos	52
3.4	Alinhamento inicial por consenso de amostra	52
3.5	Filtro de partículas	55
3.6	Conclusões	57
4	METODOLOGIA	59
4.1	Definição do problema	59
4.2	Modelagem e solução	59
4.3	Simuladores e Ferramentas	61
4.3.1	Simulador V-REP	62
4.3.1.1	Operação das juntas	64
4.3.1.2	Dinâmica	64
4.3.2	Sistema Operacional de Robôs - ROS	65
4.4	Configuração do ambiente de simulação	66
4.5	Instalação de ambiente de experimentos	67
4.6	Kinect	67
4.6.1	Localização espacial usando visão	68
4.6.2	Calibração Mão para Olho	69
4.7	Manipulador UR5	70
4.7.1	Controle do manipulador	71
4.7.2	Sistema de comunicação	72
4.7.3	URScript	72
4.7.4	Controle em tempo real	73
4.7.5	Sincronização de manipuladores	74
4.8	Múltiplos Manipuladores	74
4.9	Preensão robótica	75
4.10	Poses iniciais	76
4.11	Conclusões	78
5	SIMULAÇÕES E EXPERIMENTOS	79
5.1	Cinemática inversa em malha aberta	79
5.1.1	Singularidades da inversa do Jacobiano	80
5.1.2	Pseudo inversa	80
5.1.3	Inversa amortecida	81
5.2	Cinemática Inversa por aproximação de Newton-Raphson	82
5.3	Cinemática Inversa em malha fechada	84
5.3.1	Definição do erro	85
5.3.1.1	Erro absoluto	85
5.4	Resposta ao degrau	86

5.4.1	Degrau em coordenadas cartesianas	86
5.4.1.1	Simulação	87
5.4.1.2	Experimento	89
5.4.2	Análise de resultados	90
5.4.2.1	Comparação entre simulação e experimento	90
5.5	Movimento através de singularidade	91
5.5.1	Controle cinemático sem inversa amortecida	91
5.5.2	Controle cinemático com inversa amortecida	93
5.5.3	Análise de resultados	95
5.6	Estratégias de controle	95
5.6.1	Controle Proporcional	95
5.6.1.1	Simulação do Controle Proporcional	96
5.6.1.2	Experimento com Controle Proporcional	99
5.6.2	Controle Proporcional e Integral	101
5.6.2.1	Simulação do Controle Proporcional e Integral	102
5.6.2.2	Experimento com Controle Proporcional e Integral	105
5.6.3	Controle Proporcional e Feedforward	107
5.6.3.1	Simulação do Controle Proporcional Feedforward	108
5.6.3.2	Experimento com Controle Proporcional Feedforward	111
5.7	Análise dos resultados do controle cinemático	113
5.8	Controle cinemático de múltiplos manipuladores	115
5.8.1	Poses iniciais	116
5.8.2	Sincronia	118
5.9	Rastreamento de objeto com visão computacional	118
5.9.1	Localização inicial	118
5.9.2	Resultados	121
6	CONCLUSÃO	124
	REFERÊNCIAS	127

Lista de ilustrações

Figura 0.1 – Robô doméstico da Aeolus Robots.	15
Figura 0.2 – Manipulador UR10 da Universal Robots.	15
Figura 0.3 – Robô colaborativo Yumi da ABB.	16
Figura 1.1 – Pose em 3 dimensões representado por dois sistemas de coordenadas distintos e um ponto relativo aos dois.	24
Figura 2.1 – Representação em Denavit-Hartenberg.	37
Figura 2.2 – Robô com duas articulações.	39
Figura 3.1 – Modelo de Projeção Central.	47
Figura 3.2 – Imagem Discretizada.	48
Figura 3.3 – Visão Estéreo. O ponto \mathbf{P} é projetado nas duas câmeras e, com a posição conhecida de ambas as câmeras, é possível calcular uma triangulação para localizar \mathbf{P} no espaço.	49
Figura 3.4 – Objeto impresso usado para validação do algoritmo.	53
Figura 3.5 – Modelo CAD de objeto de teste.	53
Figura 3.6 – Nuvem de Pontos do Objeto.	53
Figura 3.7 – Aproximação inicial utilizando SAC-IA.	55
Figura 3.8 – Etapas de Predição e Atualização do rastreamento Bayesiano não linear. O arco tracejado representa o fim de um laço temporal.	58
Figura 4.1 – Trajetória para controle de posição.	61
Figura 4.2 – Diagrama ilustrativo com os diferentes equipamentos e processos da solução.	62
Figura 4.3 – Exemplo de ambientação no V-REP.	63
Figura 4.4 – Diagrama ilustrativo da troca de mensagens no ROS.	65
Figura 4.5 – Exemplo de ambiente de simulação com um único UR5.	66
Figura 4.6 – Exemplo de ambiente de simulação com os dois UR5 e um objeto.	67
Figura 4.7 – Campo de visão do Kinect.	68
Figura 4.8 – Kinect em Laboratório.	69
Figura 4.9 – Exemplo de identificação de um objeto utilizando modelo e nuvem de pontos.	70
Figura 4.10–Modelo do Universal Robot 5.	71
Figura 4.11–UR5 em Laboratório	71
Figura 4.12–Diagrama de comunicação com o manipulador UR5.	73
Figura 4.13–Geometria de prensão para dois manipuladores cooperativos manipu- lando um objeto.	75
Figura 4.14–Modelos de contato: (a) pontual sem atrito, (b) pontual com atrito, (c) dedo flexível (LEON; MORALES; SANCHO-BRU, 2014).	76

Figura 4.15–Posições iniciais para geração de trajetória do objeto e dos manipuladores. A pose do objeto em seu centro de massa e as duas poses geradas nas laterais.	78
Figura 5.1 – Erro absoluto de posição para resposta ao degrau.	87
Figura 5.2 – Trajetória no eixo Z para resposta ao degrau.	88
Figura 5.3 – Erro absoluto de posição para resposta ao degrau - Experimento.	89
Figura 5.4 – Trajetória no eixo Z para resposta ao degrau - Experimento.	89
Figura 5.5 – Trajetória percorrida com singularidade, sem amortecimento.	91
Figura 5.6 – Proximidade de singularidade e fator de amortecimento, sem amortecimento.	92
Figura 5.7 – Velocidade de articulação, sem amortecimento.	92
Figura 5.8 – Trajetória percorrida com singularidade, com amortecimento.	93
Figura 5.9 – Proximidade de singularidade e fator de amortecimento, com amortecimento.	94
Figura 5.10–Velocidade de articulação, com amortecimento.	94
Figura 5.11–Erro Absoluto de Posição do controle cinemático com Controle Proporcional.	96
Figura 5.12–Erro Absoluto de Rotação do controle cinemático com Controle Proporcional.	97
Figura 5.13–Posição de Juntas do controle cinemático com Controle Proporcional.	97
Figura 5.14–Velocidade de Juntas do controle cinemático com Controle Proporcional.	98
Figura 5.15–Erro Absoluto de Posição do controle cinemático com Controle Proporcional - Experimento.	99
Figura 5.16–Erro Absoluto de Rotação do controle cinemático com Controle Proporcional - Experimento.	100
Figura 5.17–Posição de Juntas do controle cinemático com Controle Proporcional - Experimento.	100
Figura 5.18–Velocidade de Juntas do controle cinemático com Controle Proporcional - Experimento.	101
Figura 5.19–Erro Absoluto de Posição do controle cinemático com Controle Proporcional e Integral.	102
Figura 5.20–Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Integral.	103
Figura 5.21–Posição de Juntas do controle cinemático com Controle Proporcional e Integral.	103
Figura 5.22–Velocidade de Juntas do controle cinemático com Controle Proporcional e Integral.	104
Figura 5.23–Erro Absoluto de Posição do controle cinemático com Controle Proporcional e Integral - Experimento.	105

Figura 5.24–Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Integral- Experimento.	106
Figura 5.25–Posição de Juntas do controle cinemático com Controle Proporcional e Integral - Experimento.	106
Figura 5.26–Velocidade de Juntas do controle cinemático com Controle Proporcional Integral - Experimento.	107
Figura 5.27–Erro Absoluto de Posição do controle cinemático com Controle Proporcional Feedforward.	108
Figura 5.28–Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Feedforward.	109
Figura 5.29–Posição de Juntas do controle cinemático com Controle Proporcional e Feedforward.	109
Figura 5.30–Velocidade de Juntas do controle cinemático com Controle Proporcional Feedforward.	110
Figura 5.31–Erro Absoluto de Posição do controle cinemático com Controle Proporcional e Feedforward - Experimento.	111
Figura 5.32–Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Feedforward- Experimento.	112
Figura 5.33–Posição de Juntas do controle cinemático com Controle Proporcional e Feedforward - Experimento.	112
Figura 5.34–Velocidade de Juntas do controle cinemático com Controle Proporcional Feedforward - Experimento.	113
Figura 5.35–Diagrama Temporal. A comunicação inicial, aquisição de posição inicial e geração da trajetória é feito antes do controle. O laço de controle é realizado em menos de 8 milissegundos.	116
Figura 5.36–Configuração de experimento com múltiplos manipuladores.	117
Figura 5.37–Erro de posição no controle de múltiplos manipuladores.	119
Figura 5.38–Trajetória em Z percorrida por um dos dois manipuladores sincronizados.	120
Figura 5.39–Trajetória percorrida por um dos manipuladores sincronizados.	121
Figura 5.40–Rastreo de objeto utilizando visão e múltiplos manipuladores.	122
Figura 5.41–Rastreo de objeto - Trajetória XYZ.	122
Figura 5.42–Rastreo de objeto - Trajetória XY.	123
Figura 5.43–Rastreo de objeto - Trajetória XZ.	123

Lista de tabelas

Tabela 1 – Classes de Rotações: As Eulerianas, classificadas por duas rotações em torno do mesmo eixo nas três rotações, e as Cardanas, classificadas pelas rotações nos três eixos.	27
Tabela 2 – Matrizes de rotação equivalentes a determinados sistemas de orientação. Abreviações: $c_\theta := \cos(\theta)$, $s_\theta := \sin(\theta)$, $v_\theta := 1 - \cos(\theta)$	33
Tabela 3 – Denavit-Hartenberg	37
Tabela 4 – Dados do modelo usado nas simulações	65
Tabela 5 – Parâmetros UR5	71
Tabela 6 – Tempo de processamento de inversão do jacobiano	90

Introdução

Robótica se tornou uma das áreas essenciais da engenharia na automação da indústria no final do século XX e no século XXI, é uma área caracterizada pela realização de tarefas repetitivas, tarefas que requerem grande precisão ou tarefas em ambientes inóspitos. Apesar dos avanços no último século, o estudo da robótica ainda está em progresso e tem potencial de facilitar ainda mais a vida humana em diversos aspectos, desde sua utilização na indústria, em operações médicas ou mesmo em aplicações pessoais e domésticas.

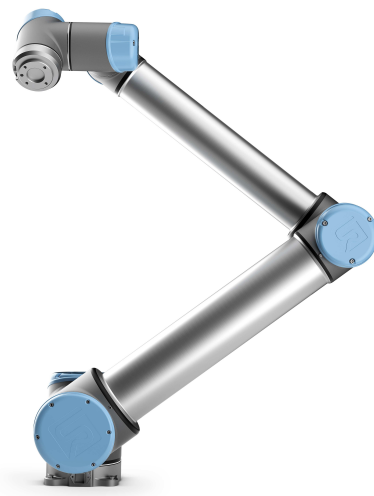
Robótica em aplicações pessoais e domésticas experimentou um forte crescimento global com uma variedade limitada de produtos de mercado: robôs de limpeza, robôs de assistência e robôs para educação. Expectativas para 2019 apontam um crescimento para até 31 milhões de robôs ativos na área, comparado a 3 milhões em 2014 (IFR, 2016). As visões de produtos futuros apontam para robôs domésticos de maior sofisticação e abrangência, como robôs assistenciais para apoio aos idosos, para ajudar com tarefas domésticas e para entretenimento. Exemplos de robôs sendo desenvolvidos no mercado incluem o robô doméstico da Aeolus, observado na Figura 0.1, responsável pela manipulação de objetos para assistência domiciliar. Assim, a demanda por sistemas robóticos que interagem com humanos tende a aumentar.

Esse aumento na demanda por sistemas robóticos está associado a crescente

Figura 0.1 – Robô doméstico da Aeolus Robots. Figura 0.2 – Manipulador UR10 da Universal Robots.



Fonte: Aeolus Robots



Fonte: Universal Robots

Figura 0.3 – Robô colaborativo Yumi da ABB.



Fonte: ABB

utilização de robôs colaborativos na indústria e pesquisa, assim classificados pois permitem que humanos entrem e trabalhem no espaço de trabalho do robô com segurança.

Um dos tipos de robôs mais utilizados nas empresas e pesquisas são os manipuladores, constituídos de uma série de juntas e conexões atuadas com um efetuator para realizar uma ou mais tarefas, comumente associadas a manipulação e transporte de objetos. Manipuladores colaborativos permitem, então, que esses robôs sejam utilizados em linhas de fábrica ou laboratórios de pesquisa sem a necessidade de serem enclausurados, permitindo um ambiente de trabalho compartilhado com seres humanos.

O uso de múltiplos manipuladores é essencial em atividades realizadas tanto na indústria quanto em aplicações domésticas, permitindo a manipulação de uma variedade de objetos que não seria possível com apenas um manipulador. Atividades que podem ser realizadas com um único manipulador podem ser melhoradas com o uso de múltiplos manipuladores. Robôs colaborativos como os manipuladores da Universal Robots (Figura 0.2) ou o robô colaborativo Yumi (Figura 0.3) foram desenvolvidos para trabalhar colaborativamente com seres humanos, dividindo o mesmo espaço de trabalho e que suportam o desenvolvimento de soluções para trabalho em conjunto com seres humanos ou outros robôs. Dito isso, um problema em aberto na robótica são técnicas de manipulação rápida de objetos e manipulação de objetos por múltiplos manipuladores, o que inclui manipulação

de objetos entre humanos e robôs.

A tarefa de manipulação de um objeto com múltiplos robôs é multidisciplinar, envolvendo áreas como identificação de objetos, planejamento de trajetória, prensão robótica e controle em espaço de trabalho, como descrito em (SMITH *et al.*, 2012). Para aplicações colaborativas, é de especial interesse que o robô consiga identificar de forma rápida o objeto a ser manipulado e consiga planejar e acompanhar o posicionamento do objeto para reagir a ambientes dinâmicos e erros humanos de manipulação. Para isso, são utilizados sensores de força, sensores de proximidade, radares e sistemas de visão computacional. Isto posto, a diminuição no custo de sistemas de visão computacional de três dimensões nos últimos anos elevou o interesse na utilização de tais sistemas como ferramentas de solução para os problemas de localização e rastreamento de objetos, como na solução proposta por (SAXENA; DRIEMEYER; NG, 2008).

O problema de manipulação de objetos vai desde a identificação do objeto, alocação das posições iniciais das garras, encontrar uma solução de distribuição dinâmica da massa do objeto, planejamento de trajetória, controle em espaço de trabalho, controle de força, rastreamento do objeto e resposta rápida a mudanças no ambiente (SICILIANO; KHATIB, 2016). Os métodos de controle atuais utilizados na manipulação de objetos normalmente envolvem decomposição de forças, conhecimento da geometria e posição do objeto, comunicação direta entre os robôs e normalmente levam em conta um ambiente estático.

Concomitantemente a isso, nos últimos anos houve uma crescente demanda do mercado por robôs colaborativos e sistemas de visão 3D mais baratos. Alguns desses novos robôs possuem a característica de terem seus controladores dinâmicos internos fechados, sendo possível apenas um controle no nível da cinemática (BENALI *et al.*, 2018). Se mostra interessante o desenvolvimento de um método de projeto e sintonia de controle cinemático para esses manipuladores, sincronização entre eles, e a capacidade de perceber, manipular e rastrear objetos no ambiente de trabalho.

Objetivos gerais

O objetivo geral desse trabalho é estudar, projetar, simular e implementar um sistema de controle em espaço de trabalho para possibilitar que múltiplos manipuladores sejam usados para manipular um objeto conhecido utilizando informações de um sistema de visão.

Objetivos específicos

- Estudo de sistemas robóticos, controle em espaço de trabalho e visão computacional.

- Revisão da literatura sobre prensão robótica, cooperação entre robôs, técnicas de controle cinemático em espaço de trabalho e visão estéreo na robótica.
- Estudar e utilizar simuladores e conjuntos de funções apropriadas a robótica, como V-REP e ROS.
- Construção de um ambiente em simulação e em laboratório para sincronia de robôs, controle cinemático, manipulação de objetos e visão computacional.
- Implementar os controladores cinemáticos encontrados na literatura e propor um método de sintonia para os mesmos.
- Obter dados de posição de manipuladores utilizando visão computacional.
- Rastrear o objeto manipulado e analisar os resultados obtidos.

Organização do trabalho

O trabalho está organizado da seguinte forma: Inicialmente, capítulos de Introdução e Revisão Bibliográfica servem de base para a apresentação do problema e das soluções atuais para o mesmo na literatura. A primeira parte do trabalho, referente aos capítulos 1 à 3 tratam da fundamentação teórica para o problema e solução proposta. Em seguida, na segunda parte do trabalho, referente aos capítulos 4 e 5, o problema e a solução são definidos e o ambiente de simulação e experimentação é detalhado, com os equipamentos utilizados e metodologia. Por fim, experimentos são realizados e métodos de controle cinemático de manipuladores são abordados para solucionar o problema da manipulação de objetos, assim como visão computacional é utilizada para rastrear o objeto manipulado. O trabalho é finalizado no capítulo 6 com uma revisão e análise do trabalho e perspectivas para trabalhos futuros.

Revisão bibliográfica

Os primeiros trabalhos na área de controle cinemático foram apresentados por (PAUL; SHIMANO; MAYER, 1981), onde a matriz Jacobiana de manipuladores e sua inversa foi definida, e uma solução numérica para o problema foi inicialmente proposta por (WOLOVICH; ELLIOTT, 1984). Por meio da matriz Jacobiana, é possível associar a velocidade do efetuador no espaço cartesiano à velocidade das juntas de um manipulador, sendo possível controlar a velocidade do efetuador no espaço cartesiano. Apesar de solucionar o problema de controle cinemático, o método não é aplicável a manipuladores redundantes e apresenta problemas de singularidade quando a matriz Jacobiana não é inversível.

Uma solução para o problema das singularidades foi apresentado por (NAKAMURA; HANAFUSA, 1986a) e (WAMPLER, 1986), que contorna o problema da singularidade ao modificar o resultado da inversa do Jacobiano na proximidade de singularidades, resultando em um desvio na trajetória inicial. Apesar de não resolver o problema da movimentação em uma singularidade, esses métodos se tornaram os mais comumente utilizados e referenciados na área de robótica, apesar de não serem os únicos a resolver o problema.

Métodos alternativos para o problema da cinemática inversa começaram com (GOLDENBERG; APKARIAN; SMITH, 1987), onde se é utilizado um operador recursivo para mapear as transformações das coordenadas cartesianas para as coordenadas de junta. A precisão do método, no entanto, depende da quantidade de passos no cálculo recursivo e o método não é recomendado para cálculos em tempo real. Em estudos mais recentes, (GALICKI, 2005) propõe uma solução de controle baseada em funções de penalidade para o problema da cinemática inversa e uma extensão da solução para evitar obstáculos, porém sua solução é sempre local a depender da descrição dos objetos por funções e uma estabilidade global não é garantida.

Em (PECHEV, 2008) e (AHMED; PECHEV, 2009), é apresentado um novo método para solução da cinemática inversa que não depende de inversão de matrizes utilizando um laço de realimentação que funciona como um filtro e evita singularidades cinemáticas, com uma comparação de desempenho entre os dois métodos. Um método similar foi apresentado em (VARGAS; LEITE; COSTA, 2013), e apesar das vantagens em singularidades e custo computacional, os métodos de Inversa Filtrada apresentados convergem para zero a uma velocidade menor do que o método clássico, e aumentar demais o valor dos ganhos do controlador pode inserir instabilidade no sistema. Outros métodos que utilizam o mesmo princípio de laço de realimentação foram apresentados em (BURRELL et al., 2016) e (DREXLER, 2017), onde diferentes estratégias para a sintonia dos controladores é

apresentada, mas o problema da estabilidade persiste. Alternativas na representação do sistema também são estudadas, como por exemplo em (FIGUEREDO et al., 2013), onde são utilizadas representações de posição e orientação utilizando quatérnions duais para resolver a cinemática inversa com um controlador robusto.

Manipuladores individuais possuem uma capacidade limitada de efetuar tarefas. O aumento na manipulabilidade, assim como a capacidade de manipular objetos pesados ou múltiplos objetos para execução de uma tarefa criaram uma necessidade na pesquisa e uso de múltiplos manipuladores desde o início das pesquisas em robótica. A pesquisa nesse tipo de sistema foi iniciada nos anos 70 e se estende até hoje devido a complexidade do sistema e sua aplicação a novos cenários, como cenários dinâmicos e com objetos desconhecidos.

Trabalhos iniciais na área, apresentados por (NAKANO et al., 1974) e (MASON, 1981), consideravam estratégias de mestre-escravo e controle de força para manipulação de objetos. A abordagem mestre-escravo, no entanto, depende de uma baixa impedância do manipulador escravo para um rastreamento de trajetória suave, e possui problemas em inverter o papel de mestre e escravo a depender da aplicação, o que levou ao desenvolvimento subsequente de controladores que não utilizassem a abordagem mestre-escravo. Outro importante trabalho inicial foi o de (COX; RACKERS; TESAR, 1995), que apresentou os primeiros experimentos de controle automático na área, enfatizando o fato da redundância de atuadores em operações com manipuladores e dividindo os tipos de experimentos com múltiplos manipuladores em manipulação de um único objeto rígido, manipulação de múltiplos objetos e manipulação de objetos flexíveis.

A área de múltiplos manipuladores ganhou mais força na década de 2010, onde (SMITH et al., 2012) fez uma revisão sistemática dos estudos realizados com múltiplos robôs, desde abordagens de controle clássico a abordagens descentralizadas. O artigo divide a grande área de pesquisa de manipuladores duais em categorias de modelagem, controle, planejamento e apreensão, aprendizado e visão e controle visual. É demonstrado que, apesar dos avanços e das diferentes abordagens para o controle dos manipuladores, o problema geral ainda não foi resolvido devido a falta de uma abordagem generalista. O artigo é finalizado com um estudo das tendências futuras na área e serviu de base para uma série de novas pesquisas no controle de múltiplos manipuladores.

Para controlar a manipulação de um objeto corretamente, dois objetivos de controle são normalmente definidos, o controle de trajetória e o controle de força aplicada. Muitas vezes esses objetivos se opõem, por exemplo, quando para seguir uma trajetória com erro mínimo a força aplicada tem que ser muito maior do que a definida, e um equilíbrio entre esses dois objetivos tem que ser alcançado. Os primeiros trabalhos apresentados por (UCHIYAMA; IWASAWA; HAKOMORI, 1987), (UCHIYAMA; DAUCHEZ, 1988) e (UCHIYAMA; DAUCHEZ, 1992) desenvolveram a modelagem dinâmica utilizada por boa parte das técnicas atuais, se baseando em *bastões virtuais* para simular uma cadeia

cinemática única entre os manipuladores e o objeto. Dito isso, as estratégias de controle desenvolvidas para múltiplos manipuladores procuram minimizar o erro de trajetória sem aplicar uma deformação no objeto, utilizando estratégias de controle híbrido (CHOU; YANG; LIN, 2014) (REN *et al.*, 2017) (CHEN; WANG; LIN, 2018), onde dois controladores separados são desenvolvidos e integrados, ou de controle de impedância e admitância (ALYAHMADI; HSIA, 2000) (HECK *et al.*, 2013) (KASERER; GATTRINGER; MÜLLER, 2016) (BJERKENG *et al.*, 2014) (YAN; MU, 2016), onde um controlador central é projetado para controlar força e posição com base na dinâmica ou cinemática do manipulador.

No mesmo tópico, Benali *et al* (BENALI *et al.*, 2018) apresentaram um método de controle cinemático de múltiplos manipuladores que envolve uso de cinemática inversa diferencial e uso de um vetor de forças normal ao objeto para o controle de força e posição em uma trajetória previamente definida. Manipuladores comerciais atuais, como o utilizado em (BENALI *et al.*, 2018) e nesse trabalho, possuem sua malha de controle dinâmico fechada e permitem apenas um controle cinemático do manipulador.

No uso de visão com robótica, Sasaki *et al* (SASAKI *et al.*, 1995) (SASAKI *et al.*, 1995) (SASAKI *et al.*, 1997) foi pioneiro na pesquisa ao utilizar um sistema de visão para obter a geometria do objeto, além de um algoritmo para definir as posições específicas de posição dos robôs após uma estimativa da massa e centro de massa do objeto. Porém, os robôs sempre estão em comunicação uns com os outros e os problemas de prensão e rastreamento em si não são abordados, apenas de posicionamento inicial. A prensão proposta no problema é muito simplificada, sendo similar a uma paleteira.

Já na área de identificação e prensão de objetos desconhecidos com manipuladores, Saxena *et al* (SAXENA; DRIEMEYER; NG, 2008) publicou resultados amplamente citados de um algoritmo de aprendizado de máquina onde, usando um banco de dados de objetos 3D mundanos a aplicações domésticas, foi possível treinar uma rede neural para aprender boas posições de prensão em um ambiente com múltiplos objetos desconhecidos. Boa parte das técnicas atuais de prensão rápida tiveram como base esse trabalho, e as técnicas atuais conseguem identificar e agarrar um objeto em menos de dez segundos. O trabalho porém, foca apenas na identificação de candidatos a prensão e leva em conta apenas um único manipulador.

Entre as técnicas mais recentes, Qujiang Lei *et al* (LEI; WISSE, 2016) propôs um método utilizando nuvem de pontos e uma análise de candidatos a posição dos dedos que leva em consideração área de contato, forças e momentos em objetos. Em (LEI; MEIJER; WISSE, 2017), uma pesquisa sobre os principais métodos de prensão de objetos desconhecidos foi feita e um novo método foi proposto. Ainda assim, esses métodos levam em conta o uso de um único manipulador para identificar os candidatos a prensão.

Com múltiplos manipuladores, (VAHRENKAMP *et al.*, 2009) apresentou um método inovador de identificação e rastreamento de objetos para utilização com um robô humanoide

com múltiplos manipuladores. No entanto, o método de reconhecimento e localização de objetos é baseado em localização de texturas e não tratam de oclusão ou semi oclusão dos objetos selecionados. Mais recentemente, (CLAUDIO; SPINDLER; CHAUMETTE, 2016) implementou um algoritmo de controle cinemático com sensoriamento de visão em malha fechada que permite o correto posicionamento do manipulador mesmo sem uma calibração da câmera ou do manipulador. O sistema de visão utilizado, no entanto, utiliza as mesmas técnicas baseadas em textura e imagens em duas dimensões já previamente selecionadas para localização no ambiente. (RASTEGARPANAH; MARTURI; STOLKIN, 2017) utilizou um controle mestre escravo e localização baseada em imagens em duas dimensões para realizar um controle cinemático e dinâmico do objeto, ainda sem rastreo e baseado em informações sem oclusão de textura. (QU et al., 2017) apresentou um método baseado em marcadores utilizando um sistema de visão em duas dimensões para desenvolver uma trajetória para execução de uma complexa atividade com um robô humanoide. O sistema rastreia os marcadores utilizados, porém a identificação e os objetos são previamente selecionados. Ainda, recentemente, (SCHWARZ et al., 2018) treinou uma rede neural de aprendizado profundo para o reconhecimento de objetos pré determinados para identificação de posições para um efetuador de sucção segurar o objeto. Por fim, (FARIA, 2016) desenvolveu um controle cinemático com controle de força para um robô Baxter de dois braços. O autor utilizou um sistema de visão para localizar a posição inicial de uma caixa para prensão. O trabalho não aborda calibração mão para câmera nem localização global dos manipuladores por possuir um referencial global. A utilização da visão foi simplificada apenas para localização inicial da posição de prensão com ajuda de pontos pré determinados.

Nenhum dos métodos acima é capaz de identificar um objeto ocluso ou semi ocluso, em parte graças a necessidade de uma identificação de textura previamente definida, o que limita sua aplicação a atividades previamente definidas e em ambiente controlado. No entanto, (ARAUJO et al., 2017) apresentou um método de identificação e rastreo de um efetuador de um manipulador baseado em modelos e nuvem de pontos. A utilização de modelos, apesar de uma identificação inicial computacionalmente mais complexa que a identificação de textura, torna possível a identificação de objetos semi oclusos e não depende de texturas.

A identificação, rastreo e manipulação de objetos com múltiplos manipuladores é um problema em aberto na robótica, e soluções para esse problema ajudariam no desenvolvimento de tecnologia com ampla gama de operações. Esse trabalho, trata do desenvolvimento de um sistema de controle cinemático que inclui um método de identificação e rastreo de objetos baseado em visão computacional para viabilizar a manipulação de objetos segundo uma trajetória pré definida.

1 Representação de posição e orientação

Para a apresentação dos conteúdos básicos de robótica e visão computacional, é necessária uma compreensão da representação espacial de objetos no \mathbb{R}^3 . Esse capítulo trata das definições espaciais de posição e orientação utilizadas no resto do trabalho.

Primeiro é dada a definição de pose e pose relativa, com as representações de posição em três dimensões dada em notação de Euler e matrizes ortonormais. Em seguida é discutido singularidades e diferentes tipos de sistemas que definem a localização de um ponto, como vetor-ângulo e quatérnions.

O material aqui descrito é baseado no encontrado em (CORKE, 2016), (SICILIANO; KHATIB, 2016) e (HAMILTON, 2000).

1.1 Pose e pose relativa

Para representar completamente um objeto no espaço tridimensional, são necessárias seis dimensões, sendo três para representar sua posição e três para representar sua orientação. Quanto a posição, a dimensão é linear e infinita, ou seja, um acréscimo ao seu valor move o objeto na direção do eixo continuamente. Acréscimos na orientação, no entanto, são finitos. Logo o objeto voltará a orientação original, ou seja, a dimensão é curva. Se faz necessário um tratamento diferente no tratamento da posição e rotação (CORKE, 2016).

Uma pose é definida como a representação da localização de um objeto no espaço tridimensional, e consiste na posição e orientação do objeto.

Uma pose é representada por ξ . Seja duas poses referenciais A e B . Uma pose relativa ${}^A\xi_B$ descreve uma pose B com relação a A . A pose relativa ${}^A\xi_B$ também pode ser interpretada como descrevendo uma translação, do ponto A ao ponto B . Se o expoente inicial não é definido, se assume que a translação é relativa ao sistema de coordenadas de referência, definido por O . Uma pose pode ser representada por

$$\xi = (\mathbf{p}, \mathbf{R}), \quad (1.1)$$

onde \mathbf{p} é um vetor que representa a posição e \mathbf{R} é uma matriz de rotações que define a rotação da pose.

Uma característica importante de poses relativas é que elas podem ser *compostas*. Se um sistema de coordenadas pode ser representado por outro sistema de coordenadas

por uma pose relativa, eles podem ser aplicados sequencialmente como

$${}^A\xi_C = {}^A\xi_B \oplus {}^B\xi_C, \quad (1.2)$$

em que uma pose C relativa a A pode ser obtida compondo as poses relativas de A para B e de B para C . O operador \oplus representa essa composição de poses. A composição de poses age diferente para posição e rotação. Sejam poses representadas pela Equação 1.1, o operador de composição de poses representa as seguintes operações

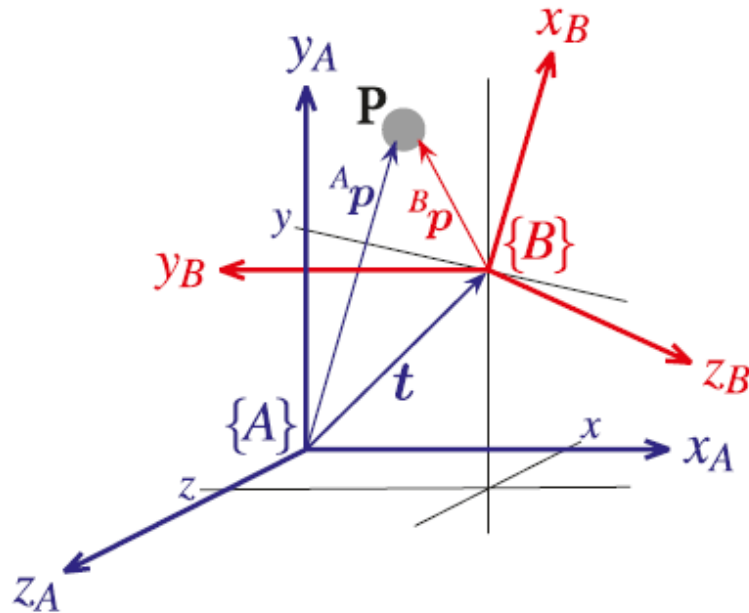
$$({}^A\mathbf{p}_C, {}^A\mathbf{R}_C) = ({}^A\mathbf{p}_B, {}^A\mathbf{R}_B) \oplus ({}^B\mathbf{p}_C, {}^B\mathbf{R}_C), \quad (1.3)$$

$${}^A\mathbf{p}_C = {}^A\mathbf{p}_B + {}^B\mathbf{p}_C, \quad (1.4)$$

$${}^A\mathbf{R}_C = {}^A\mathbf{R}_B {}^B\mathbf{R}_C. \quad (1.5)$$

Um exemplo pode ser visto na Figura 1.1, onde um mesmo ponto P é visto pelo sistema A e pelo sistema B .

Figura 1.1 – Pose em 3 dimensões representado por dois sistemas de coordenadas distintos e um ponto relativo aos dois.



Fonte: (CORKE, 2016)

Os tópicos a seguir abordam representações de ξ utilizadas no trabalho. Inicialmente uma representação da posição em três dimensões é feita, e em seguida a orientação é tratada. Após isso, é introduzido o conceito de matrizes homogêneas e quaternions.

1.2 Posição

Uma posição geométrica em três dimensões pode ser representada por geometria Euclidiana. Um sistema de coordenadas Cartesianas é definido pelos eixos x , y e z , ortogonais entre si, e é tipicamente desenhado com o eixo x na horizontal, o y na vertical e o z ortogonal ao desenho. O ponto de interseção é definido como a origem. Vetores unitários paralelos aos eixos são representados por $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ e $\hat{\mathbf{z}}$. Um ponto \mathbf{p} é representado pelas suas coordenadas (x, y, z) ou por um vetor

$$\mathbf{p} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}, \quad (1.6)$$

uma transformação de posição pode ser feita como uma adição vetorial em sistemas de coordenadas de eixos paralelos. Dado um sistema de coordenadas A com eixos x_A , y_A e z_A e um sistema de coordenadas B com eixos x_B , y_B e z_B , se $\hat{\mathbf{x}}_A = \hat{\mathbf{x}}_B$, $\hat{\mathbf{y}}_A = \hat{\mathbf{y}}_B$ e $\hat{\mathbf{z}}_A = \hat{\mathbf{z}}_B$, então uma transformação de B para A pode ser representada por

$$\begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} + \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \quad (1.7)$$

Ou seja, uma transformação de posição em três dimensões é definida como uma soma vetorial em um referencial comum. Como exemplo, seja um ponto ${}^B\mathbf{p}$ pertencente ao sistema de coordenadas B . Sua representação em A , ${}^A\mathbf{p}$, pode ser representada por

$${}^A\mathbf{p} = {}^A\mathbf{t}_B {}^B\mathbf{p}, \quad (1.8)$$

onde ${}^A\mathbf{t}_B$ representa a transformação homogênea de translação de A para B .

1.3 Rotação

O *teorema de rotação de Euler* mostra que qualquer rotação no \mathbb{R}^3 pode ser considerada uma sequência de rotações sobre eixos de coordenadas diferentes. O teorema também assegura que qualquer rotação pode ser representada por não mais do que três rotações sobre os eixos, desde que a sequência de duas delas não seja sobre o mesmo eixo. Além disso, rotações no \mathbb{R}^3 não são comutativas, ou seja, a ordem na qual as rotações são aplicadas fazem diferença no valor final de rotação.

Foram desenvolvidos diversos métodos para representar rotações. A utilizada nesse trabalho é a mesma utilizada em (CORKE, 2016), utilizando Matriz de Rotação Ortonormal. Também são utilizadas brevemente as representações em vetor-ângulo e Quatérnions, apresentadas em seguida.

1.3.1 Matriz de rotação ortonormal

A orientação de um sistema de coordenadas pode ser representada por meio de seus vetores unitários expressos em termos de um sistema de coordenadas de referência. Cada vetor unitário tem três elementos e eles formam as colunas de uma matriz $\mathbf{R}_{3 \times 3}$ ortonormal ${}^A\mathbf{R}_B$

$$\begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} = {}^A\mathbf{R}_B \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}, \quad (1.9)$$

que transforma o vetor a respeito do sistema de coordenadas B para um vetor a respeito de A .

A matriz \mathbf{R} tem as seguintes propriedades (CORKE, 2016):

- É uma matriz ortonormal, uma vez que cada uma de suas colunas é um vetor unitário e as colunas são ortogonais.
- As colunas são vetores unitários que definem os eixos de rotação do sistema de coordenadas B com respeito a A e são, por definição, de módulo unitário e ortogonais.
- A matriz pertence a um grupo ortogonal especial de dimensão 3, ou $\mathbf{R} \in SO(3) \subset \mathbb{R}^{3 \times 3}$. Isso significa que o produto de duas matrizes dentro do grupo também pertencem ao grupo, assim como sua inversa.
- Seu determinante é 1, o que significa que o módulo de um vetor não muda depois da transformação. Ou seja, $\|{}^B\mathbf{p}\| = \|{}^A\mathbf{p}\|, \forall \theta$ (para todo θ).
- A inversa é a mesma que a transposta, ou seja, $\mathbf{R}^{-1} = \mathbf{R}^T$.

Podemos então representar a matriz por

$$\mathbf{R} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix} = \begin{pmatrix} n & o & a \end{pmatrix}, \quad (1.10)$$

Onde n , o e a são os vetores que representam as direções dos eixos x , y e z do sistema de coordenadas com relação ao sistema de referência.

Sendo assim, como a primeira coluna representa o eixo x , a segunda representa o eixo y e a terceira representa o eixo z , as rotações em torno desses eixos se dão por

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}, \quad (1.11)$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}, \quad (1.12)$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (1.13)$$

O teorema de rotação de Euler requer sucessivas rotações em torno de três eixos de modo que duas rotações sucessivas não podem acontecer em torno do mesmo eixo. Isso significa que qualquer orientação pode ser representada como uma sequência de multiplicações das matrizes \mathbf{R}_x , \mathbf{R}_y e \mathbf{R}_z . Existem, então, duas classes de sequências de rotação: Eulerianas e Cardanas, em nome de Euler e Cardano, respectivamente.

As rotações Eulerianas envolvem repetição, não sucessiva, de rotações sobre um eixo em particular. Enquanto rotações Cardanas são caracterizadas por rotações em torno dos três eixos. Essas rotações podem ser vistas na Tabela 1.

Tabela 1 – Classes de Rotações: As Eulerianas, classificadas por duas rotações em torno do mesmo eixo nas três rotações, e as Cardanas, classificadas pelas rotações nos três eixos.

Rotações	
Eulerianas	Cardanas
XYX	XYZ
XZX	XZY
YXY	YZX
YZY	YXZ
ZXZ	ZXY
ZYZ	ZYX

1.4 Matriz de transformação homogênea

Um vetor $\mathbf{p} = [x, y, z]$ é escrito de forma homogênea como $\tilde{\mathbf{p}} \in \mathbb{P}^3$, $\tilde{\mathbf{p}} = [x_1, x_2, x_3, x_4]$, onde $x = x_1/x_4$, $y = x_2/x_4$, $z = x_3/x_4$ e $x_4 \neq 0$. Normalmente, $x_4 = 1$ e se representa $\tilde{\mathbf{p}} = (x, y, z, 1)$.

Vetores homogêneos tem a importante propriedade de que $\tilde{\mathbf{p}}$ é equivalente a $\lambda\tilde{\mathbf{p}}$ para todo $\lambda \neq 0$, escrito $\tilde{\mathbf{p}} \simeq \lambda\tilde{\mathbf{p}}$. Isso significa que $\tilde{\mathbf{p}}$ representa o mesmo ponto no espaço independente do fator de escala.

Uma transformação homogênea é aquela que transforma os pontos de um sistema de coordenadas referencial A em um sistema de coordenadas B e pode ser representada por

$${}^A\tilde{\mathbf{p}} = \begin{pmatrix} {}^A\mathbf{R}_B & \mathbf{t} \\ 0_{1 \times 3} & 1 \end{pmatrix} {}^B\tilde{\mathbf{p}} = {}^A\mathbf{T}_B {}^B\tilde{\mathbf{p}}, \quad (1.14)$$

onde $\mathbf{t} \in \mathbb{R}^3$ é um vetor que define a origem do vetor B com relação ao vetor A e ${}^A\mathbf{R}_B$ é uma matriz ortonormal 3×3 que descreve a orientação dos eixos de B com relação ao sistema de coordenadas A .

A matriz ${}^A\mathbf{T}_B$ é uma matriz de transformação homogênea 4×4 que pertence ao grupo Euclidiano especial de dimensão 3, ou $\mathbf{T} \in SE(3) \subset \mathbb{R}^{4 \times 4}$.

Outra notação comumente usada é

$$\mathbf{T} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (1.15)$$

A matriz de transformação (1.15) é comumente usada em robótica e visão computacional, e será utilizada nesse trabalho.

Uma matriz de transformação homogênea tem a propriedade de composição das poses previamente discutido. Isso significa que, por exemplo, dado um sistema de coordenadas C , e sistemas de coordenadas A e B , as seguintes equações são válidas

$${}^A\mathbf{T}_C = {}^A\mathbf{T}_B {}^B\mathbf{T}_C, \quad (1.16)$$

$${}^B\mathbf{T}_C = {}^A\mathbf{T}_B^{-1} {}^A\mathbf{T}_C. \quad (1.17)$$

1.5 Singularidades

Um dos problemas com as representações de orientação utilizando três ângulos é a presença de singularidades. Singularidades em robótica são definidas como uma condição causada pelo alinhamento colinear de dois ou mais eixos resultando em movimentos e velocidades imprevisíveis. Em representações de Euler, isso ocorre quando o eixo de rotação do termo do meio se torna paralelo ao eixo de rotação do primeiro ou terceiro termo.

Considere o exemplo em (CORKE, 2016), onde se tem um sistema de rotação representado pela sequência Cardana YZX , dado por

$$\mathbf{R}(\theta_p, \theta_r, \theta_z) = \mathbf{R}_y(\theta_p)\mathbf{R}_z(\theta_r)\mathbf{R}_x(\theta_y), \quad (1.18)$$

no caso em que $\theta_r = \frac{\pi}{2}$, a seguinte identidade se aplica

$$\mathbf{R}_y(\theta)\mathbf{R}_z\left(\frac{\pi}{2}\right) = \mathbf{R}_z\left(\frac{\pi}{2}\right)\mathbf{R}_x(\theta), \quad (1.19)$$

o que leva a

$$\mathbf{R}(\theta_p, \theta_r, \theta_z) = \mathbf{R}_y(\theta_p)\mathbf{R}_z(\theta_r)\mathbf{R}_x(\theta_y) = \mathbf{R}_z\left(\frac{\pi}{2}\right)\mathbf{R}_x(\theta_p + \theta_y), \quad (1.20)$$

não é possível adquirir informações referentes a rotação no eixo y . Um grau de liberdade foi perdido.

Todas as representações de orientação de três ângulos, sejam Eulerianas ou Cardanas, sofrem o problema de singularidade quando dois eixos consecutivos se alinham. Para eliminar esse problema, é necessária a utilização de representações de orientações diferentes. A introdução de um quarto termo elimina o problema de singularidade.

1.6 Representação vetor-ângulo

Dois sistemas de coordenadas de orientação arbitrária estão relacionados por uma única rotação ao redor de um eixo no espaço.

Seja a matriz de rotação $\mathbf{R}(\theta_x, \theta_y, \theta_z)$, daqui pra frente referida apenas como \mathbf{R} , existe um autovetor e um autovalor relacionados a essa matriz \mathbf{R} de forma que

$$\mathbf{R}\omega = \lambda\omega, \quad (1.21)$$

onde ω é o autovetor correspondente ao autovalor λ . Para o caso em que $\lambda = 1$

$$\mathbf{R}\omega = \omega, \quad (1.22)$$

o que implica que o autovetor não é modificado pela rotação. Existe apenas um vetor onde isso acontece, o autovetor no qual a rotação ocorre.

Uma matriz de rotação ortonormal sempre tem um autovalor real $\lambda = 1$ e um par de autovalores complexo $\lambda = \cos(\theta) \pm i \sin(\theta)$ onde θ é o ângulo de rotação.

O problema inverso, converter uma representação vetor-ângulo para uma matriz de rotação, é alcançado pela fórmula de rotação de Rodrigues (SICILIANO; KHATIB, 2016)

$$\mathbf{R} = \mathbf{I}_{3 \times 3} + \sin(\theta)[\hat{\omega}]_{\times} + (1 - \cos(\theta))[\hat{\omega}]_{\times}^2, \quad (1.23)$$

onde $[\hat{\omega}]_{\times}$ é a matriz simétrica oblíqua, dada por

$$\mathbf{S}(\mathbf{v}) = [\hat{\omega}]_{\times} = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}, \mathbf{v} = (x, y, z), \quad (1.24)$$

que é sempre singular, e sua transposta é igual a seu valor negativo. Qualquer matriz pode ser representada como a soma de uma matriz simétrica oblíqua e uma matriz simétrica (CORKE, 2016).

Essa representação de rotação é dada por quatro números, três do vetor de rotação e um do ângulo de rotação. Ela utiliza bem menos números que os nove de uma matriz de rotação.

Melhor ainda, se o vetor direção for representado por um vetor unitário $\hat{\omega}$, o valor de θ pode estar codificado como um escalar que multiplica esse vetor, dado por $\theta\hat{\omega}$. Essas representações são eficientes em termos de armazenamento de dados, porém implicam em problemas pois não são definidas quando $\theta = 0$.

1.7 Quatérnions

Dado o problema de singularidades para representações de orientações com três ângulos apresentado anteriormente, surgiu uma necessidade de um sistema de representação que não contenha tal problema. A representação em quaternions (HAMILTON, 2000) é extremamente útil na robótica e resulta em representações de orientação que não sofrem de singularidades como ângulos de Euler e ângulos fixos sofrem (SICILIANO; KHATIB, 2016).

Um quatérnion ϵ é definido como

$$\epsilon = \epsilon_0 + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k, \quad (1.25)$$

onde $\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3$ são escalares, chamados de parâmetros de Euler, e i, j e k são operadores. Esses operadores satisfazem as seguintes regras combinacionais

$$\begin{aligned} ii = jj = kk &= -1, \\ ij = k, jk = i, ki = j, \\ ji = -k, kj = -i, ik = -j. \end{aligned} \quad (1.26)$$

A soma de dois quaternions é dada pela adição de suas componentes separadamente, então os operadores atuam como separadores. Ainda se tem

$$\mathbf{0} = 0 + 0i + 0j + 0k, \quad (1.27)$$

$$\mathbf{I} = 1 + 0i + 0j + 0k, \quad (1.28)$$

$$\mathbf{I}\epsilon = \epsilon \quad (1.29)$$

$$\begin{aligned} \mathbf{i} &= 0 + 1i + 0j + 0k, \\ \mathbf{j} &= 0 + 0i + 1j + 0k, \\ \mathbf{k} &= 0 + 0i + 0j + 1k, \end{aligned} \quad (1.30)$$

e um quatérnion pode ser escrito na forma

$$\epsilon = \epsilon_0 + \epsilon_1 \mathbf{i} + \epsilon_2 \mathbf{j} + \epsilon_3 \mathbf{k}, \quad (1.31)$$

Produtos de quatérnions são associativos e distributivos, mas não comutativos. Dado dois quatérnions \mathbf{a} e \mathbf{b} , seu produto é dado por

$$\begin{aligned} \mathbf{ab} &= a_0 b_0 - a_1 b_1 - a_2 b_2 - a_3 b_3 \\ &+ (a_0 b_1 + a_1 b_0 + a_2 b_3 - a_3 b_2) i \\ &+ (a_0 b_2 + a_2 b_0 + a_3 b_1 - a_1 b_3) j \\ &+ (a_0 b_3 + a_3 b_0 + a_1 b_2 - a_2 b_1) k. \end{aligned} \quad (1.32)$$

O conjugado de um quatérnion é definido como

$$\tilde{\epsilon} = \epsilon_0 - \epsilon_1 i - \epsilon_2 j - \epsilon_3 k, \quad (1.33)$$

de forma que

$$\epsilon \tilde{\epsilon} = \tilde{\epsilon} \epsilon = \epsilon_0^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 \quad (1.34)$$

Por fim, um quatérnion unitário pode ser definido como

$$\epsilon \tilde{\epsilon} = 1. \quad (1.35)$$

Um vetor é definido em notação de quatérnion quando $\epsilon_0 = 0$. Um vetor $\mathbf{p} = [p_x, p_y, p_z]^T$ é expresso em notação de quatérnion como $\mathbf{p} = p_x i + p_y j + p_z k$.

Finalmente, para um quatérnion unitário ϵ , a operação $\epsilon \mathbf{p} \tilde{\epsilon}$ realiza uma rotação do vetor \mathbf{p} na direção $[\epsilon_1, \epsilon_2, \epsilon_3]^T$.

Quatérnions unitários são também relacionados com representações vetor-ângulo, onde ϵ_0 corresponde ao ângulo de rotação, e ϵ_1, ϵ_2 e ϵ_3 definem o vetor de rotação.

Uma representação de pose comumente utilizada na robótica é do vetor quatérnion (CORKE, 2016), dada por

$$\mathbf{p} = (\mathbf{v}, \mathbf{q}), \quad (1.36)$$

onde a posição é representada pelo vetor \mathbf{v} e a rotação pelo quatérnion \mathbf{q} .

1.8 Transformações de sistemas de orientação

Dependo da abordagem de um problema, pode ser necessário a conversão entre diferentes sistemas de rotação. Uma base comum para essas conversões são matrizes de rotação. Na Tabela 2, retirada de (SICILIANO; KHATIB, 2016), é possível observar as transformações de diferentes sistemas de coordenadas para matrizes de rotação.

Tabela 2 – Matrizes de rotação equivalentes a determinados sistemas de orientação. Abreviações: $c_\theta := \cos(\theta)$, $s_\theta := \sin(\theta)$, $v_\theta := 1 - \cos(\theta)$

Sistema de Orientação	Matriz Rotacional
Ângulos de Euler ZYX $(\alpha, \beta, \gamma)^T$	$\mathbf{R} = \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{pmatrix}$
vetor-ângulo $\theta \hat{\omega}$	$\mathbf{R} = \begin{pmatrix} \omega_x^2 v_\theta + c_\theta & \omega_x \omega_y v_\theta - \omega_z s_\theta & \omega_x \omega_z v_\theta + \omega_y s_\theta \\ \omega_x \omega_y v_\theta + \omega_z s_\theta & \omega_y^2 v_\theta + c_\theta & \omega_y \omega_z v_\theta - \omega_x s_\theta \\ \omega_x \omega_z v_\theta - \omega_y s_\theta & \omega_y \omega_z v_\theta + \omega_x s_\theta & \omega_z^2 v_\theta + c_\theta \end{pmatrix}$
Quatérnion unitário $(\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3)^T$	$\mathbf{R} = \begin{pmatrix} 1 - 2(\epsilon_2^2 + \epsilon_3^2) & 2(\epsilon_1 \epsilon_2 - \epsilon_0 \epsilon_3) & 2(\epsilon_1 \epsilon_3 + \epsilon_0 \epsilon_2) \\ 2(\epsilon_1 \epsilon_2 + \epsilon_0 \epsilon_3) & 1 - 2(\epsilon_1^2 + \epsilon_3^2) & 2(\epsilon_2 \epsilon_3 - \epsilon_0 \epsilon_1) \\ 2(\epsilon_1 \epsilon_3 - \epsilon_0 \epsilon_2) & 2(\epsilon_2 \epsilon_3 + \epsilon_0 \epsilon_1) & 1 - 2(\epsilon_1^2 + \epsilon_2^2) \end{pmatrix}$

1.8.1 Matrizes de rotação para ângulos de Euler ZYX

Seja uma matriz de rotação da forma

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \quad (1.37)$$

sua transformação para ângulos de Euler $ZYX(\alpha, \beta, \gamma)^T$ se da por:

$$\beta = \tan^{-1}(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}), \quad (1.38)$$

$$\alpha = \tan^{-1}\left(\frac{r_{21}}{\cos(\beta)}, \frac{r_{11}}{\cos(\beta)}\right), \quad (1.39)$$

$$\gamma = \tan^{-1}\left(\frac{r_{32}}{\cos(\beta)}, \frac{r_{33}}{\cos(\beta)}\right). \quad (1.40)$$

1.8.2 Matrizes de rotação para vetor-ângulo

Com uma matriz de rotação na forma de 1.37, uma transformação para vetor-ângulo $\theta\hat{\omega}$ se da por

$$\theta = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right), \quad (1.41)$$

$$\hat{\omega} = \frac{1}{2\sin(\theta)} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}. \quad (1.42)$$

1.8.3 Matrizes de rotação para quatérnions unitários

Por fim, seja uma matriz de rotação na forma de 1.37 e um quatérnion dado por 1.33, temos

$$\epsilon_0 = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}}, \quad (1.43)$$

$$\epsilon_1 = \frac{r_{32} - r_{23}}{4\epsilon_0}, \quad (1.44)$$

$$\epsilon_2 = \frac{r_{13} - r_{31}}{4\epsilon_0}, \quad (1.45)$$

$$\epsilon_3 = \frac{r_{21} - r_{12}}{4\epsilon_0}. \quad (1.46)$$

1.9 Conclusões

O capítulo tratou da definição de poses para representação de posição e orientação de um ponto ou objeto no espaço tridimensional, utilizando principalmente a notação de matrizes ortonormais, que tem a característica de poderem ser compostas. Ainda levando em conta a representação, diferentes métodos de representação foram abordados, utilizados nesse trabalho, e suas qualidades e limitantes.

Foi abordado o problema de singularidades em algumas dessas representações, que para aplicações de robótica são um limitante considerável. Por fim, foi abordado transformações entre os diferentes sistemas de coordenadas, de modo que uma pose possa ser definida de modo equivalente independente do sistema, desde que essa pose não se encontre em uma singularidade no sistema onde foi apresentada.

2 Manipulador série

Manipuladores na robótica podem ser classificados em manipuladores paralelo ou manipuladores em série. Manipuladores série consistem em uma única base e uma série de juntas atuadas ligadas por articulações, enquanto um manipulador paralelo é composto de dois ou mais manipuladores série cujos efetuadores estão anexados a uma mesma base, como em manipuladores lineares (*pick and place*).

Manipuladores série são amplamente utilizados na indústria e em pesquisa. O tipo mais comum de manipulador robótico é um braço com seis graus de liberdade composto por uma série de conexões rígidas e juntas atuadas (CORKE, 2016). Esse tipo de braço é utilizado nesse trabalho.

Sendo mais específico, um manipulador série é definido como um robô com uma série de juntas atuadas ligadas por articulações, comumente rígidas, que utiliza seu efetuador ou ferramenta para realizar tarefas de manipulação e/ou força.

Uma abordagem sobre a cinemática de um manipulador, as relações geométricas entre os ângulos das juntas do robô e a pose do efetuador será estudada. Um estudo das relações de velocidades entre as coordenadas de juntas e coordenadas cartesianas é realizado. Em seguida uma abordagem rápida sobre a dinâmica física do manipulador é introduzida.

A fundamentação desse capítulo é baseada nos materiais encontrados em (CORKE, 2016) e (SICILIANO; KHATIB, 2016).

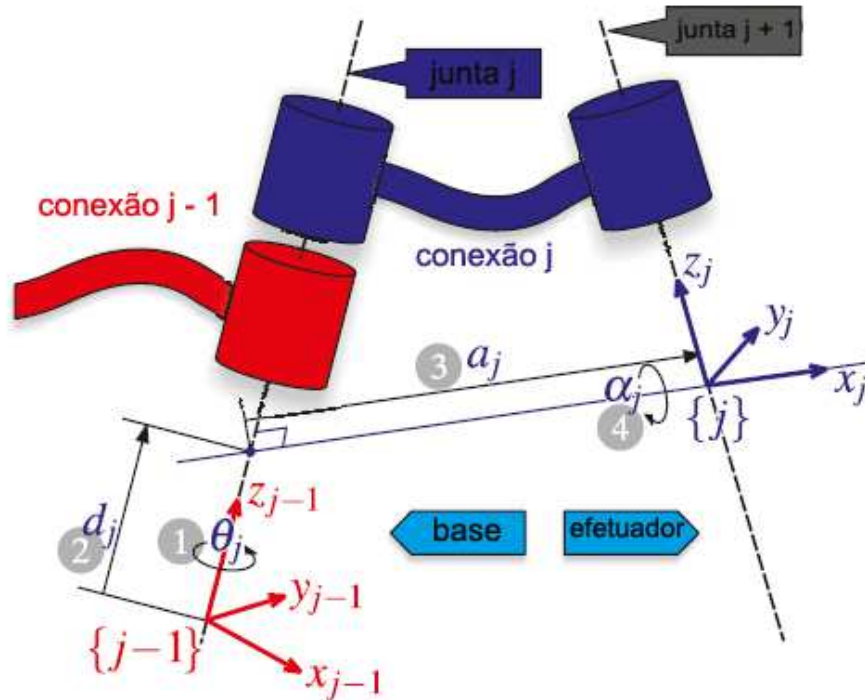
2.1 Descrição geométrica

Uma maneira sistemática de descrever a geometria de uma série de articulações é usar a notação Denavit-Hartenberg, definida a seguir:

Notação Denavit-Hartenberg: Para um manipulador com N juntas numeradas de 1 a N , existem $N + 1$ conexões, numeradas de 0 a N . A junta j conecta a conexão $l - 1$ a conexão l e as move relativa uma a outra. Disso, se tira que uma conexão l conecta a junta j a junta $j + 1$. A conexão 0 é a base do robô, tipicamente fixa, e a conexão N é a que transporta o efetuador.

Na notação de Denavit-Hartenberg, uma conexão define a relação espacial entre dois eixos de junta, como na Figura 2.1. Uma conexão é determinada com quatro parâmetros. A relação entre dois sistemas de coordenadas de conexões é de seis parâmetros, três para translação e três para rotação. Os parâmetros de Denavit-Hartenberg são sintetizados na

Figura 2.1 – Representação em Denavit-Hartenberg.



Fonte: Adaptado de (CORKE, 2016).

Tabela 3.

Tabela 3 – Denavit-Hartenberg

Parâmetros de Denavit-Hartenberg		
Ângulo de junta	θ_j	Ângulo entre o eixo x_{j-1} e o eixo x_j rotacionado no eixo z_{j-1} .
Deslocamento da conexão	d_j	Distância entre a origem da conexão $j - 1$ ao eixo x_j saindo do eixo z_{j-1} .
Tamanho da conexão	a_j	Distância entre os eixos z_{j-1} e z_j ao longo do eixo x_j ; para eixos que se interceptam, é paralelo a $\hat{z}_{j-1} \times \hat{z}_j$.
Ângulo de conexão	α_j	Ângulo do eixo z_{j-1} ao eixo z_j pelo eixo x_j .

O sistema de coordenadas j é anexado a ponta da conexão j . O eixo z do sistema de coordenadas j é alinhado com o eixo da junta $j + 1$.

A transformação de sistemas de coordenadas da conexão $j - 1$ para o sistema de coordenadas j é feita por meio de rotações e translações elementares, dado por

$${}^{j-1}\xi_j(\theta_j, d_j, a_j, \alpha_j) = \mathbf{R}_z(\theta_j) \oplus \mathbf{T}_z(d_j) \oplus \mathbf{T}_x(a_j) \oplus \mathbf{R}_x(\alpha_j), \quad (2.1)$$

que pode ser expandido em forma de matriz homogênea como

$${}^{j-1}\mathbf{A}_j = \begin{pmatrix} \cos(\theta_j) & -\sin(\theta_j)\cos(\alpha_j) & \sin(\theta_j)\sin(\alpha_j) & a_j\cos(\theta_j) \\ \sin(\theta_j) & \cos(\theta_j)\cos(\alpha_j) & -\cos(\theta_j)\sin(\alpha_j) & \alpha_j\sin(\theta_j) \\ 0 & \sin(\alpha_j) & \cos(\alpha_j) & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.2)$$

Os parâmetros α_j e a_j são sempre constantes. Para o caso de uma junta de revolução, θ_j é a variável da junta e d_j é constante, enquanto para uma junta prismática, d_j é variável, θ_j é constante e $\alpha_j = 0$. A coordenada variável pode ser generalizada no tipo de junta como q_j .

Para um robô de N eixos, o conjunto de coordenadas de junta generalizadas $\mathbf{q} \in \mathcal{C}$, onde $\mathcal{C} \subset \mathbb{R}^N$ é chamado de espaço de juntas ou espaço de configuração. Para um robô de seis juntas de revolução, utilizado nesse trabalho, as coordenadas de junta $\mathcal{C} \subset (\mathbb{S}^1)^N$ são chamadas de ângulos de junta.

2.2 Cinemática

Cinemática é o ramo da mecânica que estuda o movimento de um corpo, ou sistema de corpos, sem considerar sua massa ou as forças que atuam nele.

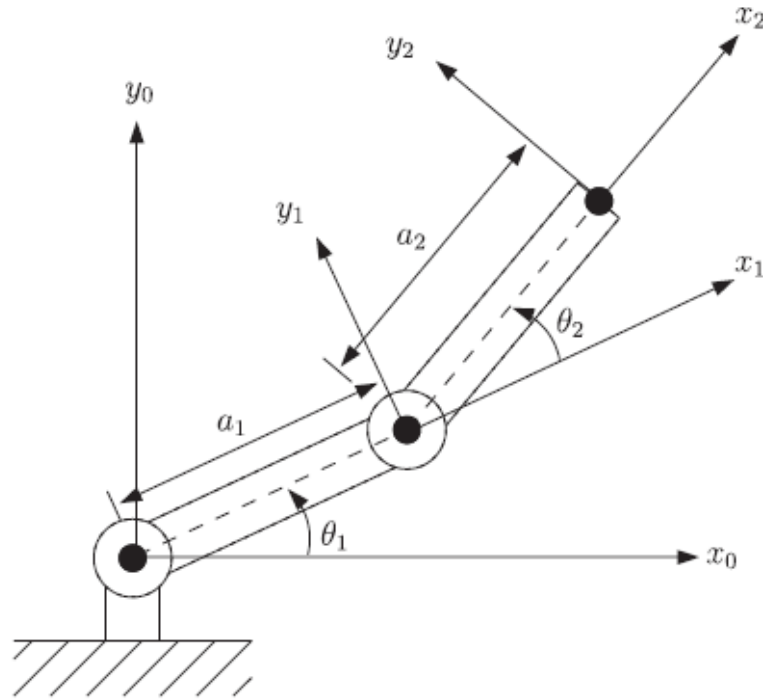
Um braço mecânico é uma cadeia cinemática aberta composta por juntas e articulações. As juntas podem ser classificadas de várias maneiras, sendo as mais comuns as juntas translacionais, prismáticas ou cilíndricas. O movimento dessa junta muda a pose relativa das conexões à qual ela se conecta. A base da série de conexões geralmente é fixa, enquanto a ponta é livre para se movimentar no espaço e possui a ferramenta utilizada para realizar o trabalho que se deseja.

Nesse âmbito, se define *cinemática direta* como a relação de posição de juntas, ou ângulo de juntas, no caso de um robô apenas com juntas rotacionais, que determina a posição do efetuador no espaço. *Cinemática inversa*, por sua vez, é o processo que determina quais posições das juntas são necessárias para que o efetuador se encontre em determinada posição do espaço (CORKE, 2016).

2.2.1 Cinemática direta

Considere um robô de duas articulações, como na Figura 2.2. A pose do seu efetuador pode ser descrita como uma sequência de poses relativas, sendo uma rotação em torno do primeiro eixo e uma translação no sentido da conexão, seguido de outra rotação e uma translação, como

Figura 2.2 – Robô com duas articulações.



Fonte: (SICILIANO; KHATIB, 2016)

$${}^0\xi_N = \mathbf{R}_z(q_1) \oplus \mathbf{T}_x(a_1) \oplus \mathbf{R}_z(q_2) \oplus \mathbf{T}_x(a_2). \quad (2.3)$$

Um robô como esse tem dois graus de liberdade e algumas posições são alcançáveis com duas soluções no espaço de juntas. A configuração espacial desse robô é $\mathcal{C} = \mathbb{S}^1 \times \mathbb{S}^1$. Isso é suficiente para efetuar um espaço de trabalho $\mathcal{T} \subset \mathbb{R}^2$ uma vez que $\dim(\mathcal{C}) = \dim(\mathcal{T})$. No entanto, se orientação for considerada, temos $\dim(\mathcal{C}) < \dim(\mathcal{T})$ e o robô é sub atuado.

Um robô típico que trabalha em três dimensões tem uma área de trabalho $\mathcal{T} \subset SE(3)$ que permite uma posição e orientação arbitrária do efetuador. Para isso, sua configuração espacial deve ser $\dim(\mathcal{C}) > \dim(\mathcal{T})$, o que é alcançado com um robô de seis ou mais juntas. O mesmo procedimento anterior pode ser feito para determinar a cinemática direta de um robô de seis juntas, mas se torna trabalhoso a medida que o número de juntas aumenta. Para simplificar isso, a notação Denavit-Hartenberg é utilizada.

Em termos da notação, a função das coordenadas de junta é simplesmente a composição das poses relativas de cada articulação

$${}^0\xi_N = \mathcal{K}(\mathbf{q}; \theta, d, a, \alpha) = {}^0\xi_1 \oplus {}^1\xi_2 \cdots {}^{N-1}\xi_N. \quad (2.4)$$

2.2.2 Cinemática inversa

O problema inverso da cinemática direta, ou seja, encontrar um valor no espaço de juntas para uma pose ξ do efetuador, é chamado de cinemática inversa. A cinemática inversa pode ser escrita de forma funcional como

$$\mathbf{q} = \mathcal{K}^{-1}(\xi), \quad (2.5)$$

e no geral é uma função que não possui solução única, ou seja, diversos vetores de coordenadas de juntas \mathbf{q} resultam na mesma pose de efetuador. Isso é fácil de ser observado ao verificar a Equação 2.2. A inversa de funções trigonométricas implica em diversas condições de ângulos precisarem ser satisfeitas, e mais de um ângulo pode ser utilizado para encontrar a solução.

Duas soluções gerais podem ser utilizadas para determinar a cinemática inversa. Inicialmente, uma solução analítica pode ser determinada utilizando métodos algébricos e geométricos. No entanto, seu equacionamento numérico se torna cada vez mais complexo a medida que o número de juntas robóticas aumenta, e para alguns manipuladores seriais, tal solução analítica não existe. Como alternativa, uma solução numérica pode ser usada.

A solução numérica pode ser vista como um problema de otimização, no qual, ao se ajustar as coordenadas de juntas na cinemática direta, uma solução cada vez mais próxima da pose desejada pode ser encontrada. A otimização procura minimizar o erro entre a solução de cinemática direta e a pose desejada ξ^*

$$\mathbf{q}^* = \arg \min_{\mathbf{q} \in \mathbb{Q} \subset \mathbb{R}^N} \|\mathcal{K}(\mathbf{q}) \ominus \xi^*\|, \quad (2.6)$$

onde $\mathbf{q} \in \mathbb{Q} \subset \mathbb{R}^N$ é o conjunto de N ângulos de junta no espaço de junta que são solução para a cinemática. Aqui, o operador \ominus é definido como o inverso do operador \oplus , subtraindo um vetor na posição e realizando uma multiplicação pela transposta da matriz de rotação para definir a rotação. Com múltiplas soluções, a solução encontrada utilizando esse método depende da escolha inicial de \mathbf{q} , uma vez que o método de minimização converge para a solução mais próxima.

2.3 Velocidade cinemática

O efetuador de um robô se move no espaço cartesiano com uma velocidade de translação e rotação, ou velocidade espacial. Essa velocidade, no entanto, é consequência das velocidades individuais das juntas do robô.

2.3.1 Jacobiano geométrico

A pose em terceira dimensão do efetuador, $\xi \in SE(3)$, tem uma taxa de variação representada por um vetor de velocidade espacial com as três velocidades translacionais e três velocidades rotacionais. Seja $\mathbf{p} \in \mathbb{R}^6$ a posição do efetuador no espaço cartesiano, definido por

$$\mathbf{p} = [x, y, z, \phi, \theta, \rho]^T. \quad (2.7)$$

Essa posição no espaço cartesiano, como demonstrado pela cinemática direta, é função do vetor de juntas \mathbf{q}

$$\mathbf{p} = \mathcal{K}(\mathbf{q}), \quad (2.8)$$

que é uma função não linear. Agora, seja a velocidade no espaço cartesiano definida como

$$\frac{d\mathbf{p}}{dt} = \dot{\mathbf{p}} = [\dot{x}, \dot{y}, \dot{z}, \omega_\phi, \omega_\theta, \omega_\rho]^T, \quad (2.9)$$

o que leva a

$$\frac{d\mathbf{p}}{dt} = \frac{d\mathcal{K}(\mathbf{q})}{d\mathbf{q}} \frac{d\mathbf{q}}{dt}, \quad (2.10)$$

e, por fim, ao Jacobiano geométrico

$$\mathbf{J}_G(\mathbf{q}) = \frac{d\mathbf{p}}{d\mathbf{q}}, \quad (2.11)$$

O Jacobiano $\mathbf{J}_G(\mathbf{q})$ é uma linearização da taxa de variação da pose cartesiana do efetuador em relação a variação das juntas. Com uma simples manipulação de (2.11), se tem

$$\dot{\mathbf{p}} = \mathbf{J}_G(\mathbf{q})\dot{\mathbf{q}}. \quad (2.12)$$

O Jacobiano relaciona a velocidade do efetuador no espaço cartesiano com a velocidade no espaço de juntas e é função das coordenadas de juntas. A matriz $\mathbf{J}_G(\mathbf{q}) \in \mathbb{R}^{6 \times N}$, onde N é o número de juntas do manipulador.

A equação (2.12) é também chamada de *Equação de Cinemática Direta Instantânea* (SICILIANO; KHATIB, 2016).

2.3.2 Jacobiano analítico

A equação (2.9) estabelece as velocidades do efetuador no espaço cartesiano como a junção das velocidades cartesianas $(\dot{x}, \dot{y}, \dot{z})$ e das velocidades angulares $(\omega_x, \omega_y, \omega_z)$. A

velocidade angular representa a variação angular da referência do efetuador com relação a referência de base. Entretanto, para engenharia é muito mais intuitivo usar a relação de variação angular da referência do efetuador com relação a ele mesmo. Assim, é possível relacionar a velocidade angular ω com a velocidade angular da referência do efetuador como

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \mathbf{B}(\phi, \theta, \rho) \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\rho} \end{pmatrix}, \quad (2.13)$$

onde (ϕ, θ, ρ) são as rotações nos eixos (x, y, z) do efetuador de acordo com o padrão aeronáutico, utilizando os ângulos de Euler ZYX .

A matriz $\mathbf{B} \in \mathbb{R}^{3 \times 3}$ representa a transformação de rotação da base para o efetuador. Definindo

$$\mathbf{\Gamma} = (\phi, \theta, \rho)^T, \quad (2.14)$$

temos

$$\mathbf{B}(\mathbf{\Gamma}) = {}^O \mathbf{R}_{EE}. \quad (2.15)$$

Assim, definindo o vetor de velocidades do efetuador no espaço cartesiano como

$$\dot{\mathbf{p}}' = [\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\rho}], \quad (2.16)$$

temos, do mesmo modo que na Equação 2.12,

$$\dot{\mathbf{p}}' = \mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}, \quad (2.17)$$

onde o Jacobiano analítico $\mathbf{J}_A(\mathbf{q})$ é relacionado com o Jacobiano Geométrico $\mathbf{J}_G(\mathbf{q})$ por

$$\mathbf{J}_A(\mathbf{q}) = \begin{pmatrix} \mathbf{I}^{3 \times 3} & \mathbf{0}^{3 \times 3} \\ \mathbf{0}^{3 \times 3} & \mathbf{B}^{-1}(\mathbf{\Gamma}) \end{pmatrix} \mathbf{J}_G(\mathbf{q}). \quad (2.18)$$

2.4 Dinâmica

Dinâmicas de robôs tratam da movimentação do robô relacionando a atuação das juntas, suas acelerações e as forças envolvidas. Equações dinâmicas de movimento formam a base para um número de algoritmos computacionais que tratam do design mecânico, de simulações e do controle do robô (SICILIANO; KHATIB, 2016). O movimento do efetuador

é uma composição da movimentação de cada conexão, que por sua vez são movimentados pelas forças e torques aplicados pelas juntas (CORKE, 2016).

Como controle dinâmico não é o foco desse trabalho, essa sessão abordará brevemente os seguintes tópicos, seguindo a estrutura de (CORKE, 2016): Primeiro um típico modelo e controle independente de junta é apresentado, enfatizando as dificuldades de um controle independente dado a ação de forças externas. Em seguida, é apresentado as equações da dinâmica de corpo rígido de um manipulador, com as forças de acoplamento, atrito, alavancas e suas interações.

2.4.1 Modelo e controle de junta independente

O modelo completo do sistema de cada junta do manipulador comumente envolve o motor para gerar torque, uma caixa de transmissão para amplificar o torque e reduzir o efeito da carga, e um *encoder* para realimentar posição e velocidade ao sistema. A inércia da carga é variável pois, do ponto de vista da junta, diferentes posições das juntas vão configurar diferentes centros de massa, forças centrífugas, etc.

De modo simplificado, o torque gerado pelo motor é proporcional a corrente aplicada, que é controlada pelo driver com um amplificador ao se aplicar um tensão u

$$i_m = K_a u, \quad (2.19)$$

$$\tau_m = K_m i_m, \quad (2.20)$$

e, dado atrito, massa das conexões e caixa de transmissão, a equação de torque do motor pode ser escrita como

$$K_m K_a u - B' \omega - \tau'_C(\omega) - \frac{\tau_d(\mathbf{q})}{G} = J' \dot{\omega} \quad (2.21)$$

Além disso, B' , τ'_C e J' são efetivamente o atrito viscoso total, o atrito de Coulomb total e a inercia total do motor, caixa de transmissão e carga. Eles são definidos pelos coeficientes constantes (m) e os coeficientes da carga (l), afetados pela caixa de transmissão, de coeficiente G ,

$$B' = B_m + \frac{B_l}{G^2}, \tau'_C = \tau_{C,m} + \frac{\tau'_{C,l}}{G}, J' = J_m + \frac{J_l}{G^2} \quad (2.22)$$

A equação 2.21 pode ser linearizada, igualando suas constantes aditivas a zero

$$J' \dot{\omega} + B' \omega = K_m K_a u, \quad (2.23)$$

e aplicando a transformação de Laplace

$$sJ'\Omega(s) + B'\Omega(s) = K_m K_a U(s), \quad (2.24)$$

onde $\Omega(s)$ e $U(s)$ são as transformações de Laplace dos sinais $\omega(t)$ e $u(t)$. Temos então, a função de transferência

$$\frac{\Omega(s)}{U(s)} = \frac{K_m K_a}{J's + B'}, \quad (2.25)$$

relacionando a velocidade do motor a entrada de controle, com um único polo em $s = -\frac{B'}{J'}$.

Assim, comumente o controle individual da junta é realizado por um controlador PI de entrada

$$u = (K_p + \frac{K_i}{s})(\dot{q}^* - \dot{q}), \quad (2.26)$$

ou um controle com *feedforward* de velocidade, dado por

$$u = K_p(\dot{q}^* - \dot{q}) + \dot{q}^* \quad (2.27)$$

2.4.2 Dinâmica de corpo rígido

Considere um motor que atua em uma junta de revolução j de um manipulador série. Da notação de Denavit-Hartenberg (2.2), sabemos que a junta j conecta as conexões $j - 1$ e j . O motor exerce um torque que faz com que a conexão j acelere rotacionalmente, mas também exerce um torque de reação na conexão $j - 1$. A ação da gravidade em cada conexão e as forças giroscópicas das conexões rotacionando também influenciam cada junta. A inércia experimentada pelo motor é uma função da configuração das conexões e de suas velocidades e acelerações (CORKE, 2016).

A força atuante sobre uma série de conexões de um manipulador pode ser escrita da seguinte forma:

$$\mathbf{Q} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{J}_{\mathbf{G}}(\mathbf{q})^T \mathbf{W}, \quad (2.28)$$

onde \mathbf{q} , $\dot{\mathbf{q}}$ e $\ddot{\mathbf{q}}$ são, respectivamente, o vetor de coordenadas de juntas, as velocidades das juntas e as acelerações de juntas, \mathbf{M} é matriz de inércia do espaço de juntas, \mathbf{C} é a matriz de Coriolis e forças de acoplamento, \mathbf{F} é a matriz de forças de atrito, \mathbf{G} é a carga gravitacional, e \mathbf{Q} é vetor de forças atuantes generalizadas atuando sobre as coordenadas generalizadas \mathbf{q} . O último termo \mathbf{W} representa uma alavanca aplicada ao efetuador em forças de juntas, enquanto $\mathbf{J}_{\mathbf{G}}$ é a matriz de jacobianos do manipulador.

A Equação 2.28 descreve a dinâmica de corpo rígido do manipulador. Dado pose, velocidade e aceleração, é possível computar as forças ou torques necessários a serem aplicados nas juntas. Essas equações podem ser encontradas com métodos como as equações de movimento de Euler e segunda lei de Newton, ou com uma abordagem baseada em energia, como as equações de Lagrange. Os elementos das matrizes \mathbf{M} , \mathbf{C} , \mathbf{F} , e \mathbf{G} são funções dos parâmetros cinemáticos das conexões $(\theta_j, d_j, a_j, \alpha_j)$ e parâmetros inerciais. Os parâmetros inerciais incluem a massa da conexão m_j , o centro de massa em relação ao seu sistema de coordenadas e as conexões a qual se conecta r_j , os momentos de inércia da conexão com relação ao centro de massa, o coeficiente de inércia do motor, o coeficiente de atrito do motor, o atrito de Coulomb, taxa de redução de transmissão, etc.

2.5 Conclusões

Esse capítulo tratou de apresentar de maneira geral os conteúdos necessários para a utilização de manipuladores série. Primeiro um manipulador série foi definido utilizando a notação de Denavit-Hartenberg e como a série de juntas articuladas de um manipulador pode ser representada por uma série de poses relativas dessas juntas, de modo que a posição do efetuador possa ser encontrada com a composição dessas poses, na cinemática direta.

O problema contrário, a cinemática inversa, foi abordado em seguida, levando em conta suas dificuldades particulares e o fato de que nem sempre é possível uma solução analítica para um determinado manipulador, sendo necessário uma abordagem numérica. Nisso, a cinemática de velocidade foi apresentada e foi definido o jacobiano geométrico e analítico, que relaciona as velocidades das juntas do manipulador com a velocidade cartesiana do efetuador. O jacobiano é uma importante ferramenta tanto na cinemática inversa quanto no controle cinemático em espaço de trabalho, utilizado nesse trabalho.

Por fim, foi realizada uma breve abordagem sobre a dinâmica do manipulador e as forças envolvidas no controle de seu movimento e dos motores, para introdução do tema.

3 Visão computacional

Visão computacional é a área da computação que extrai informações de sensores de sistemas de visão. É uma área já estabelecida, onde diversas técnicas foram desenvolvidas em processamento de imagens para essa extração. A redução no custo de sistemas de visão e o aumento de poder de processamento permitiu a utilização cada vez maior de visão computacional em conjunto com a robótica.

Nesse capítulo, uma introdução em visão computacional é abordada, com definições de imagens e representações espaciais em imagens. Em seguida, é abordado a utilização de sistemas de visão estérea e nuvem de pontos, finalizando com localização de objetos baseado em modelos e um tratamento de filtro de partículas para nuvem de pontos.

As definições de imagens, visão estérea e nuvem de pontos são baseadas em (CORKE, 2016), enquanto o casamento das nuvens de pontos com filtro de partículas é baseado em (ARAUJO, 2017).

3.1 Imagens

Uma imagem de uma foto é uma superfície bidimensional que é a projeção, por meio de uma câmera, do mundo tridimensional. A informação de profundidade é perdida e não é possível diferenciar um objeto grande à distância ou um objeto menor e mais próximo em uma imagem. Essa transformação de três a duas dimensões é chamada de projeção em perspectiva.

Um modelo clássico usado em visão computacional é o da Figura 3.1. Os raios de luz convergem na origem do sistema de coordenadas da câmera C e uma imagem é projetada no plano da imagem localizado em $z = f$. O eixo z intersecta o plano da imagem no ponto principal, ou ponto de foco, que é a origem do sistema de coordenadas da imagem 2D.

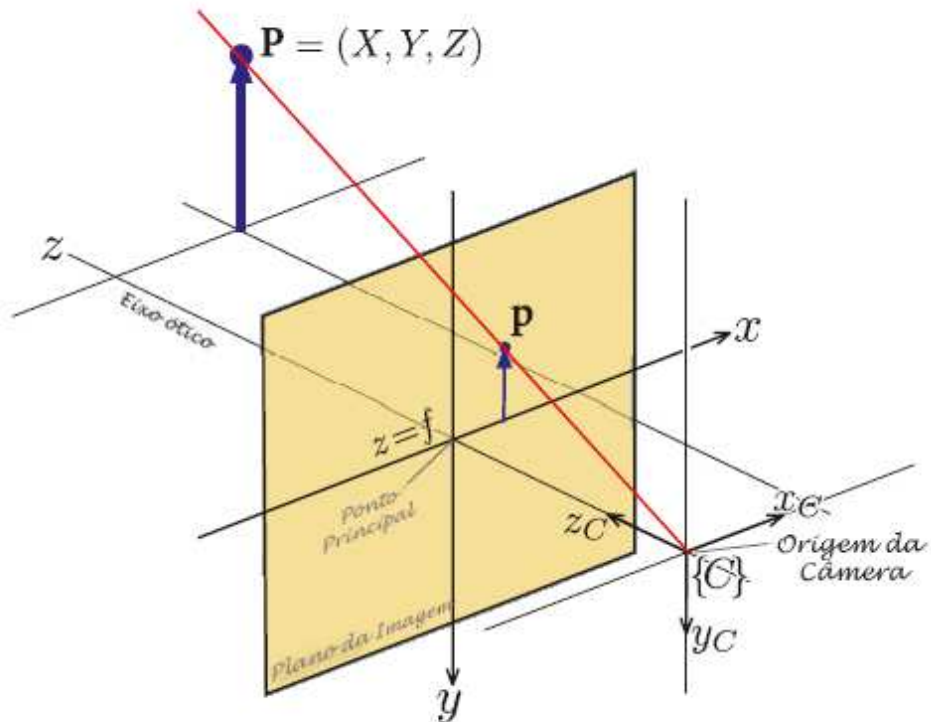
Um ponto nas coordenadas globais $\mathbf{P} = (X, Y, Z)$ é projetado para o ponto de imagem $\mathbf{p} = (x, y)$ por

$$x = f \frac{X}{Z}, y = f \frac{Y}{Z}, \quad (3.1)$$

que é uma transformação de projeção, ou projeção de perspectiva.

O plano da imagem é uma matriz $W \times H$ de elementos sensíveis à luz que correspondem diretamente aos elementos da imagem (ou pixels) da imagem como mostrado na Figura 3.2. As coordenadas de pixel são um vetor de duas dimensões (u, v) de inteiros

Figura 3.1 – Modelo de Projeção Central.



Fonte: Adaptado de (CORKE, 2016).

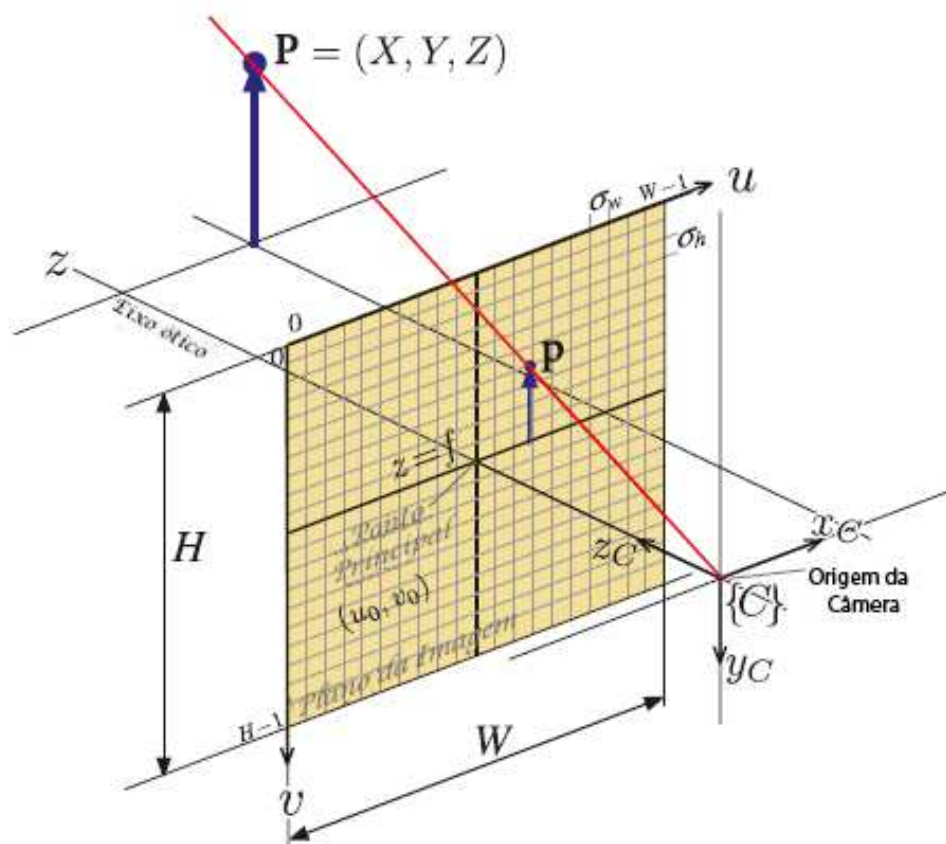
não negativos nessa matriz. Por convenção, a origem da imagem está no canto superior esquerdo do plano.

Cada pixel tem valores determinados com a convenção de cores utilizada para representar a imagem. Por exemplo, no sistema RGB cada pixel tem seu valor de intensidade de vermelho, verde e azul, enquanto no sistema HSV as cores são representadas por matiz, saturação e valor. Mais informações sobre representações de luz e cor podem ser encontradas em (CORKE, 2016) ou livros específicos de tratamento de imagens.

Assim, uma imagem é definida como uma matriz $M \in \mathbb{R}^{W \times H \times n}$, sendo n o número de valores utilizados para representar o pixel no sistema (no caso de RGB, três valores, para as cores vermelho, verde e azul). Essa discretização da imagem em uma matriz pode ser vista na Figura 3.2.

Uma vez que uma imagem foi expressa em termos matemáticos, é possível processar e interpretar as informações obtidas nela. Tratamento e processamento de imagens é uma área bastante abrangente e desenvolvida na literatura, e mais informações sobre o tema podem ser encontradas em livros específicos.

Figura 3.2 – Imagem Discretizada.



Fonte: Adaptado de (CORKE, 2016).

3.2 Visão estéreo

Uma das formas de recuperar a estrutura tridimensional do mundo é a utilização de visão estéreo. É uma técnica onde, a partir de duas imagens de pontos de vista diferentes, é possível obter essa estimaco de profundidade, como mostrado na Figura 3.3. Com o conhecimento da pose das duas câmeras e duas imagens delas obtidas, é possível utilizar geometria epipolar e realizar uma triangulao para encontrar a profundidade daquele ponto no espao em trs dimenses. O conjunto de todos os pontos obtidos no espao é chamado de nuvem de pontos.

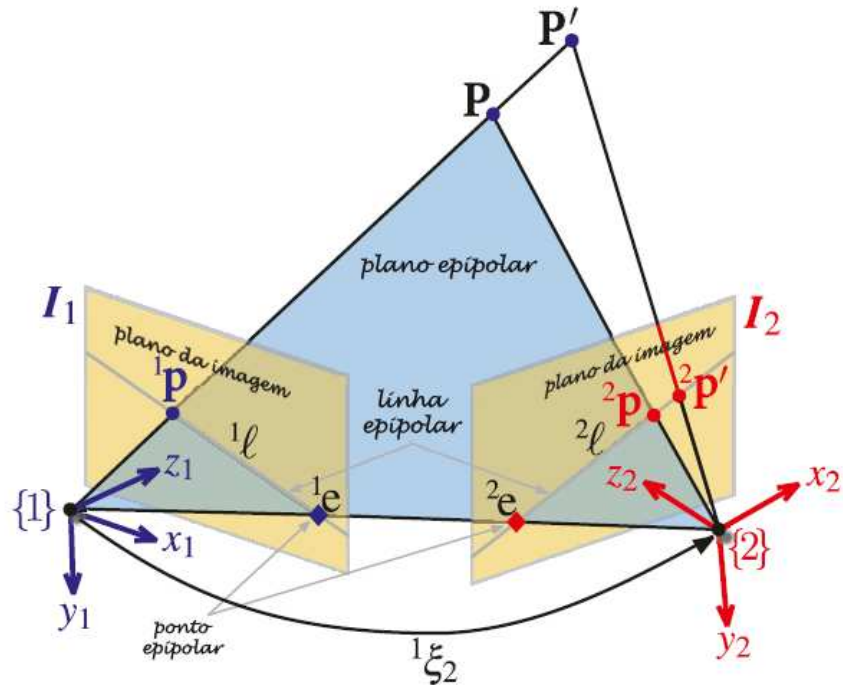
3.2.1 Geometria epipolar

Seja um ponto P no espao, observado a partir de dois pontos de vista diferentes, como na Figura 3.3. Existe uma relao geomtrica que relaciona a pose dos dois pontos de vista e o ponto P . Os dois pontos de vista diferentes podem ser representados por duas câmeras fixas em dois locais ou uma câmara em movimento tirando uma foto de dois pontos de vista diferentes. O centro de cada câmara ($\{1\}$ e $\{2\}$) e o ponto global P

definem um plano no espaço - o plano epipolar. O ponto global \mathbf{P} é projetado nos planos de imagem das duas câmeras nos pontos ${}^1\mathbf{p}$ e ${}^2\mathbf{p}$ (CORKE, 2016).

Na Figura 3.3, o ponto da imagem ${}^1\mathbf{e}$ é a projeção da pose da câmera dois visto pela câmera um. De mesmo modo, o ponto ${}^1\mathbf{p}$ é uma projeção do ponto \mathbf{P} no plano da imagem da câmera um. O centro da câmera, ${}^1\mathbf{e}$ e ${}^1\mathbf{p}$ definem o plano epipolar e a linha epipolar 1l . O mesmo ocorre visto da segunda câmera. Por definição, o ponto global se localiza na interseção das duas projeções.

Figura 3.3 – Visão Estéreo. O ponto \mathbf{P} é projetado nas duas câmeras e, com a posição conhecida de ambas as câmeras, é possível calcular uma triangulação para localizar \mathbf{P} no espaço.



Fonte: Adaptado de (CORKE, 2016).

Assim, dado um ponto em uma imagem, seu conjugado é forçado a se encontrar em uma determinada linha em outra imagem. Isso pode ser expresso por

$${}^2\tilde{\mathbf{p}}^T \mathbf{F} {}^1\tilde{\mathbf{p}} = 0, \tag{3.2}$$

onde ${}^1\tilde{\mathbf{p}}$ e ${}^2\tilde{\mathbf{p}}$ são os pontos ${}^1\mathbf{p}$ e ${}^2\mathbf{p}$ expressos na forma de matriz homogênea e $\mathbf{F} \subset \mathbb{R}^{3 \times 3}$ é a matriz fundamental.

A matriz fundamental relaciona a geometria dos dois pontos de vista e pode ser usada pra relacionar pontos de duas imagens. Sabendo a posição das duas câmeras no

espaço e seus parâmetros intrínsecos, é possível descobrir em qual posição do espaço aquele ponto se encontra por meio de triangulação (CORKE, 2016).

Cada ponto \mathbf{P} em uma imagem corresponde a um raio no espaço

$$\mathbf{P} = \alpha \mathbf{d} + \mathbf{P}_0, \forall \alpha > 0, \quad (3.3)$$

onde \mathbf{P}_0 é o ponto de foco da câmera e $\mathbf{d} \in \mathbb{R}^3$, $\|\mathbf{d}\| = 1$ é um vetor unitário na que determina a direção da linha onde o ponto \mathbf{P} se encontra. Com a projeção dos dois vetores sobre determinado ponto, a interseção dos dois vetores nos dá a posição espacial do ponto.

3.2.2 Nuvem de pontos

O resultado da geometria epipolar utilizando as duas imagens resulta em um conjunto de pontos globais tridimensionais \mathbf{P}_i chamados nuvem de pontos. Uma vez obtida a nuvem de pontos, é necessário extrair algum significado conciso de milhares ou milhões de pontos para poder usar essa informação em aplicações de robótica. Algumas das informações que podem ser processadas são a definição de planos na nuvem de pontos, ou a correlação entre dois conjuntos de pontos nessa nuvem (CORKE, 2016).

Planos são comuns em situações reais, sendo possível, por exemplo, localizar chão ou paredes com essa informação. Quanto a conjuntos de pontos, considere um modelo de algum objeto representado por um conjunto de pontos em 2 ou 3 dimensões em relação ao sistema de coordenadas global. Agora, considere um exemplo desse objeto com uma pose diferente. É possível observar um conjunto de pontos diferente para o objeto. A tarefa é determinar a pose relativa ξ que irá transformar os pontos do modelo para os pontos de dados observados, combinando os dois conjuntos de pontos (CORKE, 2016).

Assim, dado dois conjuntos de coordenadas de pontos: o modelo $\mathbf{M}_i \in \mathbb{R}^n, i \in [1, N_M]$ e alguns dados observados ruidosos $\mathbf{D}_j \in \mathbb{R}^n, j \in [1, N_D]$ determinam o movimento do corpo rígido do sistema de coordenadas dos dados para o sistema de coordenadas do modelo

$${}^D\xi_M^* = \xi \in \mathbb{T} \subset \mathbb{R}^6 \underset{\arg \min}{\sum_{i,j}} \|\mathbf{D}_j - \xi \bullet \mathbf{M}_i\|, \quad (3.4)$$

onde o operador \bullet indica que a transformação foi aplicada a todo o conjunto de pontos de \mathbf{M}_i e $\xi \in \mathbb{T}$ representa o conjunto de transformações válidas para solução. Esse é um problema em que é preciso estabelecer correspondência direta entre os pontos nos dois conjuntos, sendo computacionalmente custoso e difícil devido aos dados ruidosos. Há uma abordagem alternativa chamada ponto iterado mais próximo ou ICP. Para cada conjunto de pontos \mathbf{D}_j , é encontrado iterativamente um conjunto de pontos do modelo correspondente \mathbf{M}_i que são assumidos como o mais próximo da localização real, ou seja,

\mathbf{M}_i que minimiza $\|\mathbf{M}_i - \mathbf{D}_j\|$. Essa correspondência não é única e, em geral, vários pontos em um conjunto podem ser associados a um único ponto no outro conjunto e, conseqüentemente, alguns pontos não terão correspondentes (CORKE, 2016). O sensor retorna apenas um subconjunto de pontos no modelo, por exemplo, um escâner a laser pode ver a frente, mas não a parte de trás de um objeto (CORKE, 2016).

Em robótica, o problema é muitas vezes considerado como corresponder um modelo \mathbf{M} de um objeto tridimensional que queremos para os dados observados \mathbf{D} de um sensor. Pode-se calcular uma translação que faz com que os centroides das duas nuvens de pontos coincidam

$$\bar{\mathbf{M}} = \frac{1}{N_M} \sum_{i=1}^{N_M} \mathbf{M}_i, \quad (3.5)$$

$$\bar{\mathbf{D}} = \frac{1}{N_D} \sum_{j=1}^{N_D} \mathbf{D}_j, \quad (3.6)$$

de onde se calcula um deslocamento

$$\mathbf{t} = \bar{\mathbf{D}} - \bar{\mathbf{M}}. \quad (3.7)$$

A correspondência entre os pontos é calculada em seguida. Para cada nuvem de pontos \mathbf{D}_j é encontrado a nuvem de pontos mais próxima do modelo \mathbf{M}_i . Em seguida, a matriz de momentos é calculada

$$\mathbf{W} = \sum_{i,j} (\mathbf{M}_i - \bar{\mathbf{M}})(\mathbf{D}_j - \bar{\mathbf{D}})^T \quad (3.8)$$

que codifica a rotação entre os dois conjuntos de pontos. A decomposição do valor singular é

$$\mathbf{W} = \mathbf{U} \Sigma \mathbf{V}^T \quad (3.9)$$

de onde a matriz de rotação é determinada como

$$\mathbf{R} = \mathbf{V} \mathbf{U}^T \quad (3.10)$$

Assim, a pose estimada relativa entre as duas nuvens de pontos é $\xi_{\Delta} \sim (\mathbf{R}, \mathbf{t})$ e a nuvem de pontos do modelo são transformados de modo que estejam mais próximos da nuvem de pontos de dados

$$\mathbf{M}_i \leftarrow \xi \bullet \mathbf{M}_i, \forall i, \quad (3.11)$$

$$\xi \leftarrow \xi \oplus \xi_{\Delta}, \quad (3.12)$$

e o processo é repetido até convergir. As correspondências utilizadas são improváveis de terem sido todas corretas e, portanto, a estimativa da orientação relativa entre os conjuntos é apenas uma aproximação (CORKE, 2016).

3.3 Localização de objetos baseada em modelos

Diferentes técnicas podem ser utilizadas para identificação e rastreamento do objeto de manipulação. Técnicas como identificação de textura específica, classificação de clusters ou identificação baseada em modelos.

Na abordagem de projeto baseado em modelos, faz-se uso de modelos para representar os elementos de um sistema e as relações entre eles e pode-se trabalhar em um nível de abstração maior ao se utilizar representações que independem do ambiente. O uso de um modelo total para o objeto fornece informação tridimensional completa para seu rastreamento. Isso contribui para mitigar a dificuldade na estimativa da orientação do objeto em oclusão (ARAUJO, 2017).

Entre as vantagens do trabalho com modelos é a desassociação de texturas e paletas de cores com o objeto e a possibilidade de ter o objeto em oclusão parcial (ARAUJO, 2017).

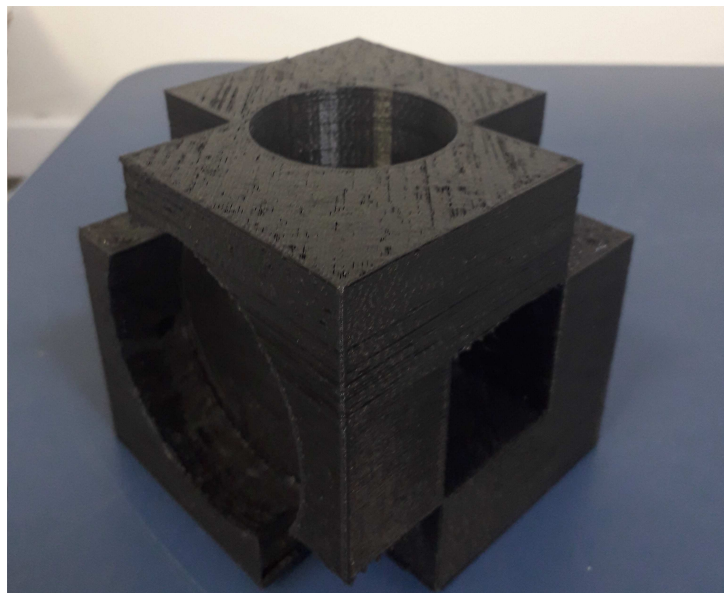
Os modelos de objetos utilizados são primeiramente transformados para nuvem de pontos antes de serem processados pelo sistema. Esse procedimento pode ser feito com aplicativos gratuitos, como o *CloudCompare* ou *FreeCad*. Um dos objetos utilizados para identificação, seu modelo e nuvem de pontos pode ser visto nas figuras 3.4, 3.5 e 3.6.

3.4 Alinhamento inicial por consenso de amostra

O alinhamento inicial de consenso de amostra (SAC-IA) é um método que procura alinhar duas ou mais nuvens de pontos correspondentes a visões parciais distintas, possivelmente sobrepostas, de uma mesma cena, de forma consistente e ótima. Esta tarefa é o registro 3D de uma nuvem de pontos e é formulada como um problema de otimização que busca a matriz de transformação homogênea que alinhe duas nuvens de pontos, minimizando uma determinada função de erro (ARAUJO, 2017).

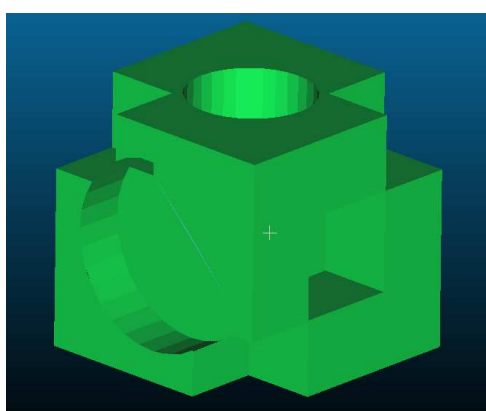
Para o trabalho desenvolvido, o alinhamento consiste em alinhar uma nuvem de pontos que representa o modelo do objeto a ser manipulado com o conjunto de pontos que representam o objeto na nuvem de pontos adquirida pelos sensores. Um tratamento prévio

Figura 3.4 – Objeto impresso usado para validação do algoritmo.

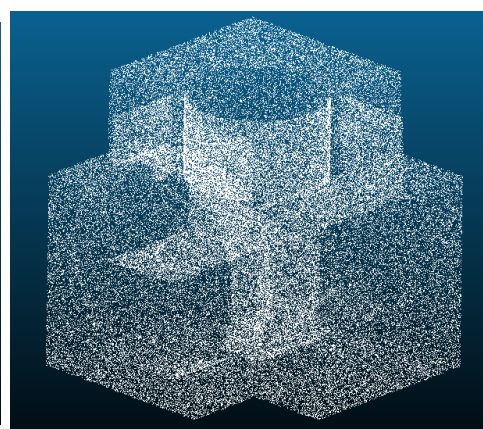


Fonte: Próprio Autor.

Figura 3.5 – Modelo CAD de objeto de teste. Figura 3.6 – Nuvem de Pontos do Objeto.



Fonte: Próprio Autor.



Fonte: Próprio Autor.

da nuvem de pontos é realizado de forma a isolar o grupo de pontos que representam o objeto na nuvem. Mais detalhes sobre esse tratamento podem ser vistos no Capítulo 4.

Este método é fundamentado na definição de um tipo específico de características denominado histograma de características de ponto (PFH). O algoritmo SAC-IA, na verdade, opera nas nuvens de pontos extraindo os chamados histograma rápido de características de ponto (FPFH), que constituem em uma otimização computacional dos PFH.

Histograma de características de pontos (PFH) tem como objetivo identificar um espaço de características no qual pontos 3D dispostos em superfícies geométricas primitivas possam ser facilmente identificados. Assim, o poder discriminatório deste espaço de características deve ser alto o suficiente para que os pontos da mesma superfície sejam conjuntamente classificados.

Os histogramas constituem características locais invariantes à pose e, para um dado ponto p , eles representam as propriedades do modelo da superfície a ele subjacente. O cálculo de um PFH em um ponto p requer o conhecimento de suas coordenadas 3D e de estimativas das normais à superfície.

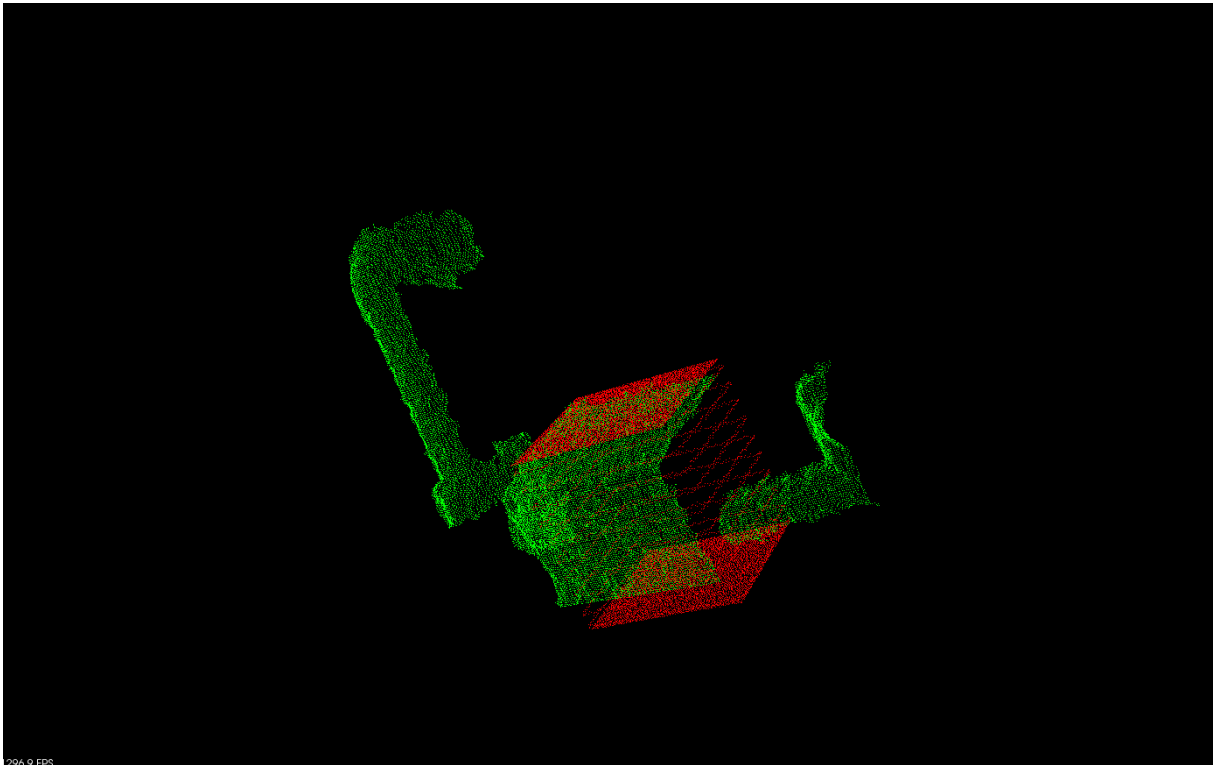
A complexidade computacional do algoritmo histograma de características de ponto para uma dada nuvem com n pontos é de $O(n \cdot k^2)$, onde k é o número de vizinhos para cada ponto na nuvem. Densas vizinhanças de pontos podem se tornar um gargalo de processamento significativo no registro 3D. Para aplicações mais rápidas, os histogramas rápidos de características de pontos (FPFH) são versões simplificadas dos PFH, onde a complexidade computacional é reduzida para $O(n \cdot k)$, porém mantendo a maior parte do poder discriminativo dos histograma de características de pontos (ARAÚJO, 2017).

O algoritmo SAC-IA é um método de Consenso de amostras e tem como objetivo a manutenção das mesmas relações geométricas entre os pontos correspondentes, sem que seja preciso testar todas as possíveis combinações do conjunto de correspondências. A abordagem pelo consenso de amostra é adotada para evitar não só alta complexidade computacional mas também o encontro de um mínimo local. O SAC-IA executa buscas rápidas em um exaustivo espaço de correspondências FPFH para encontrar uma boa solução de alinhamento, que pode, por sua vez, ser posteriormente ajustada usando um método de otimização não-linear. No contexto deste trabalho, o algoritmo fornece a primeira suposição em relação à localização do objeto a ser rastreado no ambiente. Um exemplo de alinhamento inicial com um objeto em cubo pode ser observado na Figura 3.7.

Sejam duas nuvens de pontos P e Q a serem comparadas. Para o SAC-IA, é feito um levantamento de correspondências candidatas, classificadas da seguinte maneira (ARAÚJO, 2017):

- Selecione s amostras de pontos em P , de modo que as distâncias entre cada par seja

Figura 3.7 – Aproximação inicial utilizando SAC-IA.



Fonte: Próprio Autor.

superior a um parâmetro de distância mínima d_{min} definido pelo usuário.

- Para cada ponto de amostra, encontre a lista dos pontos em Q cujos histogramas são similares ao da amostra. Destes, selecione um aleatoriamente e o assuma como sendo o ponto correspondente à amostra.
- Compute a transformação rígida definida pelos pontos de amostra e seus correspondentes e calcule uma métrica de erro que traduza a qualidade da transformação.

Esses passos são repetidos e a transformação que apresentar a melhor métrica de erro é armazenada como alinhamento inicial.

3.5 Filtro de partículas

Após o alinhamento inicial, o ICP é utilizado para convergir para o mínimo local mais próximo e, juntamente de um filtro de partículas, rastrear o objeto em tempo real.

O método utilizado para rastreamento, exemplificado pelo filtro de partículas adotado, é o chamado rastreamento de múltiplas hipóteses (Multiple Hypothesis Tracking - MHT), cuja ideia básica é a de manter várias estimativas a cada frame, de modo que, caso

a melhor delas falhe, o sistema ainda seja capaz de manter o rastro a partir das restantes (ARAÚJO, 2017).

O algoritmo MHT funciona como princípio de uma filtragem Bayesiana, que visa o cálculo de uma função de densidade de probabilidade (fdp) a posteriori para o estado de um sistema a partir de toda informação disponível, incluindo observações realizadas. A biblioteca PCL oferece suporte para o desenvolvimento de filtros de partícula, facilitando a aplicação do método no projeto. O filtro representa uma distribuição de probabilidade arbitrária a partir de amostras ponderadas, extraídas de outra distribuição de probabilidade, denominada densidade de importância. Os pesos utilizados representam as probabilidades de ocorrência de cada amostra.

O filtro é definido em duas etapas: Na primeira, o filtro coleta amostras de uma dada distribuição e as pondera de acordo com seu grau de ocorrência. Em seguida, fazendo uso de observações tomadas, é realizada uma nova amostragem e a distribuição é atualizada (ARAÚJO, 2017).

Seja $\mathbf{n}_x, \mathbf{n}_z, \mathbf{n}_v \in \mathbf{n}_n$ as dimensões dos vetores de estado, medição, ruído de processo e ruído de medição. O modelo do sistema é dado da seguinte forma: Seja $\{\mathbf{x}_k, k \in \mathbb{Q}\}$ a sequência de estados do sistema, com uma evolução no tempo discreto k dada por

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}), \quad (3.13)$$

onde $\mathbf{f}_k: \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$ é uma função não linear do estado \mathbf{x}_{k-1} e do ruído de processo dado por $\{v_{k-1}, k \in \mathbb{N}\}$.

O modelo de medições é definido como

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k), \quad (3.14)$$

onde $\mathbf{h}_k: \mathbb{R}^{n_x} \times \mathbb{R}^{n_n} \rightarrow \mathbb{R}^{n_z}$ é uma função não linear do estado \mathbf{x}_k e do ruído de medição dado por $\{\mathbf{n}_{k-1}, k \in \mathbb{N}\}$.

O rastreamento tem como objetivo buscar por estimativas filtradas do vetor de estados \mathbf{x}_k com base no conjunto de todas as medições disponíveis até o tempo k , isto é, o conjunto $\mathbf{z}_{1:k} = \{\mathbf{z}_i, i = 1, \dots, k\}$. O problema consiste, do ponto de vista Bayesiano, em calcular recursivamente um grau de crença a respeito do estado \mathbf{x}_k , para valores distintos condicionados às medições realizadas até k . Ou seja, procura-se construir a função de densidade de probabilidade $p(\mathbf{x}_k | \mathbf{z}_{1:k})$ (ARAÚJO, 2017). Para inicialização, é assumido $p(\mathbf{x}_0 | \mathbf{z}_0) \equiv p(\mathbf{x}_0)$ como a fdp a priori.

A etapa de predição faz uso do modelo do sistema (3.13) para obter a fdp a priori

por meio da equação de Chapman-Kolmogorov

$$p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k, \mathbf{x}_{k-1} | \mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}. \quad (3.15)$$

Como o modelo do sistema (3.13) é um processo de Markov de primeira ordem, $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_{1:k-1})$ equivale a $p(\mathbf{x}_k | \mathbf{x}_{k-1})$ e, utilizando a regra da cadeia, a equação 3.15 pode ser vista como

$$p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}, \quad (3.16)$$

que é o modelo probabilístico da evolução do estado, definido pela Equação 3.13 e pelas estatísticas conhecidas de \mathbf{v}_{k-1} .

Na etapa de atualização, é necessária a disponibilidade de um vetor de medições \mathbf{z}_k no instante k . As novas medições são utilizadas para modificar a fdp a priori, obtendo a fdp a posteriori desejada: $p(\mathbf{x}_k | \mathbf{z}_{1:k})$. A atualização se dá através da regra de Bayes

$$p(\mathbf{x}_k | \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})} \quad (3.17)$$

em que a constante de normalização

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) d\mathbf{x}_k \quad (3.18)$$

depende da distribuição $p(\mathbf{z}_k | \mathbf{x}_k)$, que é definida pelo modelo de medição (3.14) e pelas estatísticas conhecidas de n_{k-1} .

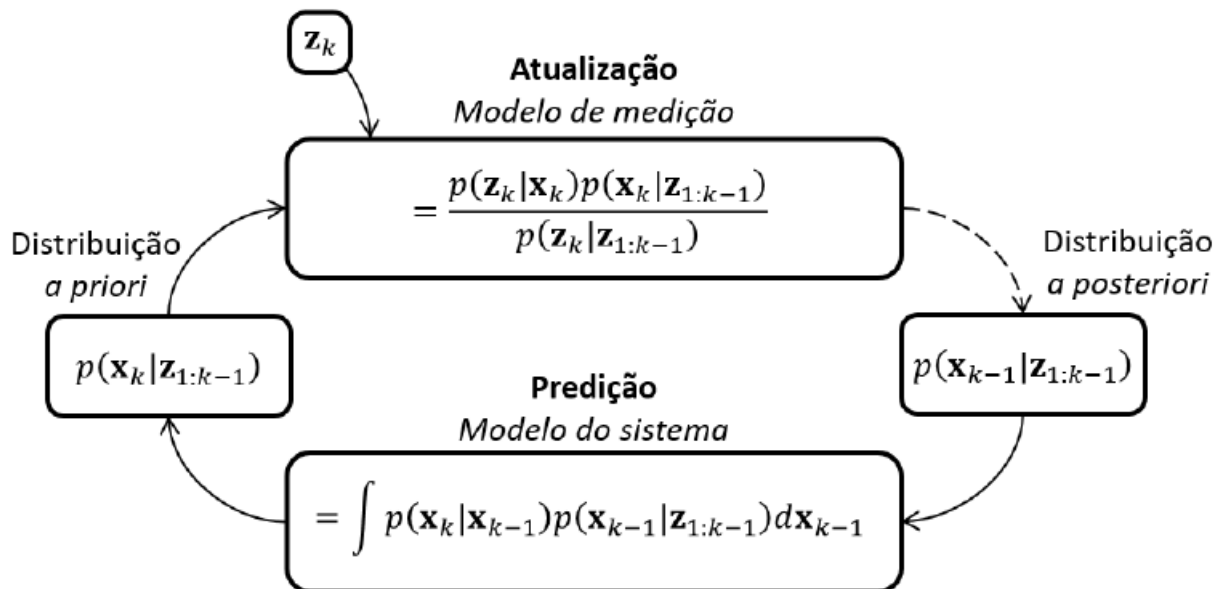
O rastreamento não linear é associado, assim, a uma solução Bayesiana ótima (ARAÚJO, 2017). A propagação recursiva da fdp a posteriori estabelece uma solução que, em geral, não pode ser resolvida analiticamente. Um diagrama do processo do filtro de partículas pode ser visto na Figura 3.8

3.6 Conclusões

Esse capítulo tratou dos conhecimentos necessários para o entendimento da utilização de visão computacional no problema e solução tratados no trabalho. Uma imagem foi definida em termos matemáticos e a utilização de múltiplas imagens para localização espacial foi brevemente estudada.

Em seguida, técnicas para o tratamento de nuvem de pontos e localização de objetos foram abordadas. As técnicas abordadas aqui não são as únicas possíveis para solução do problema, sendo essa uma área ainda em desenvolvimento e pesquisa.

Figura 3.8 – Etapas de Predição e Atualização do rastreamento Bayesiano não linear. O arco tracejado representa o fim de um laço temporal.



Fonte: (ARAÚJO, 2017)

4 Metodologia

A metodologia abrange o problema proposto, sua modelagem e solução, a construção do ambiente de simulação e experimental, e os métodos e ferramentas utilizados para a realização das simulações e experimentos necessários.

O capítulo é organizado da seguinte forma: Primeiro o problema principal é analisado e apresentado de maneira mais ampla, enfatizando os problemas menores a serem solucionados antes que o principal possa ser atacado. Em seguida, uma modelagem e solução são propostos, enfatizando as etapas do processo tanto em simulação quanto em experimentos.

4.1 Definição do problema

O problema consiste em usar dois manipuladores robóticos para manipular um objeto em uma trajetória e rastrear esse objeto no espaço cartesiano. Esse problema envolve uma série de pequenos problemas listados a seguir.

- Localização de dois robôs e do objeto perante um referencial comum.
- Rastreamento da posição em espaço de trabalho dos robôs e do objeto utilizando filtro de partículas.
- Prensão dos dois manipuladores no objeto.
- Definição de uma trajetória a ser seguida pelo objeto.
- Desenvolvimento da cinemática direta e inversa dos manipuladores.
- Controle de trajetória em espaço de trabalho dos dois manipuladores.
- Sincronização do controle dos dois manipuladores em tempo real.
- Identificação espacial do objeto pelo sistema de visão.
- Uso de métricas para quantificar as soluções.

4.2 Modelagem e solução

De modo a resolver o problema apresentado, ambientes de simulação e de experimentos foram montados. Com a utilização de dois manipuladores, um objeto de geometria

comum e um sistema de visão computacional, foi possível adquirir todos os dados possíveis para solução do problema.

Foram utilizados dois manipuladores UR5, capazes de manipular objetos de até cinco quilos, e um objeto comum em formato cúbico. A escolha do cubo se deu a facilidade de sua preensão e manipulação, além da facilidade de se modelar um objeto 3D em formato de cubo para o algoritmo de identificação utilizado na visão.

Uma trajetória para o objeto foi definida onde, mantendo uma rotação constante, o objeto é levantado do local de repouso, se move uma distância fixa em uma direção e depois realiza um semi-círculo no plano xy . A trajetória é definida em três etapas,

$$\begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \begin{cases} (x_{k-1}, y_{k-1}, z_{k-1} + v\Delta k), & \text{para } 0 < k < \frac{z_d}{v}, \\ (x_{k-1}, y_{k-1}, z_{k-1}) + v\mathbf{u}\Delta k, & \text{para } \frac{z_d}{v} < k < \frac{z_d}{v} + \frac{r}{v}, \\ (x_{k-1}, y_{k-1}, z_{k-1}) + v\mathbf{T}(k)\Delta k & \text{para } k > \frac{z_d}{v} + \frac{r}{v}, \end{cases} \quad (4.1)$$

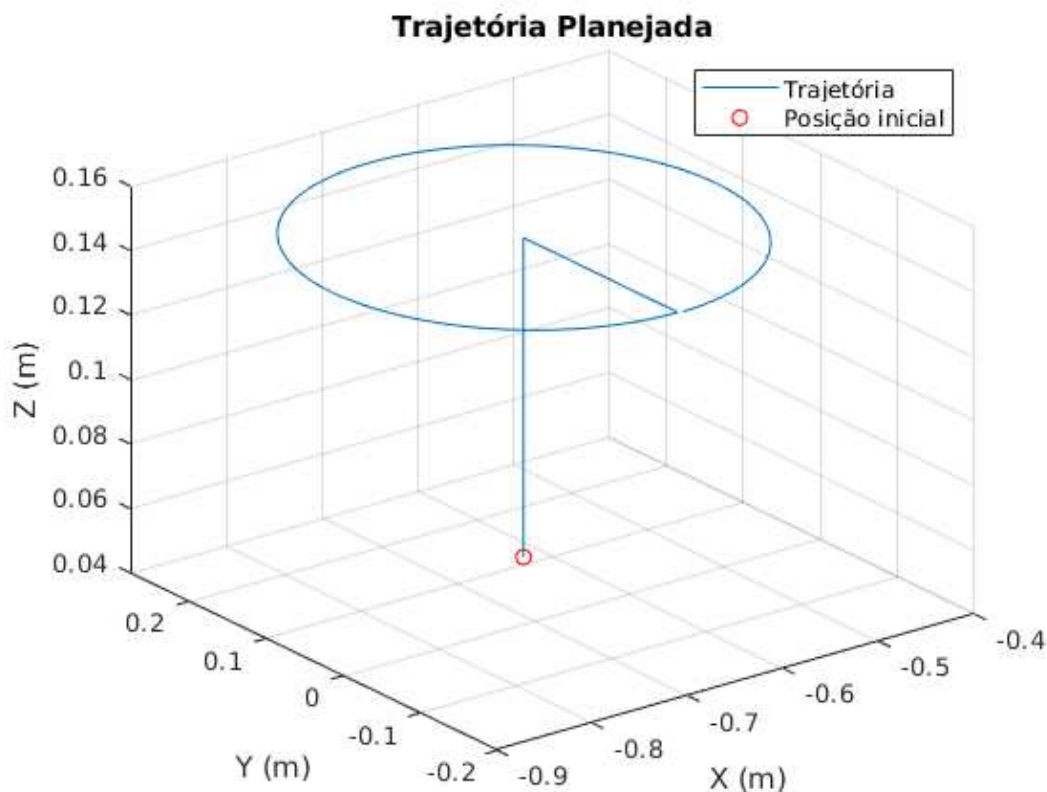
onde v é um escalar da velocidade desejada para trajetória, \mathbf{u} é um vetor direção definido para o raio inicial, e $\mathbf{T}(t)$ é o vetor tangente ao círculo desejado. z_d e r são escalares que determinam a distância inicial a ser percorrida em z e o raio do círculo da trajetória. Um exemplo da trajetória pode ser observado na Figura 4.1.

Inicialmente um ambiente de simulação foi montado consistindo de apenas um manipulador. Com os dados da tabela de Denavit-Hartenberg do UR5, algoritmos foram desenvolvidos para solucionar os problemas de cinemática direta e inversa. Em seguida, um ambiente envolvendo dois manipuladores UR5 e o objeto foi construído, e algoritmos de controle cinemático para os manipuladores foram projetados e simulados, a serem apresentados no capítulo 5. No experimento, os dois manipuladores entram em contato com o objeto por laterais contrárias no cubo, e utilizam a trajetória projetada para mover o objeto sincronicamente.

O principal fator que dita a presença de uma singularidade na trajetória é o espaço de trabalho do manipulador. Se uma posição inicial for escolhida próximo as extremidades do espaço de trabalho ou se um dos fatores z_d ou r for demasiadamente grande, a trajetória gerada saíra do espaço de trabalho do robô e implicará em uma singularidade. No controle de velocidade a trajetória foi projetada de modo a não passar por uma singularidade, mas uma simulação sobre singularidades na trajetória foi realizado e pode ser visto no Capítulo 5.

Após isso, o ambiente de experimentos foi montado. Ele consiste na utilização de dois manipuladores UR5, uma caixa representando o objeto e um sistema de visão utilizando um Kinect, apresentado na seção 4.6. O Kinect foi tomado como origem do sistema global e as posições dos manipuladores foram adquiridas experimentalmente. Um sistema de rastreamento com a biblioteca PCL e o ROS foi desenvolvido para o rastreamento do

Figura 4.1 – Trajetória para controle de posição.



Fonte: Próprio Autor.

objeto no ambiente de simulação.

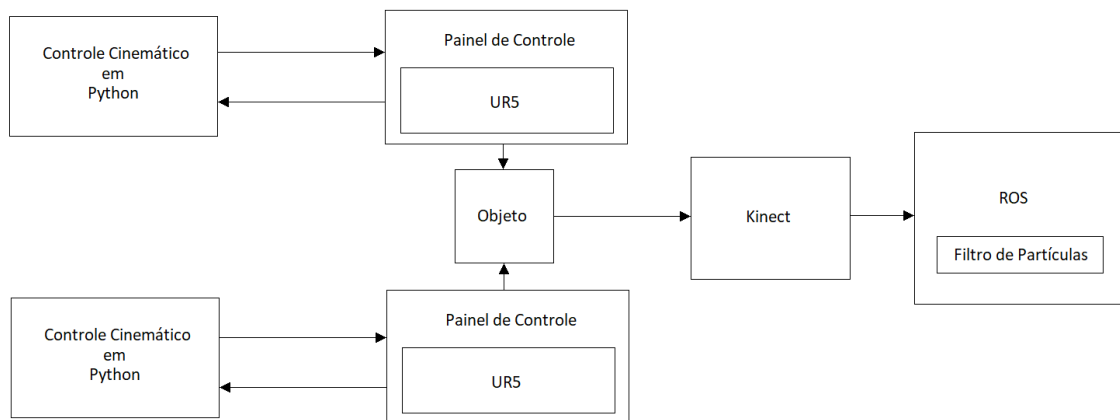
Os experimentos realizados em simulação foram replicados no ambiente experimental. Um diagrama ilustrativo da comunicação dos processos pode ser visualizado na Figura 4.2.

4.3 Simuladores e Ferramentas

Sistemas robóticos são sistemas complexos multi disciplinares que envolvem desde a modelagem do sistema mecânico, o controle da corrente e torque dos motores individualmente e acoplados, e o controle de posicionamento, velocidade e força das juntas. Atrelado a isso, sistemas de sensoriamento como sensores inerciais, sensores de força, posição, corrente, visão, etc tornam um sistema robótico em um sistema de alta complexidade.

A fim de economizar tempo, dinheiro e modularizar os sistemas, simuladores são utilizados para desenvolvimento e teste de diferentes soluções de controle para problemas específicos na robótica, como controle de juntas, controle dinâmico, controle de força ou velocidade, etc. No âmbito da robótica, simuladores como Gazebo, V-REP, OpenRAVE

Figura 4.2 – Diagrama ilustrativo com os diferentes equipamentos e processos da solução.



Fonte: Próprio Autor.

são utilizados para simular o comportamento físico do equipamento e ambiente ao redor.

De modo a validar as soluções desenvolvidas antes da implementação em hardware, um simulador foi utilizado para o desenvolvimento das soluções de controle estudadas. O simulador V-REP possui fácil integração com sistemas externos como Matlab e ROS e foi utilizado nesse trabalho.

Além disso, de modo a simplificar a sincronização e integração com o sistema de visão, o ROS (*Robotic Operating System*) foi utilizado para o tratamento dos dados da visão computacional.

4.3.1 Simulador V-REP

O simulador de robôs V-REP possui ambiente de desenvolvimento integrado e é baseado em uma arquitetura de controle distribuído: cada objeto / modelo pode ser controlado individualmente através de um script incorporado, um plugin, um nó ROS, um cliente de API remoto ou uma solução personalizada. Isso torna o V-REP muito versátil e ideal para aplicações multi-robô. Os controladores podem ser escritos em C, C++, Python, Java, Lua, Matlab ou Octava (COPPELIA..., 2017).

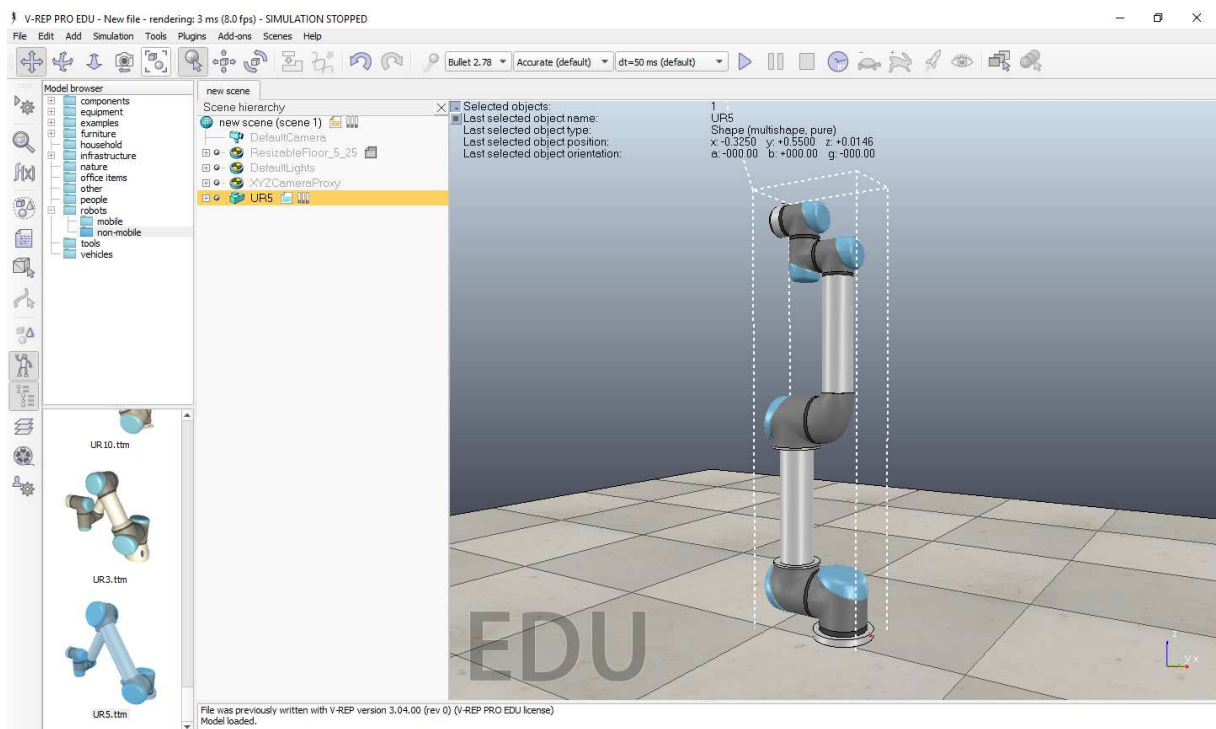
O V-REP é usado para desenvolvimento de algoritmos rápidos, simulações de automação de fábrica, prototipagem e verificação rápidas, educação relacionada à robótica, monitoramento remoto, verificação de segurança, etc.

O simulador é equipado com as opções de motores de simulação física *Bullet* ou ODE (*Open Dynamics Engine*), a mesma utilizada por outros simuladores, como o Gazebo, e oferece uma gama de opções de modelagem e controle, como a disponibilidade de utilização de sensores de força, distância ou visão, e controle direto das juntas por força,

velocidade ou um controle próprio.

O motor de simulação física permite o controle de características como passo de simulação, peso de objetos, atrito, iluminação, etc. Apesar da capacidade de controlar o passo de simulação, com um mínimo de dez milissegundos por passo, os cálculos de controle físicos são realizados a uma frequência dez vezes maior que o passo de cálculo determinado.

Figura 4.3 – Exemplo de ambientação no V-REP.



Fonte: Próprio Autor.

O V-REP foi utilizado juntamente do Matlab por meio de uma API remota, onde as informações de posição e dos sensores é passada para o Matlab e o processamento foi realizado no mesmo. Por meio da API, é possível controlar a simulação em modo assíncrono, com a simulação rodando independente do uso da API, ou em modo síncrono, onde um passo de execução é dado na simulação apenas quando comandado pela API.

A API, porém, possui limitações quanto as funções disponíveis como, por exemplo, não possuir uma função para o retorno da matriz Jacobiana. No entanto, a API disponibiliza a possibilidade de execução de uma função criada em LUA dentro da simulação e retornar o resultado dessa função. Isso foi utilizado para a criação de uma função em LUA que calculasse a matriz jacobiana dos manipuladores e retornasse o resultado para o Matlab.

Um modelo de UR5 fornecido pela Universal Robots se encontra disponível no V-REP, além de modelos de sistemas de visão estéreo com nuvem de pontos. É possível ainda

escolher entre diversos simuladores de física diferentes, cada um com suas especificações.

Além disso, o V-REP permite a importação de um descritor em formato URDF (*Unified Robot Description Format*) de robôs, permitindo uma descrição mais personalizada e precisa do sistema robótico.

Um exemplo de interface do V-REP com um modelo do UR5 se encontra na Figura 4.3.

4.3.1.1 Operação das juntas

Para a simulação física, as juntas são simuladas em *Torque or force mode*. As juntas podem operar livres ou controladas em modo de força/torque, em velocidade ou posição. Em modo de operação livre, com o controle desabilitado, a junta tenta atingir a velocidade desejada o mais rápido possível, utilizando o máximo de torque determinado nas configurações da junta.

Alternativamente, as juntas podem ser controladas em modo de Controle de Posição PID, onde a força é constante e a velocidade é determinada por um controle PID, ou em modo de Mola-Amortecedor, onde a velocidade é constante e a força é controlada como um sistema massa-mola.

Nesse trabalho, as juntas foram controladas em operação livre, atingindo rapidamente a velocidade desejadas, e o controle de velocidade foi realizado externamente.

4.3.1.2 Dinâmica

A vantagem do uso de um simulador é que, graças aos motores físicos implementados, as equações dinâmicas (Equação 2.28) estão todas implementadas e não há a necessidade de se desenvolver as mesmas.

Isso nos dá uma vantagem com relação ao UR5 pois o mesmo tem sua dinâmica e seu controlador de baixo nível fechado, sendo possível acessar apenas a camada cinemática do mesmo. Assim, realizando um controle cinemático em simulação no V-REP, caso as equações dinâmicas sejam equivalentes, o mesmo comportamento será visto no manipulador externo.

Para o modelo de UR5 utilizado no trabalho, os dados que puderam ser coletados do URDF e do V-REP são sumarizados na Tabela 4. I_{xx} , I_{yy} , I_{zz} representam os momentos de inercia da diagonal principal da matriz de momentos de inercia. A junta, na tabela, representa não só o motor, mas o motor e a conexão que ele controla.

Tabela 4 – Dados do modelo usado nas simulações

Dados do modelo dinâmico		
Juntas	Massa	Momentos inerciais principais / Massa
Junta 1	4 kg	$I_{xx} = 4.43 \times 10^{-3}$, $I_{yy} = 4.43 \times 10^{-3}$, $I_{zz} = 7.2 \times 10^{-3}$
Junta 2	3.7 kg	$I_{xx} = 1.03 \times 10^{-2}$, $I_{yy} = 1.03 \times 10^{-2}$, $I_{zz} = 6.66 \times 10^{-3}$
Junta 3	8.39 kg	$I_{xx} = 2.27 \times 10^{-1}$, $I_{yy} = 2.27 \times 10^{-1}$, $I_{zz} = 1.51 \times 10^{-2}$
Junta 4	2.33 kg	$I_{xx} = 4.94 \times 10^{-2}$, $I_{yy} = 4.94 \times 10^{-2}$, $I_{zz} = 4.09 \times 10^{-3}$
Junta 5	1.22 kg	$I_{xx} = 1.11 \times 10^{-1}$, $I_{yy} = 1.11 \times 10^{-1}$, $I_{zz} = 2.19 \times 10^{-1}$
Junta 6	1.22 kg	$I_{xx} = 1.11 \times 10^{-1}$, $I_{yy} = 1.11 \times 10^{-1}$, $I_{zz} = 2.19 \times 10^{-1}$

4.3.2 Sistema Operacional de Robôs - ROS

O sistema operacional de robôs (Robot Operating System - ROS) é uma estrutura flexível para desenvolver softwares para robôs. É uma coleção de ferramentas, bibliotecas e convenções que visam simplificar a tarefa de criar um comportamento complexo e robusto em uma ampla variedade de plataformas robóticas.

O ROS foi construído desde o início para incentivar o desenvolvimento colaborativo de software para robôs. O ROS foi projetado para ser tão distribuído e modular quanto possível, para que os usuários possam usar o máximo ou o mínimo de ROS que desejarem.

O ROS é responsável pela abstração da camada de comunicação e o tratamento da troca de informações, permitindo que o desenvolvedor não se preocupe com a sincronização da comunicação dos sistemas envolvidos na tarefa.

A unidade computacional básica no sistema ROS é denominada nó (node), que é um processo a que se atribui uma determinada tarefa. A comunicação entre nós se dá pela passagem de mensagens tipadas, podendo elas ser um dado inteiro, booleano, uma estrutura de dados com tipos distintos, um conjunto de pontos tridimensionais, etc.

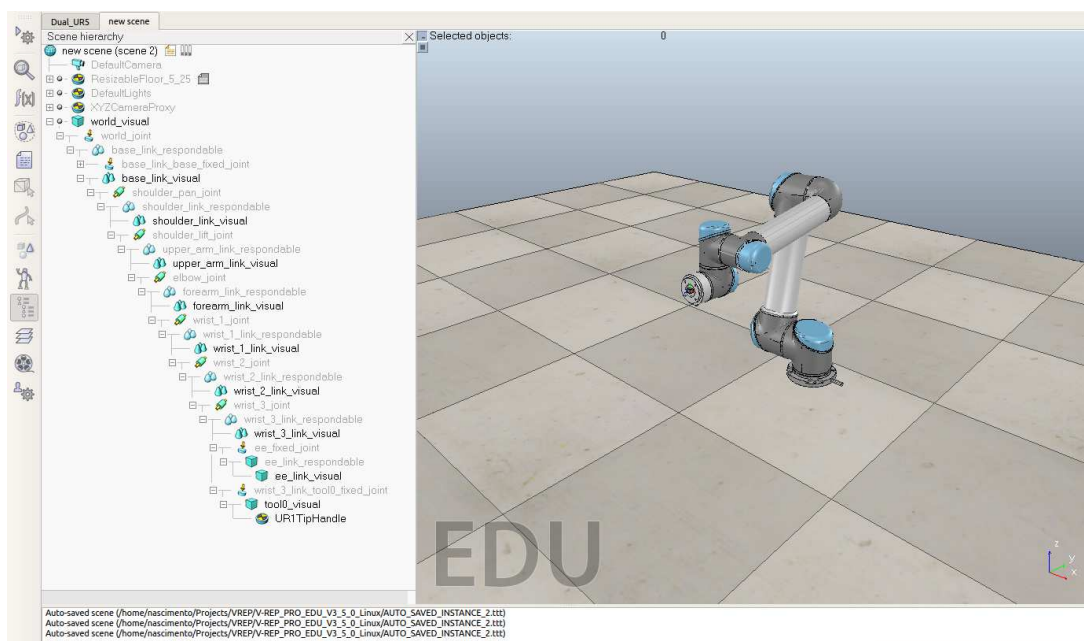
A troca dessas mensagens é feita em um tópico (topic), onde nós podem disponibilizar dados, publicando no tópico (publisher node), ou podem ser assinantes daquele tópico e extrair dados dele (subscriber node). Um diagrama ilustrativo pode ser visto na Figura 4.4.

Figura 4.4 – Diagrama ilustrativo da troca de mensagens no ROS.



Fonte: (ARAÚJO, 2017).

Figura 4.5 – Exemplo de ambiente de simulação com um único UR5.



Fonte: Próprio Autor.

Nesse trabalho, o ROS será utilizado principalmente para tratamento da nuvem de pontos dos sistemas de visão estéreo e identificação de garras, manipuladores e objetos.

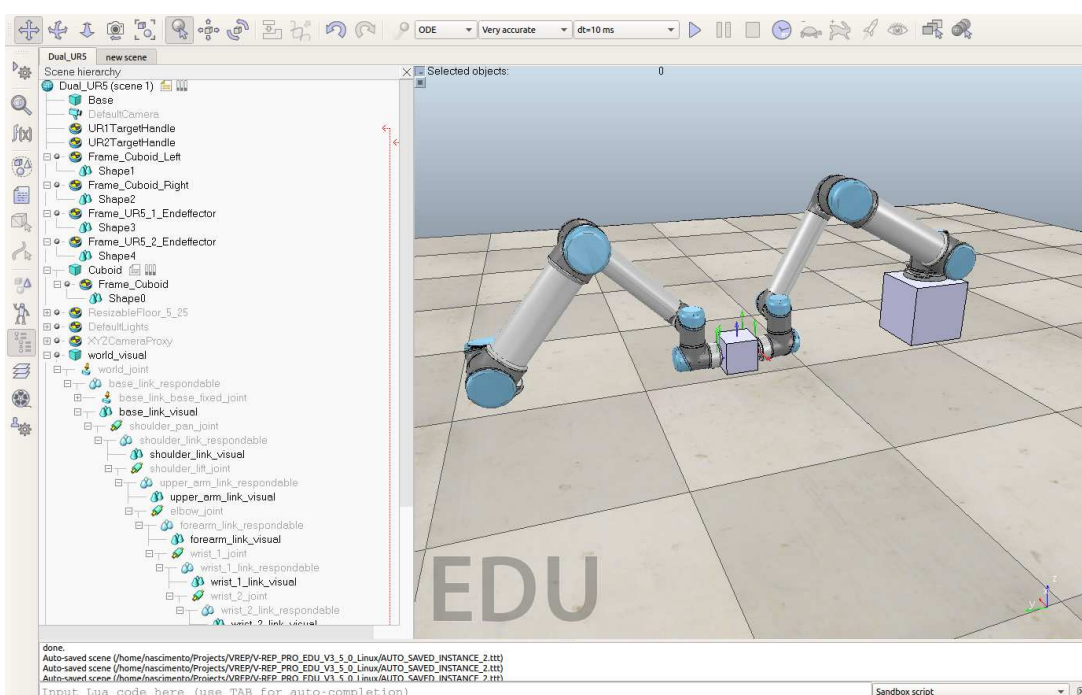
4.4 Configuração do ambiente de simulação

Com a utilização do ambiente de simulação V-REP, é possível adquirir os dados das coordenadas de juntas, coordenadas cartesianas globais e relativas, e velocidade de juntas.

Com isso, foi possível comparar o resultado do algoritmo de cinemática direta com as coordenadas relativas do efetuador do manipulador para confirmar a solução do algoritmo. Então, os dados da posição adquiridos do efetuador foram utilizados em uma solução de cinemática inversa analítica e numérica, por aproximação de Newton-Raphson, para encontrar a solução das juntas equivalente aquela posição no simulador, efetivando as soluções de cinemática inversa.

Para o controle cinemático, o modo de controle de juntas por velocidade foi utilizado no V-REP. As velocidades eram calculadas no Matlab e enviadas pela porta de comunicação síncrona com o V-REP. Os ambientes desenvolvidos podem ser observados nas figuras 4.5 e 4.6.

Figura 4.6 – Exemplo de ambiente de simulação com os dois UR5 e um objeto.



Fonte: Próprio Autor.

4.5 Instalação de ambiente de experimentos

Os braços e o sistema de visão são montados de modo a dar seguimento aos experimentos. Uma calibração dos dispositivos é feita, onde, utilizando o modelo do UR5 apresentado no Gazebo como referência, a pose dos manipuladores são estimadas com relação ao Kinect.

O sistema de visão foi projetado de forma a capturar as posições de dois manipuladores e de uma caixa por meio de nuvem de pontos, ou o mais próximo disso. Códigos para calibração dos sistemas de visão estéreo existem prontos em ROS e, com base nas especificações do Kinect, uma precisão de 5 cm é esperada.

Para rastreamento do objeto, o código base desenvolvido por (ARAUJO et al., 2017) foi modificado e utilizado, onde uma série de filtros é utilizada para isolar um setor da nuvem de pontos, e aquele setor é comparado com a nuvem de pontos de um objeto modelo a ser rastreado, de modo a adquirir sua pose inicial.

4.6 Kinect

O sistema de visão utilizado utiliza um Kinect em posição fixa no laboratório, já adquirido.

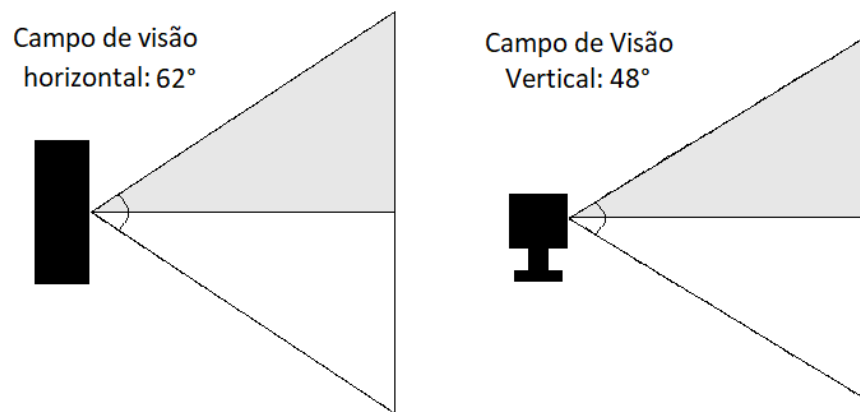
O Kinect apresenta um campo de visão de $62^\circ \times 48^\circ$, adquirindo dados entre 0.5

e 4.5 metros, com recursos de infra vermelho e câmera em cores (Figura 4.7) (KINECT, 2017).

A vantagem principal do Kinect é sua integração fácil com o ROS com o pacote *Freenect*. Uma imagem de um Kinect em laboratório pode ser vista na Figura 4.8.

Uma calibração intrínseca do Kinect em laboratório foi feita utilizando ferramentas prontas oferecidas pelo ROS e um quadro de calibração apropriado.

Figura 4.7 – Campo de visão do Kinect.



Fonte: Próprio Autor.

4.6.1 Localização espacial usando visão

Em laboratório, por meio do módulo *Freenect* integrado ao ROS e utilizando como base o trabalho em (ARAUJO et al., 2017), a localização dos manipuladores e garras no espaço de trabalho foi feita, além do objeto de experimento.

O código para o tratamento dos dados foi desenvolvido em C++ para integração com o ROS utilizando os pacotes *freenect* e bibliotecas matemáticas e de tratamento de nuvem de pontos, como a PCL (*Point Cloud Library*).

A nuvem de pontos e a pose do objeto foi adquirida e rastreada enquanto os experimentos eram realizados. Uma comparação dos dados observados pelo Kinect com os dados adquiridos pelos manipuladores foi realizada. Um exemplo de um registro de modelo pode ser visualizado na Figura 4.9.

A nuvem de pontos inicial capturada pelo Kinect é sub amostrada e tratada por filtros passa faixa nos três eixos de modo a isolar o grupo de pontos pertencentes ao objeto. Em seguida, filtros são aplicados para eliminar planos na nuvem de pontos que não pertencem ao objeto, e os grupos de pontos restantes são salvos. Após isso, o grupo

Figura 4.8 – Kinect em Laboratório.



Fonte: Próprio Autor.

de pontos que representam o objeto é escolhido manualmente para servir de base para os algoritmos de localização e rastreamento.

4.6.2 Calibração Mão para Olho

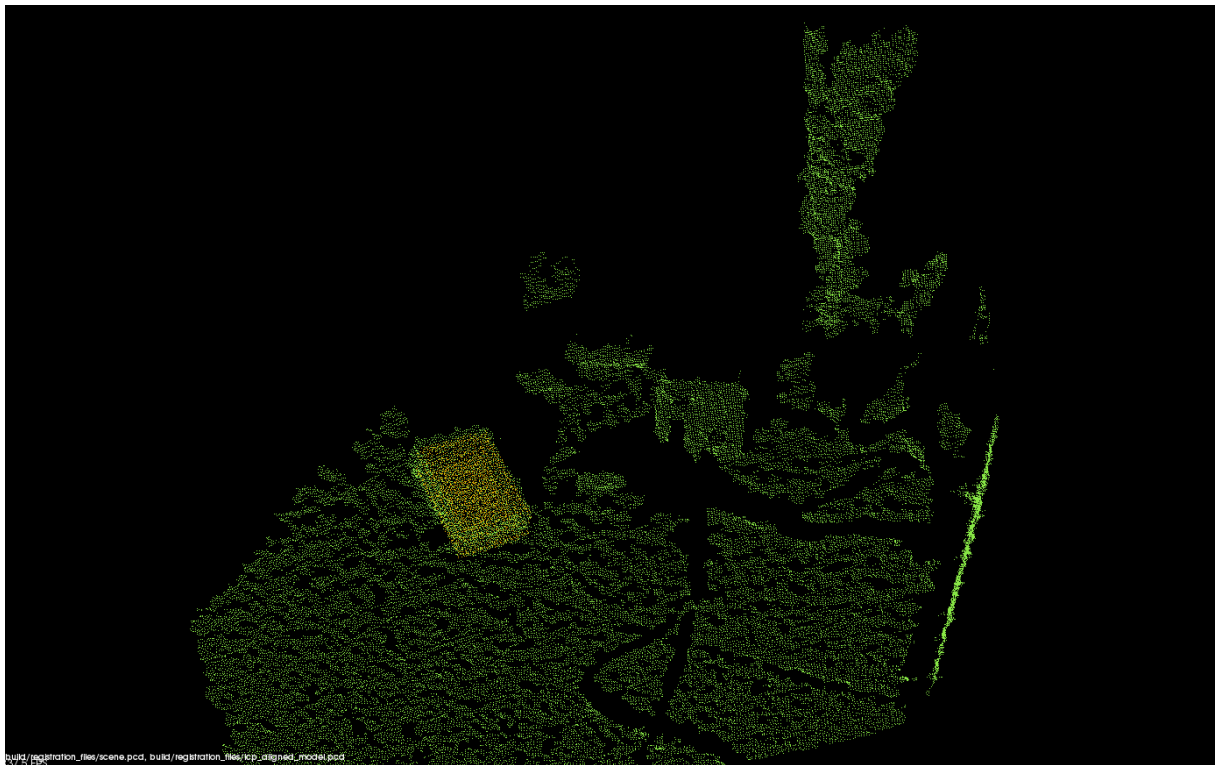
De forma a obter o posicionamento dos manipuladores, do sistema de visão e do objeto em um mesmo referencial, uma calibração mão para olho é necessário. Tomando como referencial mundial o sistema de visão do Kinect, a posição de um objeto no manipulador pode ser obtida como

$${}^C T_O = {}^C T_B {}^B T_{EE} {}^{EE} T_O, \quad (4.2)$$

onde ${}^B T_{EE}$ é dado pela cinemática direta do manipulador e ${}^{EE} T_O$ é a distância do efetuador para o centro de massa do objeto. Uma calibração mão para olho consiste em determinar a transformação ${}^C T_B$, seja por medições físicas, métodos numéricos ou pelo uso de equipamentos avançados de calibração.

Para esse trabalho, essa transformação foi encontrada heurísticamente utilizando o sistema de visão do Kinect e o modelo em URDF do UR5, que foi ajustado de modo a seu modelo 3D sobrepôr o manipulador visualizado pelo Kinect.

Figura 4.9 – Exemplo de identificação de um objeto utilizando modelo e nuvem de pontos.



Fonte: Próprio Autor.

4.7 Manipulador UR5

Os robôs colaborativos da série UR5 da Universal Robots tem capacidade de carga de até 5 kg, um raio de trabalho de 850 mm, software integrado baseado em LINUX e aceita comandos baseados em *URScript*.

O manipulador ainda conta com uma precisão de 0.1 mm para até um milhão de operações. Ele possui seis graus de liberdade, seis juntas rotativas, dezesseis entradas e saídas digitais e duas entradas e saídas digitais na caixa de controle. Possui sensores de força, acelerômetros, encoders.

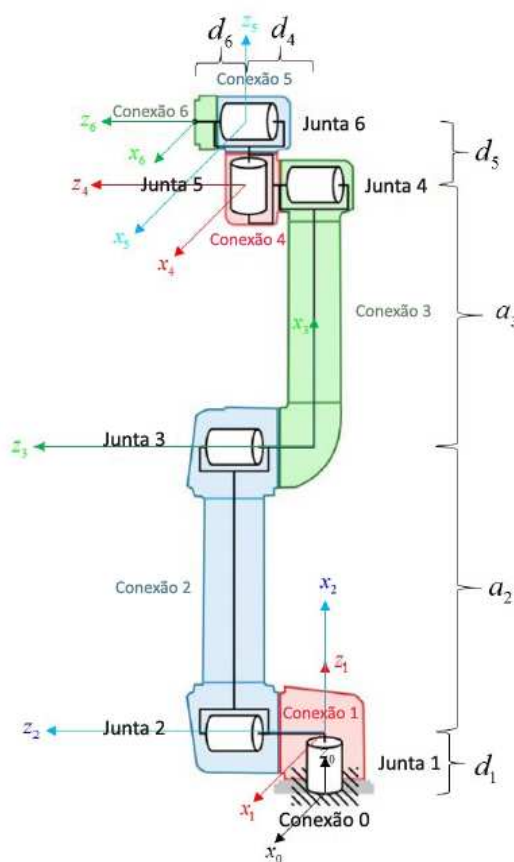
O manipulador ainda aceita comunicações via conexão Ethernet, aceitando comandos em *URScript* (UNIVERSAL..., 2017). Na comunicação com o manipulador é possível adquirir uma série de dados do mesmo, como posições atuais e de alvo de cada uma das juntas, posição e rotação do efetuador no plano cartesiano, dados de velocidade e aceleração, saídas e entradas digitais e analógicas, etc.

Uma representação do manipulador pode ser vista na Figura 4.10. Um dos manipuladores do laboratório pode ser visto na Figura 4.11. Seus parâmetros Denavit-Hartenberg podem ser vistos na Tabela 5

Tabela 5 – Parâmetros UR5

Parâmetros DH: UR5			
θ_j	d_j	a_j	α_j
θ_1	0.089459	0	1.570796327
θ_2	0	-0.42500	0
θ_3	0	-0.39225	0
θ_4	0.10915	0	1.570796327
θ_5	0.09465	0	-1.570796327
θ_6	0.0823	0	0

Figura 4.10 – Modelo do Universal Robot 5.



Fonte: Adaptado de (KEBRIA et al., 2016)

Figura 4.11 – UR5 em Laboratório



Fonte: Próprio Autor.

4.7.1 Controle do manipulador

O controle do manipulador via conexão remota foi feito via protocolo TCP/IP. Isso é, o desenvolvimento do módulo de comunicação, o estudo da linguagem *URScript* e o desenvolvimento de módulos de cinemática direta, inversa, controle de velocidade e posicionamento foi realizado.

O controle é feito tanto em ambiente de simulação quanto em laboratório. Os

manipuladores já possuem controladores internos que realizam o controle da dinâmica e dos motores. Um controle no nível de cinemática foi realizado, encontrado no capítulo 5. Para controlar velocidade, a função *SpeedJ* de URScript foi utilizada, inicialmente pela porta de tempo real, e em seguida com o protocolo *Real Time Data Exchange* (RTDE).

Os dados de pose adquiridos diretamente do UR5 trazem a orientação em forma de vetor-ângulo, e uma transformação teve que ser desenvolvida para utilização dela em forma de matriz ortonormal.

Um levantamento do tempo de processamento dos cálculos foi realizado para garantir sua resposta antes do próximo passo de controle de tempo real dos manipuladores.

O tratamento dos dados e o algoritmo de controle foi desenvolvido na linguagem de script Python, utilizando a biblioteca de processamento matemático *NumPy*.

4.7.2 Sistema de comunicação

A comunicação com o manipulador pode ser realizada de três formas diferentes. Uma comunicação mais lenta, a uma frequência de 10 Hz, pode ser realizada via TCP/IP na porta 30002. Uma comunicação em tempo real, a uma frequência de 125 Hz, pode ser realizada via TCP/IP na porta 30003. Em ambas as conexões uma *string* é enviada com um comando em URScript. Essas portas são as portas determinadas pelo fabricante para a interpretação de comandos URScript pela caixa controladora do robô.

Uma alternativa é configurar uma série de variáveis de entrada e saída utilizando o protocolo *Real Time Data Exchange* (RTDE). As variáveis tem que ser interpretadas pelo controlador do robô, e o envio de bytes diretamente tem que ser tratado pelo cliente. Um diagrama pode ser observado na Figura 4.12.

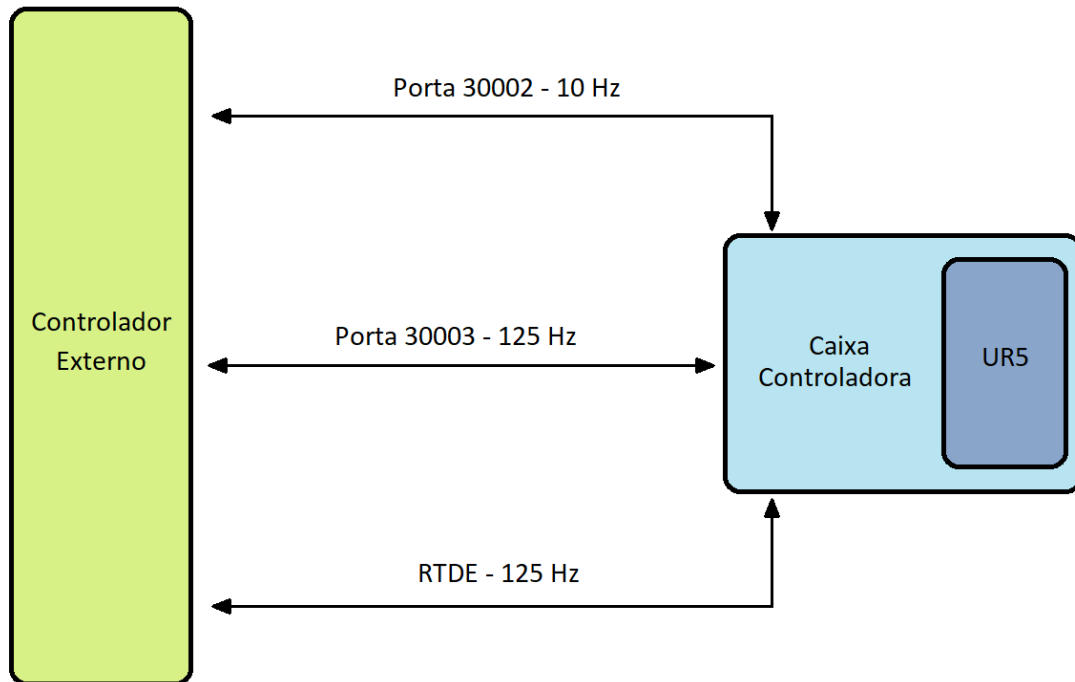
4.7.3 URScript

O controlador do manipulador Universal Robots interpreta funções em uma linguagem própria de texto fornecida pelo fabricante. A linguagem tem funções matemáticas básicas e criação de variáveis a serem interpretadas pelo controlador. Suas principais funções de movimentação incluem *movej*, *movel* e *movec*, que determinam um ponto no espaço e o controlador realiza um controle linear em espaço de juntas, espaço cartesiano ou em arco para aquele ponto.

Para controle direto de juntas, duas funções são de interesse. A função *servoj* utiliza uma lista de coordenadas de juntas para realizar um controle de posição de junta a junta. O controle realizado é interno e a velocidade da trajetória depende das coordenadas de juntas enviadas.

A função *speedj* recebe valores de velocidade para as seis juntas, com o valor de

Figura 4.12 – Diagrama de comunicação com o manipulador UR5.



Fonte: Próprio Autor.

aceleração da junta principal (a mais rápida) e um valor de tempo mantendo aquela velocidade antes de retornar a função. Quando a função é retornada, o robô imediatamente desacelera e tende a parar. Caso esse tempo não seja grande o suficiente antes de receber um novo comando, pode haver uma flutuação e ruído na velocidade das juntas.

4.7.4 Controle em tempo real

Utilizando a porta de tempo real, é possível controlar o manipulador a uma frequência de 125 Hz. Entre o envio de um sinal e outro, informações recebidas pelo robô tem que ser tratadas, o algoritmo de controle gera novas velocidades, elas são encapsuladas no comando URScript correspondente e o comando é enviado pela porta de tempo real.

A conexão é feita por Ethernet via protocolo TCP/IP. É possível configurar a conexão para ser mais rápida, sem envio de pacotes de confirmação ou redundância, mas ainda assim um atraso é introduzido pela utilização da Ethernet.

Uma alternativa é a utilização do protocolo de RTDE para envio direto de variáveis pré configuradas, evitando a interpretação e encapsulamento das funções e dados.

Uma segunda alternativa é executar o controle diretamente do sistema Linux interno a caixa controladora, evitando atrasos de comunicação em rede.

4.7.5 Sincronização de manipuladores

Uma vez que os dois manipuladores tiverem realizado a apreensão, uma sincronização baseada em trajetória foi feita entre os dois manipuladores utilizando a porta de tempo real oferecida pelo UR5. Inicialmente, para o controle de um único manipulador, a porta de tempo real do manipulador estava sendo utilizada, que possui uma frequência de resposta de 125 Hz. No entanto, ao processar o controle cinemático juntamente do processamento da nuvem de pontos ou ao tentar uma comunicação com os dois manipuladores ao mesmo tempo, um atraso cumulativo no sistema torna o controle pela porta de tempo real inviável.

Uma alternativa foi encontrada ao se configurar o protocolo de *Real-Time Data Exchange* (RTDE) do manipulador, permitindo um controle muito mais rápido ao se modificar apenas variáveis previamente configuradas. Com isso, a função de movimentação do robô utilizada, o *speedj*, é comandada em ciclo por um script criado rodando internamente na caixa controladora do robô. Contudo, do modo que é definido, o *speedj* toma como parâmetro um tempo mínimo para controle de velocidade, e que se for pequeno o suficiente a velocidade do robô constantemente vai para zero antes do próximo ciclo, o que resulta num baixo desempenho. Um tempo mínimo de nove ciclos (72 milissegundos) é necessário antes da atualização da velocidade do *speedj*, resultando em um baixo desempenho para variações bruscas na direção de velocidade.

Uma terceira alternativa, a ser explorada em trabalhos futuros, é a execução do script de controle diretamente do sistema Linux da caixa controladora do robô, eliminando qualquer atraso ou necessidade de tempo mínimo para a função de velocidade.

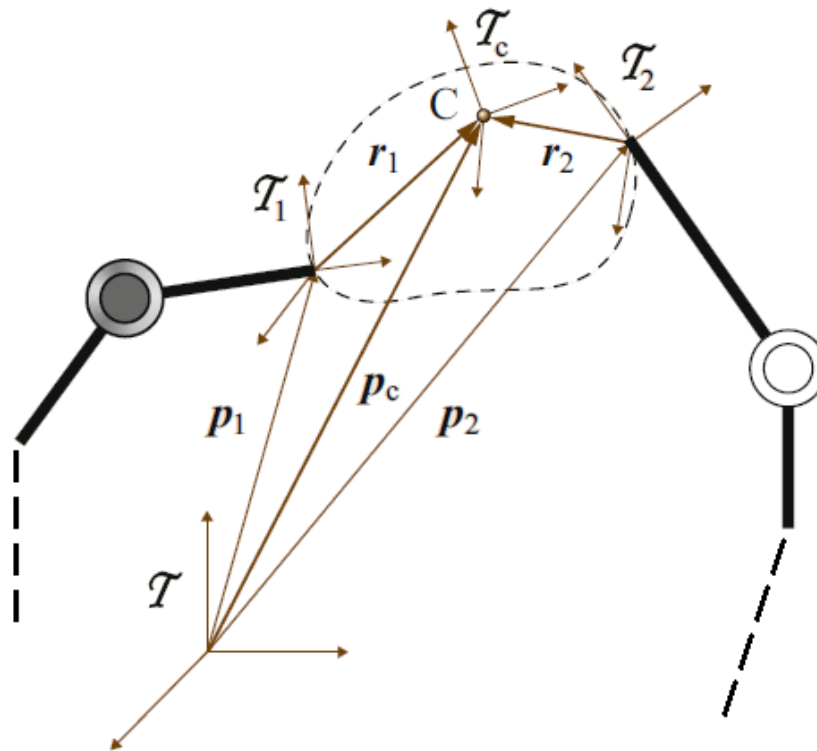
Um levantamento do tempo de resposta e do tempo de processamento foi feita para assegurar envio dos dados de controle antes do passo de tempo real do UR5.

4.8 Múltiplos Manipuladores

Um sistema cooperativo é composto de M manipuladores atuados, onde a pose $\mathbf{p}_i \in \mathbb{R}^6$ de cada manipulador i é função da cinemática direta. Seja agora, um sistema cooperativo onde múltiplos manipuladores manipulam um objeto comum, como na Figura 4.13.

Seja C uma pose de referência no objeto. As poses T_1 , T_2 e T_C representam transformações homogêneas no referencial global que denominam as posições dos efetuadores de dois manipuladores e do objeto, respectivamente. Os trabalhos de (UCHIYAMA; IWASAWA; HAKOMORI, 1987), (UCHIYAMA; DAUCHEZ, 1988) e (UCHIYAMA; DAUCHEZ, 1992) foram os primeiros a trabalhar com robótica cooperativa, introduzindo o conceito de *bastão virtual*, vetores de posição $\mathbf{r}_i \in \mathbb{R}^3$, representados na Figura 4.13 pelos vetores \mathbf{r}_1 e \mathbf{r}_2 , que conectam a pose T_C as poses T_1 e T_2 .

Figura 4.13 – Geometria de preensão para dois manipuladores cooperativos manipulando um objeto.



Fonte: (SICILIANO; KHATIB, 2016)

Se o objeto for rígido e a preensão dos manipuladores é fixa, os vetores \mathbf{r}_i são vetores constantes que executam essa transformação. Assim, a cinemática direta de cada manipulador pode determinar a pose T_C do objeto. Esse modelo só é válido caso o objeto não seja deformado. Durante o desenvolvimento desse trabalho, foi considerado que o objeto era rígido e as preensões eram fixas, de forma que a diferença de deslocamento dos efetuadores é considerada ínfima.

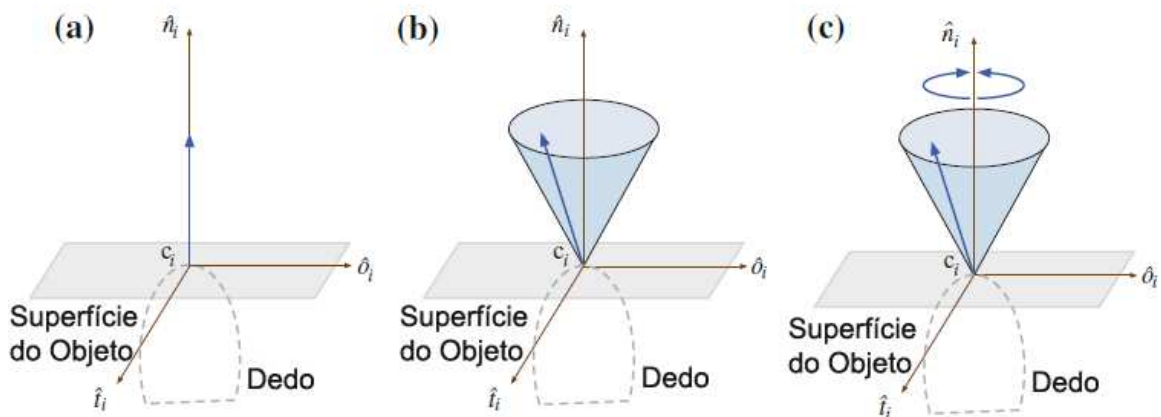
4.9 Preensão robótica

A área de preensão robótica consiste no estudo de métodos para determinar apreensão, localização e força que um determinado manipulador tem que exercer sobre um objeto para completar certa tarefa. Ela pode ser dividida em dois problemas principais, *Síntese de compressão* e *Análise de compressão*. O primeiro consiste em definir conjuntos de contatos apropriados dado um objeto e determinadas restrições nos contatos permitidos. A segunda define se a preensão é estável usando propriedades de fechamento, dado um objeto e um conjunto de contatos. Síntese de compressão está fora do escopo desse trabalho.

Dois pontos foram definidos, que possuem o mesmo vetor normal em direções

diferentes, de modo a restringir a movimentação do objeto no eixo da normal e rotações nos dois eixos ortogonais ao contato. Além disso, com o contato modelado como pontual com atrito (Fig. 4.14) as forças de atrito restringem parcialmente o movimento nos outros dois eixos e a rotação no eixo da normal.

Figura 4.14 – Modelos de contato: (a) pontual sem atrito, (b) pontual com atrito, (c) dedo flexível (LEON; MORALES; SANCHO-BRU, 2014).



Fonte: (LEON; MORALES; SANCHO-BRU, 2014)

Cada manipulador realizou a prensão independentemente da posição do outro, e uma vez que os dois tenham realizado a prensão do objeto, eles foram sincronizados.

A prensão dos efetadores foi idealizada, e não houve um tratamento específico por síntese de compressão ou análise de compressão para garantir o contato com o objeto. Os dois manipuladores realizaram a prensão independentemente e aplicaram uma força ao objeto de modo a restringir sua movimentação lateral e com um atrito grande o suficiente para vencer a força gravitacional.

4.10 Poses iniciais

As poses iniciais dos manipuladores são geradas de maneiras diferentes na simulação e nos experimentos. Na simulação, é possível obter todas as poses dos manipuladores e objetos com relação a um referencial global. Assim, sabendo as dimensões do objeto, e que seu referencial se encontra no centro de massa, duas posições em laterais opostas do objeto foram geradas, e a orientação dessas poses foi ajustada manualmente para coincidir com as orientações dos efetadores dos manipuladores, ilustrado na Figura 4.15. As posições são geradas da seguinte maneira

$$\mathbf{T}_{R1} = (\mathbf{T}_O) \mathbf{T}_L, \quad (4.3)$$

$$\mathbf{T}_{R2} = (\mathbf{T}_O) (\mathbf{T}_L)^{-1}, \quad (4.4)$$

onde \mathbf{T}_{R1} e \mathbf{T}_{R2} são as poses iniciais dos dois robôs e \mathbf{T}_O é a pose do objeto. É assumido um sistema de bastões virtuais como na Figura 4.13, onde as preensões são fixas e o objeto é rígido. \mathbf{T}_L , então, é a transformação que representa o vetor \mathbf{r}_i de translação da metade da largura do objeto na direção de uma das laterais, dado pelo vetor $\mathbf{r} = [x_l, y_l, z_l]$, que é um vetor unitário de direção multiplicado pela metade do valor da lateral l , $\mathbf{r} = \hat{\mathbf{r}}_2^l$, e uma matriz de rotação T_R que transforma as duas referências para o referencial de rotação dos robôs,

$$\mathbf{T}_L = \mathbf{T}_R \begin{pmatrix} 1 & 0 & 0 & x_l \\ 0 & 1 & 0 & y_l \\ 0 & 0 & 1 & z_l \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (4.5)$$

Uma vez gerada essas poses globalmente, a trajetória é gerada em coordenadas globais e em seguida transformada para o referencial de cada manipulador segundo a seguinte fórmula

$${}^B\mathbf{T}_t = ({}^O\mathbf{T}_B^{-1}) {}^O\mathbf{T}_t \quad (4.6)$$

onde B é a base do manipulador, O é a origem do sistema de coordenadas global e t são as poses da trajetória. A equação transforma todas as poses da trajetória para o referencial do manipulador.

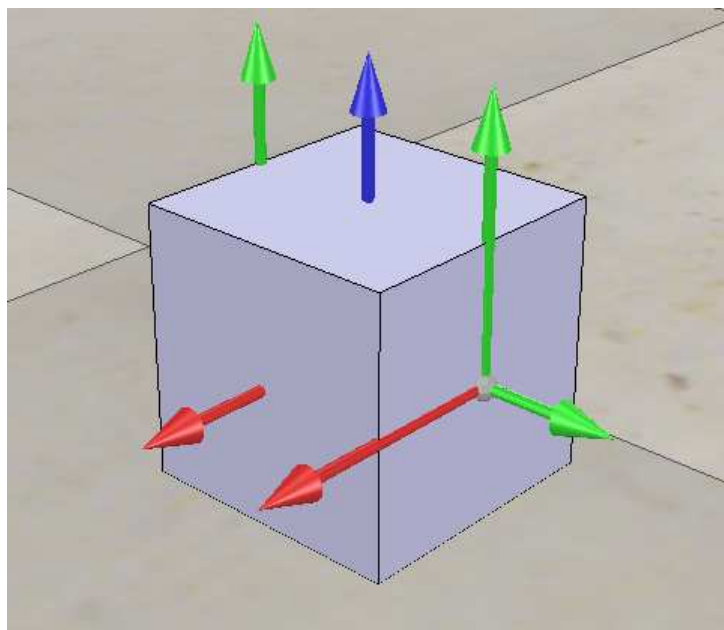
Na etapa experimental, o inverso ocorre. Os manipuladores são posicionados em contato com o objeto e seu centro de massa e pose são estimados com relação a um dos manipuladores sabendo as dimensões do mesmo. As trajetórias já são geradas diretamente dos referenciais dos manipuladores, sendo modificado apenas a direção do raio e da circunferência, uma vez que os dois manipuladores estão orientados a 180 graus um do outro,

$$\mathbf{T}_O = \mathbf{T}_{R1} (\mathbf{T}_L)^{-1}. \quad (4.7)$$

A calibração mão para olho então é utilizada para transformar a pose estimada do objeto do referencial de um dos manipuladores para o referencial global (no caso, o Kinect).

É importante ressaltar que existe uma incerteza associada tanto a estimação do centro de massa do objeto quanto a calibração mão para olho, e esse sistema foi aqui utilizado apenas para validar o rastreamento do objeto com o filtro de partículas.

Figura 4.15 – Posições iniciais para geração de trajetória do objeto e dos manipuladores. A pose do objeto em seu centro de massa e as duas poses geradas nas laterais.



Fonte: Próprio Autor.

4.11 Conclusões

O capítulo abordou inicialmente o problema a ser solucionado no trabalho, enfatizando o escopo e as limitações do mesmo. Dentro dessas limitações, uma solução foi proposta utilizando dois manipuladores colaborativos e um sistema de visão em profundidade.

O trabalho prosseguiu com a definição da trajetória utilizada, das ferramentas de simulação e da etapa experimental, entrando em detalhes no funcionamento das mesmas.

Uma descrição extensiva do funcionamento do UR5 foi realizada, enfatizando seu sistema de comunicação e como ele interfere na solução proposta, e por fim apresentou soluções para as etapas iniciais de prensão, posição inicial e calibração mão para olho para o sistema.

5 Simulações e experimentos

Movimentos com manipuladores robóticos são tipicamente especificados com relação ao espaço de trabalho cartesiano em termos da posição do efetuador no tempo, velocidade e aceleração. De acordo com (WIT; SICILIANO; BASTIN, 1996), a estratégia comum para controle cinemático é dividida em dois estágios: Inversão cinemática das variáveis em espaço cartesiano, e em seguida um projeto de controle no espaço de juntas.

Esse capítulo apresenta a inversão cinemática utilizando o Jacobiano, juntamente com problemas de singularidade e estratégias para contornar esses problemas, e em seguida um controle de posição, velocidade e trajetória é apresentado, com definições do erro, modelagem matemática em malha fechada e três estratégias clássicas envolvendo controle proporcional, proporcional integral e controle feedforward.

Em seguida, o controle da trajetória com os dois manipuladores é apresentado, junto da sincronização na comunicação. A identificação e rastreamento do objeto é apresentada com o controle dos manipuladores e os resultados são apresentados.

As estratégias de inversão do Jacobiano, definição do erro, modelagem matemática e controle de posição são apresentadas em (WIT; SICILIANO; BASTIN, 1996). As estratégias de controle são baseadas no material encontrado em (CORKE, 2016) e (SICILIANO; KHATIB, 2016).

Todas as simulações para controle cinemático foram feitas utilizando V-REP, *Matlab* e Lua, enquanto os experimentos de controle cinemático foram feitos utilizando *numPy*.

O ROS foi utilizado com a biblioteca PCL em C++ para implementação do sistema de visão e filtro de partículas.

5.1 Cinemática inversa em malha aberta

Assuma que a trajetória a ser seguida é definida por $(p(t), \dot{p}(t))$. O objetivo é encontrar uma trajetória no espaço de juntas $(q(t), \dot{q}(t))$ que reproduza essa trajetória.

É possível estabelecer um mapeamento linear entre velocidades no espaço de junta e no espaço cartesiano. Relembrando a equação 2.12, temos

$$\dot{\mathbf{p}}(\mathbf{t}) = \mathbf{J}_a(\mathbf{q})\dot{\mathbf{q}}(\mathbf{t}), \quad (5.1)$$

ou, de forma reduzida,

$$\dot{\mathbf{p}} = \mathbf{J}\dot{\mathbf{q}}(\mathbf{t}), \quad (5.2)$$

onde $\mathbf{J}(\mathbf{q})$ pode representa o Jacobiano analítico. A equação diferencial 5.2 pode ser usada para resolver o problema da cinemática inversa. De 5.2, temos

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{p}}, \quad (5.3)$$

que é uma solução de mínimos quadrados para 5.2.

As velocidades de juntas $\dot{\mathbf{q}}(t)$ podem ser obtidas resolvendo a cinemática inversa diferencial para a configuração atual, após isso, um integrador no tempo pode ser utilizado para encontrar as posições de juntas $\mathbf{q}(t)$, dado conhecimento das condições iniciais. Essa estratégia é baseada no Jacobiano do manipulador e pode ser aplicada em qualquer estrutura de manipulador, na condição de que a inversa do Jacobiano possa ser encontrada.

5.1.1 Singularidades da inversa do Jacobiano

Existem casos onde a inversa do Jacobiano na equação 5.3 não pode ser encontrada. Esses casos ocorrem quando a matriz do Jacobiano $\mathbf{J}(q)$ possui posto nulo.

Quando o manipulador é sub atuado, ou seja, possui menos que seis graus de liberdade, sua matriz Jacobiana será da forma $\mathbf{J}(q) \in \mathbb{R}^{6 \times N}$, $N < 6$, e a inversa da matriz não é possível.

Além disso, existem configurações no espaço de juntas onde, para o efetuador, o movimento individual de duas juntas resultam no mesmo movimento do efetuador, normalmente quando essas juntas estão alinhadas. Quando isso acontece, a coluna das duas juntas se tornará igual no Jacobiano, resultando em uma matriz de posto nulo.

Para contornar esses problemas, os métodos de *Pseudo Inversa* e *Inversa Amortecida* do Jacobiano foram desenvolvidos.

5.1.2 Pseudo inversa

Para os casos onde o manipulador é sub atuado, uma solução para equação 5.2 pode ser obtida ao se utilizar a pseudo inversa \mathbf{J}^\dagger da matriz \mathbf{J} . Essa matriz satisfaz as condições de Moore-Penrose

$$\begin{aligned} \mathbf{J}\mathbf{J}^\dagger\mathbf{J} &= \mathbf{J}, & \mathbf{J}^\dagger\mathbf{J}\mathbf{J}^\dagger &= \mathbf{J}^\dagger, \\ (\mathbf{J}\mathbf{J}^\dagger)^T &= \mathbf{J}\mathbf{J}^\dagger, & (\mathbf{J}^\dagger\mathbf{J})^T &= \mathbf{J}^\dagger\mathbf{J}, \end{aligned} \quad (5.4)$$

e a solução para a equação 5.2 pode ser escrita como

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \dot{\mathbf{p}}, \quad (5.5)$$

que resulta em uma solução de mínimos quadrados com norma mínima.

Se a matriz de Jacobiano tem posto completo, a pseudo inversa pode ser calculada como

$$\mathbf{J}^\dagger = \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}, \quad (5.6)$$

é possível observar que, se o Jacobiano \mathbf{J} é quadrado, a pseudo inversa 5.6 é reduzida para a inversa comum \mathbf{J}^{-1} .

5.1.3 Inversa amortecida

Na vizinhança de configurações singulares, o uso da pseudo inversa não é adequado e uma solução numérica robusta é alcançada ao se utilizar a inversa amortecida. Ela é baseada na solução da equação modificada

$$\mathbf{J}^T \dot{\mathbf{p}} = (\mathbf{J}^T \mathbf{J} + \lambda^2 \mathbf{I}) \dot{\mathbf{q}}, \quad (5.7)$$

no lugar da equação (5.2). O fator λ na equação 5.7 é chamado de fator de amortecimento. Quando $\lambda = 0$, a equação 5.7 se reduz a 5.2. A solução 5.7 pode ser escrita como

$$\dot{\mathbf{q}} = \mathbf{J}^T ((\mathbf{J}\mathbf{J}^T)^{-1} + \lambda^2 \mathbf{I}) \dot{\mathbf{p}}, \quad (5.8)$$

que pode ser escrita na forma equivalente

$$\dot{\mathbf{q}} = \mathbf{J}^\#(\mathbf{q}) \dot{\mathbf{p}}. \quad (5.9)$$

A solução 5.9 satisfaz a condição

$$\min_{\dot{\mathbf{q}}} \|\dot{\mathbf{p}} - \mathbf{J}\dot{\mathbf{q}}\|^2 + \lambda^2 \|\dot{\mathbf{q}}\|^2, \quad (5.10)$$

que demonstra uma permuta entre a condição de mínimos quadrados e norma mínima. Nesse sentido, é essencial uma escolha efetiva do valor de λ . Baixos valores vão resultar em uma acurácia maior, mas pouca robustez na vizinhança de singularidades, isso é, na vizinhança de singularidades a inversa do jacobiano tende rapidamente a valores infinitos, enquanto valores altos resultam em baixa acurácia mesmo se soluções mais acuradas fossem possível.

O fator de amortecimento λ resulta em uma aproximação na solução de mínimos quadrados, assim, usar um valor constante para λ pode ser inadequado para obter um bom desempenho em todo o espaço de trabalho do manipulador. Uma escolha apropriada é ajustar λ em função de uma medida de proximidade a singularidade na configuração atual do manipulador.

Uma medida de manipulabilidade foi definida por (YOSHIKAWA, 1985), é dada por

$$w = \sqrt{\det |\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})|}, \quad (5.11)$$

onde w , chamado de medida de manipulabilidade, da uma medida da proximidade da configuração \mathbf{q} com uma singularidade. Quanto mais próximo de zero é w , mais próximo ele é de uma singularidade e em uma singularidade se tem $w = 0$.

Uma região de singularidade pode ser definida como base para menor singularidade possível de \mathbf{J} : Fora da região a solução exata é utilizada, enquanto dentro da região o fator de amortecimento é introduzido para se obter a solução aproximada. O fator de amortecimento precisa ser escolhido para garantir a continuidade da velocidade $\dot{\mathbf{q}}$ na transição na borda da região de singularidade. Borda da região de singularidade é a região no espaço onde $w \approx w_0$, sendo w_0 um fator que determina o limiar de quando as velocidades estão aumentando rapidamente devido a inversa do jacobiano, ao se aproximar de uma singularidade.

Sem perda de generalidade, para um manipulador com seis graus de liberdade, o fator de amortecimento pode ser escolhido como proposto por (NAKAMURA; HANAFUSA, 1986b)

$$\lambda = \begin{cases} \lambda_0(1 - \frac{w}{w_0})^2, & \text{para } w < w_0 \\ 0, & \text{para } w \geq w_0 \end{cases} \quad (5.12)$$

onde λ_0 é uma constante de fator de escala, w é a manipulabilidade da configuração e w_0 é o fator que determina o limite da região de singularidade.

5.2 Cinemática Inversa por aproximação de Newton-Raphson

Uma forma de encontrar a cinemática inversa de um ponto específico para um manipulador é utilizar a *cinemática inversa numérica por aproximação de Newton-Raphson* (LYNCH; PARK, 2017). Seja a cinemática direta definida como na Equação 2.8,

$$\mathbf{p} = \mathcal{K}(\mathbf{q}), \quad (5.13)$$

a cinemática inversa para um ponto no espaço, \mathbf{p}_d , pode ser definida como encontrar um conjunto de ângulos \mathbf{q}_d que satisfaçam a equação

$$\mathbf{p}_d - \mathcal{K}(\mathbf{q}_d) = 0. \quad (5.14)$$

Uma aproximação inicial, \mathbf{q}_i , pode ser escolhida para satisfazer a Equação 5.14. Se essa aproximação não for exata, é possível expandir a Equação 5.13 por série de Taylor para encontrar

$$\mathbf{p}_d = \mathcal{K}(\mathbf{q}_d) = \mathcal{K}(\mathbf{q}_i) + \left. \frac{d\mathcal{K}}{d\mathbf{q}} \right|_{\mathbf{q}_i} (\mathbf{q}_d - \mathbf{q}_i) + \dots. \quad (5.15)$$

Ignorando os termos de alta ordem, a Equação 5.15 pode ser resumida em

$$\mathbf{p}_d = \mathcal{K}(\mathbf{q}_d) = \mathcal{K}(\mathbf{q}_i) + \mathbf{J}(\mathbf{q}_i)\Delta\mathbf{q}, \quad (5.16)$$

que pode ser escrita como

$$\mathbf{p}_d - \mathcal{K}(\mathbf{q}_i) = \mathbf{J}(\mathbf{q}_i)\Delta\mathbf{q}. \quad (5.17)$$

Resolvendo a Equação 5.17 para $\Delta\mathbf{q}$, encontra-se

$$\mathbf{J}^{-1}(\mathbf{q}_i)(\mathbf{p}_d - \mathcal{K}(\mathbf{q}_i)) = \Delta\mathbf{q}. \quad (5.18)$$

Agora, é possível utilizar aproximação de Newton-Raphson para encontrar \mathbf{q}_d (LYNCH; PARK, 2017). Definindo $\mathbf{e} = (\mathbf{p}_d - \mathcal{K}(\mathbf{q}))$, enquanto $\|\mathbf{e}\| > \epsilon$ para um ϵ pequeno, o primeiro passo do cálculo numérico é, dado uma aproximação inicial \mathbf{q}_0 ,

$$\mathbf{e}_0 = (\mathbf{p}_d - \mathcal{K}(\mathbf{q}_0)), \quad (5.19)$$

se $\|\mathbf{e}_0\| > \epsilon$, prossegue-se com

$$\mathbf{q}_1 = \mathbf{q}_0 + \mathbf{J}^{-1}(\mathbf{q}_0)\mathbf{e}_0, \quad (5.20)$$

e o processo é repetido até convergir, da forma

$$\mathbf{e}_i = (\mathbf{p}_d - \mathcal{K}(\mathbf{q}_i)), \quad (5.21)$$

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \mathbf{J}^{-1}(\mathbf{q}_i)\mathbf{e}_i, \quad (5.22)$$

um pseudo código para a aplicação dessa estratégia pode ser observado no Algoritmo 1, onde as funções *forwardKinematics* e *getJacobian* são funções que aplicam a cinemática direta e calculam o Jacobiano, respectivamente.

Algoritmo 1: Cinemática Inversa por Newton Raphson

```

Data:  $\mathbf{p}_d$ ,  $\mathbf{q}_0$ ,  $\epsilon$ , MaxInteractions
Result:  $\mathbf{q}_d$ 
Inicialização;
 $\mathcal{K}(\mathbf{q}_0) = \text{forwardKinematics}(\mathbf{q}_0)$ ;
 $\mathbf{e} = \mathbf{p}_d - \mathcal{K}(\mathbf{q}_0)$ ;
if  $\mathbf{e} < \epsilon$  then
  | return  $\mathbf{q}_0$ ;
end
IntCount = 1;
 $\mathbf{q}_i = \mathbf{q}_0$ ;
while  $\mathbf{e} < \epsilon$  do
  |  $\mathbf{J}(\mathbf{q}_i) = \text{getJacobian}(\mathbf{q}_i)$ ;
  |  $\mathbf{q}_{i+1} = \mathbf{q}_i + \mathbf{J}^{-1}(\mathbf{q}_i)\mathbf{e}$ ;
  |  $\mathcal{K}(\mathbf{q}_i) = \text{forwardKinematics}(\mathbf{q}_{i+1})$ ;
  |  $\mathbf{e} = \mathbf{p}_d - \mathcal{K}(\mathbf{q}_i)$ ;
  |  $\mathbf{q}_i = \mathbf{q}_{i+1}$ ;
  | IntCount + = 1;
  | if IntCount > MaxInteractions then
  | | break;
  | end
end
return  $\mathbf{q}_i$ ;

```

5.3 Cinemática Inversa em malha fechada

Reconstrução em malha aberta das variáveis de junta através de integração numérica inevitavelmente conduz a flutuação na solução e a erros no espaço cartesiano. Um algoritmo de cinemática inversa em malha fechada pode ser projetado para corrigir esse erro, baseado no erro cartesiano entre a pose desejada \mathbf{p}_d e atual \mathbf{p} do efetuador, como $\mathbf{p}_d - \mathbf{p}$.

Nesse ponto, a equação cinemática 5.2 é considerada na forma

$$\dot{\mathbf{p}} = \mathbf{J}_a(\mathbf{q})\dot{\mathbf{q}}, \quad (5.23)$$

pois a definição do erro no espaço cartesiano requer a utilização do Jacobiano analítico em vez do geométrico. Nisso, o algoritmo mais simples de malha fechada toma a forma de um controle proporcional

$$\dot{\mathbf{q}} = \mathbf{J}_a^*(\mathbf{q})(\mathbf{p}_d - \mathbf{p}), \quad (5.24)$$

onde \mathbf{J}_a^* é a inversa do Jacobiano que pode ser definida como inversa pura, pseudo inversa ou inversa amortecida. O erro de uma pose estática vai convergir para zero pelos mínimos quadrados.

5.3.1 Definição do erro

O algoritmo de cinemática inversa em 5.24 utiliza o Jacobiano analítico uma vez que as variáveis de erro de posição e orientação são definidas no espaço cartesiano. Tratamentos diferentes são necessários para os dois tipos de erro.

O erro de posição é dado diretamente pela diferença cartesiana das posições, dado por

$$\mathbf{e}_p = \mathbf{p}_{ed} - \mathbf{p}_e, \quad (5.25)$$

onde \mathbf{p}_{ed} e \mathbf{p}_e representam a posição desejada e atual do efetuador, respectivamente.

A da rotação, no entanto, não é dada de forma direta. Seja $\mathbf{R}_{ed} = \begin{pmatrix} n_{ed} & o_{ed} & a_{ed} \end{pmatrix}$ a matriz de rotação desejada para a referência do efetuador, então o erro de orientação é dado por

$$\mathbf{e}_o = \mathbf{u}_r \sin(\theta), \quad (5.26)$$

onde \mathbf{u}_r é um vetor unitário e o ângulo θ descreve a rotação equivalente

$$\mathbf{R}_r(\theta) = \mathbf{R}_{ed} \mathbf{R}_e^T, \quad (5.27)$$

necessária para alinhar \mathbf{R}_e com \mathbf{R}_{ed} . Essa matriz pode ser convertida para a representação angular de três rotações utilizada no efetuador e o vetor de erro é dado por

$$\mathbf{e} = \begin{pmatrix} x_e & y_e & z_e & \phi_e & \theta_e & \rho_e \end{pmatrix}^T \quad (5.28)$$

5.3.1.1 Erro absoluto

Uma vez definido o erro pela Equação 5.28, o erro absoluto da posição e rotação podem ser definidos como

$$|\mathbf{e}_p| = \sqrt{x_e^2 + y_e^2 + z_e^2}, \quad (5.29)$$

$$|\mathbf{e}_o| = \sqrt{\phi_e^2 + \theta_e^2 + \rho_e^2}. \quad (5.30)$$

5.4 Resposta ao degrau

Nesses experimentos, a pose final a ser alcançada foi determinada apenas como um degrau de 10 centímetros nas direções x, y e z. A pose inicial escolhida foi uma longe de singularidades.

A intenção desse experimento é determinar a velocidade de resposta dos controladores internos do robô a um degrau na posição e validar o algoritmo de cinemática inversa em malha fechada.

Um pseudo código pode ser observado no Algoritmo 2, onde a função *GetInverseJacobian* varia de acordo com o método de inversa utilizado. O erro calculado é visto no Algoritmo 3. As funções *getCartesianPose* e *getJointVector* se comunicam com o robô para adquirir o vetor de pose cartesiana e o vetor de juntas, respectivamente. No caso do UR5, a pose cartesiana retornada tem sua rotação representada em notação vetor-ângulo, e uma transformação tem que ser feita para utilização em notação matricial. A função *getInverseJacobian* é a responsável por fazer o cálculo da inversão do jacobiano, de acordo com o método definido (inversa, pseudo inversa ou inversa amortecida). Por fim, a função *sendSpeedToRobot* é a responsável por abrir a comunicação com o robô e enviar a velocidade calculada (por comando *speedj* ou o conjunto de valores para o RTDE).

Algoritmo 2: Resposta ao Degrau

```

Data:  $\mathbf{p}_d$ 
/* Inicialização - Dados da pose desejada */
Robot R1;
/* Dados da pose atual */
 $\mathbf{p}_e = \text{getCartesianPose}(R1);$ 
 $\mathbf{e} = \text{computeErro}(\mathbf{p}_d, \mathbf{p}_e);$ 
/* Inicio */
TempoTotal = T;
Tempo = 0;
while Tempo < TempoTotal do
     $\mathbf{p}_e = \text{getCartesianPose}(R1);$ 
     $\mathbf{q} = \text{getJointVector}(R1);$ 
     $\mathbf{J}_{\text{inv}} = \text{getInverseJacobian}(\mathbf{p}_e, \mathbf{q});$ 
     $\mathbf{e} = \text{computeErro}(\mathbf{p}_d, \mathbf{p}_e);$ 
     $\dot{\mathbf{q}} = \mathbf{J}_{\text{inv}}\mathbf{e};$ 
    sendSpeedToRobot(R1,  $\dot{\mathbf{q}}$ );
end

```

5.4.1 Degrau em coordenadas cartesianas

Os gráficos apresentam as coordenadas de juntas e cartesianas, e as velocidades de juntas e velocidades cartesianas para o degrau no eixo z com os diferentes métodos de

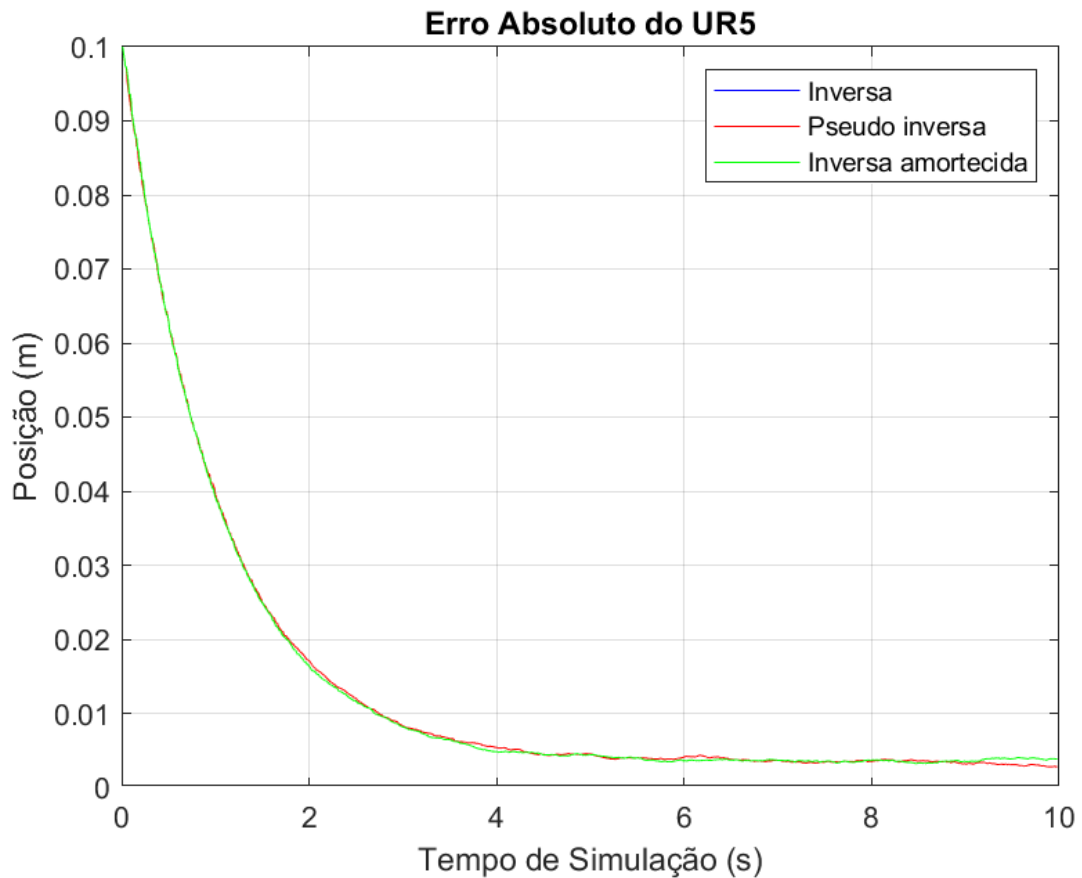
Algoritmo 3: Cálculo do Erro - *computeErro()*

Data: $\mathbf{p}_d, \mathbf{p}_e$
 $\mathbf{e}_p = \mathbf{p}_d(1:3) - \mathbf{p}_e(1:3);$
 $\mathbf{R}_{ed} = \text{rotationToMatrix}(\mathbf{p}_d(4:6));$
 $\mathbf{R}_e = \text{rotationToMatrix}(\mathbf{p}_e(4:6));$
 $\mathbf{R}_r(\theta) = \mathbf{R}_{ed}\mathbf{R}_e^T;$
 $\mathbf{e}_o = \text{matrixToRotation}(\mathbf{R}_r(\theta));$
 $\mathbf{e} = (\mathbf{e}_p, \mathbf{e}_o);$
return \mathbf{e}

inversa de Jacobiano. Foram realizados simulações e experimentos para cada método, com os resultados a seguir. O ponto inicial escolhido para aplicação dos degraus na simulação e nos experimentos foi um com posição $\mathbf{p} = [-0.6410, 0.0794, 0.0462]$. Esse ponto foi escolhido pois é o mesmo ponto inicial utilizado na simulação de controle de trajetória.

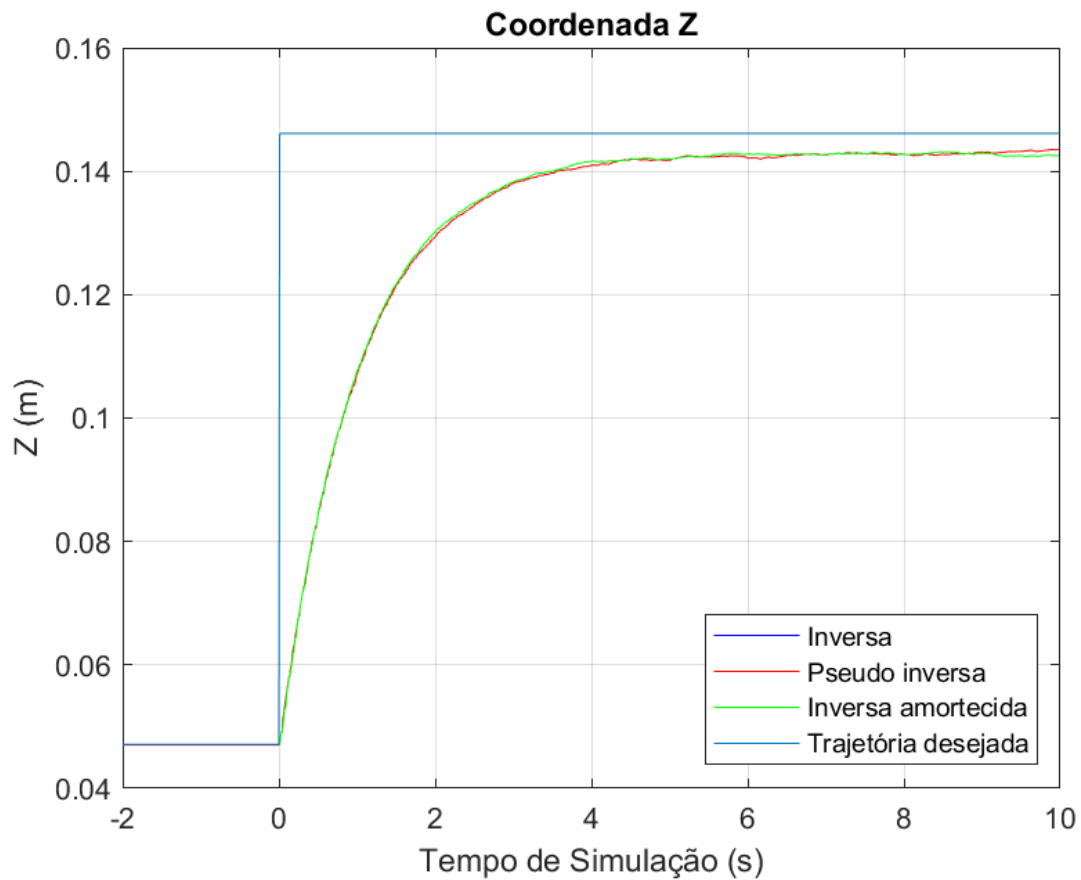
5.4.1.1 Simulação

Figura 5.1 – Erro absoluto de posição para resposta ao degrau.



Fonte: Próprio Autor.

Figura 5.2 – Trajetória no eixo Z para resposta ao degrau.

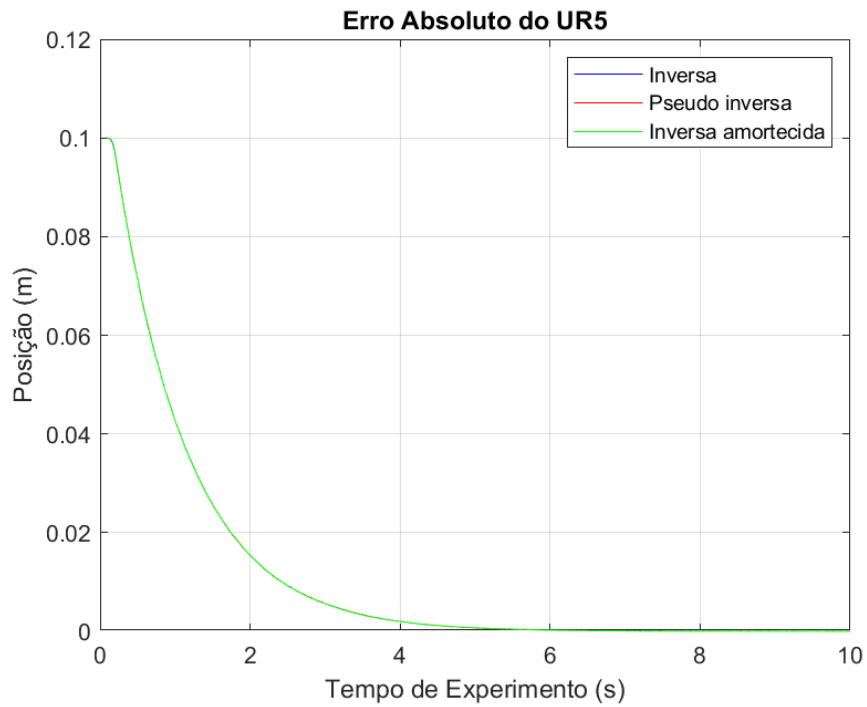


Fonte: Próprio Autor.

É possível observar nas simulações que os resultados são praticamente os mesmos. Isso faz sentido do ponto de vista que a matriz jacobiana do manipulador tem posto completo e o degrau não passa por nenhuma singularidade. Mais especificamente, o resultado da inversa comum e da inversa amortecida se sobrepõe, uma vez que sem singularidades a inversa amortecida é reduzida a inversa, enquanto há uma pequena disparidade no resultado com a pseudo inversa graças a arredondamento e truncamento nas operações para calcular a pseudo inversa.

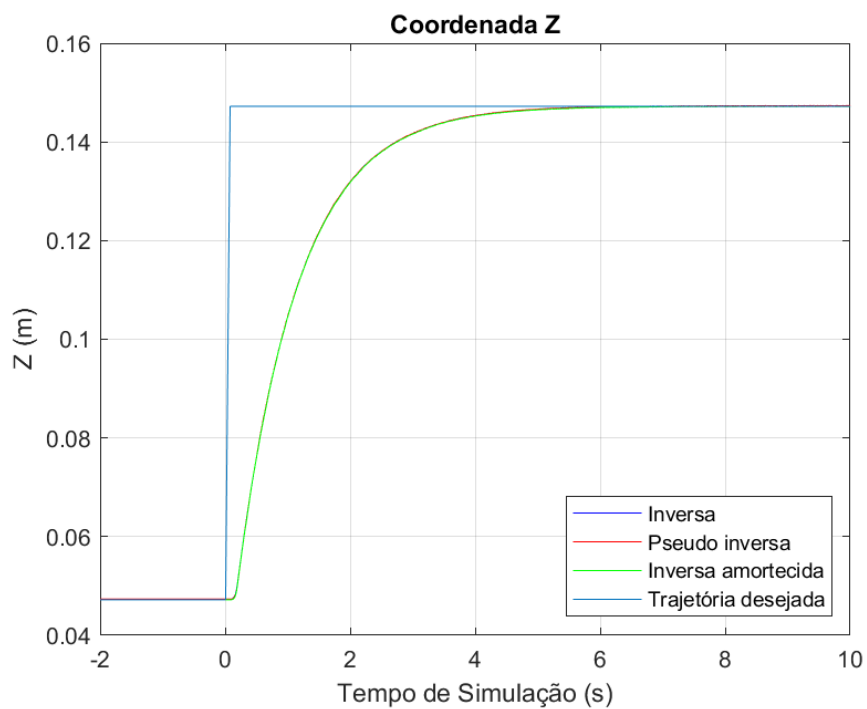
5.4.1.2 Experimento

Figura 5.3 – Erro absoluto de posição para resposta ao degrau - Experimento.



Fonte: Próprio Autor.

Figura 5.4 – Trajetória no eixo Z para resposta ao degrau - Experimento.



Fonte: Próprio Autor.

Nos experimentos, o comportamento se repete, como esperado. É possível observar, no entanto, que nas simulações há um erro de regime permanente, enquanto nos experimentos esse erro é eliminado. Os resultados das inversas no experimento são quase os mesmos, com uma variação insignificante que pode ser atribuída a ruído na aquisição dos dados. Esse resultado é esperado pela mesma razão que era esperado na simulação, a matriz jacobiana do robô tem posto completo e a trajetória não passa por uma singularidade.

5.4.2 Análise de resultados

Os métodos de inversa, pseudo inversa e inversa amortecida do Jacobiano se mostram equivalentes para trajetórias que não passam por uma singularidade. Uma média do cálculo de dez inversas de cada método foi realizado e pode ser observado na Tabela 6. Esse tempo de processamento foi obtido utilizando a biblioteca *numpy* em um sistema Linux 16.04 com o mínimo de processos ativos. Para o caso da inversa amortecida, foi considerado o cálculo de λ no tempo de processo.

Tabela 6 – Tempo de processamento de inversão do jacobiano

Método	Tempo
Inversa	2.722×10^{-5} s
Pseudo Inversa	1.582×10^{-4} s
Inversa Amortecida	1.213×10^{-4} s

O cálculo da pseudo inversa é necessário sempre que o manipulador não for de norma completa (sua matriz jacobiana tem posto completo), e nesse caso foi calculado utilizando a função *pinv* do *numpy*, o que pode explicar seu tempo de processamento maior do que a inversa amortecida.

Apesar de quase cinco vezes mais lento que a inversa direta, as qualidades da inversa amortecida em evitar singularidades faz dela a melhor escolha para aplicações em tempo real.

5.4.2.1 Comparação entre simulação e experimento

Para todos os métodos, é possível observar um erro de regime permanente na simulação para um degrau em coordenadas cartesianas, enquanto nos testes experimentais esse erro não ocorre.

De fato, ao analisar a Equação 5.24, para um único ponto no espaço não há erro de regime permanente pois a inversa do jacobiano atua como um gradiente levando o erro ao mínimo local de zero pelos mínimos quadrados.

O erro em regime permanente para as simulações revela uma falha no modelo cinemático do simulador V-REP, refletida ao adquirir os dados do jacobiano do mesmo.

Esse erro associado ao Jacobiano pode ser associado a um truncamento ou arredondamento nos dados, um ruído atribuído pelo próprio V-REP ou uma incerteza associada a posição de referência do efetuador utilizada para adquirir o jacobiano.

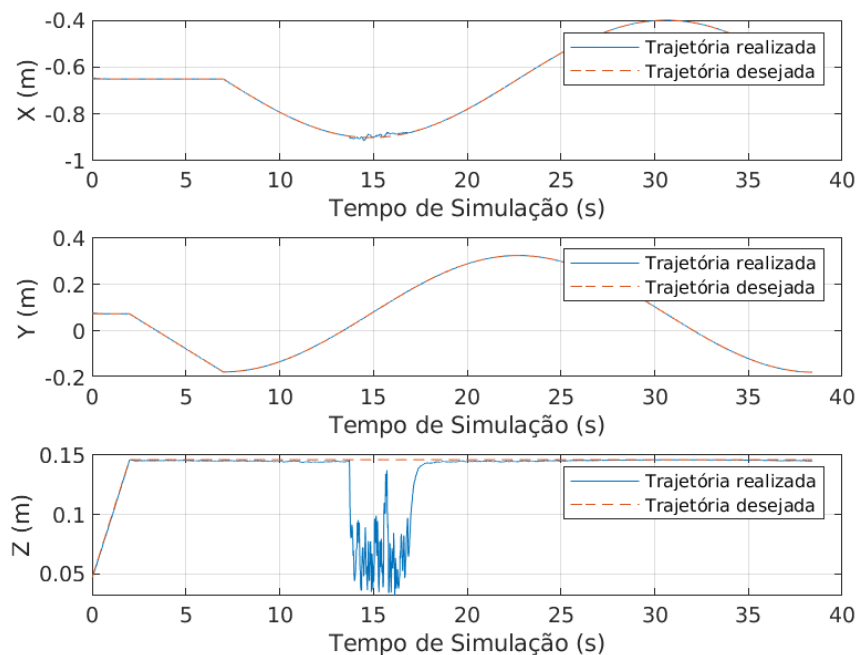
5.5 Movimento através de singularidade

De forma a exemplificar o efeito de uma singularidade na trajetória, foi realizada uma simulação da trajetória base definida no Capítulo 4 com um raio $r = 0.25m$, grande o suficiente para que, graças a sua posição inicial $\mathbf{p} = [-0.6410, 0.0794, 0.0462]$, parte da trajetória se encontre fora do espaço de trabalho do robô, pois no limite da trajetória o vetor estará a $0.9m$ da base do robô, além do seu alcance de $0.85m$. O robô foi controlado utilizando a estratégia de controle proporcional feedforward, detalhado na Seção 5.6. O fluxograma e algoritmos de controle também são os mesmos da seção 5.6.

A mesma trajetória foi executada com e sem a inversa amortecida em simulação, de modo que as velocidades das juntas não estão fisicamente limitadas pelos motores do robô real.

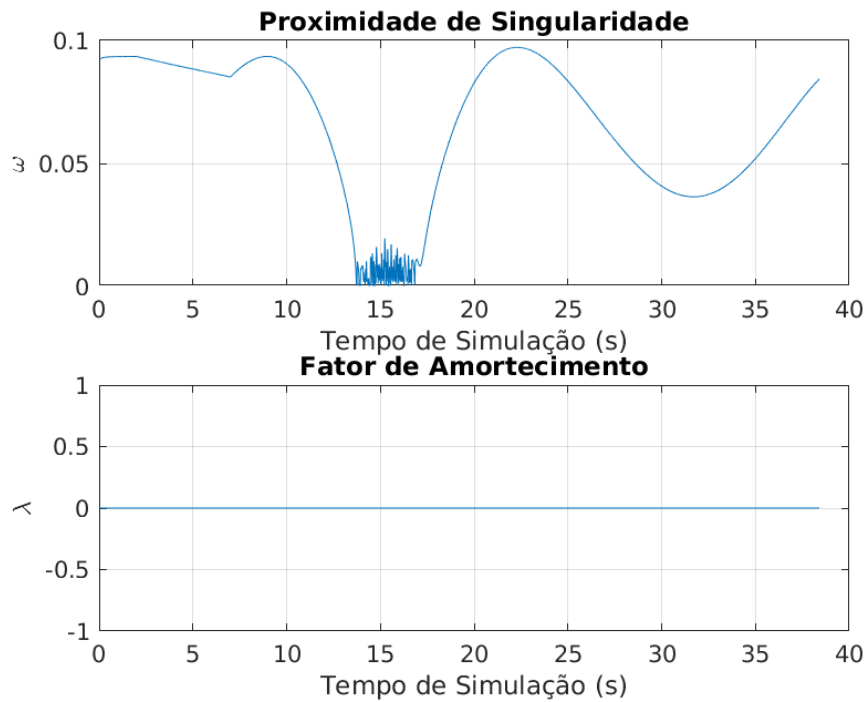
5.5.1 Controle cinemático sem inversa amortecida

Figura 5.5 – Trajetória percorrida com singularidade, sem amortecimento.



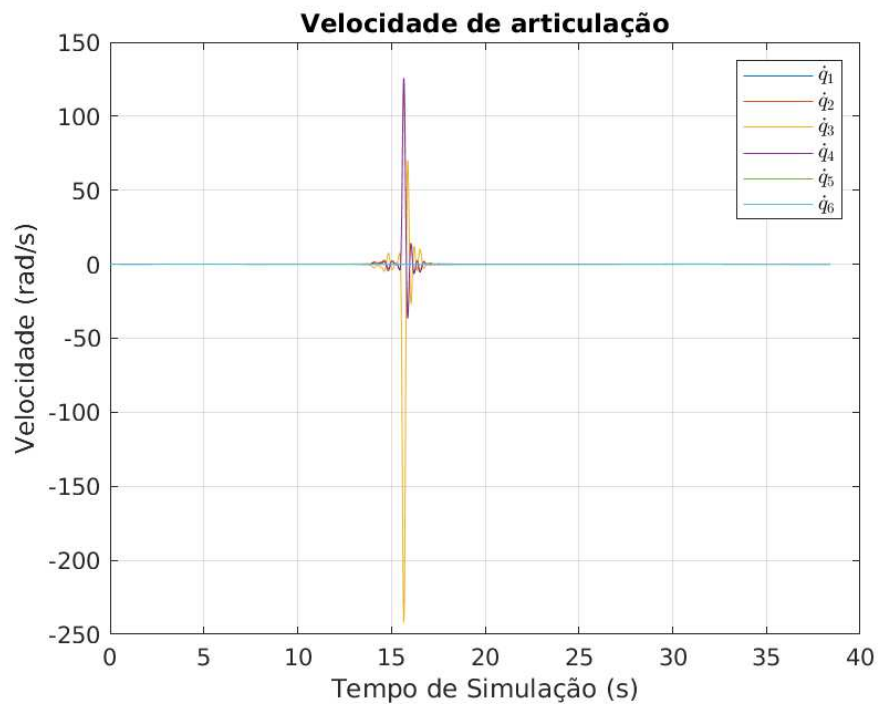
Fonte: Próprio Autor.

Figura 5.6 – Proximidade de singularidade e fator de amortecimento, sem amortecimento.



Fonte: Próprio Autor.

Figura 5.7 – Velocidade de articulação, sem amortecimento.

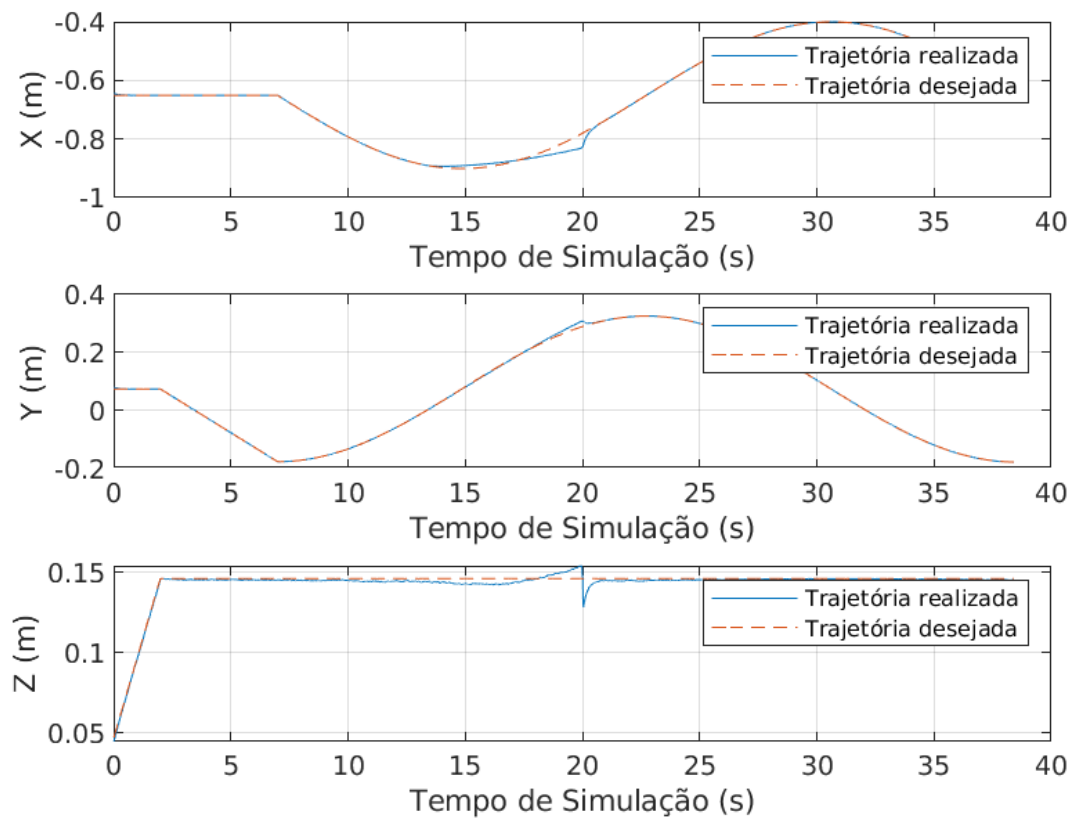


Fonte: Próprio Autor.

Vale aqui ressaltar as velocidades exorbitantes encontradas na Figura 5.7, alcançando velocidades, que não seriam possíveis em ambiente experimental, exorbitantes devido a singularidade no cálculo da inversa do jacobiano.

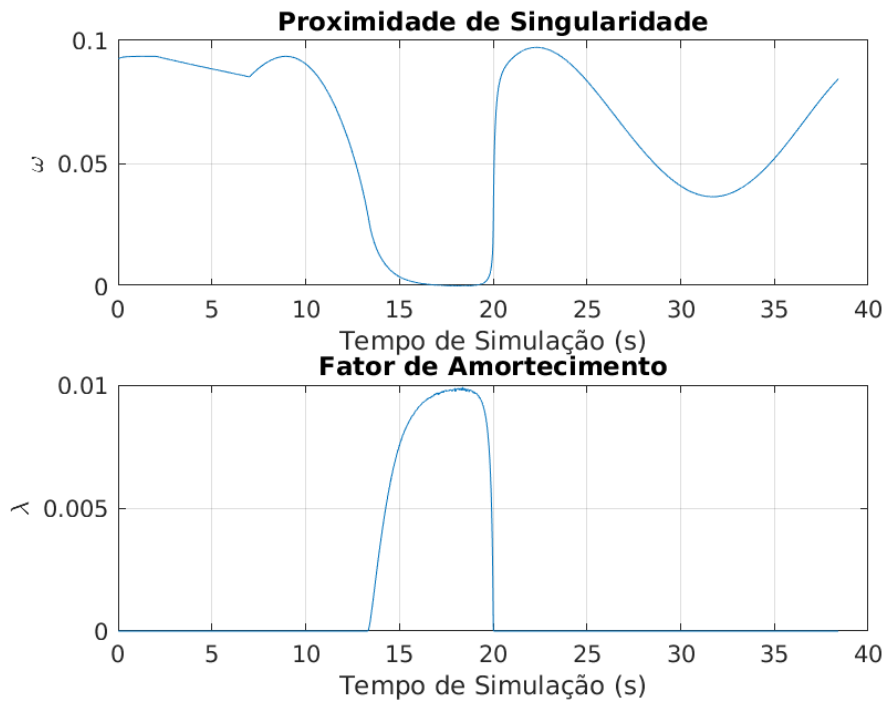
5.5.2 Controle cinemático com inversa amortecida

Figura 5.8 – Trajetória percorrida com singularidade, com amortecimento.



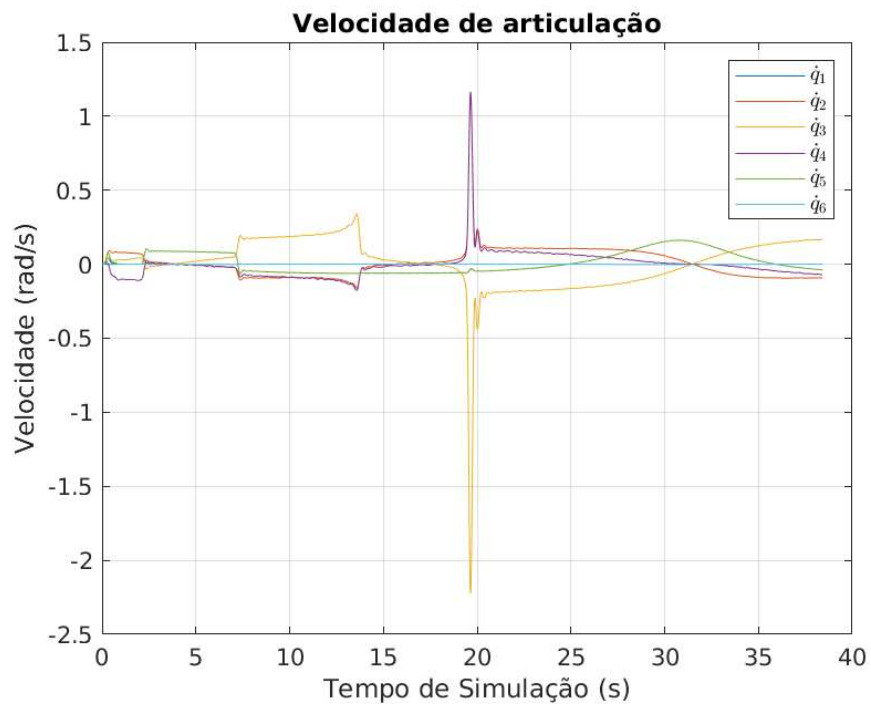
Fonte: Próprio Autor.

Figura 5.9 – Proximidade de singularidade e fator de amortecimento, com amortecimento.



Fonte: Próprio Autor.

Figura 5.10 – Velocidade de articulação, com amortecimento.



Fonte: Próprio Autor.

5.5.3 Análise de resultados

É possível observar nas Figuras 5.6 e 5.10 quando o manipulador se aproxima da área de singularidade, ao tentar se mover para uma área fora da área de trabalho no qual ele atua, rapidamente a manipulabilidade se aproxima de zero, e sem um fator de amortecimento, não só o robô não alcança a trajetória (Figura 5.5) como a inversa do jacobiano resulta em sua velocidade aumentar para valores altíssimos em um espaço de tempo quase instantâneo (Figura 5.7).

Em contrapartida, ao introduzir um fator de amortecimento (Figura 5.9), o manipulador faz um desvio da trajetória planejada não alcançável (Figura 5.8) com uma variação muito menor do que o mesmo percurso sem amortecimento, porém com as velocidades das articulações muito mais contidas (Figura 5.10), de modo a não introduzir comportamentos excepcionais nas juntas devido a inversa do jacobiano.

Na simulação, o fator de amortecimento λ e a manipulabilidade mínima ω_0 foram escolhidas de forma a evitar o máximo possível o desvio na trajetória e de forma suave. Os valores foram $\lambda = 0.01$ e $\omega_0 = 0.02$. Diferentes técnicas para uma escolha otimizada desses parâmetros são apresentadas na literatura, mas seu estudo está fora do escopo do trabalho.

5.6 Estratégias de controle

Dado uma trajetória no espaço cartesiano definida por $(\mathbf{p}_d, \dot{\mathbf{p}}_d)$ e a cinemática com Jacobiano analítico apresentada na equação 5.23, uma série de estratégias de controle em malha fechada podem ser desenvolvidas. As mais utilizadas são as de *Controle Proporcional*, *Controle Proporcional e Integral* e *Controle Proporcional e Feedforward*.

Pela análise realizada com os métodos de inversão do Jacobiano, o método escolhido para utilização nas estratégias de controle foi o da *Inversa Amortecida*. Mesmo sem a presença de singularidades na trajetória, o custo computacional extra da inversa amortecida não afeta os algoritmos de controle de forma significativa e é o método mais seguro.

5.6.1 Controle Proporcional

O sistema em malha fechada apresentado na equação 5.24 pode ser modificado para incluir um ganho proporcional, na forma

$$\dot{\mathbf{q}} = \mathbf{J}_a^*(\mathbf{q})\mathbf{K}_p(\mathbf{p}_d - \mathbf{p}) \quad (5.31)$$

onde $\mathbf{K}_p \in \mathbb{R}^{m \times m}$ é uma matriz diagonal definida positiva. m é função da quantidade de juntas do robô. No caso do UR5, $\mathbf{K}_p \in \mathbb{R}^{6 \times 6}$.

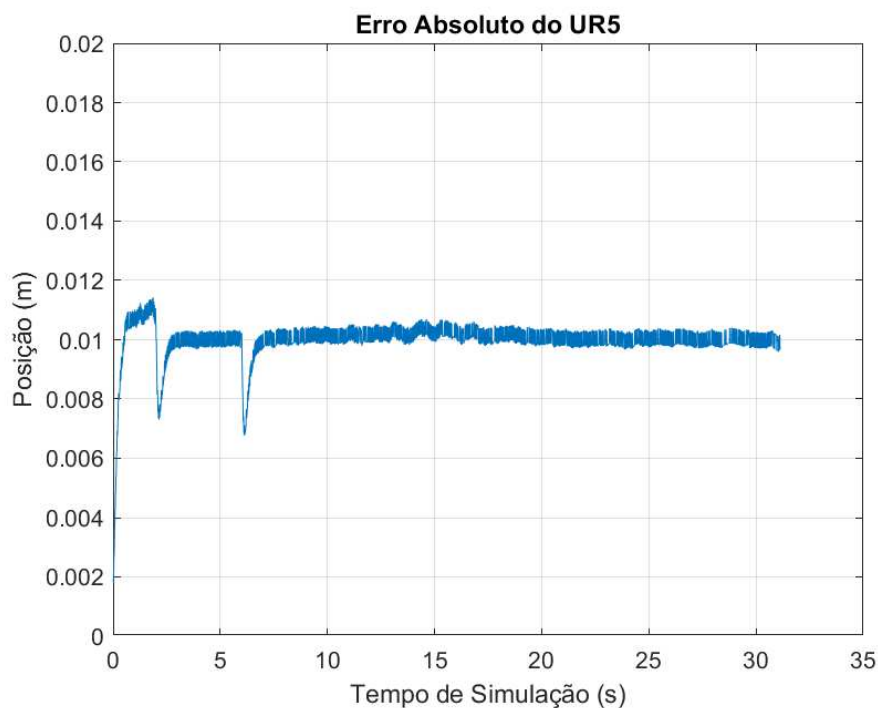
O erro de uma pose estática vai convergir para zero a uma taxa que depende dos ganhos de \mathbf{K} . Porém, essa malha é baseada em controle de realimentação negativa clássica. Ter erro de pose zero enquanto acompanha uma trajetória significaria uma demanda de velocidade zero, que faz o sistema contraditório. Formalmente, o sistema é do Tipo 1, pois possui um integrador para se obter a posição atual, no que decorre que ele possui um erro constante para uma entrada de degrau.

Esse erro pode ser resolvido de duas formas, adicionando um integrador no erro proporcional ou utilizando a estratégia feedforward.

Para as simulações e experimentos a seguir, o ganho utilizado foi $\mathbf{K}_p = 5\mathbf{I}$.

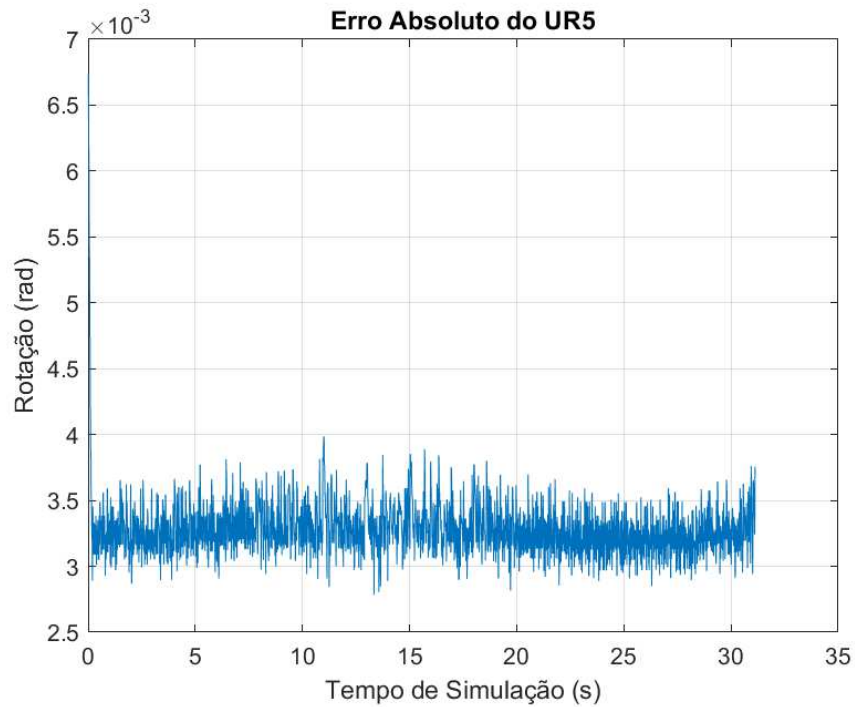
5.6.1.1 Simulação do Controle Proporcional

Figura 5.11 – Erro Absoluto de Posição do controle cinemático com Controle Proporcional.



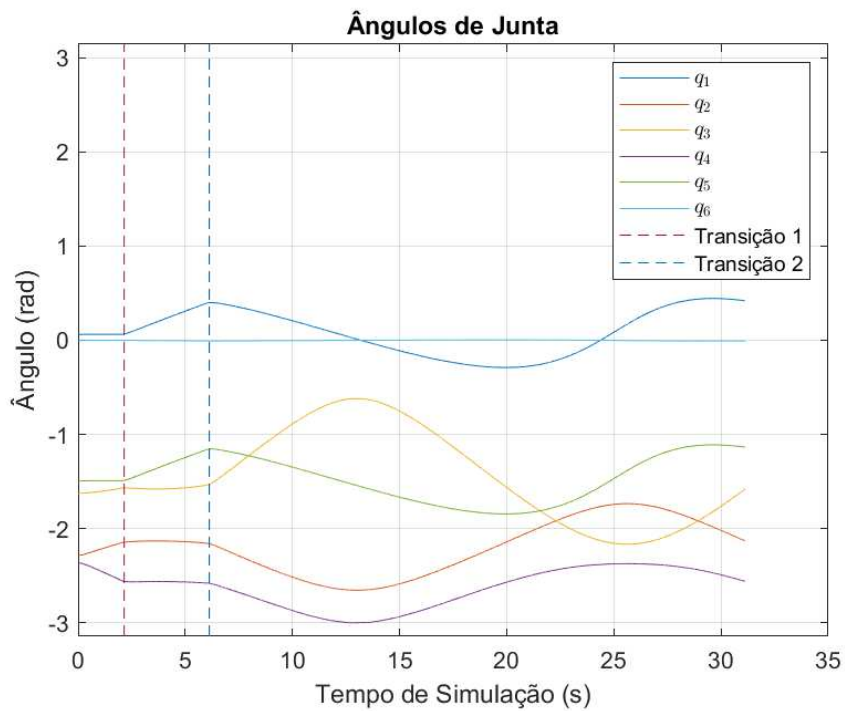
Fonte: Próprio Autor.

Figura 5.12 – Erro Absoluto de Rotação do controle cinemático com Controle Proporcional.



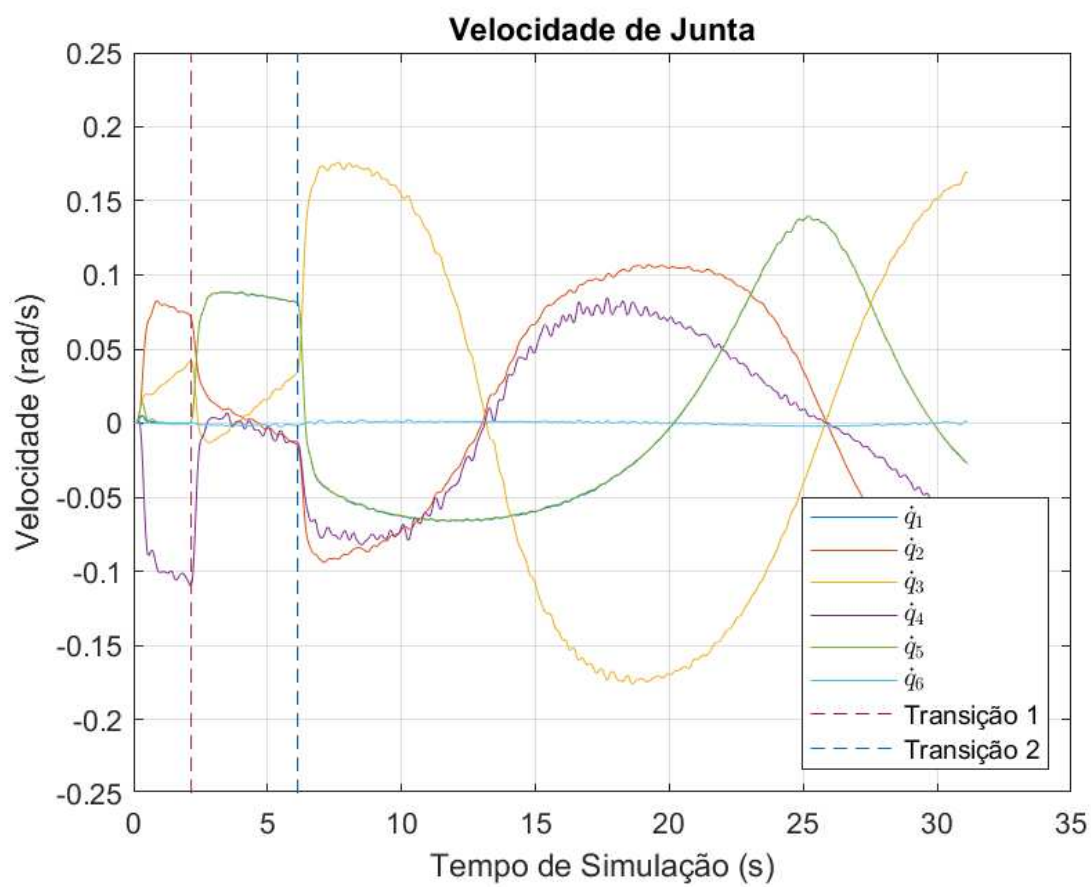
Fonte: Próprio Autor.

Figura 5.13 – Posição de Juntas do controle cinemático com Controle Proporcional.



Fonte: Próprio Autor.

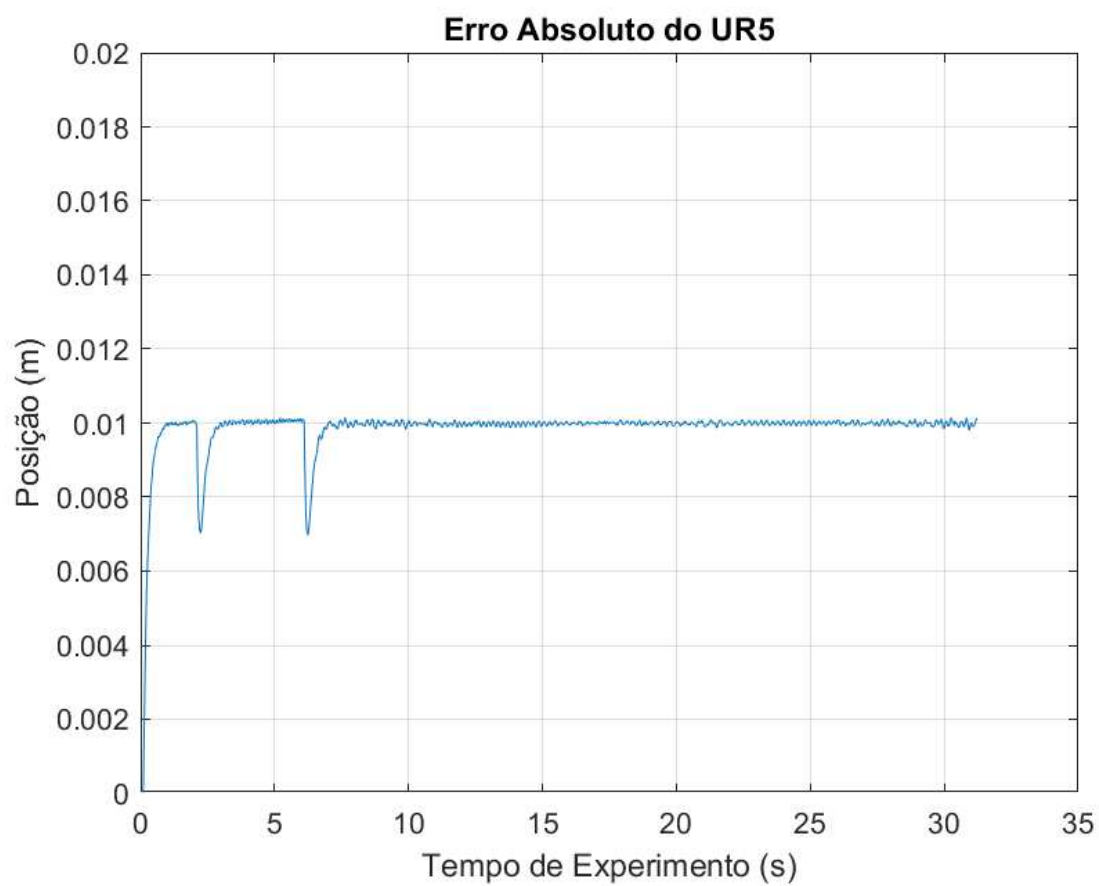
Figura 5.14 – Velocidade de Juntas do controle cinemático com Controle Proporcional.



Fonte: Próprio Autor.

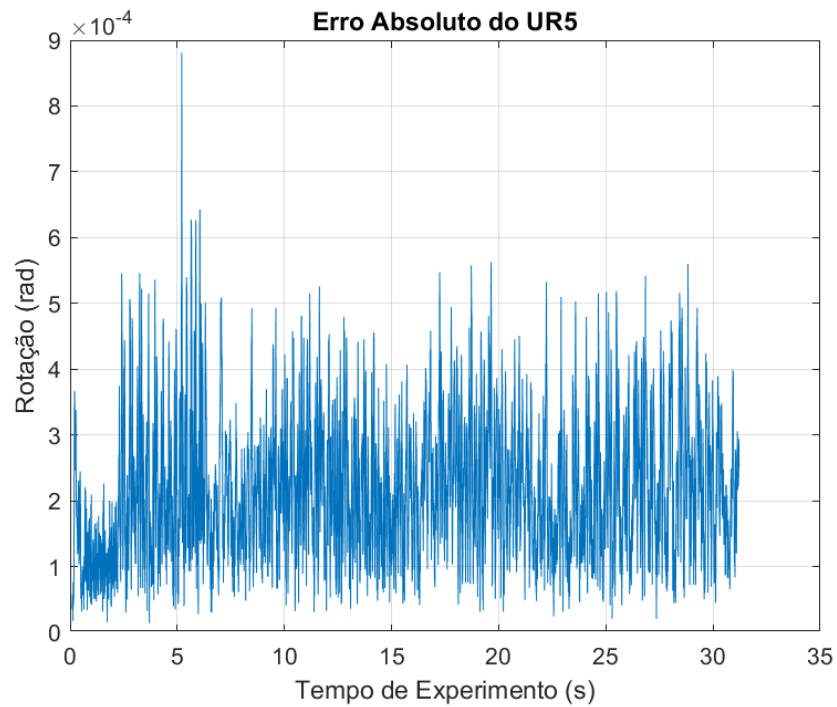
5.6.1.2 Experimento com Controle Proporcional

Figura 5.15 – Erro Absoluto de Posição do controle cinemático com Controle Proporcional - Experimento.



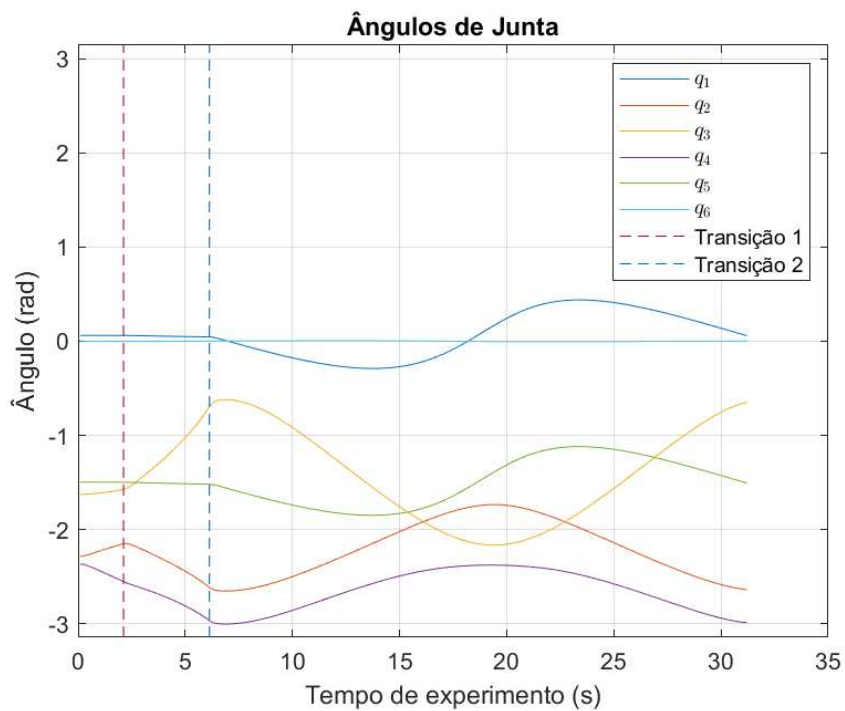
Fonte: Próprio Autor.

Figura 5.16 – Erro Absoluto de Rotação do controle cinemático com Controle Proporcional - Experimento.



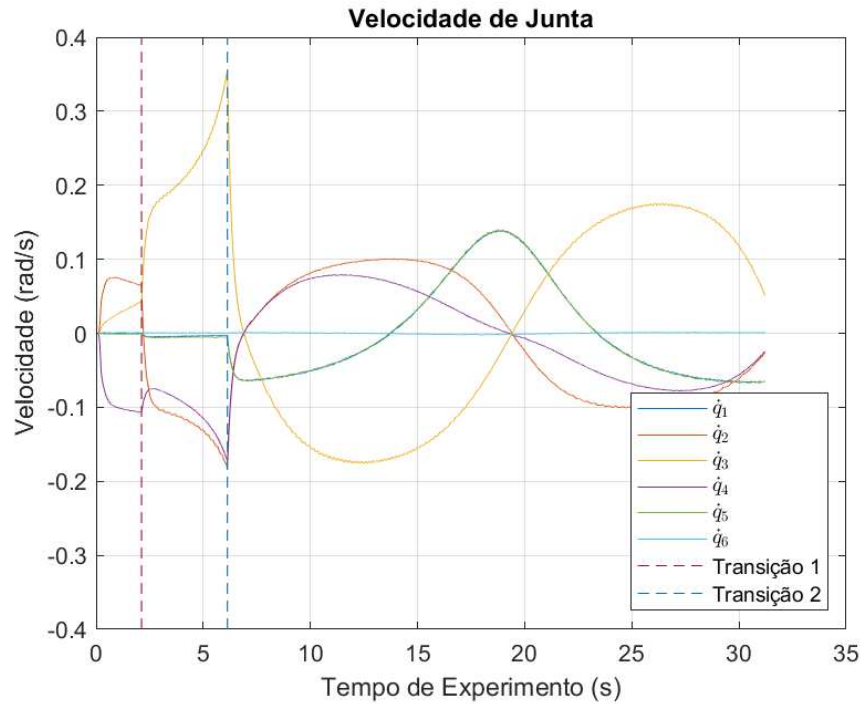
Fonte: Próprio Autor.

Figura 5.17 – Posição de Juntas do controle cinemático com Controle Proporcional - Experimento.



Fonte: Próprio Autor.

Figura 5.18 – Velocidade de Juntas do controle cinemático com Controle Proporcional - Experimento.



Fonte: Próprio Autor.

5.6.2 Controle Proporcional e Integral

Adicionando um integrador ao sistema definido em 5.31, obtém-se

$$\dot{\mathbf{q}} = \mathbf{J}_a^*(\mathbf{q})(\mathbf{K}_p \mathbf{e} + \mathbf{K}_i \int_0^t \mathbf{e} dt) \quad (5.32)$$

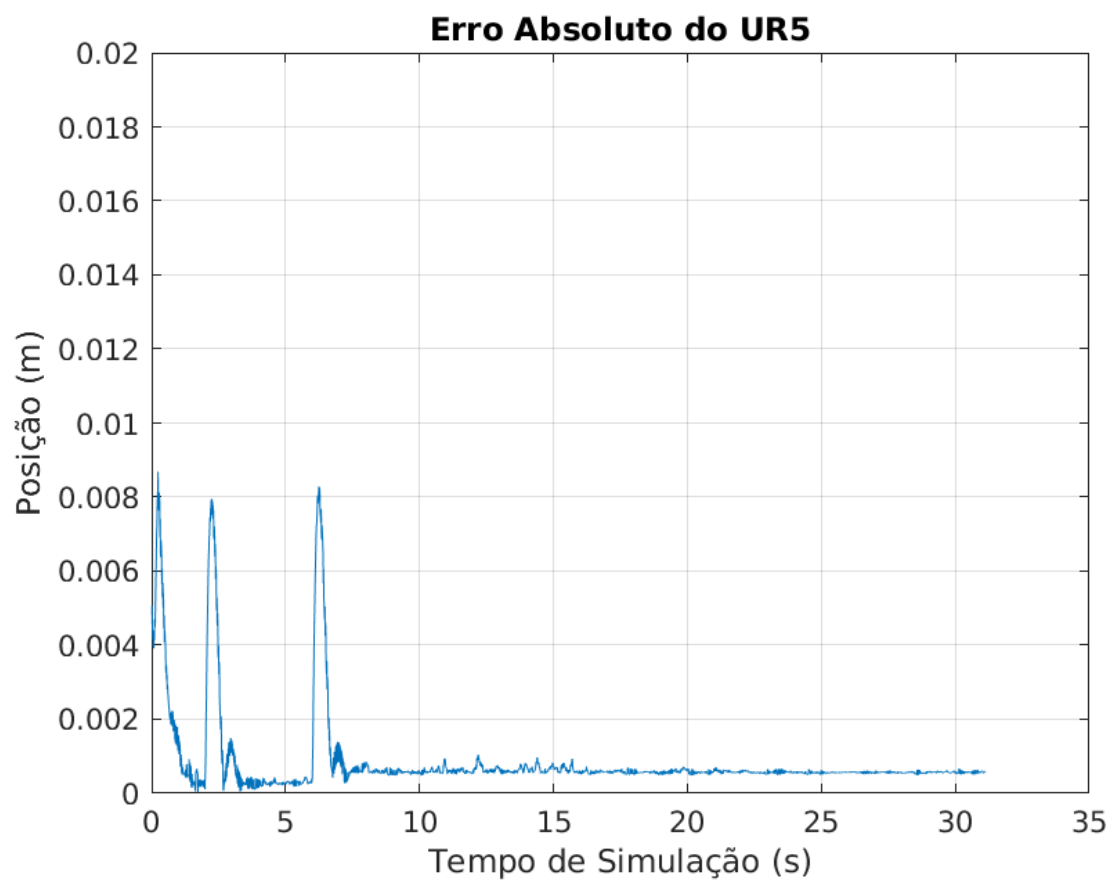
onde $\mathbf{K}_i \in \mathbb{R}^{m \times m}$ é uma matriz diagonal definida positiva.

A adição do termo integrador garante uma convergência para o erro zero. Contudo, agora é necessário o ajuste de duas matrizes de parâmetros de controle, e a adição de um termo integrador pode implicar em saturação e *integral windup*, necessitando de mais adições a malha de controle. Uma estratégia mais simples e efetiva é a de *Controle Proporcional e Feedforward*.

Para as simulações e experimentos a seguir, o ganho utilizado foi $\mathbf{K}_p = 5\mathbf{I}$ e $\mathbf{K}_i = 0.25\mathbf{I}$.

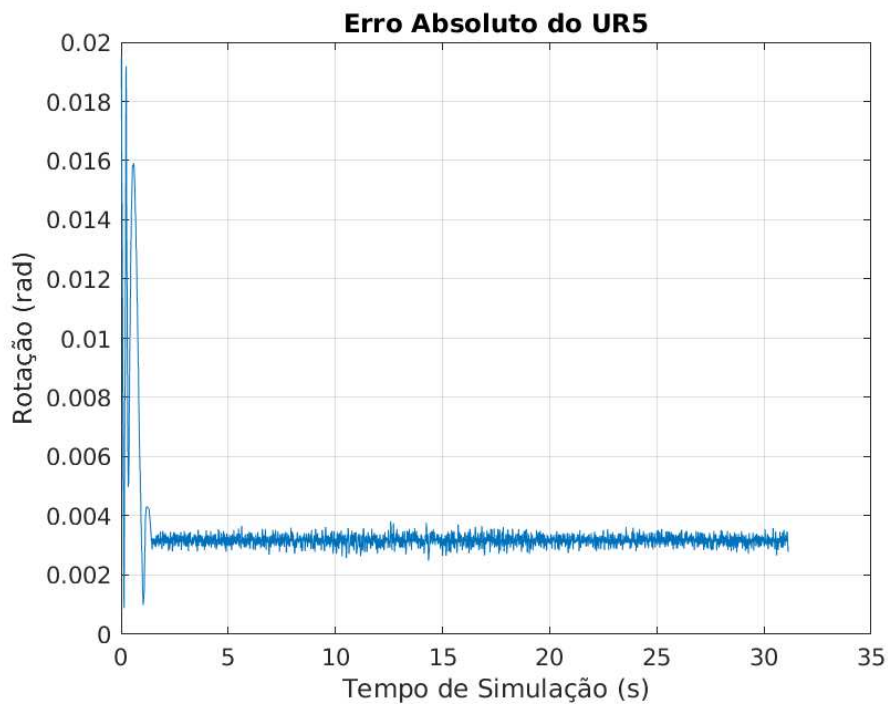
5.6.2.1 Simulação do Controle Proporcional e Integral

Figura 5.19 – Erro Absoluto de Posição do controle cinemático com Controle Proporcional e Integral.



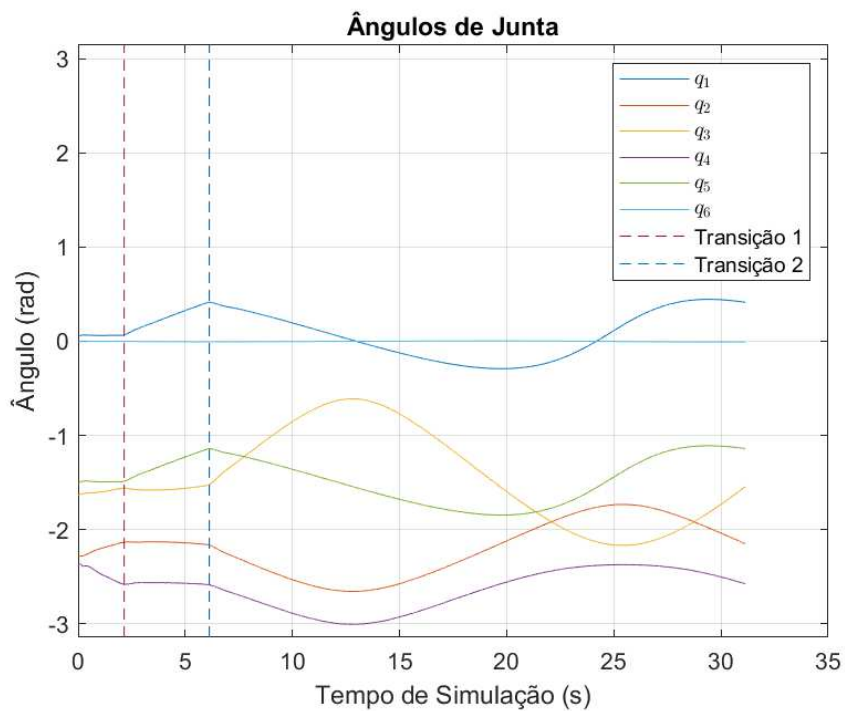
Fonte: Próprio Autor.

Figura 5.20 – Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Integral.



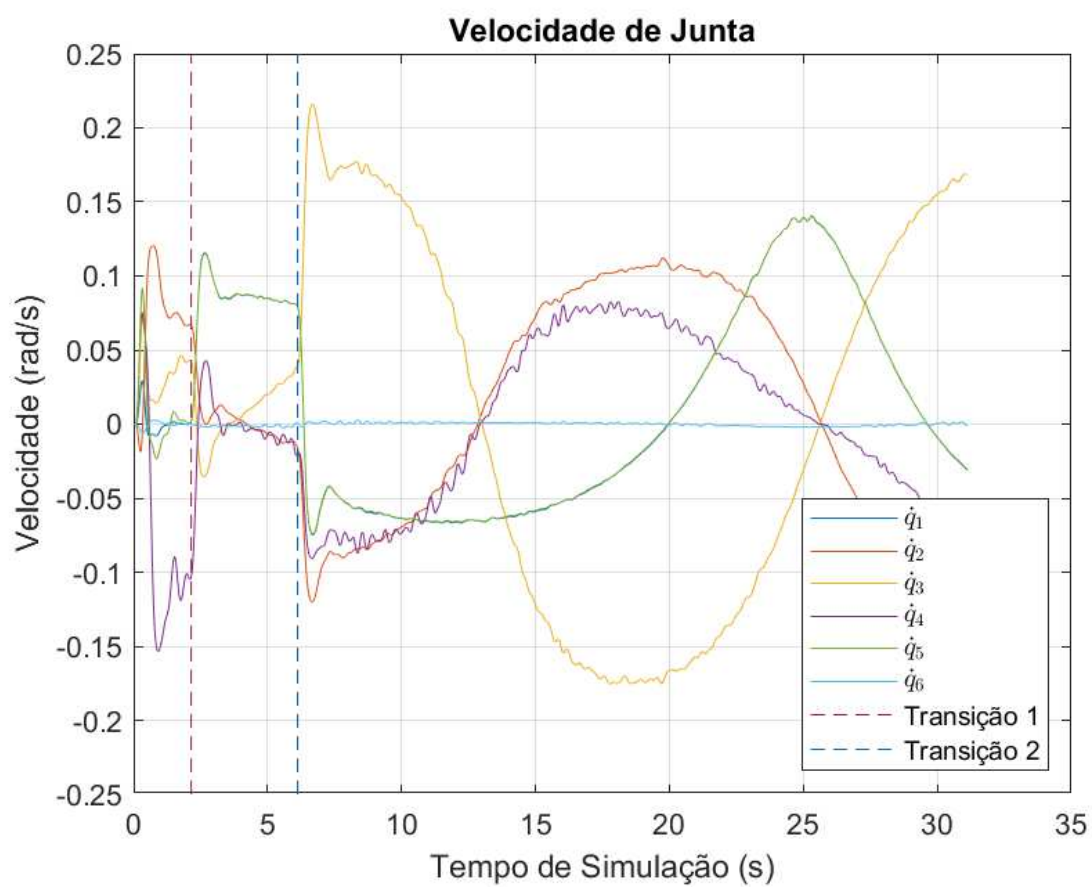
Fonte: Próprio Autor.

Figura 5.21 – Posição de Juntas do controle cinemático com Controle Proporcional e Integral.



Fonte: Próprio Autor.

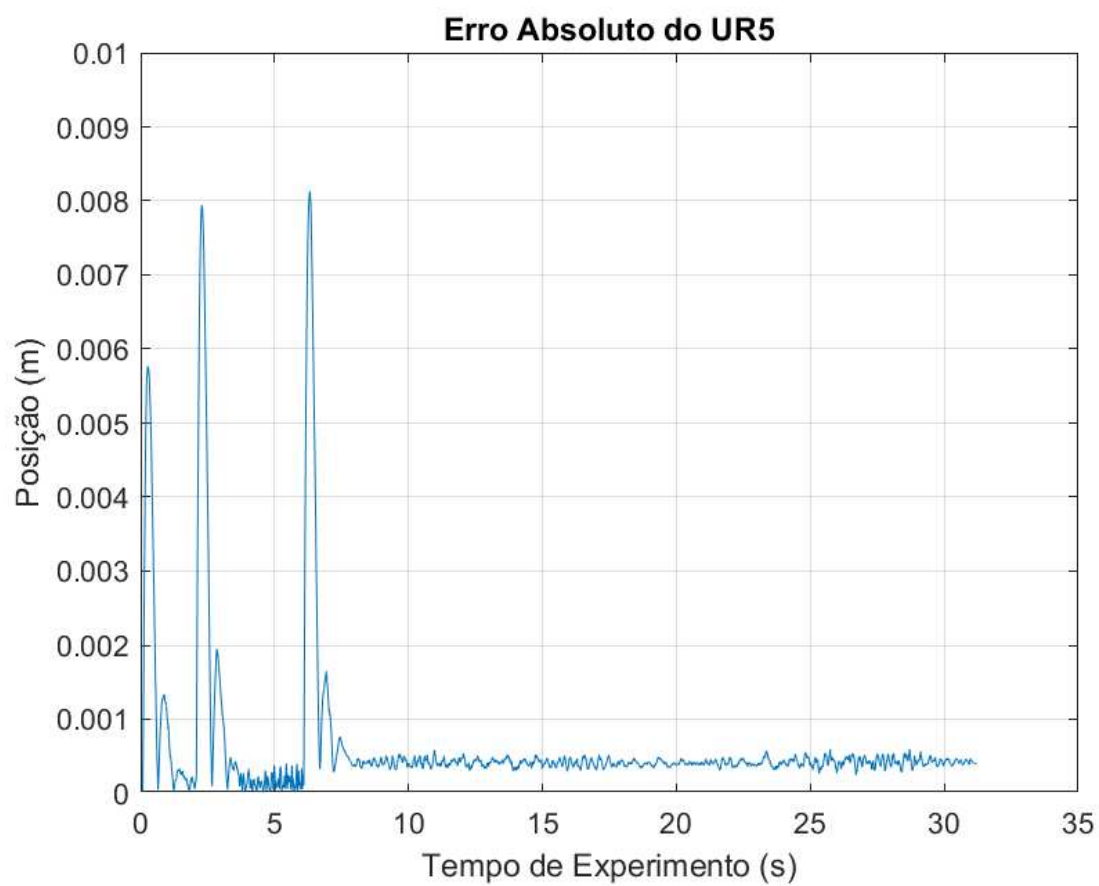
Figura 5.22 – Velocidade de Juntas do controle cinemático com Controle Proporcional Integral.



Fonte: Próprio Autor.

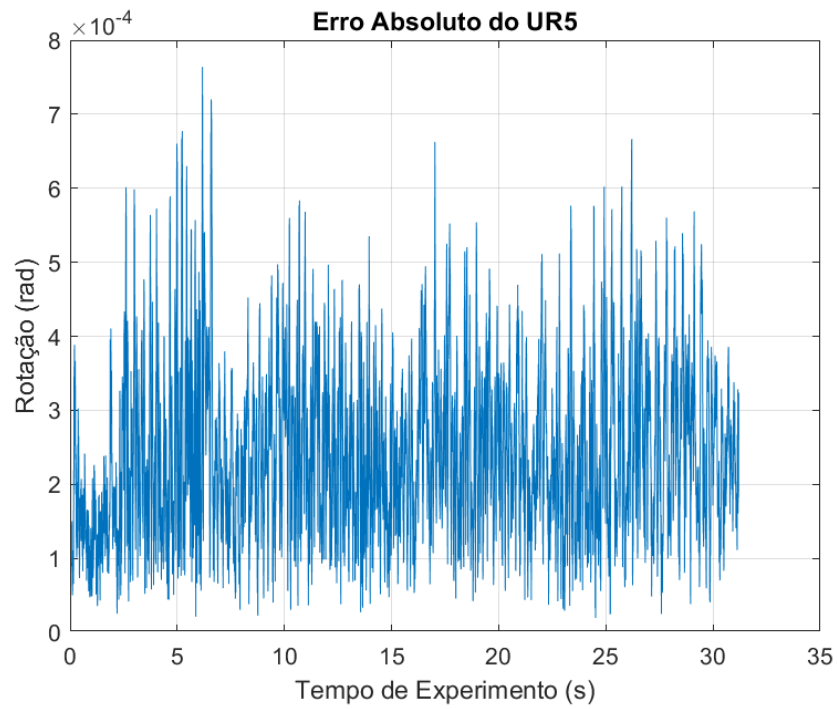
5.6.2.2 Experimento com Controle Proporcional e Integral

Figura 5.23 – Erro Absoluto de Posição do controle cinemático com Controle Proporcional e Integral - Experimento.



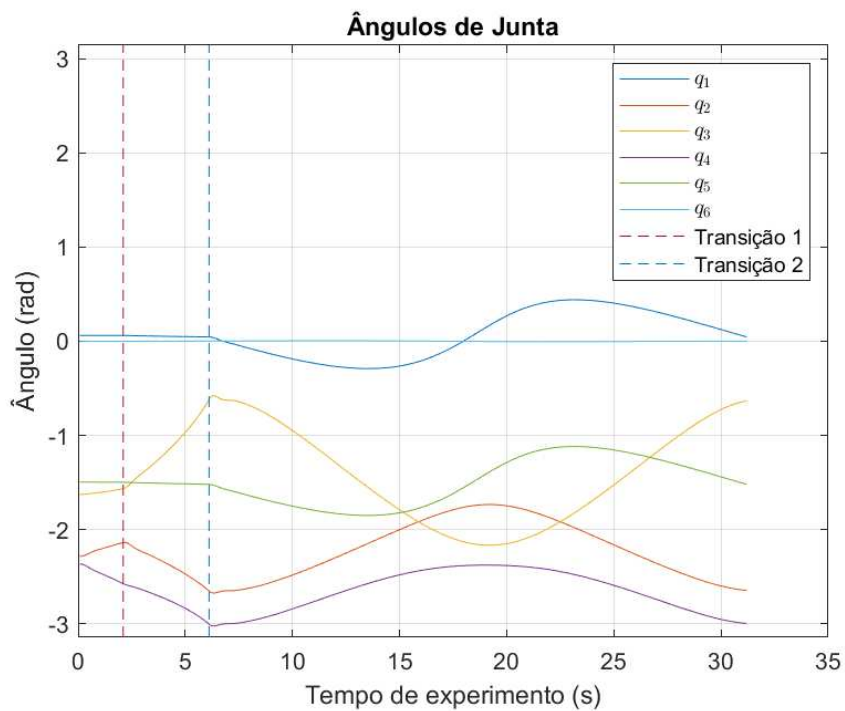
Fonte: Próprio Autor.

Figura 5.24 – Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Integral- Experimento.



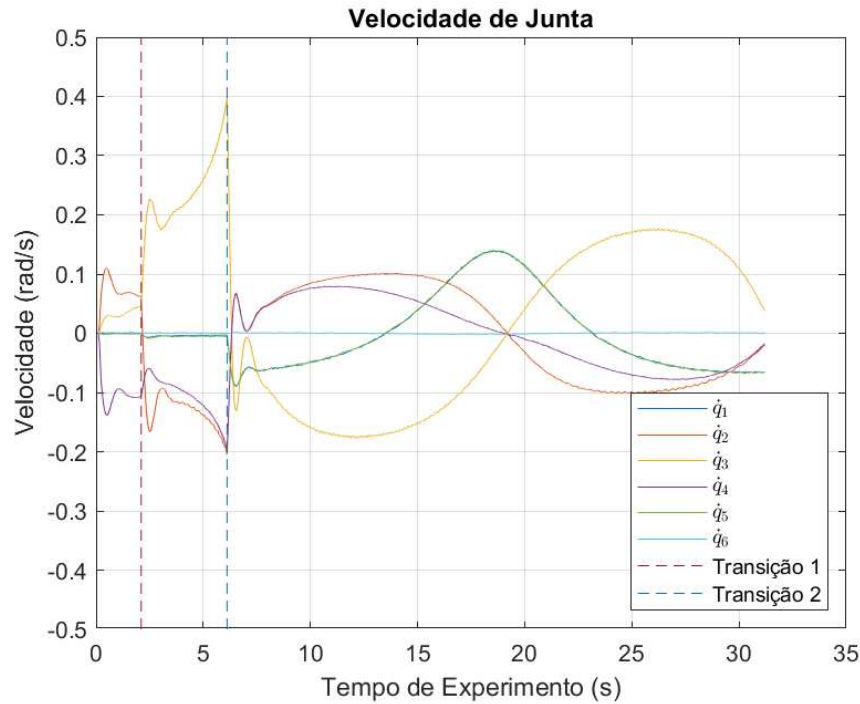
Fonte: Próprio Autor.

Figura 5.25 – Posição de Juntas do controle cinemático com Controle Proporcional e Integral - Experimento.



Fonte: Próprio Autor.

Figura 5.26 – Velocidade de Juntas do controle cinemático com Controle Proporcional Integral - Experimento.



Fonte: Próprio Autor.

5.6.3 Controle Proporcional e Feedforward

Considere a seguinte adição a equação 5.24

$$\dot{\mathbf{q}} = \mathbf{J}_a^*(\mathbf{q})(\dot{\mathbf{p}} + \mathbf{K}_p(\mathbf{p}_d - \mathbf{p})) \quad (5.33)$$

substituindo a equação 5.33 na equação 5.23 resulta em

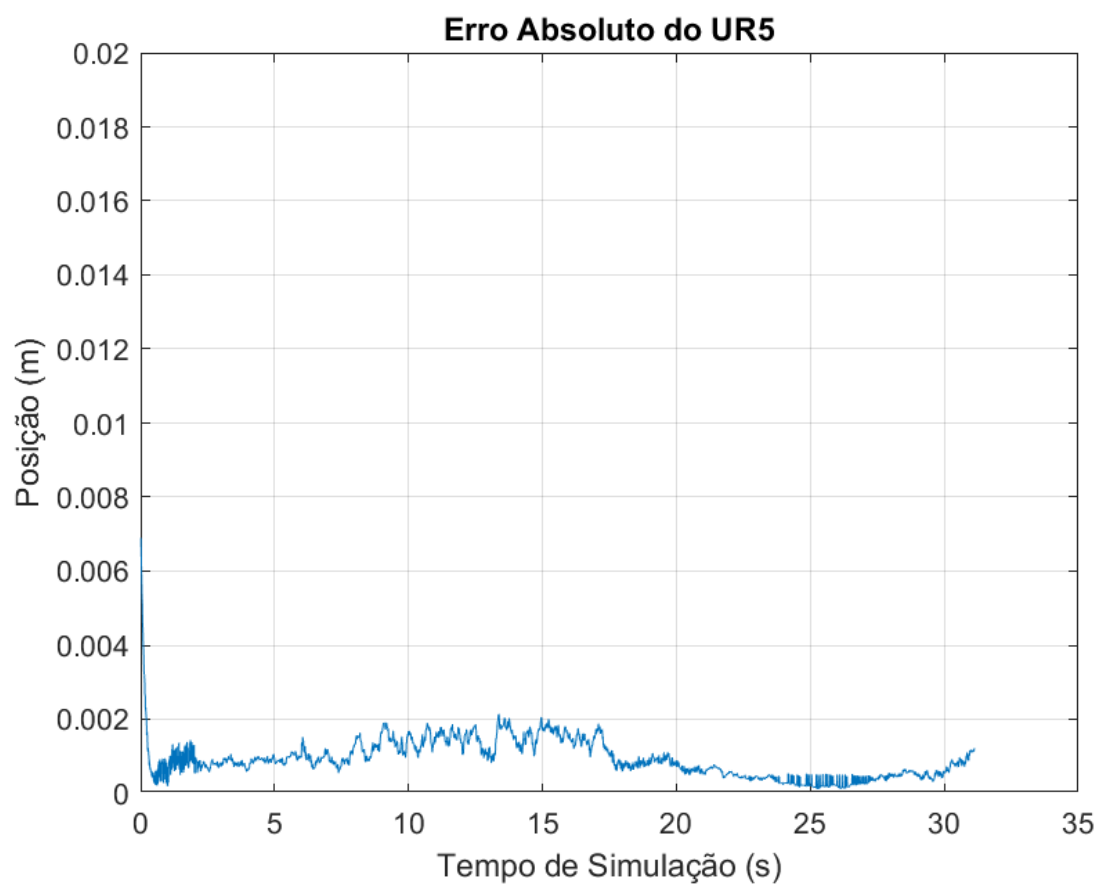
$$(\dot{\mathbf{p}}_d - \dot{\mathbf{p}}) + \mathbf{K}(\mathbf{p}_d - \mathbf{p}) = 0 \quad (5.34)$$

O sistema é assintoticamente estável, o erro ao longo da trajetória converge para zero a uma taxa dependente dos ganhos de \mathbf{K} .

Para as simulações e experimentos a seguir, o ganho utilizado foi $\mathbf{K}_p = 5\mathbf{I}$.

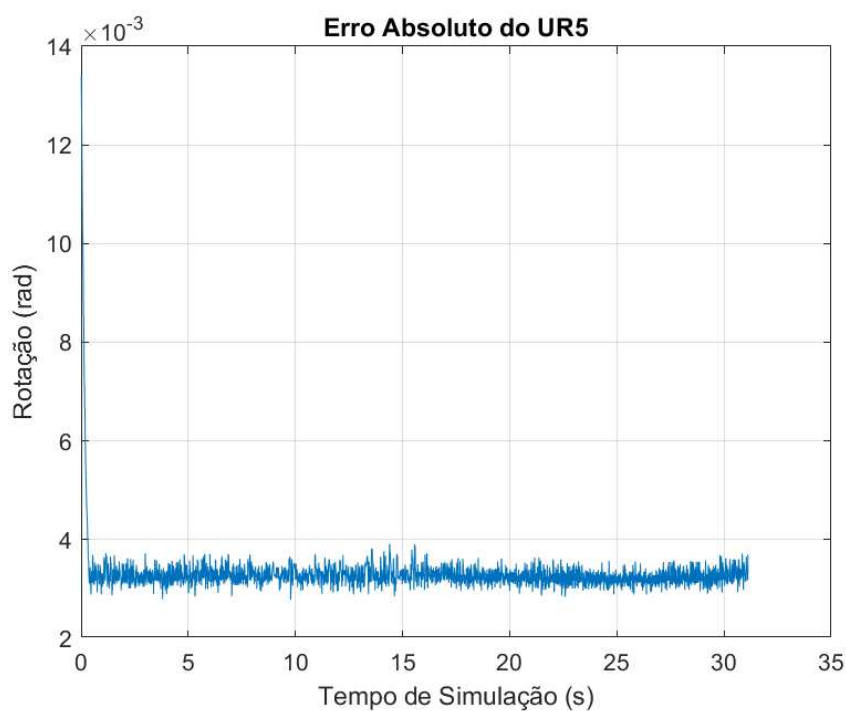
5.6.3.1 Simulação do Controle Proporcional Feedforward

Figura 5.27 – Erro Absoluto de Posição do controle cinemático com Controle Proporcional Feedforward.



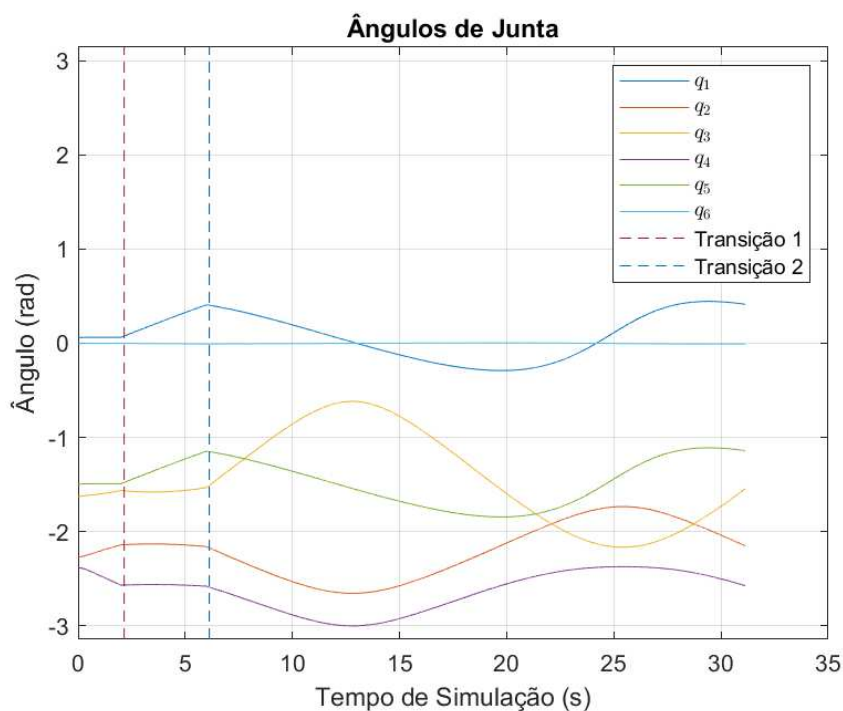
Fonte: Próprio Autor.

Figura 5.28 – Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Feedforward.



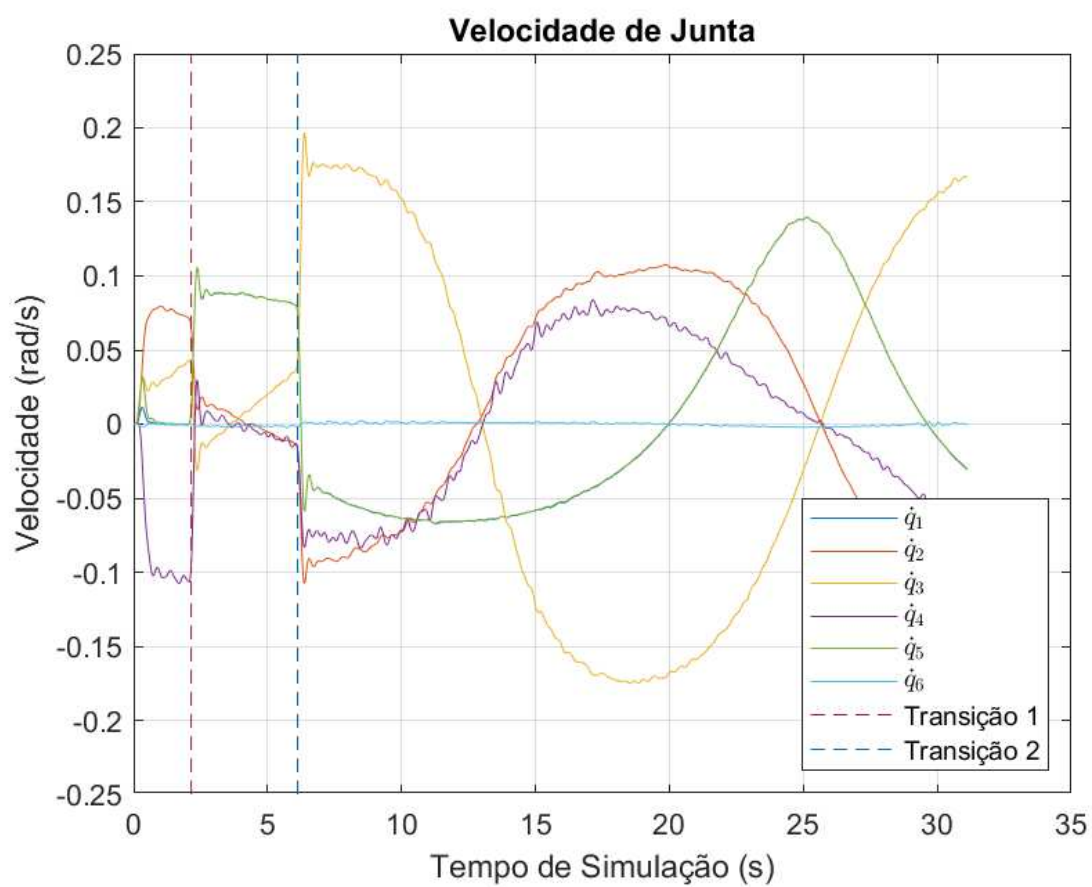
Fonte: Próprio Autor.

Figura 5.29 – Posição de Juntas do controle cinemático com Controle Proporcional e Feedforward.



Fonte: Próprio Autor.

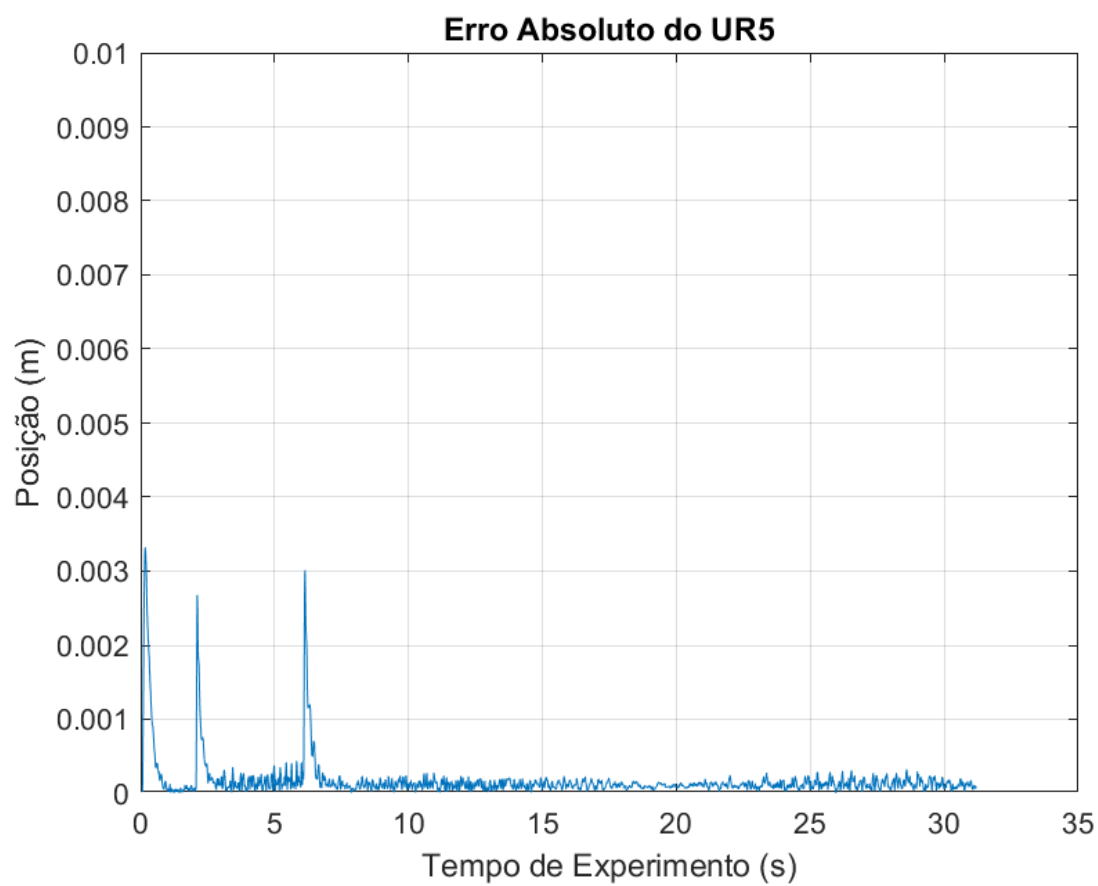
Figura 5.30 – Velocidade de Juntas do controle cinemático com Controle Proporcional Feedforward.



Fonte: Próprio Autor.

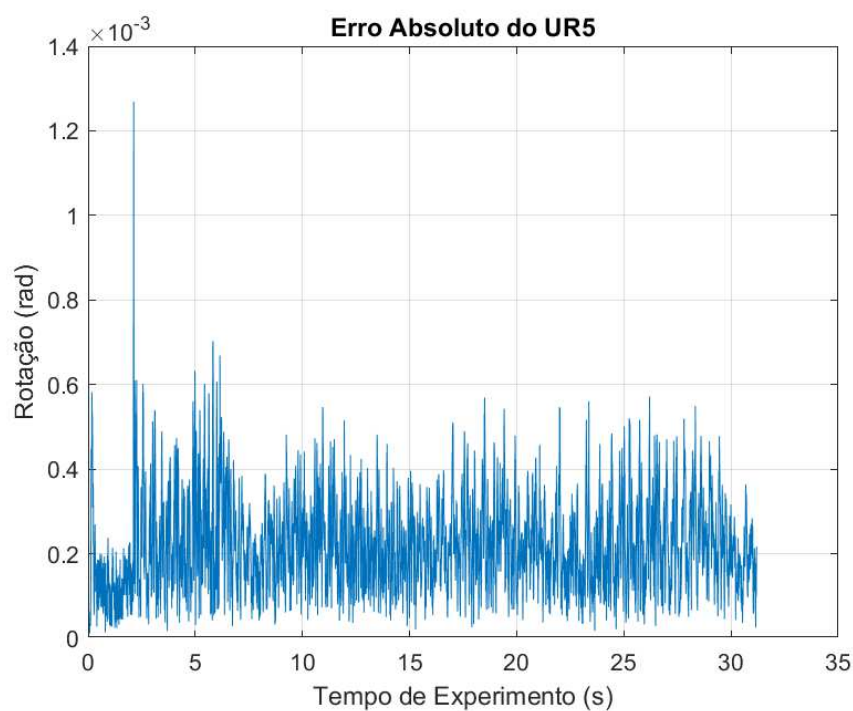
5.6.3.2 Experimento com Controle Proporcional Feedforward

Figura 5.31 – Erro Absoluto de Posição do controle cinemático com Controle Proporcional e Feedforward - Experimento.



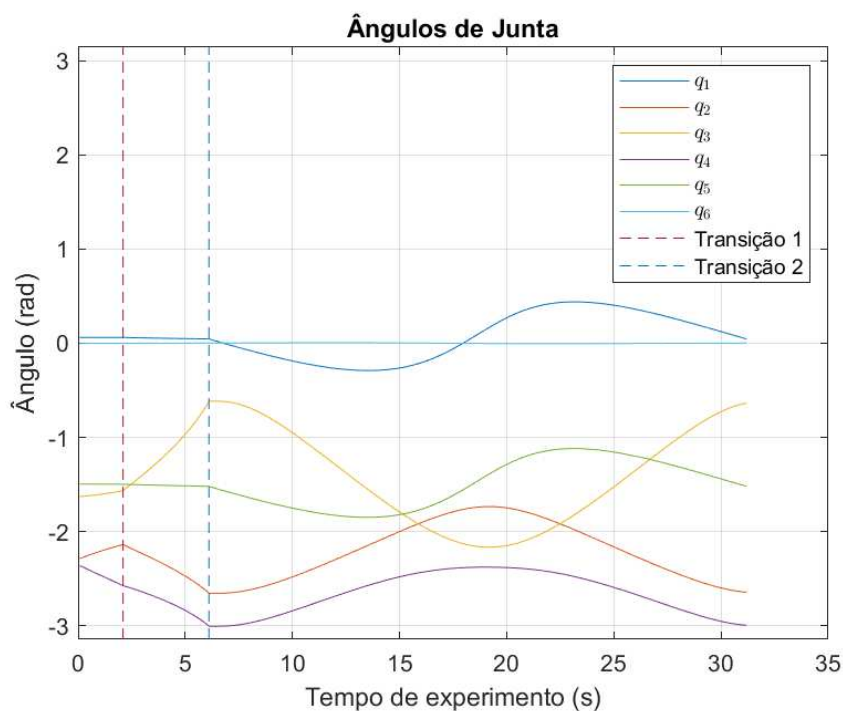
Fonte: Próprio Autor.

Figura 5.32 – Erro Absoluto de Rotação do controle cinemático com Controle Proporcional e Feedforward- Experimento.



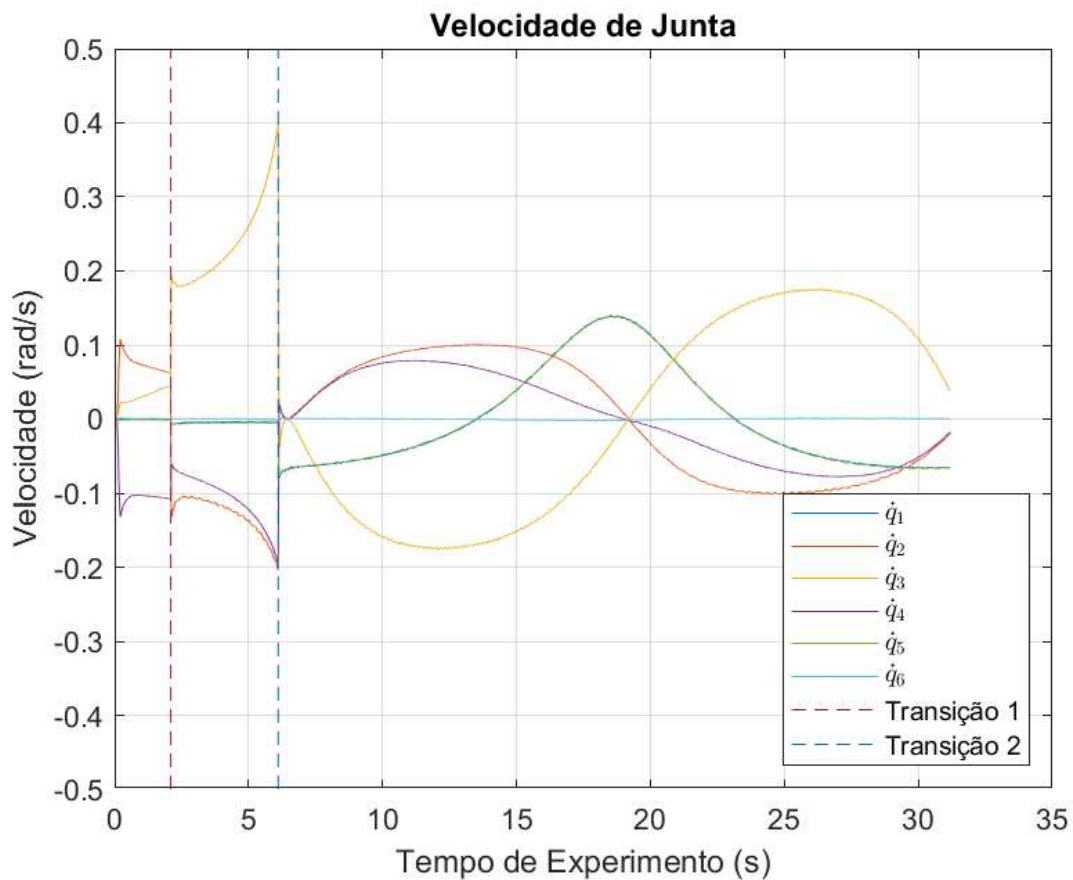
Fonte: Próprio Autor.

Figura 5.33 – Posição de Juntas do controle cinemático com Controle Proporcional e Feedforward - Experimento.



Fonte: Próprio Autor.

Figura 5.34 – Velocidade de Juntas do controle cinemático com Controle Proporcional Feedforward - Experimento.



Fonte: Próprio Autor.

5.7 Análise dos resultados do controle cinemático

Os resultados dos experimentos condizem com os resultados esperados pelas simulações e controles desenvolvidos.

Os gráficos apresentados nas seções anteriores evidenciam o erro estacionário constante de um centímetro em um controle proporcional (Figuras 5.11 e 5.15), enquanto mostram que com um controle proporcional e integral ou um proporcional com feedback ele tende a zero (Figuras 5.19, 5.23, 5.27 e 5.31).

Apesar disso, o controle PI mostra uma resistência maior a mudanças de trajetória e uma oscilação nas inflexões da mesma (Figura 5.23). As transições onde ocorre uma inflexão na trajetória estão marcadas no texto com linhas tracejadas. No momento onde a mudança no vetor direção é constante, na circunferência da trajetória, esse erro causado pelo controle integral acaba resultando em um erro constante de posição que faz com que o erro estacionário do sistema não atinja zero. Isso pode ser ajustado com o ganho integral

ou com estratégias para evitar saturação, mas um equilíbrio entre desempenho geral e desempenho nos pontos de mudança de direção tem que ser encontrado.

Ainda mais, a trajetória apresenta dois pontos de inflexão onde o vetor direção muda bruscamente e a derivada de aceleração cresce consideravelmente. Tanto em simulação quanto nos experimentos, as limitações dos controles dos motores fazem com que esses picos de erro ocorram. Essas mudanças bruscas podem ser observadas nos gráficos de velocidades das juntas ao redor dos dois e seis segundos (linhas tracejadas nas Figuras 5.18, 5.26, 5.34), respectivamente. Como a velocidade escolhida para a trajetória foi de cinco centímetros por segundo, isso corresponde exatamente aos momentos de inflexão de acordo com a fórmula desenvolvida para a trajetória.

Graças a essas inflexões, o controlador PI apresenta picos muito maiores do que o controle feedforward (oito e três milímetros, respectivamente)(Figura 5.26) devido a sua componente integral. O controle feedforward se mostrou o mais eficiente a levar o erro estacionário para zero e o menos afetado pelas inflexões na trajetória. No entanto, o uso da trajetória como mecanismo direto de controle resulta em um impacto muito menor do controle proporcional geral, e para estratégias de controle onde a trajetória real do robô tem que ser rapidamente modificada para evitar um obstáculo isso pode resultar em uma dificuldade a mais no controle. Apesar disso, o controle feedforward foi utilizado para a próxima etapa dos experimentos.

Outro ponto a ser observado é o comportamento das velocidades entre os dois e seis segundos nos experimentos. Especificamente nesses experimentos, a direção para o raio inicial definida na segunda parte da trajetória é em direção a base do primeiro manipulador (consequentemente é a direção oposta no ponto de vista do segundo manipulador), e a proximidade do manipulador com a base, ou seja, a proximidade com uma singularidade, faz as velocidades das juntas crescerem em magnitude (Figuras 5.18, 5.26, 5.34). No limite de uma singularidade, a magnitude da velocidade das juntas tende a infinito se não for controlada, resultando em um sistema instável.

De modo a ilustrar o controle, pseudo códigos para o controle podem ser encontrados nos Algoritmos 4 e 5. Aqui, a função *computeError* é a mesma do Algoritmo 3, enquanto a função *getInverseJacobian* tem a mesma função que nos algoritmos anteriores, de calcular a inversa do jacobiano de acordo com o método escolhido. No caso dos testes de controle, foi escolhido a inversa amortecida. Por fim, a função *ControlStrategy* implementa a equação de controle selecionada (sem a aplicação da inversão do jacobiano) de acordo com o tipo de controle escolhido, para depois, com a multiplicação da inversa do jacobiano, a velocidade ser finalmente calculada.

Um diagrama temporal pode ser visto na Figura 5.35.

Algoritmo 4: Laço de Controle - *controlLoop()*

```

Data:  $\mathbf{p}_d, \dot{\mathbf{p}}_d, \mathbf{p}_e, \mathbf{q}$ 
Result:  $\dot{\mathbf{q}}$ 
/* Inicialização - Definição dos Ganhos */
 $\mathbf{K}_p = K_p \mathbf{I};$ 
 $\mathbf{K}_i = K_i \mathbf{I};$  /* Apenas para controle PI */
/* Cálculo do Erro, como no Algoritmo 3 */
 $\mathbf{e} = \text{computeError}(\mathbf{p}_d, \mathbf{p}_e);$ 
 $\mathbf{e}_i += \mathbf{e};$  /* Apenas para controle PI */
/* Controle */
 $\mathbf{J}_{\text{inv}} = \text{getInverseJacobian}(\mathbf{p}_e, \mathbf{q});$  /* Inversa Amortecida */
ErroControl = controlStrategy(type = StrategyType,  $\mathbf{K}_p, \mathbf{K}_i, \mathbf{e}, \mathbf{e}_i$ );
 $\dot{\mathbf{q}} = \mathbf{J}_{\text{inv}} \text{ErroControl};$ 
return  $\dot{\mathbf{q}}$ ;

```

Algoritmo 5: Controle Cinemático

```

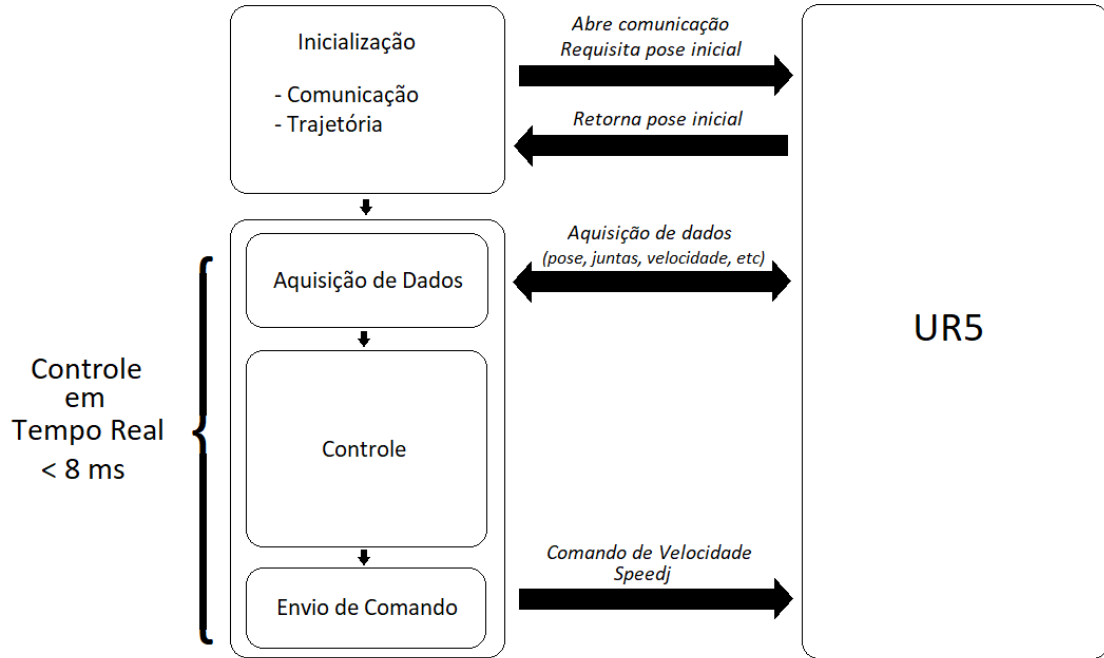
Data:  $\mathbf{p}_0$ , VariaveisTrajetoria, Robo, PassoDeControle
/* Inicialização - Geração da Trajetória */
(Trajectoria, TempoMax) = generateTrajectory( $\mathbf{p}_0$ , VariaveisTrajetoria,
PassoDeControle);
TrajetoriaVelocidade = getTrajectorySpeed(Trajectoria, PassoDeControle); /* Apenas
Controle FeedForward */
Tempo = 0;
/* Inicialização - Loop de Controle */
while Tempo < TempoMax do
     $\mathbf{p} = \text{getCartesianPose}(\text{Robo});$ 
     $\mathbf{q} = \text{getJointVector}(\text{Robo});$ 
     $\dot{\mathbf{q}} = \text{controlLoop}(\text{Trajectoria}(\text{Tempo}), \text{TrajectoriaVelocidade}(\text{Tempo}), \mathbf{p}, \mathbf{q});$ 
    enviarVelocidadeAoRobo(Robo,  $\dot{\mathbf{q}}$ );
    Tempo += PassoDeControle;
end

```

5.8 Controle cinemático de múltiplos manipuladores

Em seguida, o controle cinemático foi aplicado a múltiplos manipuladores segurando uma caixa. A preparação experimental pode ser visto na Figura 5.36. Dois experimentos distintos foram realizados: No primeiro, a caixa foi levantada de um altura mais baixa por 60 centímetros, e o foco do experimento foi na sincronização dos manipuladores. A prensão robótica foi realizada previamente ao início do experimento. No segundo experimento, a caixa foi levantada de uma altura inicial maior por 20 centímetros e a trajetória do objeto foi rastreada utilizando visão computacional.

Figura 5.35 – Diagrama Temporal. A comunicação inicial, aquisição de posição inicial e geração da trajetória é feito antes do controle. O laço de controle é realizado em menos de 8 milissegundos.



Fonte: Próprio Autor.

5.8.1 Poses iniciais

As poses iniciais dos manipuladores e a estimação da pose inicial do objeto foi feita como introduzido nas seções 4.8 e 4.10, onde se assumiu que o objeto é rígido e a preensão dos manipuladores é fixa e não acarreta em movimentação do objeto. Dado a pose inicial do primeiro manipulador como ${}^B\mathbf{T}_{EE}$, a pose do objeto é encontrada como

$${}^B\mathbf{T}_O = {}^B\mathbf{T}_{EE} \mathbf{r}_1, \tag{5.35}$$

onde \mathbf{r}_1 representa a transformação fixa do bastão virtual que transforma a pose do contato ao centro de gravidade.

O módulo do vetor posição da transformação do bastão virtual \mathbf{r}_1 é calculado como a metade do tamanho da lateral do objeto, ou seja, seja \mathbf{p}_{r1} o vetor de posição dessa transformação. Então

$$\|\mathbf{p}_{r1}\| = \frac{l}{2}, \tag{5.36}$$

e sua direção é dada pela normal ao contato do manipulador com o objeto. Como o contato com o efetuador do UR5 representa o vetor direção z , o vetor posição da

Figura 5.36 – Configuração de experimento com múltiplos manipuladores.



Fonte: Próprio Autor.

transformação é dado por

$$\mathbf{P}_{\mathbf{r}1} = \begin{bmatrix} 0 \\ 0 \\ \frac{l}{2} \end{bmatrix}. \quad (5.37)$$

A rotação da transformação do bastão virtual \mathbf{r}_1 foi tratada de forma direta com base no conhecimento de quais faces do objeto estão em contato com o manipulador. Se no contato com o manipulador o vetor z aponta para o centro de massa do objeto, e é assumido que o vetor z do centro de massa do objeto aponta parte de cima do objeto, uma rotação de 90 graus em torno do eixo y da transformação inicial fixou a rotação do objeto apontando para cima. Como o objeto é um paralelepípedo, não houve preocupação em terminar se o objeto estava de frente ou de trás com relação ao kinect, uma vez que virtualmente é o mesmo modelo. Assim, seja $\mathbf{R}_{\mathbf{r}1}$ a matriz de rotação de \mathbf{r}_1 , ela é dada por

$$\mathbf{R}_{\mathbf{r}1} = \mathbf{R}_z(90^\circ). \quad (5.38)$$

Uma vez calculada a pose do objeto, sua trajetória pode ser calculada juntamente da trajetória dos dois manipuladores.

5.8.2 Sincronia

No que se refere a sincronia, o controle de múltiplos manipuladores pela porta ethernet infere em um acúmulo de atraso no sistema de controle, não comportado pela porta de tempo real 30003. O protocolo RTDE foi desenvolvido e utilizado no experimento, mas pela natureza da função em URscript *speedj*, um tempo mínimo de 10 ciclos de tempo real (80 milissegundos) foi utilizado no controlador interno, o que introduz um erro no controle sempre que há uma mudança na direção da trajetória. Esse erro está enfatizado nas Figuras 5.37 e 5.38, e a trajetória pode ser visualizada na Figura 5.39.

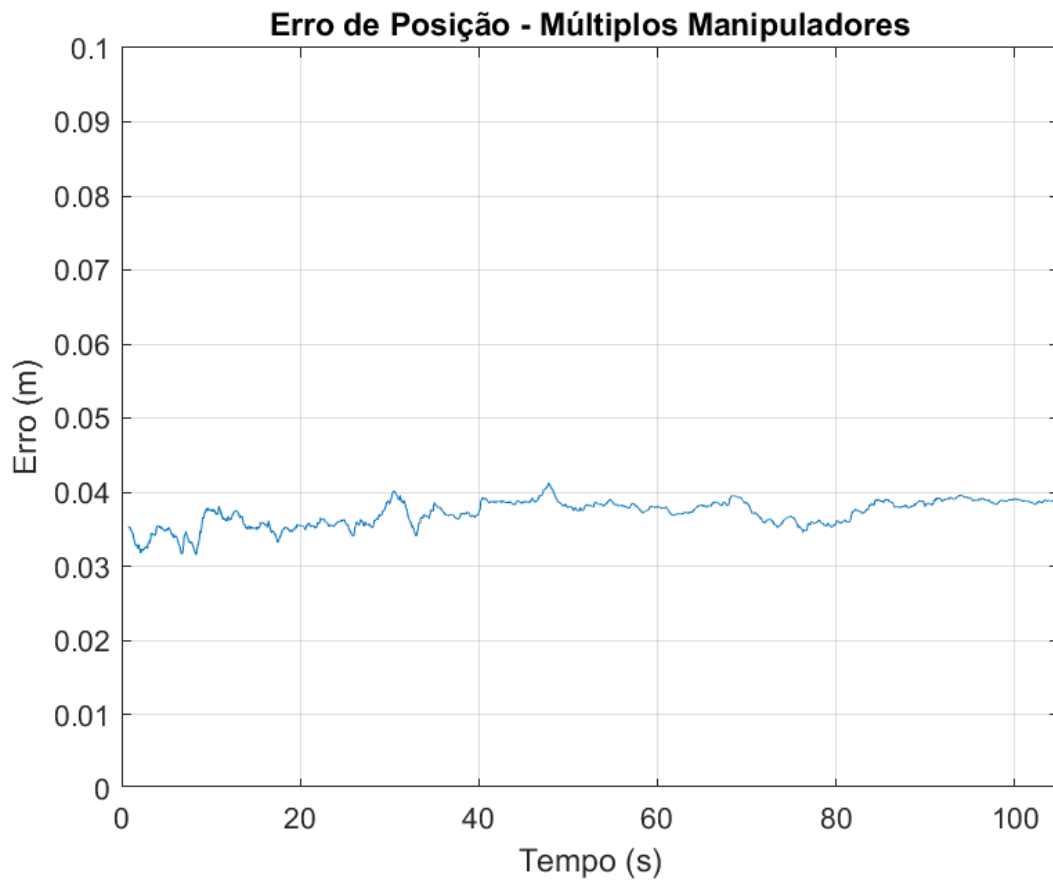
5.9 Rastreo de objeto com visão computacional

O último experimento realizado foi o de controle cinemático de múltiplos manipuladores com o rastreo do objeto utilizando visão computacional. Inicialmente a posição inicial do objeto foi adquirida ao se realizar uma comparação com um modelo 3D, utilizando PCL e ROS, e em seguida o objeto foi rastreado durante o controle cinemático.

5.9.1 Localização inicial

Inicialmente uma única nuvem de pontos da cena atual é utilizada para localizar o objeto. Primeiro a cena é registrada e a cena e a nuvem de pontos do modelo do objeto

Figura 5.37 – Erro de posição no controle de múltiplos manipuladores.

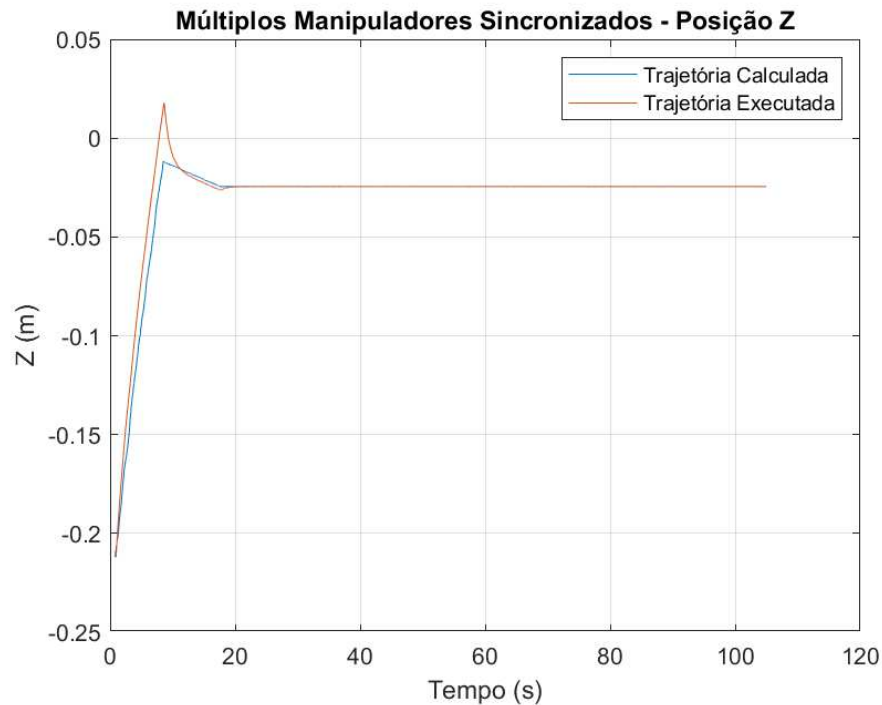


Fonte: Próprio Autor.

são carregados na memória. Em seguida uma série de etapas se segue na nuvem de pontos da cena para encontrar a posição inicial do objeto na cena

- **Filtro passa faixa em X, Y e Z:** Um filtro passa faixa é aplicado na nuvem de pontos da cena para isolar ao máximo a nuvem de pontos que representa o objeto com a nuvem de pontos do modelo. O faixa onde o filtro é aplicado foi determinada previamente de acordo com a cena. Esse filtro é aplicado três vezes, uma para cada eixo.
- **Subamostragem da nuvem de pontos da cena e do modelo:** O processo de localização é computacionalmente custoso, então uma sub amostragem é realizada na nuvem de pontos restante da cena e na nuvem de pontos do modelo. Essa sub amostragem afeta diretamente a performance da localização inicial.
- **Geração das normais da cena e do modelo:** Com as nuvens restantes, um algoritmo disponível no PCL é utilizado para computar as normais dos pontos na nuvem de pontos.

Figura 5.38 – Trajetória em Z percorrida por um dos dois manipuladores sincronizados.

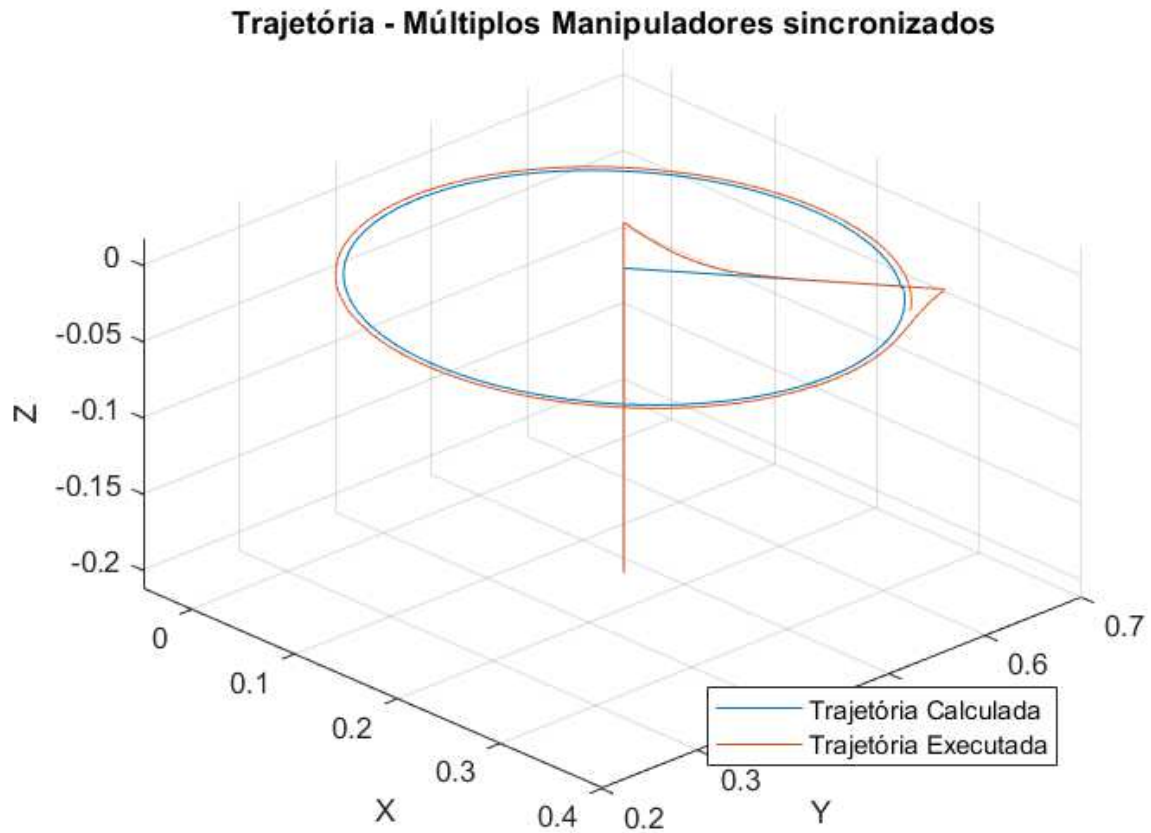


Fonte: Próprio Autor.

- **Segmentação de planos e objetos pequenos:** Com as normais computadas, pequenos objetos e planos que não representam o objeto podem ser segmentados para isolar ainda mais a nuvem de pontos. Dos grupos restantes de pontos, o que representa a nuvem de pontos do objeto é escolhido manualmente e os outros descartados.
- **Geração do histograma rápido de características de pontos:** Com as normais e o grupo correto de pontos, o histograma de características é gerado.
- **Alinhamento inicial por consenso de amostra:** Com o histograma de características, um laço é utilizado para o alinhamento inicial, retornando a transformação estimada inicial.
- **Melhoria no alinhamento com ICP:** O alinhamento inicial é melhorado com um algoritmo de ICP.

Uma vez que a transformação da câmera para o objeto é computada, a transformação é utilizada no algoritmo de filtro de partículas para rastrear o objeto. A frequência de atualização da pose pelo filtro de partículas foi calculada como aproximadamente 0.08 segundos, dez vezes mais lenta que a atualização do controlador dos manipuladores. Esse tempo não é fixo, no entanto, sendo uma média da quantidade de transformações salvas durante o período do experimento.

Figura 5.39 – Trajetória percorrida por um dos manipuladores sincronizados.



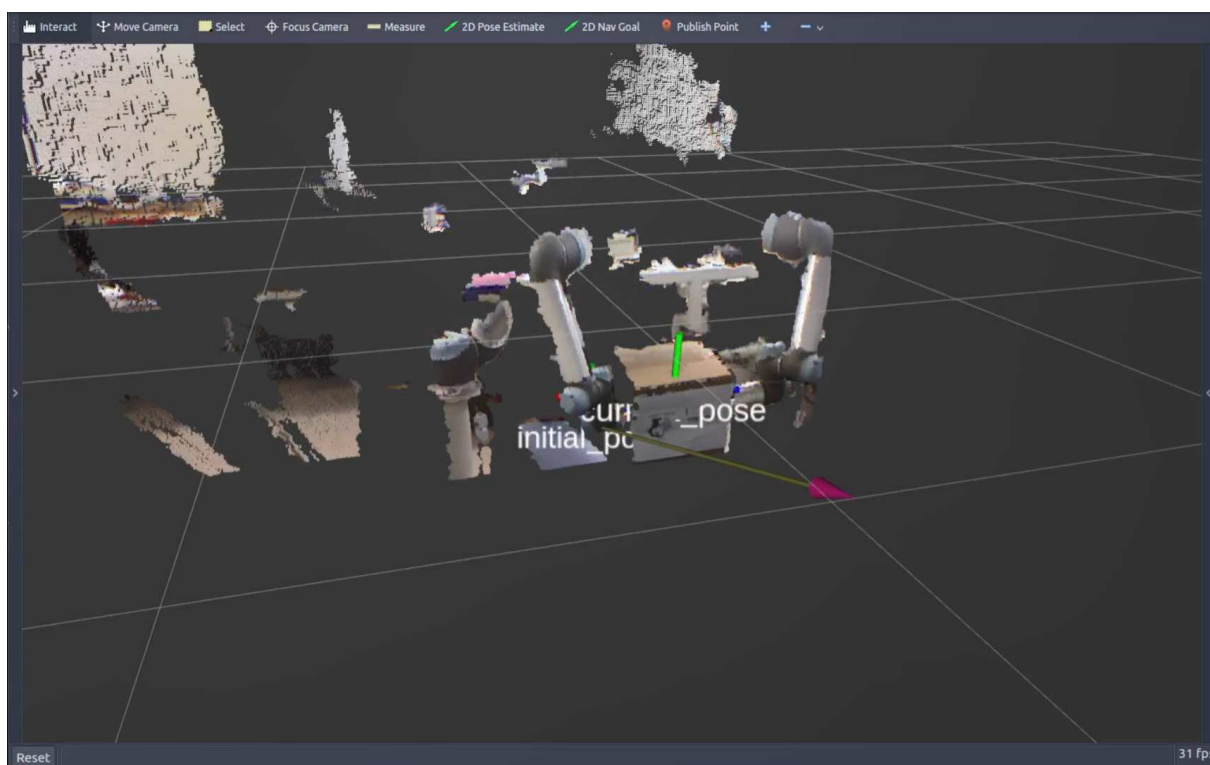
5.9.2 Resultados

A precisão do sistema de rastreamento se mostrou insatisfatória mesmo ao se levar em conta que a precisão do sistema de visão era muito menor, de cerca de cinco centímetros, do que o dos manipuladores.

A escolha do objeto manipulado, um paralelepípedo, afetou a performance dos algoritmos de ICP e Filtro de Partículas devido a simetria associada às faces do mesmo. O rastreamento da rotação do objeto era facilmente perdido se o mesmo rotacionasse, e um objeto com características marcantes e sem simetria é o ideal para utilização dos algoritmos.

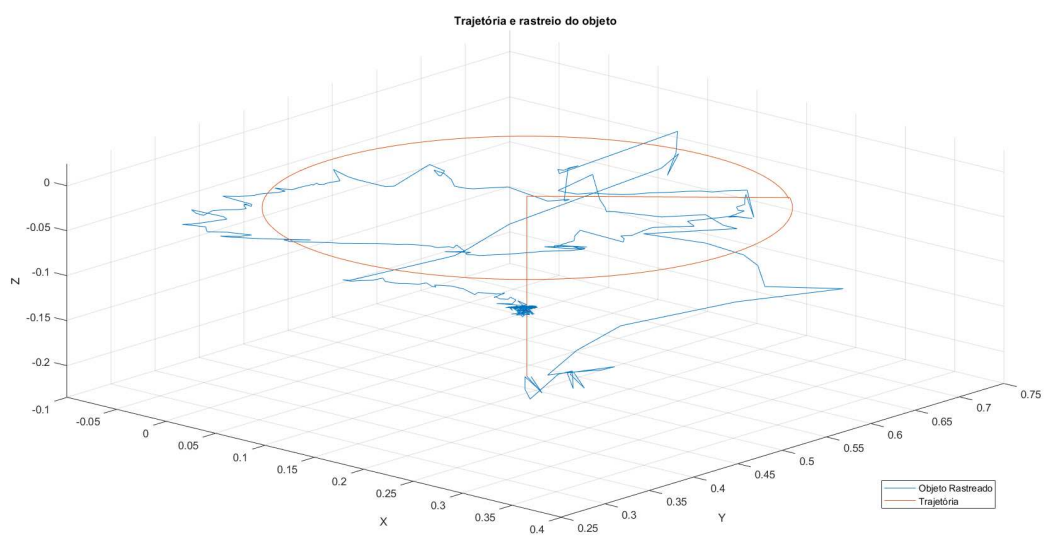
Uma amostra do sistema visualizado no ROS pode ser vista na Figura 5.40, e um exemplo do baixo desempenho em rastrear o objeto pode ser visto nas Figuras 5.41, 5.42 e 5.43.

Figura 5.40 – Rastreo de objeto utilizando visão e múltiplos manipuladores.



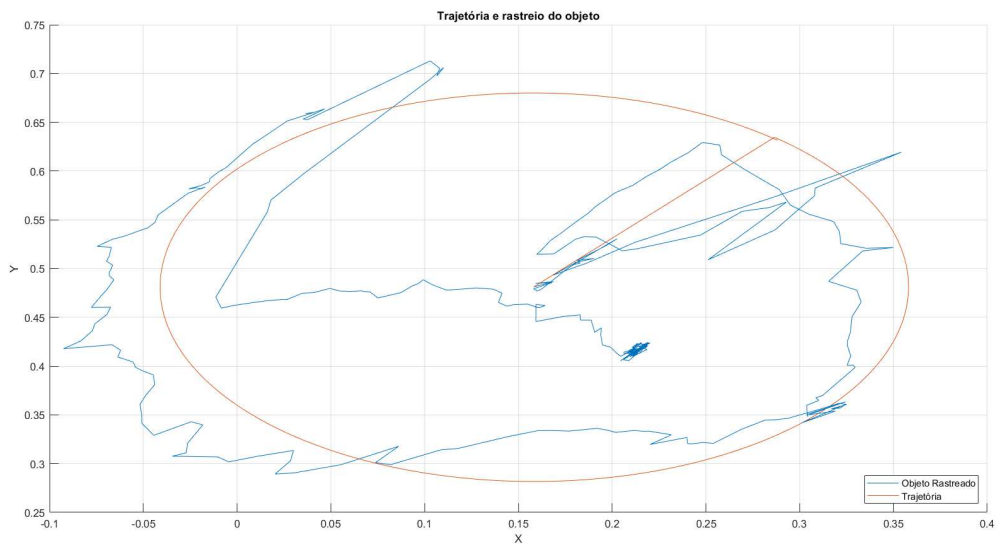
Fonte: Próprio Autor.

Figura 5.41 – Rastreo de objeto - Trajetória XYZ.



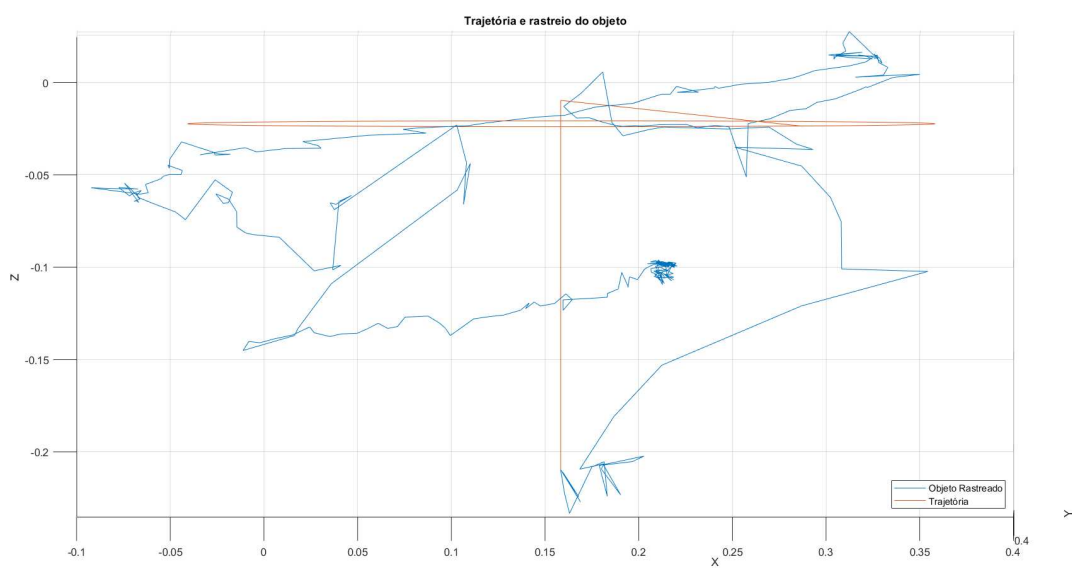
Fonte: Próprio Autor.

Figura 5.42 – Rastreo de objeto - Trajetória XY.



Fonte: Próprio Autor.

Figura 5.43 – Rastreo de objeto - Trajetória XZ.



Fonte: Próprio Autor.

6 Conclusão

O problema de manipulação de objetos com múltiplos manipuladores usando visão foi abordado, onde o problema geral foi apresentado e em seguida uma modelagem do problema foi feita para o desenvolvimento de uma solução. Uma solução foi apresentada para o problema no qual a pose inicial do objeto é estimada utilizando visão computacional e dois manipuladores movem o objeto em uma trajetória pré definida, com o rastreamento feito pelo sistema de visão.

Algoritmos de cinemática direta e inversa, e cinemática direta e inversa diferenciais foram desenvolvidos para o controle de dois manipuladores UR5 em uma trajetória pré definida. No que diz respeito a cinemática inversa, foi explorado os efeitos do uso de diferentes métodos de inversão de Jacobiano e a importância desses métodos.

Três sistemas de controle foram desenvolvidos em simulação e em experimentos para resolver o problema da cinemática inversa diferencial para rastreamento de trajetória, sendo eles um método de *Controle Proporcional*, *Controle Proporcional Integral* e *Controle Proporcional Feedforward*, enfatizando as vantagens e desvantagens de cada um deles. Os controles desenvolvidos foram testados em simulação e em experimentação utilizando dois manipuladores UR5.

As vantagens e desvantagens do uso de um controle PI ou um controle Feedforward foram analisadas, ressaltando o problema da saturação no controle PI, onde ele responde lentamente a mudanças no vetor direção da trajetória, enquanto o controle Feedforward apresenta o problema de boa parte do controle vir da trajetória a ser seguida, o que torna o problema computacionalmente mais complexo para trajetórias que sejam criadas em tempo real. Ainda sobre os experimentos, as inflexões causadas pela trajetória escolhida mostraram os limites de cada controlador e os limites dos próprios controladores de baixo nível encontrados nos manipuladores.

Enquanto o controlador P mostrou um erro estacionário de um centímetro, nos controladores PI e Feedforward o erro estacionário tende a zero. A saturação do integrador resultou em erros de até oito milímetros nas inflexões da trajetória, comparado a um erro máximo de três milímetros do controlador Feedforward.

Dois manipuladores foram sincronizados, e as dificuldades na sincronização devido ao sistema de comunicação foi abordada. A solução atualmente abordada de utilizar a função *speedj* em thread interna ao robô em uma conexão RTDE resultou em atrasos de até 72 milissegundos para as mudanças de direção na trajetória, resultando em um erro de estado estacionário de aproximadamente 3.5 centímetros. Utilizando as técnicas desenvolvidas para um manipulador, os dois manipuladores movimentaram o objeto na

trajetória definida mesmo com o erro adicional do sistema de comunicação.

O objeto foi rastreado durante a trajetória utilizando visão computacional, porém uma combinação de baixa resolução do sensor e baixo desempenho do algoritmo resultou em uma taxa de erro de ordens de magnitude maior do que a precisão dos manipuladores. Além da taxa de aquisição dos dados ser menor que a dos manipuladores, necessitando uma interpolação de dados com relação aos dados dos manipuladores.

Para trabalhos futuros, um controle de força na prensão aplicada ao objeto pode ser projetado juntamente do controle de posição, permitindo um rastreamento de trajetória com força controlada no objeto, evitando deformações e estresse no objeto e robô. Métodos de se realizar isso incluem controle de impedância e admitância, podendo ser feito a nível cinemático ou dinâmico. Para controle de impedância, existe a possibilidade de se controlar a força sem a utilização de um sensor de força em malha fechada para realimentação da malha.

A sincronização dos manipuladores pode ser melhorada com a introdução da malha de controle agindo diretamente na caixa controladora do robô, evitando assim problemas de comunicação. Alternativas para utilização do *speedj* podem ser estudadas, como a utilização da função *servoj* para controle direto de posicionamento ou uma forma de contornar o problema de tempo do *speedj*.

Formas de melhorar o rastreamento do objeto incluem a utilização de um sistema de visão com maior resolução, múltiplas câmeras para uma visualização completa do objeto, a utilização de filtros ativos para melhorar a qualidade da localização e treinamento em aprendizado de máquina para uma caracterização mais precisa do objeto.

Outros trabalhos futuros podem incluir a caracterização de objetos e análise de síntese utilizando algoritmos de aprendizado profundo para identificação automática do objeto e de seus candidatos a prensão.

Métodos automáticos de calibração olho mão podem ser utilizados ou desenvolvidos para melhorar a precisão do posicionamento do manipulador com relação ao sistema de visão. Um desses métodos inclui usar um modelo do efetuador, rastrear esse modelo com nuvem de pontos e utilizar mínimos quadrados para encontrar a posição precisa do manipulador com relação ao sistema de visão.

Com o problema da sincronização em tempo real resolvido e uma boa precisão no rastreamento do objeto, a trajetória pré definida pode ser modificada em tempo real para evitar colisões com base em um controle em malha fechada com a posição do objeto e dos manipuladores.

Um dos manipuladores pode ser substituído por um braço humano e o outro manipulador funcionar em um sistema mestre-escravo guiando sua trajetória e o controle de força aplicado ao objeto em função do rastreamento do objeto e das forças aplicadas pelo

braço humano.

Por fim, o trabalho cumpriu com seu propósito de controle de múltiplos manipuladores para manipulação de um objeto e o rastreamento desse objeto utilizando visão computacional. O trabalho representa um passo na solução do problema geral de manipulação de objetos desconhecidos e cooperação homem máquina, que se demonstra um problema complexo e multidisciplinar da engenharia robótica atual.

Referências

- AHMED, S. M.; PECHEV, A. N. Performance analysis of fik and dls inverse kinematics using six degree of freedom manipulator. *IEEE International Conference on Robotics and Biomimetics*, 2009. Citado na página 19.
- ALYAHMADI, A. S.; HSIA, T. C. Internal force-based impedance control of dual-arm manipulation of flexible objects. *International Conference on Robotics and Automation*, 2000. Citado na página 21.
- ARAUJO, A. C. *Sistema de rastreamento de um manipulador robótico e detecção do estado de seu efetuador final*. 2017. Monografia (Bacharel em Engenharia Elétrica), UFCG (Universidade Federal de Campina Grande), Campina Grande, Brasil. Citado 7 vezes nas páginas 46, 52, 54, 56, 57, 58 e 65.
- ARAUJO, A. C.; LIMA, A. M. N.; MUTHUSAMY, R.; MATTILA, J. Model-based robotic hand tracking and gripper state determination. *14rd Latin American Robotics Symposium*, 2017. Citado 3 vezes nas páginas 22, 67 e 68.
- BENALI, K.; BRETHÉ, J.-F.; GUÉRIN, F.; GORKA, M. Dual arm robot manipulator for grasping boxes of different dimensions in a logistics warehouse. *IEEE International Conference on Industrial Technology (ICIT)*, 2018. Citado 2 vezes nas páginas 17 e 21.
- BJERKENG, M.; SCHRIMPF, J.; MYHRE, T.; PETTERSEN, K. Y. Fast dual-arm manipulation using variable admittance control: Implementation and experimental results. *IEEE International Conference on Intelligent Robots and Systems*, 2014. Citado na página 21.
- BURRELL, T.; MONTAZERI, A.; MONK, S.; TAYLOR, C. J. Feedback control-based inverse kinematics solvers for a nuclear decommissioning robot. *International Federation of Automatic Control*, 2016. Citado na página 19.
- CHEN, B.-H.; WANG, Y.-H.; LIN, P.-C. A hybrid control strategy for dual-arm object manipulation using fused force/position errors and iterative learning. *IEEE International Conference on Advanced Intelligent Mechatronics*, p. 39–44, 2018. Citado na página 21.
- CHOU, C.-K.; YANG, W.-T.; LIN, P.-C. Dual-arm object manipulation by a hybrid controller with kalman-filter-based inputs fusion. *International Automatic Control Conference*, 2014. Citado na página 21.
- CLAUDIO, G.; SPINDLER, F.; CHAUMETTE, F. Vision-based manipulation with the humanoid robot romeo. *IEEE International Conference on Humanoid Robots*, 2016. Citado na página 22.
- COPPELIA Robotics Website. 2017. <<http://www.coppeliarobotics.com/>>. Acessado: 11 de Dezembro de 2017. Citado na página 62.
- CORKE, P. I. *Robotics, vision and control*. 2. ed. Berlin: Springer, 2016. Citado 19 vezes nas páginas 23, 24, 26, 29, 30, 33, 36, 37, 38, 43, 44, 46, 47, 48, 49, 50, 51, 52 e 79.

- COX, D. J.; RACKERS, K.; TESAR, D. Cooperative manipulation experiments using a dual-arm robot. *Annual Conference on IEEE Industrial Electronics*, 1995. Citado na página 20.
- DREXLER, D. A. Closed-loop inverse kinematics algorithm with implicit numerical integration. *International Symposium on Applied Machine Intelligence and Informatics*, 2017. Citado na página 19.
- FARIA, R. de O. *Hybrid Kinematic Control of Dual-Arm Cooperative Robots for Object Manipulation*. 2016. Dissertação de Mestrado (Engenharia Elétrica), UFRJ (Universidade Federal do Rio de Janeiro), Rio de Janeiro, Brasil. Citado na página 22.
- FIGUEREDO, L.; ADORNO, B.; ISHIHARA, J.; BORGES, G. Robust kinematic control of manipulator robots using dual quaternion representation. *IEEE International Conference on Robotics and Automation*, 2013. Citado na página 20.
- GALICKI, M. Control based solution to inverse kinematics for mobile manipulators using penalty functions. *Journal of Intelligent and Robotic Systems*, 2005. Citado na página 19.
- GOLDENBERG, A. A.; APKARIAN, J. A.; SMITH, H. W. A new approach to kinematic control of robot manipulators. *Journal of Dynamic Systems, Measurement, and Control*, v. 109, 1987. Citado na página 19.
- HAMILTON, W. R. On quaternions, or on a new system of imaginaries in algebra. *Philosophical Magazine*, 2000. Citado 2 vezes nas páginas 23 e 31.
- HECK, D.; KOSTIC, D.; DENASI, A.; NIJMEIJER, H. Internal and external force-based impedance control for cooperative manipulation. *European Control Conference*, 2013. Citado na página 21.
- IFR. *31 million robots helping in households worldwide by 2019*. 2016. <<https://ifr.org/ifr-press-releases/news/31-million-robots-helping-in-households-worldwide-by-2019>>. Acessado: 03 de Julho de 2019. Citado na página 15.
- KASERER, D.; GATTRINGER, H.; MÜLLER, A. Spatial impedance control of two cooperative industrial manipulators. In: *Proceedings in Applied Mathematics and Mechanics*. Weinheim: WILEY VCH Verlag GmbH Co., 2016. v. 16. Citado na página 21.
- KEBRIA, P. M.; AL-WAIS, S.; ABDI, H.; NAHAVANDI, S. Kinematic and dynamic modelling of ur5 manipulator. *IEEE International Conference on Systems, Man, and Cybernetics*, 2016. Citado na página 71.
- KINECT. 2017. <<https://developer.microsoft.com/pt-br/windows/kinect/hardware>>. Acessado: 11 de Dezembro de 2017. Citado na página 68.
- LEI, Q.; MEIJER, J.; WISSE, M. A survey of unknown object grasping and our fast grasping algorithm-c shape grasping. *3rd International Conference on Control, Automation and Robotics*, 2017. Citado na página 21.
- LEI, Q.; WISSE, M. Object grasping by combining caging and force closure. *14th International Conference on Control, Automation, Robotics and Vision*, 2016. Citado na página 21.

- LEON, B.; MORALES, A.; SANCHO-BRU, J. *From Robot To Human Grasping Simulation*. Suíça: Springer International Publisher, 2014. v. 1. Citado 2 vezes nas páginas 11 e 76.
- LYNCH, K. M.; PARK, F. C. *Modern Robotics: Mechanics, Planning, and Control*. 1. ed. New York: Cambridge University Press, 2017. Citado 2 vezes nas páginas 82 e 83.
- MASON, M. T. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 11, 1981. Citado na página 20.
- NAKAMURA, Y.; HANAFUSA, H. Inverse kinematic solutions with singularity robustness for manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, v. 108, 1986. Citado na página 19.
- NAKAMURA, Y.; HANAFUSA, H. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, v. 108, 1986. Citado na página 82.
- NAKANO, E.; OZAKI, S.; ISHIDA, T.; KATO, I. Cooperational control of the anthropomorphous manipulator "melarm". *Internacional Symposium on Industrial Robots*, v. 4, p. 251–260, 1974. Citado na página 20.
- PAUL, R. P.; SHIMANO, B.; MAYER, G. E. Differential kinematic control equations for simple manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 11, 1981. Citado na página 19.
- PECHEV, A. N. Inverse kinematics without matrix inversion. *IEEE International Conference on Robotics and Automation*, 2008. Citado na página 19.
- QU, J.; ZHANG, F.; FU, Y.; GUO, S. Multi-cameras visual servoing for dual-arm coordinated manipulation. *Robotica*, v. 35, p. 2218–2237, 2017. Citado na página 22.
- RASTEGARPANAH, A.; MARTURI, N.; STOLKIN, R. Autonomous vision-guided bi-manual grasping and manipulation. *IEEE Workshop on Advanced Robotics and its Social Impacts*, 2017. Citado na página 22.
- REN, Y.; CHEN, Z.; LIU, Y.; GU, Y.; JIN, M.; LIU, H. Adaptive hybrid position/force control of dual-arm cooperative manipulators with uncertain dynamics and closed-chain kinematics. *Journal of the Franklin Institute*, v. 354, 2017. Citado na página 21.
- SASAKI, J.; OTA, J.; MAEDA, Y.; AIYAMA, Y.; ARAI, T. Initial grasping strategy for an unknown object by cooperative mobile robots. *IEEE International Conference on Robotics and Automation*, 1995. Citado na página 21.
- SASAKI, J.; OTA, J.; NISHIDA, G.; YAMASHITA, A.; AIYAMA, Y.; ARAI, T. Estimating the center of gravity of an object using tilting by multiple mobile robots. *IEEE International Conference on Intelligent Robots and Systems*, 1997. Citado na página 21.
- SASAKI, J.; OTA, J.; YOSHIDA, E.; KURABAYASHI, D.; ARAI, T. Cooperating grasping of a large object by multiple mobile robots. *IEEE International Conference on Robotics and Automation*, 1995. Citado na página 21.

- SAXENA, A.; DRIEMEYER, J.; NG, A. Y. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, v. 27, p. 157–173, 2008. Citado 2 vezes nas páginas 17 e 21.
- SCHWARZ, M.; LENZ, C.; GARCÍA, G. M.; KOO, S.; PERIYASAMY, A. S.; SCHREIBER, M.; ; BEHNKE, S. Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing. *IEEE International Conference on Robotics and Automation*, 2018. Citado na página 22.
- SICILIANO, B.; KHATIB, O. *Springer Handbook of Robotics*. Berlin: Springer, 2016. v. 2. Citado 11 vezes nas páginas 17, 23, 30, 31, 33, 36, 39, 41, 42, 75 e 79.
- SMITH, C.; KARAYIANNIDIS, Y.; NALPANTIDIS, L.; GRATAL, X.; QI, P.; DIMAROGONAS, D. V.; KRAGIC, D. Dual arm manipulation—a survey. *Robotics and Autonomous Systems*, 2012. Citado 2 vezes nas páginas 17 e 20.
- UCHIYAMA, M.; DAUCHEZ, P. A symmetric hybrid position/force control scheme for the coordination of two robots. *IEEE International Conference on Robotics and Automation*, Philadelphia, PA, USA, USA, 1988. Citado 2 vezes nas páginas 20 e 74.
- UCHIYAMA, M.; DAUCHEZ, P. Symmetric kinematic formulation and non-master/slave coordinated control of two-arm robots. *Advanced Robotics*, v. 7, 1992. Citado 2 vezes nas páginas 20 e 74.
- UCHIYAMA, M.; IWASAWA, N.; HAKOMORI, K. Hybrid position/force control for coordination of a two-arm robot. *IEEE International Conference on Robotics and Automation*, Raleigh, NC, USA, USA, 1987. Citado 2 vezes nas páginas 20 e 74.
- UNIVERSAL Robots Website. 2017. <<https://www.universal-robots.com>>. Acessado: 11 de Dezembro de 2017. Citado na página 70.
- VAHRENKAMP, N.; BOGE, C.; WELKE, K.; ASFOUR, T.; WALTER, J.; DILLMANN, R. Visual servoing for dual arm motions on a humanoid robot. *IEEE-RAS International Conference on Humanoid Robots*, 2009. Citado na página 21.
- VARGAS, L. V.; LEITE, A. C.; COSTA, R. R. Kinematic control of robot manipulators using filtered inverse. *Mediterranean Conference on Control and Automation*, 2013. Citado na página 19.
- WAMPLER, I. C. W. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 16, 1986. Citado na página 19.
- WIT, C. C. de; SICILIANO, B.; BASTIN, G. *Theory of Robot Control*. Berlin: Springer, 1996. Citado na página 79.
- WOLOVICH, W. A.; ELLIOTT, H. A computational technique for inverse kinematics. *Conference on decision and control*, 1984. Citado na página 19.
- YAN, L.; MU, Z. Coordinated compliance control of dual-arm robot for payload manipulation: Master-slave and shared force control. *IEEE International Conference on Intelligent Robots and Systems*, 2016. Citado na página 21.
- YOSHIKAWA, T. Manipulability and redundancy control of robotic mechanisms. *IEEE International Conference on Robotics and Automation*, 1985. Citado na página 82.