

**Universidade Federal de Santa Catarina  
Centro de Blumenau  
Departamento de Engenharia de  
Controle e Automação e Computação**



**Anderson Cordeiro de Souza**

**Geração automática de trajetórias em robôs colaborativos**

**Blumenau**

**2021**

**Anderson Cordeiro de Souza**

## **Geração automática de trajetórias em robôs colaborativos**

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Controle e Automação.  
Orientador: Prof. Dr. Leonardo Mejia Rincon

Universidade Federal de Santa Catarina  
Centro de Blumenau  
Departamento de Engenharia de  
Controle e Automação e Computação

Blumenau  
2021

**Anderson Cordeiro de Souza**

# **Geração automática de trajetórias em robôs colaborativos**

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

## **Comissão Examinadora**

---

Prof. Dr. Leonardo Mejia Rincon  
Universidade Federal de Santa Catarina  
Orientador

---

Prof. Dr. Daniel Alejandro Ponce Saldias  
Universidade Federal de Santa Catarina

---

Prof. Dr. Julio Cesar Frantz  
Centro Universitário de Brusque -  
UNIFEDE

Blumenau, 1 de outubro de 2021

Dedico este trabalho aos meus pais, que nunca mediram esforços para me auxiliar,  
graças a eles que hoje posso concluir o meu curso.

# Agradecimentos

Agradeço a Deus pelo dom da vida, por ter me dado saúde e forças para superar cada etapa da minha caminhada.

Agradeço aos meus pais, Adilson e Valéria, por me proporcionarem educação, humildade, caráter e por todo apoio e ajuda que me forneceram com seus esforços, permitindo que eu pudesse chegar até aqui.

Agradeço ao meu orientador Professor Dr. Leonardo Mejia Rincon, que acreditou em mim em 2019 para realização de um projeto de iniciação científica, e desde então viemos trabalhando juntos, agradeço por suas orientações e parte de seus conhecimentos passados à mim.

Aproveito para agradecer todos os professores que participaram da minha caminhada, fornecendo o conhecimento e tempo necessário para que eu chegassem onde estou hoje. Agradeço em especial o professor Daniel Girardi o qual acreditou na minha capacidade desde o começo da faculdade, abrindo portas para pesquisas e fazendo mediação entre os donos da empresa Atto Educacional e eu, o qual tive muitos aprendizados.

Também Agradeço em especial o professor Daniel Almeida Fagundes, por suas palavras de sabedoria quando reprovei em sua matéria de Física II, a conversa e os conselhos ditos pelo professor me proporcionaram crescer pessoalmente e academicamente.

Ao professor Daniel Alejandro Ponce Saldías o meu muito obrigado, pois foi quem confiou a mim a responsabilidade de apresentar um trabalho no congresso MEC3F em Foz do Iguaçu, proporcionando minha primeira participação em congressos, o qual obteve a conquista de projeto destaque.

A todos os amigos que fiz neste período dentro e fora da faculdade, os quais são exemplos de resiliência, persistência e dedicação, deixo minha gratidão por proporcionarem momentos e conversas, de evolução e de diversão. Agradeço a todos os colegas que dividi turma em algum momento dessa jornada.

A todos os professores e alunos que compraram um vidro de mel ou auxiliaram na divulgação, ajudando na realização de um sonho e um objetivo para que eu pudesse chegar aqui hoje, deixo o meu muito obrigado.

Por fim, agradeço a Universidade Federal de Santa Catarina por ter me possibilitado a tantas oportunidades e experiências, por sua estrutura e seus laboratórios.

*"Nenhum saber é saber completo."*  
(Galileu Galilei)

# Resumo

Uma das principais preocupações na área industrial é a segurança dos trabalhadores, principalmente quando trabalham diretamente com máquinas ou robôs industriais. A robótica colaborativa tem um papel importante para que robôs e humanos trabalhem juntos e acidentes sejam evitados. Este trabalho propõe uma abordagem de geração automática de trajetórias em manipuladores para que evitem colisões com objetos e principalmente com humanos. A aplicação envolve a identificação de pontos específicos do corpo do trabalhador que são considerados como obstáculos do ponto de vista do manipulador.

Através do *software* de simulação CoppeliaSim, linguagem de programação LUA e da biblioteca de cinemática inversa simIK, foi desenvolvida uma metodologia que identifica quando o obstáculo encontra-se no caminho da trajetória do manipulador, possibilitando a geração de novas trajetórias de evasão através de cálculos e combinações de vetores de posição e matrizes de transformação homogênea.

Como principal resultado do projeto pode-se citar a implementação e validação da metodologia no robô Kuka LBR iiwa 7, o qual é um robô colaborativo, conseguindo realizar com alta precisão os desvios necessários do operador sem colidir.

Adicionalmente, a metodologia proposta é aplicada em outros tipos de manipuladores com ajustes nos parâmetros do algoritmo, sendo estes o ABB IRB 4600 e o robô Sawyer, os quais apresentaram resultados satisfatórios em termos de geração de novas trajetórias.

**Palavras-Chave:** 1.Geração de trajetórias. 2.Obstáculos. 3.CoppeliaSim. 4.Manipulador. 5.Matrizes de transformação homogênea.

# Abstract

One of the main concerns in the industrial area is the safety of workers, especially when they work directly with industrial machines or robots. Collaborative robotics has an important role so that robots and humans work together and accidents are avoided. This work proposes an automatic trajectory generation approach in manipulators to avoid collisions with objects and mainly with humans. The application involves identifying specific points on the worker's body that are considered obstacles from the manipulator's point of view.

Through the simulation *software* CoppeliaSim, LUA programming language, and the inverse kinematics library simIK, a methodology that identifies when the obstacle is in the path of the manipulator trajectory was developed, enabling the generation of new evasive trajectories through calculations and combinations of position vectors and homogeneous transformation matrices.

The main result of the project is the implementation and validation of the methodology in the Kuka LBR iiwa 7 robot, which is a collaborative robot, being able to perform with high precision the necessary deviations of the operator without colliding.

Additionally, the proposed methodology is applied to other types of manipulators with adjustments to the algorithm parameters, these being the ABB IRB 4600 and the Sawyer robot, which presented satisfactory results in terms of generating new trajectories.

**Keywords:** 1.Trajectory generation. 2.Obstacles. 3.Manipulator. 4.CoppeliaSim. 5.Homogeneous transformation matrix.

# Listas de figuras

Figura 1 – Quantidade de robôs industriais no mundo . . . . .	14
Figura 2 – Robô da ABB com seis eixos . . . . .	18
Figura 3 – Robô SCARA com quatro eixos . . . . .	19
Figura 4 – Espaço de trabalho do Robô Kuka LBr iiwa . . . . .	20
Figura 5 – Sistema cartesiano ortogonal $O_{xyz}$ . . . . .	21
Figura 6 – Vetor $\vec{r}$ de uma partícula no espaço. . . . .	21
Figura 7 – Matriz homogênea. . . . .	23
Figura 8 – Trajetória linear entre dois pontos. . . . .	25
Figura 9 – Robô colaborativo com divisão de tarefas. . . . .	27
Figura 10 – Robô colaborativo em trabalhos diferentes. . . . .	27
Figura 11 – Robô colaborativo Kuka LBr iiwa . . . . .	28
Figura 12 – Obstáculos presentes no trabalho. . . . .	28
Figura 13 – KUKA LBr iwaa no simulador CoppeliaSim. . . . .	30
Figura 14 – Interface do simulador CoppeliaSim. . . . .	31
Figura 15 – Ícone de acesso a interface de programação em LUA. . . . .	32
Figura 16 – Fluxograma das etapas realizadas. . . . .	33
Figura 17 – Dummy no CoppeliaSim. . . . .	34
Figura 18 – Pontos no espaço representados por Dummies. . . . .	35
Figura 19 – Efetuador Final ROBOTIQ 2F-85 acoplado no Kuka LBR iiwa 7 . . .	36
Figura 20 – Hierarquia dos <i>dummies</i> , <i>Tip</i> e <i>Target</i> em relação ao Kuka LBR iiwa 7. .	36
Figura 21 – Vetor distância entre dois pontos. . . . .	38
Figura 22 – Sentido das componentes do vetor $\vec{d}$ . . . . .	39
Figura 23 – Posição dos pontos de obstáculos no operador. . . . .	41
Figura 24 – Esfera virtual de segurança. . . . .	42
Figura 25 – Identificação de obstáculo através de vetor com vista superior. . . . .	42
Figura 26 – Geração de trajetória superior linear. . . . .	43
Figura 27 – Criação de uma nova posição através da matriz homogênea. . . . .	44
Figura 28 – Resultado da matriz homogênea em Y. . . . .	45
Figura 29 – Resultado da matriz homogênea em Z. . . . .	46
Figura 30 – Resultado da soma de matrizes. . . . .	47
Figura 31 – Sentido positivo do Sistema de coordenadas fixado no obstáculo. . . .	48
Figura 32 – Sentido $x$ da aproximação do manipulador em relação ao obstáculo. . .	49
Figura 33 – Sentido $z$ da aproximação do manipulador em relação ao obstáculo. . .	49
Figura 34 – Ações de rotação através dos parâmetros de ângulo e $\varepsilon$ . . . . .	50
Figura 35 – Nova trajetória com braço do operador estático. . . . .	52

Figura 36 – Realização de novas trajetórias com obstáculos em posições diferentes.	53
Figura 37 – Nova trajetória com identificação de dois obstáculos simultâneos. . . . .	54
Figura 38 – Criação de trajetória com obstáculo dinâmico. . . . .	54
Figura 39 – Aplicação da metodologia usando um robô ABB IRB 4600. . . . .	55
Figura 40 – Comprovação da aplicação da metodologia por outro sentido de identificação. . . . .	56
Figura 41 – Manipulador ABB IRB 4600 com obstáculo dinâmico. . . . .	56
Figura 42 – Criação de nova trajetória aplicada para robô Sawyer. . . . .	57

# **Lista de tabelas**

Tabela 1 – Dimensões alcançáveis do Kuka LBr iiwa . . . . .	20
Tabela 2 – Nome dos simuladores e seus desenvolvedores. . . . .	29
Tabela 3 – Diferença entre linguagens de programação. . . . .	30
Tabela 4 – Sentido de giro através dos parâmetros de ângulo e $\varepsilon$ . . . . .	50

# **Lista de Siglas e Abreviaturas**

IFR	<i>International Federation of Robotics</i>
RIA	<i>Robotic Industries Association</i>
ISO	<i>International Organization for Standardization</i>
GdL	<i>Graus de Liberdade</i>
CHR	<i>Colaboração Humano-Robô</i>

# Sumário

1	INTRODUÇÃO . . . . .	14
1.1	Objetivos . . . . .	15
1.1.1	Objetivo geral . . . . .	15
1.1.2	Objetivos específicos . . . . .	16
1.2	Estrutura do documento . . . . .	16
2	REVISÃO DE LITERATURA . . . . .	17
2.1	Robótica . . . . .	17
2.2	Robótica industrial . . . . .	17
2.2.1	Manipulador robótico . . . . .	18
2.2.1.1	Graus de liberdade . . . . .	19
2.2.1.2	Espaço de trabalho . . . . .	19
2.2.2	Vetores no espaço . . . . .	20
2.2.3	Matriz de transformação . . . . .	22
2.2.3.1	Matriz de translação . . . . .	22
2.2.3.2	Matriz de rotação . . . . .	22
2.2.3.3	Matriz de transformação homogênea . . . . .	23
2.2.4	Cinemática de manipuladores . . . . .	24
2.2.5	Cinemática direta . . . . .	24
2.2.6	Cinemática inversa . . . . .	24
2.2.7	Trajetórias . . . . .	25
2.3	Robótica Colaborativa . . . . .	26
2.3.1	Colaboração Humano-Robô . . . . .	26
2.3.2	Obstáculos e Colisões . . . . .	27
2.4	Simuladores Robóticos . . . . .	28
2.4.1	Simulador CoppeliaSim . . . . .	29
2.4.1.1	Linguagem de Programação LUA . . . . .	30
3	METODOLOGIA . . . . .	31
3.1	Interface de desenvolvimento e simulação . . . . .	31
3.2	Desenvolvimento da simulação . . . . .	32
3.3	Identificação dos pontos de destino e movimento do manipulador em trajetória linear . . . . .	34
3.3.1	Dummy . . . . .	34
3.3.2	Posição e orientação no espaço . . . . .	34

3.3.3	Posição do efetuador final do manipulador . . . . .	35
3.3.4	Cinemática inversa no CoppeliaSim . . . . .	37
3.3.5	Cálculo da distância entre pontos no espaço . . . . .	37
3.3.6	Próximos pontos e avanço do efetuador final do manipulador	38
3.4	Obstáculos e suas posições . . . . .	40
3.4.1	Esfera de segurança . . . . .	40
3.4.2	Cálculo de identificação dos obstáculos . . . . .	41
3.5	Geração de trajetória do manipulador . . . . .	43
3.5.1	Geração de trajetória de elevação linear . . . . .	43
3.5.2	Geração de trajetórias através da matriz de transformação homogênea . . . . .	43
4	<b>RESULTADOS . . . . .</b>	52
4.1	Simulação com o manipulador Kuka LBR iiwa 7 . . . . .	52
4.1.1	Geração de trajetórias com obstáculos estáticos . . . . .	52
4.1.2	Manipulador desviando de dois pontos estáticos com esferas de segurança concatenadas . . . . .	53
4.1.3	Manipulador desviando dinamicamente do operador . . . . .	53
4.2	Simulação com o manipulador ABB IRB 4600 . . . . .	55
4.3	Simulação com o manipulador Sawyer . . . . .	56
5	<b>CONCLUSÕES . . . . .</b>	58
5.1	Considerações finais . . . . .	59
5.2	Trabalhos futuros . . . . .	59
	<b>REFERÊNCIAS BIBLIOGRÁFICAS . . . . .</b>	60

# 1 Introdução

O mercado de robôs industriais está em grande crescimento, segundo a *International Federation of Robotics (IFR)* o uso de robôs nas indústrias ultrapassou a quantidade de 2,7 milhões de unidades no ano de 2019, sendo o maior nível da história. Em 10 anos o aumento mundial de robôs industriais foi maior que 150%, indo de 1,021 milhões de robôs em 2009 até 2,722 milhões em 2019. A expansão desse mercado pode ser acompanhada através da Figura 1.

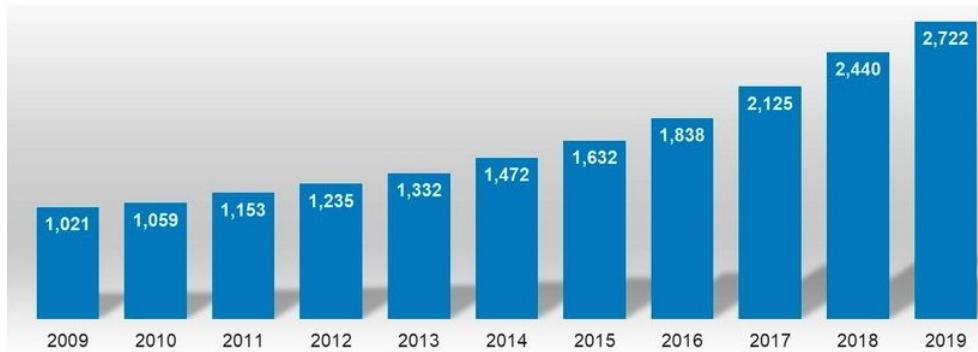


Figura 1 – Quantidade de Robôs industriais no mundo [1]

O conceito de Indústria 4.0 está presente no contexto da robótica industrial, uma vez que essa abordagem gera uma aproximação muito maior entre robôs e humanos para trabalharem juntos. Segundo a *International Organization for Standardization (ISO)* [2], com os novos avanços da tecnologia, há um potencial crescente para aproximar com segurança o poder e a precisão dos robôs da capacidade criativa e de resolução de problemas dos humanos, aumentando assim a produtividade.

Através da integração de humanos e robôs industriais trabalhando no mesmo espaço de trabalho, alguns acidentes podem acontecer [3][4][5]. Segundo Chinniah [6], estudos mostraram que 72% das lesões ou acidentes aconteceram diretamente com o operador do robô, e 28% dos acidentes aconteceram com trabalhadores da manutenção ou programadores.

Para diminuir a possibilidade de acidentes entre robôs e humanos, o conceito de robôs colaborativos tem surgido nos últimos anos. Um robô colaborativo é um robô que pode trabalhar ao lado dos seres humanos em total segurança [7]. Estes robôs são capazes de ajudar operadores a realizar atividades manuais em sistemas de manufatura moderna, combinando a capacidade humana com os pontos fortes das máquinas inteligentes [8].

Embora os robôs colaborativos apresentem algumas medidas de segurança que permitem a implementação de aplicações seguras, essas geralmente sofrem alterações quando integradas ao setor industrial [9]. As medidas de segurança dos robôs colaborativos são dadas pela detecção de contato do robô com objetos ou pessoas, fazendo com que o robô reduza a velocidade e força evitando choques mais bruscos e tornando-o mais sensível [10].

Um dos principais desafios da robótica colaborativa é o de realizar a identificação da proximidade entre robô e operador acompanhado o desvio do manipulador em relação ao trabalhador, evitando assim a colisão. Na tentativa de contribuir na solução desse desafio alguns autores têm desenvolvido algumas propostas, Flacco et. al. [11] utilizaram uma câmera Kinect para calcular as distâncias entre os pontos de referência do robô e do humano. Estes mesmos autores utilizaram as distâncias para gerar vetores repulsivos que são usados para controlar o robô durante a execução de uma tarefa de movimento genérico. Por outra parte, Khatib [12] apresenta o conceito clássico de campo de potencial artificial para evitar obstáculos em tempo real. Em um novo estudo, Flacco et. al. [13] estudaram o campo de potencial artificial, ao tempo que desenvolveram técnicas de prevenção de colisão para robôs redundantes. Outra abordagem é a geração de trajetórias livres de colisões para um robô que compartilha o espaço de trabalho com humanos utilizando uma rede neural, a qual cria pontos de caminhos evitando obstáculos dinâmicos no espaço [14].

Neste trabalho abordaremos o problema de geração automática de trajetórias ao identificar obstáculos presentes sobre a trajetória original de um manipulador industrial. Os conceitos apresentados por outros autores de calcular distâncias entre os pontos de referência, e usar essas distâncias para criar vetores são abordados de forma similar neste trabalho, entretanto, o método de geração de novas trajetórias é diferente dos autores citados anteriormente, fazendo com que a geração de uma nova trajetória seja determinada em volta do obstáculo, utilizando o mesmo como referência e ponto central dos cálculos. Os cálculos para geração de novas trajetórias englobam matrizes de transformação homogêneas e vetores de posição, possibilitando o robô rotacionar em torno do operador. Inicialmente a abordagem e testes são referentes aos robôs colaborativos, contudo, com ajustes de alguns parâmetros existe a possibilidade de implementar a geração de novas trajetórias em robôs não colaborativos, aumentando a segurança nas áreas industriais. Apesar que o trabalho pretende ser aplicado em robôs físicos futuramente, os testes foram realizados em softwares de simulação, por conta da praticidade de realizar ajustes sem danificar o robô.

## 1.1 Objetivos

### 1.1.1 Objetivo geral

O objetivo geral deste trabalho é, desenvolver uma proposta metodológica para a geração automática de trajetórias em um manipulador robótico colaborativo para evitar colisões com operadores humanos no ambiente de trabalho.

### 1.1.2 Objetivos específicos

Para que o objetivo geral seja atingido, o trabalho deve alcançar os respectivos objetivos específicos:

- Analisar trabalhos correlatos ao assunto.
- Implementar um sistema para que um robô industrial percorra pontos de destino pré-determinados pelo usuário.
- Determinar uma estratégia que permita identificar automaticamente a aproximação do efetuador final do manipulador com um objeto
- Propor um modelo adequado para rotação e translação de partículas no espaço.
- Avaliar a melhor estratégia de geração de novas trajetórias.
- Validar a metodologia proposta através de simulações.

## 1.2 Estrutura do documento

O presente documento foi dividido em cinco capítulos a fim de apresentar a fundamentação teórica, os materiais e métodos utilizados na construção metodológica e os resultados obtidos.

No Capítulo 1, uma breve introdução e contextualização do problema são apresentadas junto com os objetivos geral e específicos.

No Capítulo 2, é realizada uma breve revisão da literatura apresentando os conceitos que serão utilizados no desenvolvimento da pesquisa. Além disso, também são apresentados alguns simuladores que facilitam a aplicação e testes dos conceitos apresentados.

No Capítulo 3, é apresentada a metodologia utilizada, descrevendo os cálculos matemáticos aplicados no desenvolvimento do algoritmo de trajetórias proposto.

No Capítulo 4, apresentam-se os resultados obtidos através de testes executados em ambiente de simulação.

Por fim, o Capítulo 5, contém as conclusões deste trabalho e algumas propostas para trabalhos futuros utilizando novas abordagens para identificação dos seres humanos através de visão computacional.

## 2 Revisão de literatura

### 2.1 Robótica

“Robótica é a arte, a base de conhecimento e o *know-how* de concepção, aplicação e uso de robôs em atividades humanas” [15]. A ficção científica tem gerado uma concepção, algumas vezes incorreta, que robôs não são confiáveis para estarem no mesmo espaço que os humanos. Entretanto, os robôs facilitam os trabalhos e atividades no cotidiano, sendo elas dentro de indústrias, da própria casa ou até mesmo em áreas hospitalares.

Robótica, além da mecânica para o desenvolvimento da estrutura de sistemas robótizados, também é uma área interdisciplinar, que envolve elétrica, controle, programação e eletrônica, abrindo portas para que novas pesquisas sejam desenvolvidas no aperfeiçoamento destes sistemas.

### 2.2 Robótica industrial

A robótica industrial usufrui do estudo dos robôs que detém como finalidade auxiliar nos processos industriais. Com novas pesquisas os robôs estão se tornando cada vez mais uteis, pois são mais ágeis, velozes, precisos e também com uma maior flexibilidade [16].

Os robôs industriais ajudam principalmente em atividades que são perigosas para os seres humanos, sendo elas, pegar peças quentes ou frias na manufatura ou até mesmo peças muito pesadas.

A primeira definição de robô industrial foi proposta em 1979 pela *Robot Institute of America*, (atualmente *Robotic Industries Association RIA*), que definia estes equipamentos como:

*“Um robô industrial é um manipulador multifuncional reprogramável, capaz de movimentar materiais, peças, ferramentas ou dispositivos especiais, de acordo com trajetórias variáveis, programado para realizar diversas tarefas”.*

A ISO (International Organization for Standardization) também apresentou uma definição de robô industrial através da norma ISO 8373, como citado a seguir:

*“um robô industrial é um manipulador com 3 ou mais eixos, com controle automático, reprogramável, multi-aplicação, móvel ou não, destinado ao uso em aplicações de automação industrial”.*

Neste ponto é importante destacar que as duas definições supracitadas abordam o conceito de reprogramabilidade, que segundo a ISO pode ser entendida como: “os mo-

vimentos programados ou funções auxiliares que podem ser alterados sem a modificação física do sistema”. A ISO entende ainda uma modificação física como sendo, “a modificação da estrutura mecânica ou do sistema de controle (alterações na medida de memória são excluídas: disco, fita, ROM, etc)”.

### 2.2.1 Manipulador robótico

Manipulador robótico é o corpo do robô, que é constituído por eixos rígidos, que por sua vez são conectados por juntas e outros elementos estruturais do robô. Sem outros elementos tais como o controlador, o Teachpendant e os sistemas de comunicação e programação, o manipulador por si só não é um robô [15]. O conjunto de elementos estruturais do manipulador permite o movimento do manipulador robótico em um espaço de trabalho que é alterado conforme o tipo do manipulador. Os manipuladores robóticos tem diferentes estruturas e diferentes atuações, na Figura 2 é apresentado um manipulador de seis eixos da ABB que é utilizado na automatização de fábricas.



Figura 2 – Robô da ABB com seis eixos [17].

Um outro tipo comum de manipulador que pode ser apresentado, é o do tipo SCARA (do Ingles *Selective Compliance Assembly Robot Arm*) que pode ser visto na Figura 3 . Este tipo de manipulador consiste de quatro eixos e não possui movimentos de inclinação e de guinada igual os robôs de seis eixos [18].



Figura 3 – Robô SCARA com quatro eixos [19].

#### 2.2.1.1 Graus de liberdade

O número de graus de liberdade (GdL) que um manipulador possui é o número de variáveis de posição independentes que teriam de ser especificadas para localizar todas as partes do mesmo [16]. Os manipuladores podem ser classificados pela quantidade dos seus GdL da seguinte maneira.

- Manipuladores com 6 GdL, são chamados de Propósito Geral.
- Manipuladores com mais de 6 GdL, são chamados de Redundantes.
- Manipuladores com menos de 6 GdL, são chamados de SubAtuados.

Para posicionar um ponto no espaço é necessário conhecer as coordenadas no eixos  $x$ ,  $y$ , e  $z$ , entretanto para obter todas as informações da localização deste ponto, além da sua posição é fundamental conhecer também a sua orientação, assim, para que um manipulador possa atingir esse ponto dentro da sua área de trabalho, ele precisa ter 6 GdL [15], três definindo a posição do objeto, e mais três definindo a sua orientação.

#### 2.2.1.2 Espaço de trabalho

O espaço de trabalho representa o volume de pontos que o efetuador final do manipulador consegue acessar. Sua forma e volume dependem da estrutura do manipulador, juntamente com os limites mecânicos de cada uma das juntas [20].

Para que seja possível utilizar o manipulador em uma determinada tarefa, é importante verificar qual o espaço de trabalho do mesmo. Neste trabalho o robô utilizado é do modelo Kuka LBr iiwa que contém sete graus de liberdade e seu alcance pode chegar até 800 mm,

seu espaço de trabalho pode ser visto na Figura 4. Além disso, todas as medidas do espaço de trabalho do manipulador Kuka LBr iiwa, podem ser encontradas na Tabela Tabela 1.

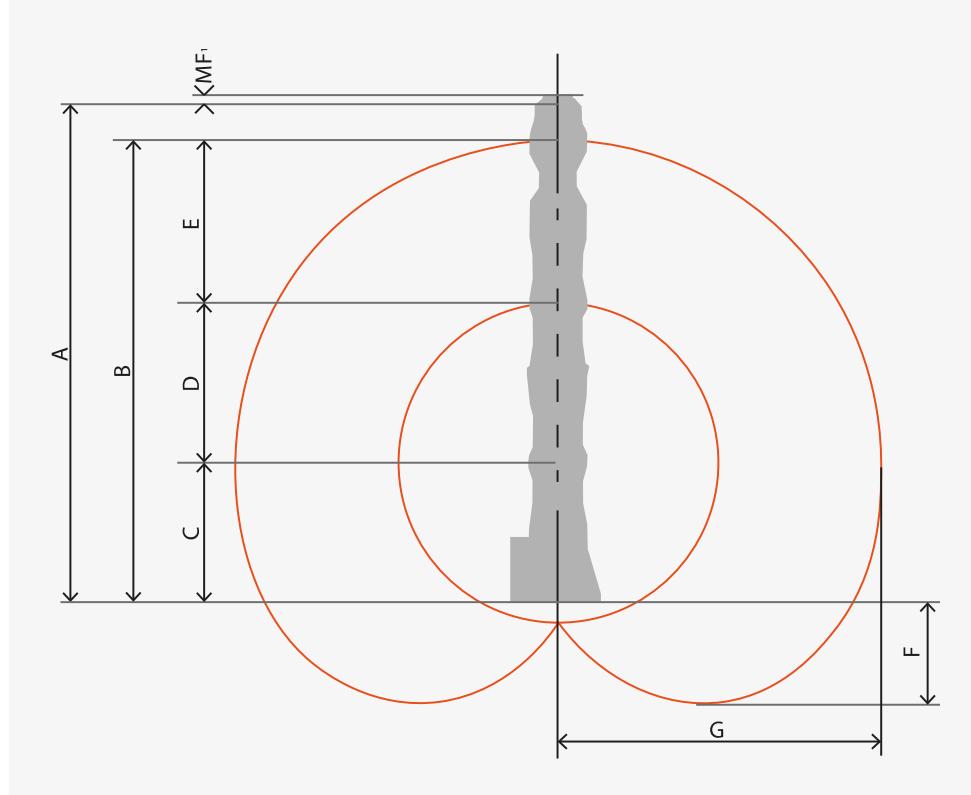


Figura 4 – Espaço de trabalho do Robô Kuka LBr iiwa [21]

Tabela 1 – Dimensões alcançáveis do Kuka LBr iiwa [21].

Referência	Dimensão
Dimensão A	1,266 mm
Dimensão B	1,140 mm
Dimensão C	340 mm
Dimensão D	400 mm
Dimensão E	400 mm
Dimensão F	260 mm
Dimensão G	800 mm
Volume	$1,8m^3$

## 2.2.2 Vetores no espaço

Vetores no espaço tem a base canônica  $\{\vec{i}, \vec{j}, \vec{k}\}$  sendo a que determina o sistema cartesiano ortogonal de  $Oxyz$  [22], onde é dado por três retas orientadas, perpendiculares duas a duas [23]. Esse sistema é representado por três vetores unitários com origem no ponto O conforme visto na Figura 5, onde a representação de cada vetor é determinada

por um ponto de referência e um ponto de destino, traçando uma reta com uma seta na ponta apontando para o ponto de destino, essa seta representa o sentido do vetor. Cada um dos vetores da Figura 5 definem os três eixos cartesianos, sendo eles o eixo  $O_x$  condizente ao vetor  $\vec{i}$ , o eixo  $O_y$ , condizente ao vetor  $\vec{j}$  e por fim o eixo  $O_z$ , condizente com o vetor  $\vec{k}$ .

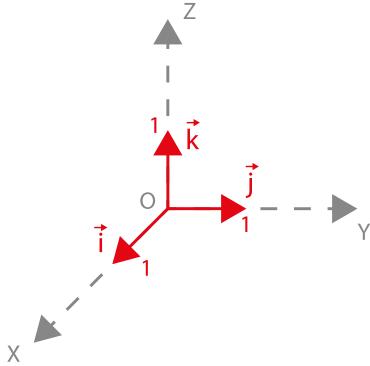


Figura 5 – Sistema cartesiano ortogonal  $O_{xyz}$

Um ponto no espaço afastado do ponto de referência  $O$  pode ser representado por um vetor posição  $\vec{r}$  que pode ser denotado na forma

$$\vec{r} = x\vec{i} + y\vec{j} + z\vec{k}$$

onde  $x\vec{i}, y\vec{j}$  e  $z\vec{k}$  são as componentes vetoriais de  $\vec{r}$ , e as componentes  $x, y$  e  $z$  são as componentes escalares. As componentes escalares representam a posição de uma partícula referente à origem ao longo dos eixos de coordenadas [24]. Por exemplo, a Figura 6 apresenta um elemento com vetor posição dado por

$$\vec{r} = 2\vec{i} + 4\vec{j} + 2\vec{k}$$

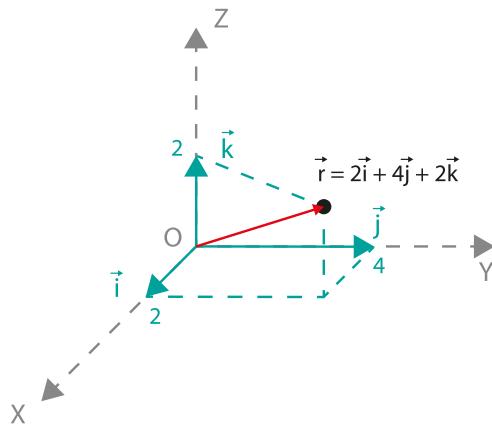


Figura 6 – Vetor  $\vec{r}$  de uma partícula no espaço.

O conceito de vetor espacial neste trabalho é de grande importância, pois a metodologia desenvolvida utiliza mais que uma partícula no espaço, de forma que algumas operações

vetoriais também são realizadas com o intuito de identificar o sentido das componentes vetoriais.

### 2.2.3 Matriz de transformação

Na álgebra linear, as transformações lineares podem ser representadas por matrizes. Se  $T$  é uma transformação linear de  $\mathbb{R}^n$  para  $\mathbb{R}^m$  e  $\vec{x}$  é um vetor coluna, então:

$$T(\vec{x}) = A\vec{x} \quad (2.1)$$

Na expressão acima, a matriz  $A$  é uma matriz de transformação que permite que transformações lineares arbitrárias sejam exibidas em um formato consistente e apropriado para cálculos. A representação matricial destes elementos também permite que as transformações sejam concatenadas facilmente.

Matrizes de transformação são usadas em vários campos do conhecimento como visão computacional, ciências físicas, computação gráfica e robótica. Apesar que existem muitos tipos de matrizes de transformação como as de compressão, reflexão e cisalhamento, em robótica as matrizes mais comumente usadas são a matriz de translação pura, a matriz de rotação pura em torno de um eixo e as matrizes de transformação homogênea [15]. Estas três principais matrizes são descritas na sequência:

#### 2.2.3.1 Matriz de translação

É utilizada quando um referencial ou objeto está se deslocando no espaço sem alteração em sua orientação, obtendo assim uma translação pura. A matriz de translação tem tamanho de 4x4 e apenas sua última coluna tem valores variáveis que referem-se a translação do objeto, a transformação  $T$  é dada por:

$$T = \begin{bmatrix} 1 & 0 & 0 & r_x \\ 0 & 1 & 0 & r_y \\ 0 & 0 & 1 & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

as componentes de translação  $r_x$ ,  $r_y$  e  $r_z$  são componentes do vetor  $\vec{r}$  referentes aos eixos  $x$ ,  $y$  e  $z$  do sistema de referência.

#### 2.2.3.2 Matriz de rotação

Permite a rotação de um ponto no espaço em torno de um eixo do sistema de referência, assim, podem existir três tipos de rotações relacionadas aos eixos  $x$ ,  $y$  e  $z$ .

Para uma matriz que rotaciona em relação ao eixo  $x$  por exemplo, segue a seguinte estrutura matricial:

$$Rot(x, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

Para rotacionar um ponto no espaço, é necessário definir um sistema de referência. Um exemplo típico é rotacionar em relação ao sistema de referência da origem  $O$ , portanto, para a obtenção de novas coordenadas  $x'$ ,  $y'$  e  $z'$ , a matriz de rotação é multiplicada pelas coordenadas  $x, y$  e  $z$  do ponto original, representadas pela equação (2.2).

$$P_{x'y'z'} = Rot(x, \theta) \cdot P_{xyz} \quad (2.2)$$

na forma matricial:

$$\begin{bmatrix} Px' \\ Py' \\ Pz' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix}$$

### 2.2.3.3 Matriz de transformação homogênea

De maneira simplificada, uma matriz de transformação homogênea pode ser entendida como a junção de uma matriz de translação com uma matriz de rotação. O principal problema dessa junção é o fato que as matrizes de translação pura e rotação pura apresentam dimensões diferentes como mostrado anteriormente, portanto, para resolver este problema recorre-se a uma transformação de dimensão em que adiciona-se mais alguns elementos à matriz de rotação pura, tornando essa a matriz uma matriz quadrada  $4 \times 4$  sem alterar os cálculos dos outros coeficientes. Este tipo de matriz é ideal para ser utilizada nesse trabalho. A Figura 7 apresenta de forma generalizada uma matriz de transformação homogênea descrevendo cada um dos seus elementos. Nessa figura, além dos componentes de translação e rotação são apresentados os componentes de perspectiva e fator de escala, que em robótica assumem os valores de  $[0 \ 0 \ 0]$  e  $[1]$  respectivamente.

$$T = \begin{bmatrix} \text{Rotações} & & \text{Translação} \\ \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} & \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \\ \text{Perspectiva} & \text{Fator de Escala} & S \end{bmatrix}$$

Figura 7 – Matriz homogênea.

As principais matrizes de transformação homogênea utilizadas neste trabalho são as de rotação em torno aos eixos  $y$  e  $z$ , e cuja representação é apresentada nas equações 2.3 e 2.4.

$$Rot(y, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & O_n \\ 0 & 1 & 0 & O_o \\ -\sin \theta & 0 & \cos \theta & O_a \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

$$Rot(z, \theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & O_n \\ \sin \theta & \cos \theta & 0 & O_a \\ 0 & 0 & 1 & O_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

## 2.2.4 Cinemática de manipuladores

A cinemática de manipuladores é o estudo da posição e da velocidade do efetuador final e das juntas do mesmo. Quando citado o termo de posição, refere-se tanto à posição propriamente dita quanto à orientação, e quando se refere ao termo de velocidade, considera-se tanto a velocidade linear quanto a velocidade angular.

Podemos distinguir dois tipos diferentes de cinemática, sendo estas a cinemática direta e a cinemática inversa como mostrado a seguir.

## 2.2.5 Cinemática direta

Na cinemática direta, deseja-se obter a posição e velocidade do efetuador final do manipulador em função do comportamento das velocidades cinemáticas das juntas atuadas do mesmo.

## 2.2.6 Cinemática inversa

A cinemática inversa é utilizada quando temos as variáveis cinemáticas do efetuador final sendo elas um conjunto de posições e orientações, de forma que o problema é dado ao calcular todos os conjuntos possíveis de variáveis cinemáticas de junta acionadas onde é representado por posição e velocidade [25].

A cinemática inversa de um manipulador pode ser considerada um dos conhecimentos mais valiosos e indispensáveis na hora de realizar a movimentação de um robô no espaço uma vez que através da cinemática inversa é possível calcular o conjunto de ângulos nas juntas atuadas necessários para que o efetuador final do manipulador consiga alcançar um ponto com posição e orientação desejadas [16].

A cinemática inversa pode se tornar muito mais complexa do que a cinemática direta devido a possibilidade de existirem equações não lineares, múltiplas soluções ou até mesmo infinitas soluções para caso de manipuladores redundantes, inclusive, pode não

haver soluções admissíveis, tendo em vista a estrutura cinemática do manipulador [20]. Levando em consideração que a utilização dessa abordagem deve ser considerada para manipuladores seriais, pois para robôs paralelos, a cinemática inversa é mais fácil do que a cinemática direta.

### 2.2.7 Trajetórias

As trajetórias em muitas aplicações práticas são definidas através do caminho que o manipulador deve percorrer, partindo de um ponto inicial até um ponto de destino, sendo capaz de passar por vários outros pontos intermediários. O conjunto de ângulos dos eixos do manipulador são definidos basicamente pela posição do efetuador final e seus movimentos [26].

Uma maneira de movimentar o robô é utilizando uma trajetória linear entre dois pontos, conforme mostrado na Figura 8, entretanto para que isso seja praticável é necessário dividir essa reta em vários pequenos pontos e mover o robô ao longo de cada ponto intermediário [15]. Com a realização desse tipo de método é possível identificar em cada momento o movimento do manipulador.

Vale salientar que os pontos descritos anteriormente não são apenas pontos vagos no espaço, cada um desses pontos são constituídos como um sistema local de referência que fornece instantaneamente tanto a posição quanto a orientação [16].

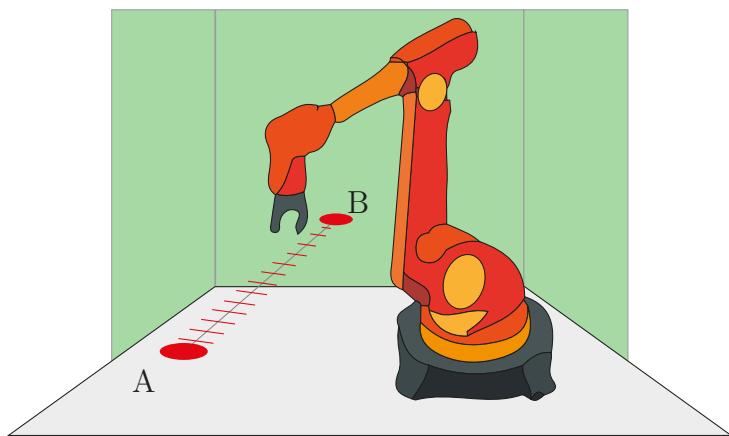


Figura 8 – Trajetória linear entre dois pontos.

Uma outra abordagem de trajetórias a ser descrito, é a trajetória no espaço articular com características controladas. Através desse enfoque a trajetória contém um determinado tempo para chegar em cada um dos pontos definidos, e esse tempo é especificado em termos da velocidade e da aceleração [20].

Nesse trabalho o foco principal é dado em trajetórias lineares entre dois pontos (Ponto de origem e ponto de destino) com velocidade constante no efetuador final do manipulador. Com o aparecimento de obstáculos no espaço de trabalho que interfiram na trajetória inicialmente calculada, a geração de novas trajetórias é de grande relevância para que

não haja colisões, contudo, essa nova trajetória gerada ainda mantém a característica da velocidade constante no efetuador final do manipulador.

## 2.3 Robótica Colaborativa

Nos últimos anos, o interesse em desenvolver manipuladores com maior flexibilidade, maior rapidez e de fácil reprogramação tem aumentado notavelmente. Isso inclui pesquisas para soluções de colaboração humano-robô (CHR), onde processos semi ou totalmente automatizados podem ser combinados com a flexibilidade e destreza humana de maneira relativamente simples [27]. O objetivo deste tipo de robótica é o de realizar atividades que sejam de difícil manuseio humano, como por exemplo: carregar itens pesados, produtos farmacêuticos ou até mesmo objetos radioativos.

Quando se fala de robótica colaborativa, é importante ter a segurança necessária para que um robô possa trabalhar juntamente com humanos, nesse contexto foi criada a norma *ABNT NBR ISO 10218-1:2018*, a qual expõe as ameaças relacionadas à efetivação dessa tecnologia e as condições para extinguí-las, juntamente com o conhecimento sobre a utilização dos robôs industriais [28].

### 2.3.1 Colaboração Humano-Robô

O comportamento humano pode ser imprevisível nas atividades de montagem, de forma que é difícil para os robôs entenderem as intenções das operações humanas [29]. Para que haja desenvolvimentos contínuos na colaboração humano-robô, são primordiais os aperfeiçoamentos adicionais em uma infinidade de disciplinas e áreas de pesquisa [30].

A colaboração humano-robô pode ser abordada de duas formas diferentes, a primeira representada pela Figura 9 onde o operador interage com o robô na realização da montagem de um equipamento, ou uma placa eletrônica por exemplo, e outra onde o operador e o robô utilizam do mesmo espaço de trabalho realizando suas atividades separadamente, porém muito próximos um do outro como pode ser visto como exemplo na Figura 10.

Como já comentado anteriormente este trabalho é realizado com o Kuka LBR iiwa que é um robô assistente de trabalho industrial inteligente, onde humanos e robôs conseguem realizar funções altamente sensíveis em estreita cooperação [31]. O KUKA LBR iiwa tem uma variedade de funções para garantir a segurança do operador como reações rápidas e detecção de variações indesejadas no seu percurso, fazendo com que o robô pare seus movimentos ou retorne para posições anteriores.

Por referir-se a um robô com colaboração humana, este robô permite realizar sua programação de percurso diretamente por softwares e com a criação de códigos, ou também através da interação humana, sendo visto um exemplo na Figura 11 onde o humano pode movimentar o robô até o ponto desejado e definir este ponto como um ponto de destino.

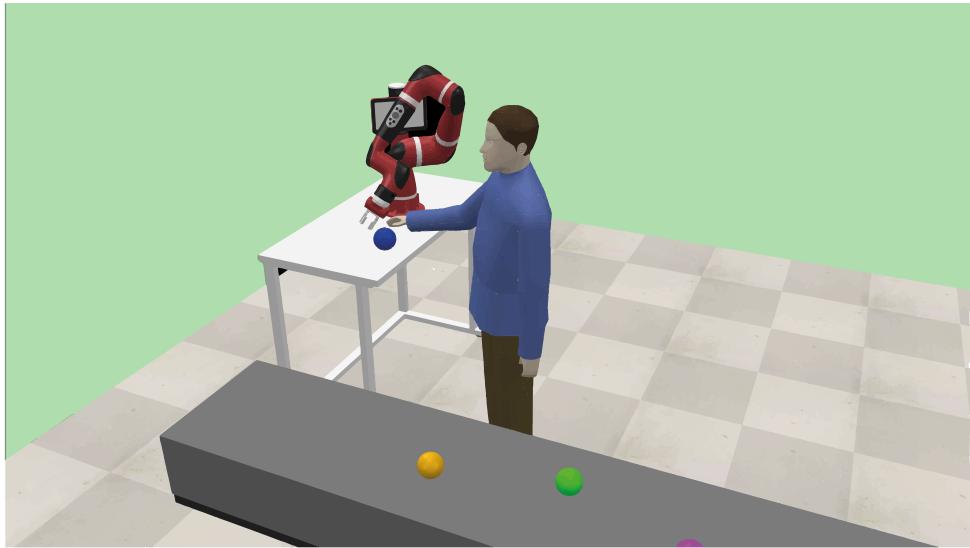


Figura 9 – Robô colaborativo com divisão de tarefas.



Figura 10 – Robô colaborativo em trabalhos diferentes.

### 2.3.2 Obstáculos e Colisões

Do ponto de vista dos robôs colaborativos que trabalham sem a interação direta com outros operários de uma linha de montagem, os seres humanos na grande maioria representam obstáculos nas suas trajetórias. Uma das premissas mais importantes na execução desse tipo de atividades é a de evitar colisões com o operadores humanos [32].

No presente trabalho, os obstáculos são definidos como sendo as mãos e antebraços do operador, representados na cor vermelha na Figura 12. Inicialmente estes pontos são utilizados como obstáculos para validação dos conceitos que serão apresentados na seção de metodologia, entretanto, para que possam ser construídos os algoritmos, realizados os testes e as devidas validações, o uso de um simulador é indispensável neste trabalho.



Figura 11 – Robô Colaborativo Kuka LBr iiwa [21].

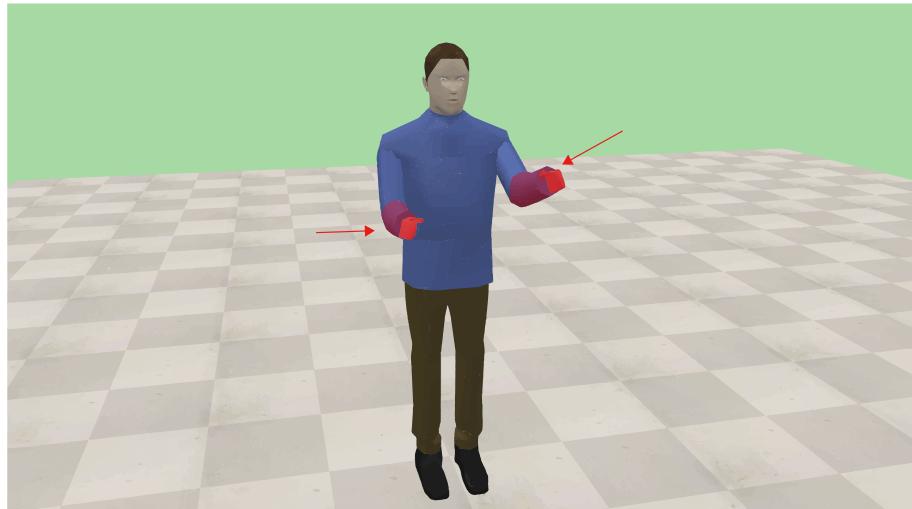


Figura 12 – Obstáculos presentes no trabalho.

## 2.4 Simuladores Robóticos

Simuladores robóticos são ferramentas úteis que auxiliam no desenvolvimento e testes de novas soluções para a área da robótica através de um software. Desenvolver um algoritmo e tentar rodar diretamente em um robô físico é algo um tanto complicado, inicialmente por precisar de um robô físico disponível para realização dos testes. Outro ponto, seria a possibilidade de queimar algum componente ou até mesmo quebrar alguma junta, elo ou peças do robô, além é claro, do custo que teria para repor as peças e o risco humano envolvido.

Para facilitar a vida de estudantes, pesquisadores, profissionais da área e até mesmo para quem prefere aprender mais sobre robótica por *hobby*, existem os simuladores, cada um com suas características e aplicações. Alguns dos simuladores têm a capacidade de transferir o código desenvolvido diretamente para o robô, para que posteriormente o comportamento físico do sistema possa ser analisado. Na Tabela 2 são apresentados alguns dos principais simuladores usados atualmente junto com os seus respectivos desenvolvedores. Neste trabalho foi utilizado especificamente o simulador CoppeliaSim que será descrito com maior detalhamento na sequência.

Tabela 2 – Nome dos simuladores e seus desenvolvedores.

Software	Desenvolvedores
Actin	Energid Technologies
Gazebo	Open Source Robotics Foundation (OSRF)
MORSE	Academic community
RoboDK	RoboDK
SimSpark	O. Obst et al. (+26)
CoppeliaSim	Coppelia Robotics
Webots	Cyberbotics
4DV-Sim	4D Virtualiz

### 2.4.1 Simulador CoppeliaSim

O simulador CoppeliaSim tem uma vasta gama de recursos para serem utilizados nos mais variados tipos de aplicações, esses recursos permitem que possamos realizar testes com uma maior precisão e eficiência dos estudos desenvolvidos. Nem todos os recursos são utilizados neste trabalho, porém o conhecimento é importante para a realização de trabalhos futuros. Alguns dos recursos mais importantes do CoppeliaSim são:

- Suporta API para 6 tipos de linguagens de programação: C, Java, Python, Matlab, Octave e Lua;
- API's remotas para que seja possível controlar os objetos através de programas externos;
- Cinemática direta e inversa para realização dos cálculos automaticamente;
- Sensores de proximidade conseguindo calcular com exatidão os objetos no simulador;

Além dos recursos citados anteriormente, o CoppeliaSim disponibiliza em seu software o robô KUKA LBr iiwa como visto na Figura 13, o qual tem todas as características de um robô real, tanto na quantidade de juntas, espaço de trabalho entre outros aspectos.

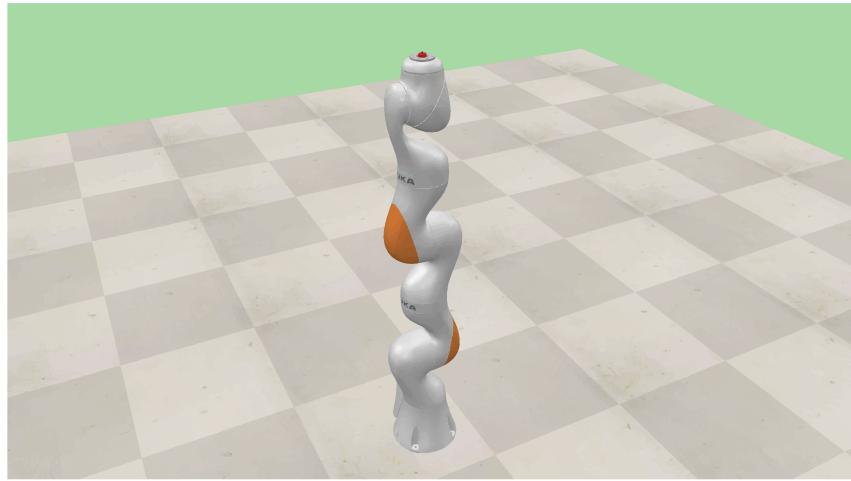


Figura 13 – KUKA LBr iwa no simulador CoppeliaSim.

Entretanto, vários dos recursos que o CoppeliaSim disponibiliza para o KUKA LBR iwa foram desabilitados neste trabalho, sendo eles: colisão, mensurabilidade e detectabilidade. Com esses recursos desativados, é possível implementar a metodologia proposta na prática e de fato verificar a autenticidade do algoritmo criado para realização dos movimentos do robô sem levar em consideração ajustes automáticos de colisão que o próprio simulador pode incluir.

#### 2.4.1.1 Linguagem de Programação LUA

Para que seja possível criar um algoritmo e simular o robô, é necessário escolher uma linguagem de programação entre as quais o CoppeliaSim tem disponível. Ao buscar uma linguagem com script poderoso, eficiente, leve e incorporável ao CoppeliaSim, encontramos a linguagem LUA, que suporta programação procedural, programação orientada a objetos, programação funcional, programação orientada a dados e descrição de dados [33].

A decisão da linguagem de programação foi baseado em algumas tópicos apresentados na Tabela 3, sendo que as principais abordagens foram o cálculo da cinemática inversa podendo ser executado pelo CoppeliaSim e também a variedade de bibliotecas disponíveis prontamente para LUA. Além disso para incorporação do código Python e MATLAB, seria necessário a utilização de uma parte do código em LUA para recebimento e envio dos valores das variáveis, o que se tornaria desnecessário para o objetivo deste trabalho.

Tabela 3 – Diferença entre linguagens de programação.

Linguagem	LUA	Python	Matlab
Cálculo da Cinemática Inversa	X	X	X
Programação incorporada no CoppeliaSim	X		
Programação procedural	X	X	X
Bibliotecas próprias do CoppeliaSim disponíveis	X	X	X

# 3 Metodologia

Neste capítulo é apresentado um pouco sobre o processo que o autor desenvolveu para geração automática de trajetórias, mostrando assim o caminho e cálculos propostos para atingir o objetivo. Esta metodologia não é única para atingir o propósito, outras formas e ferramentas podem ser utilizadas para chegar no resultado adequado, de forma que este texto possa servir como base para novos desenvolvimentos.

## 3.1 Interface de desenvolvimento e simulação

Como comentado no capítulo anterior a utilização do simulador CoppeliaSim é fundamental para a realização deste trabalho, de modo que nesta seção iremos apresentar uma breve descrição da sua interface e suas funcionalidades. Na Figura 14 as principais seções do simulador são exibidas, como, barra de menu, barra de ferramentas, tela de modelos, tela de hierarquia e a tela de projeto.

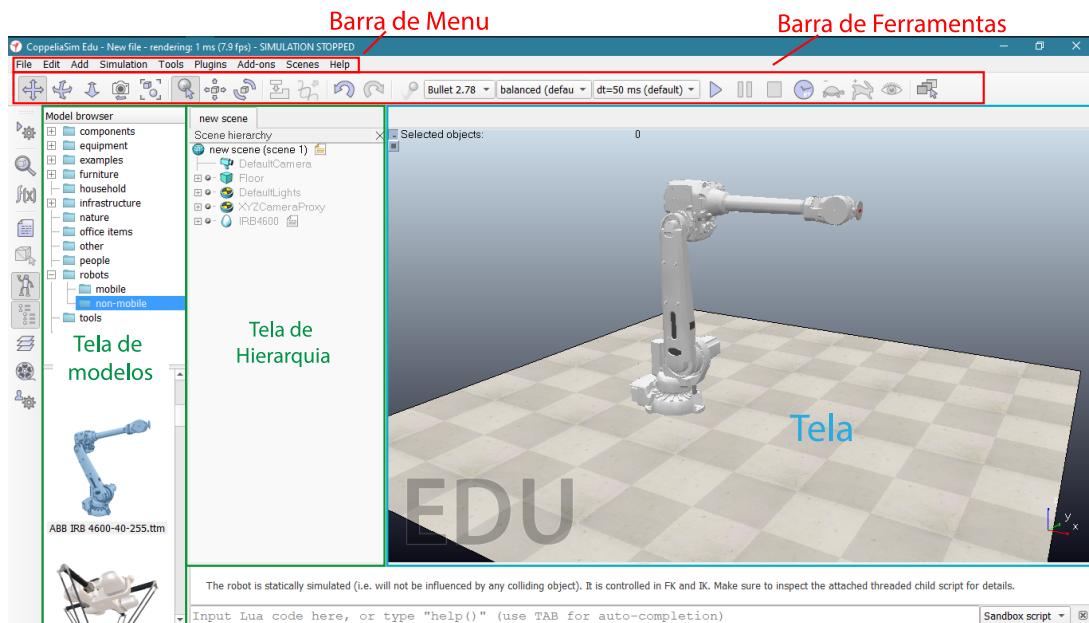


Figura 14 – Interface do simulador CoppeliaSim.

A barra de menu contém a maioria das funcionalidades e ferramentas do simulador, desde a criação de novas cenas e a adição de componentes até o acesso ao manual do usuário.

A barra de ferramentas permite ao usuário realizar mudanças na posição ou na orientação de um objeto, robô ou outro componente quando o mesmo está na tela de projeto. A barra de ferramentas ainda permite ao usuário inicializar a simulação, pausar, parar e até mesmo acelerar ou reduzir a velocidade de simulação.

A tela de modelos mostra todos os robôs, mesas, pessoas e acessórios que podem ser utilizados para realizar uma simulação. Para utilizar o objeto escolhido, o usuário precisa apenas segurar o objeto e arrastar para a tela de projeto com o *mouse*, soltando logo em seguida.

A tela de hierarquia apresenta todos os elementos que compõem uma cena na tela de projeto. Os objetos que são construídos na cena apresentam uma estrutura hierárquica, podendo assim expandir ou contrair a quantidade de itens.

Na tela de projeto, todos os componentes são colocados para gerar a simulação e visualização do que está sendo executado.

Na Figura 15 é apresentado o ícone que dá acesso a interface de programação do CoppeliaSim, o qual é programado em linguagem LUA, entretanto, através dessa interface e também de API's é possível realizar a comunicação com linguagens de programação alternativas, como python, matlab, C++, entre outras.

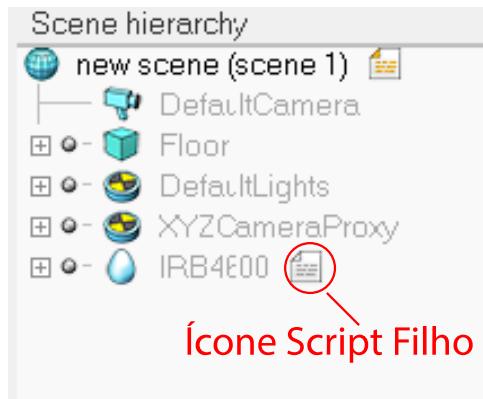


Figura 15 – Ícone de acesso a interface de programação em LUA.

## 3.2 Desenvolvimento da simulação

Ao compreender as ferramentas do CoppeliaSim citadas na seção anterior, agora é possível conhecer mais sobre o desenvolvimento e aplicabilidade empregada nesse trabalho. A Figura 16 apresenta de maneira simplificada as etapas principais do processo metodológico proposto através de um fluxograma. A primeira etapa do fluxograma identifica os pontos de destino e movimento do manipulador em trajetória linear, a segunda etapa do fluxograma aborda a identificação dos obstáculos próximos ao trajeto do manipulador, e por último, a geração de uma nova trajetória desviando de obstáculos é obtida na terceira etapa do fluxograma.

Com o objetivo de permitir a replicabilidade e aprimoramento dos procedimentos metodológicos propostos neste trabalho, cada uma das três etapas apresentadas no fluxograma é descrita na sequência junto com os cálculos vetoriais utilizados no desenvolvimento das simulações realizadas através do *software* CoppeliaSim.

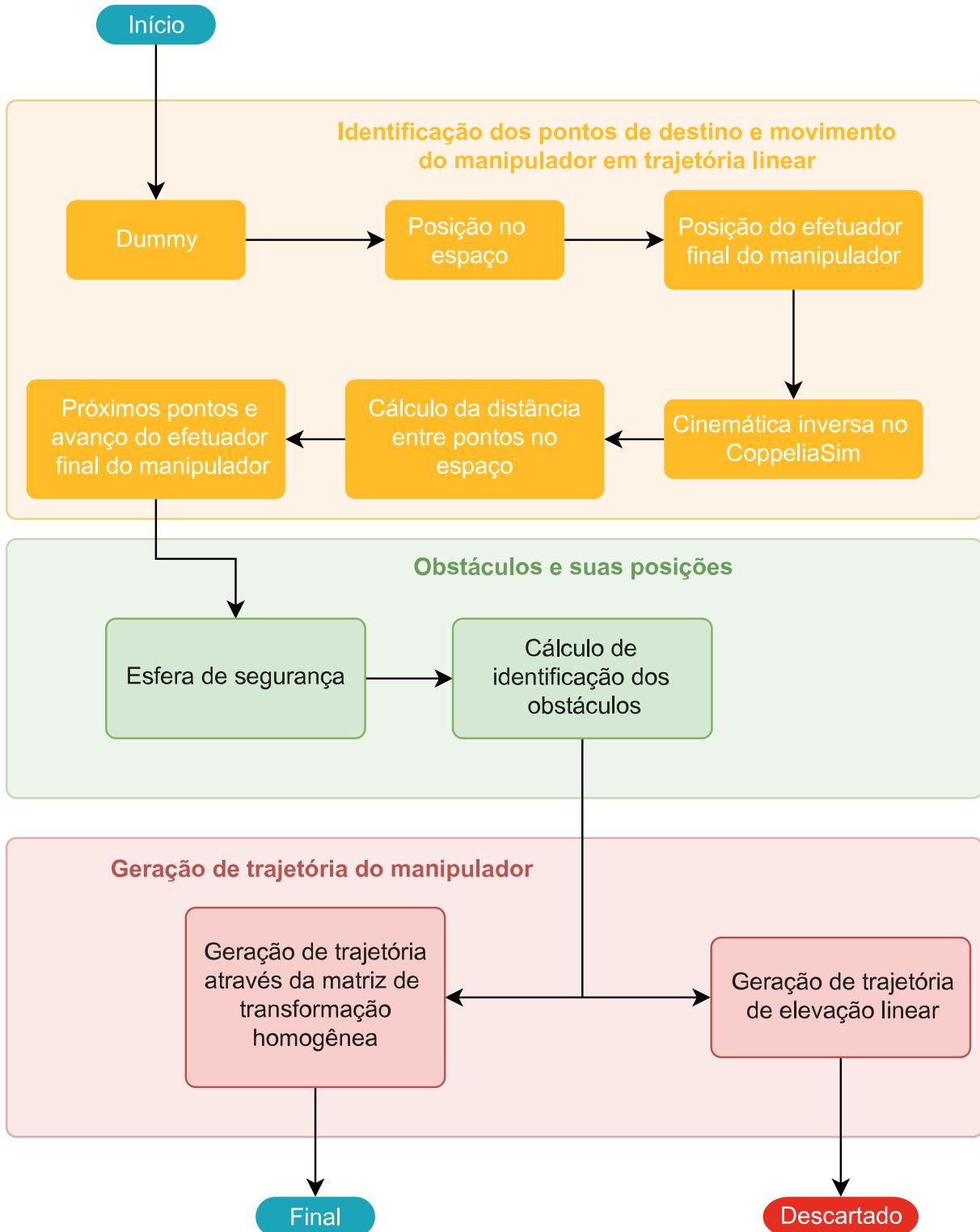


Figura 16 – Fluxograma das etapas realizadas.

### 3.3 Identificação dos pontos de destino e movimento do manipulador em trajetória linear

Nesta seção, é importante saber como identificar os pontos de obstáculos, onde o manipulador se encontra e todas as posições necessárias para que os cálculos possam ser realizados. Esta seção aborda tópicos como Dummy, Posição e orientação no espaço, Posição do efetuador final do manipulador, Cinemática inversa do CoppeliaSim, Cálculo da distância entre pontos no espaço e por fim Próximos pontos e avanço do efetuador final do manipulador.

#### 3.3.1 Dummy

Para dar início na identificação dos pontos no espaço e obter cada posição que o manipulador deve alcançar, é essencial a utilização de uma funcionalidade do CoppeliaSim chamada de *dummy*. O *dummy* como visto na Figura 17, é um ponto no espaço representado em forma de uma esfera o qual disponibiliza valores de posição e de orientação, onde a orientação é representada pelas setas, vermelha (eixo *x*), verde (eixo *y*) e azul (eixo *z*). Para cada eixo um ângulo de orientação pode ser fornecido ao simulador, modificando assim a orientação do *dummy* em relação a um referência fixo.

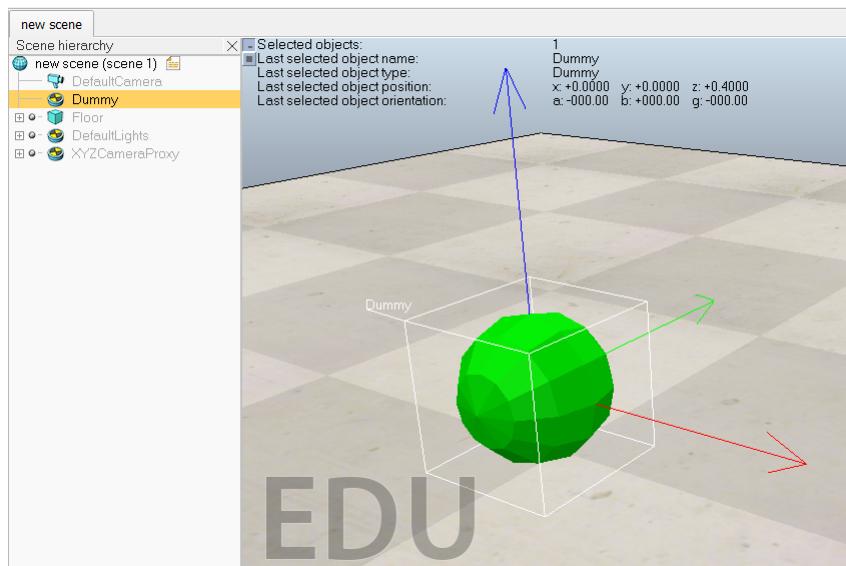


Figura 17 – Dummy no CoppeliaSim.

#### 3.3.2 Posição e orientação no espaço

Como visto anteriormente os *dummies* são utilizados para obter a posição e orientação em um ponto específico. Entretanto, tanto a posição quanto a orientação podem ser modificados conforme a aplicação. Para os testes elaborados neste trabalho foram escolhidos

quatro pontos em cima de uma mesa representados por *dummies* de cores diferentes, o *dummy* de cor verde representa o ponto inicial que o manipulador deverá alcançar para iniciar seu processo de trajetórias, os *dummies* de cor amarela representam os pontos intermediários de passagem do manipulador, por último, o *dummy* de cor vermelha representa o ponto final que o manipulador tem que alcançar para finalizar sua trajetória. Os pontos descritos são representados na Figura 18.

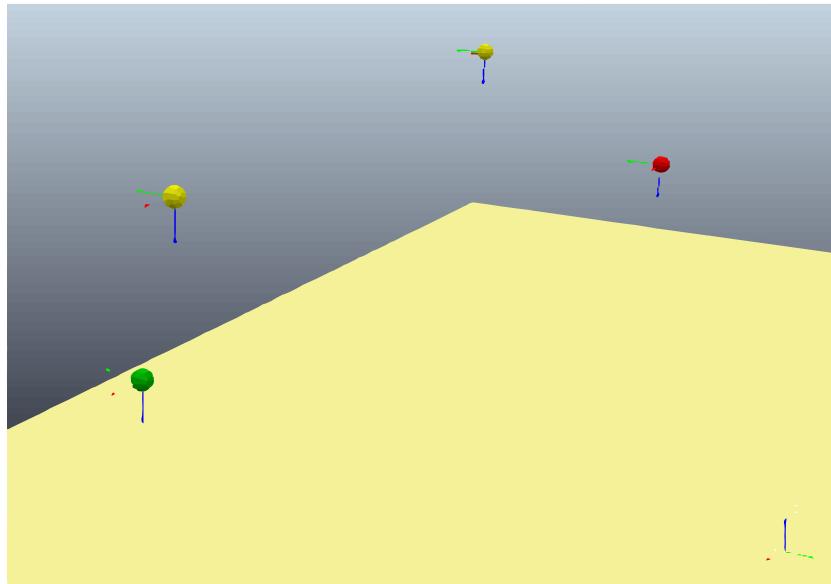


Figura 18 – Pontos no espaço representados por Dummies.

Cada *dummy* retorna sete variáveis, onde as três primeiras são as componentes de um vetor, dado pelas posições  $x$ ,  $y$  e  $z$ , as três variáveis seguintes fornecem a orientação, através dos componentes ângulares alpha ( $\alpha$ ), beta ( $\beta$ ) e gamma ( $\gamma$ ). A última variável é um fator de escala onde seu padrão tem o valor de um (1), e que não será modificado nem utilizado neste trabalho.

Através da Figura 18 percebe-se que os *dummies* estão com a orientação rotacionada 180° no eixo x em relação ao referencial fixo (apresentado no canto direito inferior) do simulador. Contudo, os quatro *dummies* estão com as orientações iguais, de forma que o manipulador ao realizar a trajetória não sofrerá alterações na orientação de seu efetuador final no percurso entre cada par de pontos.

### 3.3.3 Posição do efetuador final do manipulador

O manipulador está acompanhado de um efetuador final, neste trabalho utilizou-se o modelo ROBOTIQ 2F-85 conforme visto na Figura 19, ao realizar testes experimentais foi percebido que este efetuador é o mais adequado para se utilizar nesta simulação. Analisando a Figura 19, percebe-se que existe um *dummy* em sua extremidade, este *dummy* é chamado de *Tip\_1* neste trabalho, juntamente com este *Tip\_1* também existe outro *dummy* chamado de *Target*, inicialmente na mesma posição.

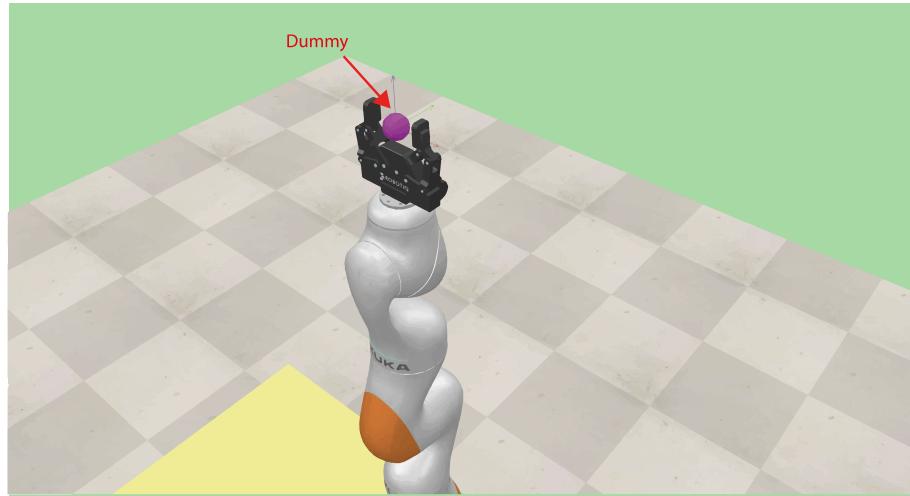


Figura 19 – Efetuador Final ROBOTIQ 2F-85 acoplado no Kuka LBR iiwa 7

Através da Figura 20 nota-se que o *Tip\_1* e *Target* estão anexados ao manipulador robótico em forma de hierarquia, de modo que a função do *Target* é receber valores da próxima posição que o manipulador deve avançar, enquanto o *Tip\_1* tem a função de guiar o manipulador para a posição que o *Target* recebeu, por meio da cinemática inversa.

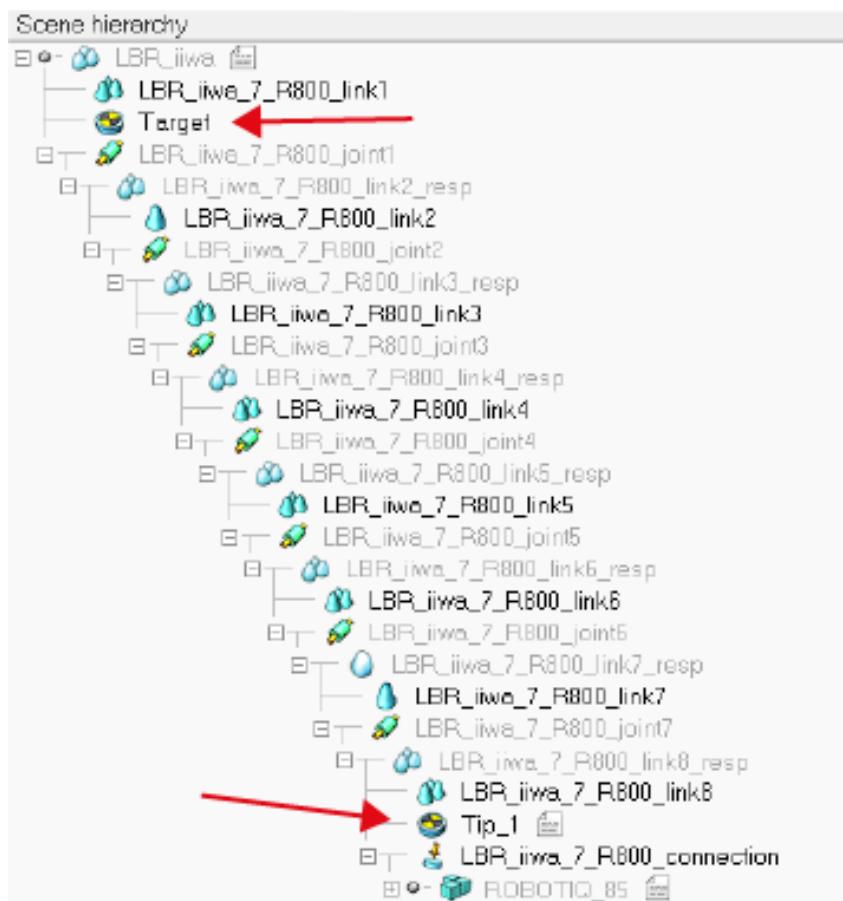


Figura 20 – Hierarquia dos *dummies*, *Tip* e *Target* em relação ao Kuka LBR iiwa 7.

### 3.3.4 Cinemática inversa no CoppeliaSim

Para realizar um movimento com o manipulador é necessário aplicar a cinemática inversa, movendo assim a extremidade do efetuador final até o ponto no espaço desejado. A biblioteca utilizada no simulador é a API simIK contendo todas as tarefas da cinemática.

Para que os cálculos da cinemática inversa sejam realizados, alguns ajustes iniciais são feitos, como a definição constante da velocidade, da aceleração e do jerk para as juntas do manipulador. Para cada constante (velocidade, aceleração e jerk) são passados quatro valores, os três primeiros referindo-se as componentes  $x$ ,  $y$ ,  $z$  e o último refere-se velocidade e aceleração angular do manipulador, sendo assim temos umas constantes em forma de *array* a serem passados para dentro de uma função da biblioteca simIK.

A função utilizada é definida como: **simIK.applyIkEnvironmentToScene()**, essa função realiza os cálculos e aplica os valores da cinemática inversa na tela em que o código está sendo executado, entretanto, essa função deve ser chamada dentro da função **sim.moveToPose()** a qual tem objetivo de mover o manipulador para os pontos calculados, é através dessa função que são passados valores da posição atual, velocidade, aceleração, jerk, próxima posição desejada e consequentemente chamada a função **simIK.applyIkEnvironmentToScene()**, entre outros dados, como as juntas do manipulador que serão movimentadas.

### 3.3.5 Cálculo da distância entre pontos no espaço

A Figura 21 exibe o ponto inicial dado por  $P_i$  e o ponto final dado por  $P_f$ . A terminologia para estes dois pontos são referentes aos cálculos vetoriais e não aos pontos de passagem do manipulador. Com os *dummies* posicionados na tela do simulador podemos traçar um vetor  $\vec{d}$  entre os dois pontos e calcular a distância através de uma subtração entre cada um dos componentes conforme apresentado nas Equações 3.1, 3.2 e 3.3.

$$dx = Pfx - Pix \quad (3.1)$$

$$dy = Pfy - Piy \quad (3.2)$$

$$dz = Pfz - Piz \quad (3.3)$$

Com auxílio das Eq. 3.1, 3.2 e 3.3 também é possível descobrir o sentido em que cada uma das componentes está apontando, caso o resultado seja menor que zero, a componente aponta no sentido negativo do sistema de referências, entretanto se o resultado for maior que zero, a componente está indicada para o sentido positivo do sistema de referências. Se a distância é zero, o manipulador não precisa se mover na componente determinada.

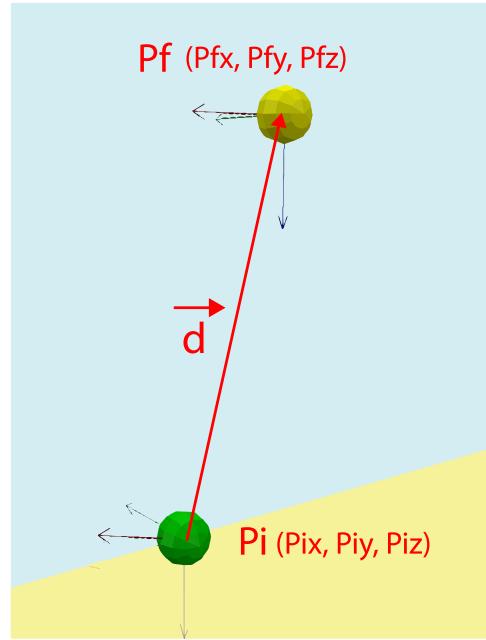


Figura 21 – Vetor distância entre dois pontos.

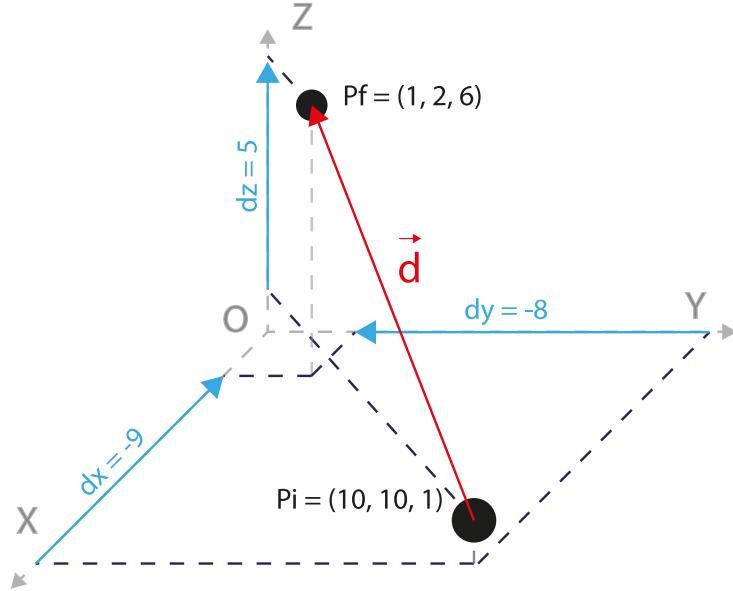
Como exemplo, a Figura 22 apresenta um ponto inicial com coordenadas  $Pi(10, 10, 1)$  e um ponto final  $Pf(1, 2, 6)$ , realizando os cálculos através das Eq. 3.1,3.2 e 3.3, temos os seguintes resultados:

$$\begin{aligned} dx &= 1 - 10 \rightarrow dx = -9 \\ dy &= 2 - 10 \rightarrow dy = -8 \\ dz &= 6 - 1 \rightarrow dz = 5 \end{aligned}$$

Com os resultados obtidos, é possível concluir que as componentes  $dx$  e  $dy$  apontam para o sentido negativo do sistema de referências, o  $dz$  aponta para o sentido positivo do sistema de referências.

### 3.3.6 Próximos pontos e avanço do efetuador final do manipulador

Nas subseções anteriores apresentamos a obtenção dos pontos no espaço, movimentação através da cinemática inversa para o manipulador e também os cálculos da distância entre dois pontos no espaço, mencionados como  $Pi$  e  $Pf$ . Porém, com todas essas informações apresentadas, ainda não é possível conhecer todas as posições que o efetuador final do manipulador irá percorrer entre  $Pi$  e o  $Pf$ . Contudo, uma maneira para resolver este problema é mostrado na Eq. 3.4, a qual divide cada componente do vetor distância em uma quantidade constante de posições ( $Q_p$ ), obtendo assim frações discretas da posição sobre a trajetória. Mediante testes experimentais a quantidade de posições foi definida

Figura 22 – Sentido das componentes do vetor  $\vec{d}$ .

em um valor de 100 posições, podendo ser modificado conforme a necessidade de precisão que o usuário deseje obter em relação à movimentação do manipulador.

$$\vec{F}p_{(Ex,Ey,Ez)} = \frac{dx, dy, dz}{Q_p} \quad (3.4)$$

A fração de posição para cada componente do vetor  $\vec{F}p_{(Ex,Ey,Ez)}$  serve para guiar o manipulador de forma linear entre a posição inicial e a posição final. Entretanto, ainda é necessário realizar o cálculo de multiplicação entre uma matriz de translação pura com o valor das componentes do vetor fração de posição, pela matriz da localização atual do manipulador para descobrir os novos valores de posição que o manipulador deverá seguir. O cálculo é realizado pela Eq. 3.5.

$$\vec{N}_{po} = \vec{T}(Fp_{Ex}, Fp_{Ey}, Fp_{Ez}) \cdot \vec{M}(Ma_x, Ma_y, Ma_z) \quad (3.5)$$

Representado na forma matricial como:

$$\vec{N}_{po} = \begin{bmatrix} 1 & 0 & 0 & Fp_{Ex} \\ 0 & 1 & 0 & Fp_{Ey} \\ 0 & 0 & 1 & Fp_{Ez} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} n_x & o_x & a_x & Ma_x \\ n_y & o_y & a_y & Ma_y \\ n_z & o_z & a_z & Ma_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

$$\vec{N}_{po} = \begin{bmatrix} n_x & o_x & a_x & Fp_{Ex} + Ma_x \\ n_y & o_y & a_y & Fp_{Ey} + Ma_y \\ n_z & o_z & a_z & Fp_{Ez} + Ma_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ao saber que a multiplicação do ponto atual é realizada por uma translação pura ( $\vec{T}$ ), a orientação do efetuador final do manipulador ( $\vec{M}$ ) não sofre alterações, dessa forma os cálculos matriciais apresentados na Eq. 3.6 são resumidos a uma soma do vetor de fração de posição com o vetor de posição atual do manipulador, criando assim um novo vetor de posição ( $\vec{N}_{po}$ ) com componentes  $x$ ,  $y$  e  $z$ . (Eq. 3.7)

$$\vec{N}_{po} = \vec{F}p + \vec{M}a \quad (3.7)$$

Com as componentes do novo vetor  $\vec{N}_{po}$  é possível mover o efetuador final do manipulador para estes pontos calculados, passando as componentes do vetor como parâmetro para a função de cinemática inversa **sim.moveToPose()**.

## 3.4 Obstáculos e suas posições

Os obstáculos, como mencionado anteriormente, são tratados como partes do corpo de um ser humano, no caso, um operador que está trabalhando juntamente com o manipulador industrial. Neste trabalho iremos utilizar um boneco em 3D dentro do CoppeliaSim para representar a pessoa que está trabalhando no mesmo espaço do manipulador. Entretanto partes do corpo da pessoa devem ser identificadas no espaço e referidos como obstáculos, na Figura 23 apresentamos quatro pontos da cor vermelha que o manipulador deve evitar ao estar realizando sua trajetória linear, de modo que quando identificados estes pontos, o manipulador seja capaz de gerar uma nova trajetória evitando a colisão. Cada um dos pontos é representado por *dummies* acoplados ao corpo do operador para a identificação das coordenadas de um ponto no espaço, os valores de orientação desses pontos não são utilizados.

### 3.4.1 Esfera de segurança

Neste etapa, é importante destacar que o fato de conhecer apenas as posições dos pontos dos obstáculos não é suficiente no processo de evasão, dessa maneira, precisamos ainda criar uma distância segura em que o manipulador consiga operar longe dos obstáculos reais. Pensando nisso, criamos esferas virtuais (Figura 24) ao redor de cada um dos pontos destacados como obstáculo. Por meio de testes experimentais o raio da esfera foi determinado em 15 centímetros (cm), entretanto, este valor pode ser modificado dependendo de quais pontos são escolhidos como obstáculos, por exemplo se o obstáculo for a cabeça do operador, o raio de segurança deve ser maior do que 15cm. Se o robô industrial utilizado for do modelo da ABB IRB 4600, um robô de grande porte, o obstáculo pode ser considerado como uma pessoa inteira e não mais um braço ou a cabeça do operador, de modo que neste caso o raio de segurança pode variar conforme o tamanho da pessoa.

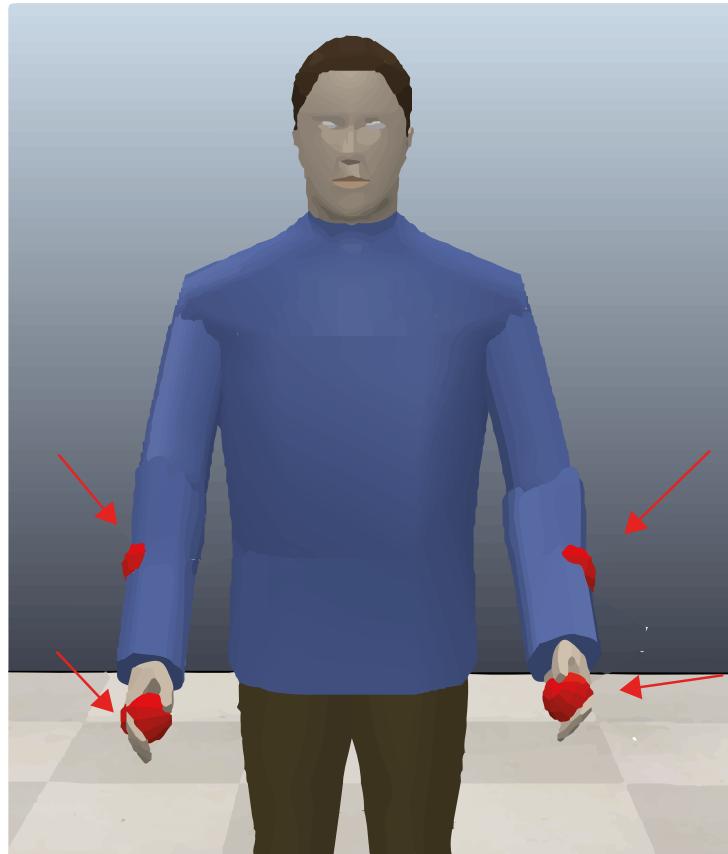


Figura 23 – Posição dos pontos de obstáculos no operador.

### 3.4.2 Cálculo de identificação dos obstáculos

O cálculo da identificação de possíveis colisões contra os obstáculos é realizado através da utilização de vetores. A detecção dos obstáculos é realizada desde a perspectiva do próprio obstáculo, facilitando a verificação das condições de colisão.

Tanto a posição dos obstáculos quanto a posição do efetuador final do manipulador são conhecidos. Um vetor iniciando no obstáculo é traçado apontando para a posição do manipulador, criando um vetor  $\vec{S}$  (Figura 25). O vetor resultante é uma subtração da posição do manipulador ( $Ma_{(x,y,z)}$ ) em relação a posição do obstáculo ( $Ob_{(x,y,z)}$ ), conforme visto na Equação 3.8.

$$\vec{S} = Ma_{(x,y,z)} - Ob_{(x,y,z)} \quad (3.8)$$

Uma vez realizado o cálculo para encontrar o valor das componentes do vetor  $\vec{S}$ , é possível comparar com o vetor  $\vec{R}$ , representando o raio da esfera de segurança. Quando todas as componentes do vetor  $\vec{S}$  são menores ou iguais aos valores das componentes do vetor  $\vec{R}$ , é identificado que o manipulador está próximo de colidir com o operador e uma mudança na trajetória do manipulador deve ser executada.

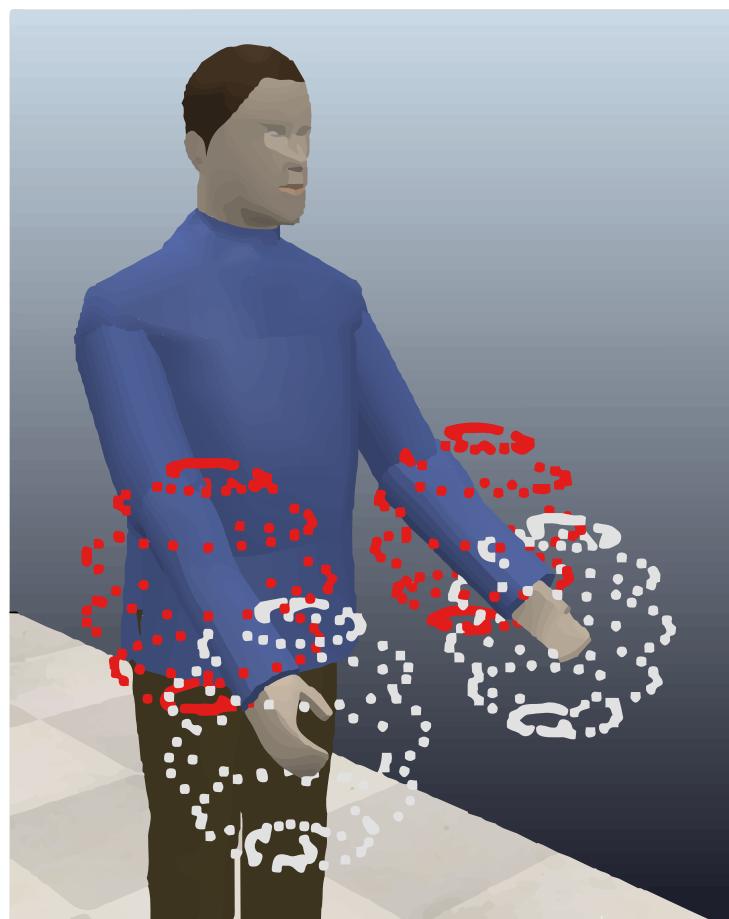


Figura 24 – Esfera virtual de segurança.

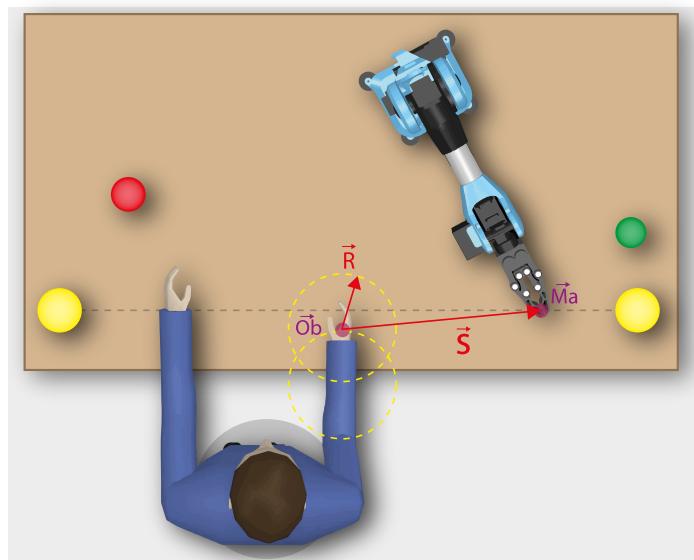


Figura 25 – Identificação de obstáculo através de vetor com vista superior.

## 3.5 Geração de trajetória do manipulador

Uma ação deve ser realizada para que o manipulador não colida com o obstáculo enquanto percorre sua trajetória linear. Esta seção aborda dois tipos de geração de trajetórias, o primeiro deles é uma trajetória simples de elevação do manipulador, o segundo tipo de geração, é o modelo com maior eficiência que também é capaz de criar a melhor trajetória para que o efetuador desvie dos obstáculos.

### 3.5.1 Geração de trajetória de elevação linear

A primeira abordagem efetuada quando o manipulador chega próximo do obstáculo, é a realização de uma elevação no efetuador passando por cima do obstáculo com uma trajetória linear conforme apresentado na Figura 26. É uma abordagem funcional utilizando apenas 1 obstáculo estático, quando o número de obstáculos aumenta e o obstáculo torna-se dinâmico, este tipo de solução começa a apresentar inconsistências na sua eficiência como a colisão com algum dos obstáculos ou não conseguindo mais encontrar o ponto de destino inicialmente estipulado.

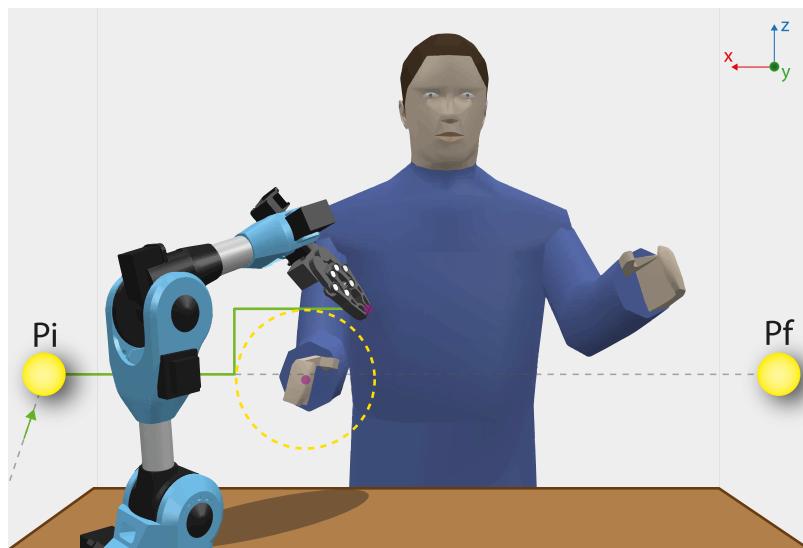


Figura 26 – Geração de trajetória superior linear.

### 3.5.2 Geração de trajetórias através da matriz de transformação homogênea

A utilização da matriz de transformação homogênea nessa aplicação é de grande importância, pois seu objetivo é gerar novas posições ao redor do obstáculo fazendo com que o manipulador siga essas posições e desvie do operador, criando assim uma nova trajetória como visto na Figura 27. Essa abordagem tem um grande potencial para desviar dinamicamente de obstáculos em movimento, buscar novas trajetórias para alcançar o ponto de

destino e afastar o máximo o manipulador do operador, mesmo quando o efetuador está no limite da sua área de trabalho.

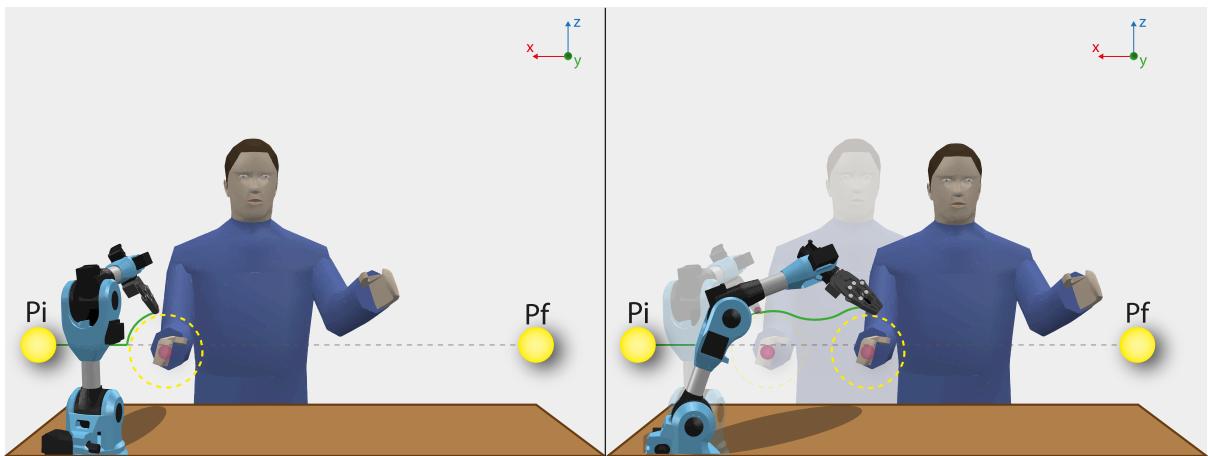


Figura 27 – Criação de uma nova posição através da matriz homogênea.

A matriz de transformação homogênea é o resultado do somatório de duas matrizes homogêneas, as quais são rotacionadas em relação a um eixo  $y$  e outra em relação a um eixo  $z$ .

Aqui o *Matlab* é utilizado para auxiliar na verificação dos cálculos das matrizes.

A primeira matriz a ser verificada é a matriz de transformação homogênea rotacionada no eixo  $y$ , a qual foi descrita pela Eq. 2.3. Os componentes de translação dessa matriz,  $O_x, O_y$  e  $O_z$  são nulos devido a estarmos interessados apenas na rotação em torno de um eixo. Para realizar essa rotação alguns parâmetros precisam ser descritos, como o ângulo inicial e final de quanto a matriz deve rotacionar, entretanto, como a rotação precisa começar de um lado do obstáculo e chegar até o outro lado, um valor de ângulo inicial e final condizente, são os ângulos de  $0^\circ$  e  $180^\circ$ .

O código abaixo foi utilizado no *Matlab* para realizar as devidas validações com os parâmetros mencionados anteriormente.

```

1 for w = 0: 3: 180
2     % Posicao de referencia de rotacao
3     tx = 2;
4     ty = 0;
5     tz = 0;
6     P = [tx;ty;tz;1];
7     Ry = [ cosd(w) 0 -sind(w) 0; 0 1 0 0; sind(w) 0 cosd(w) 0; 0 0 0 1];
8     Rot = Ry*P;
9     X = Rot(1,1);
10    Y = Rot(2,1);
11    Z = Rot(3,1);
12    plot3(X,Y,Z, '-or');
13 end

```

Ao executar o código no *Matlab* obtemos o resultado apresentado na Figura 28. Para essa validação é efetuado uma multiplicação da matriz  $y$  por um vetor posição dado pelos valores  $\vec{V_p} = [2, 0, 0]$ , esse vetor posição é o ponto de onde a rotação é iniciada. Outro detalhe destacado é em torno de qual ponto a matriz rotaciona, sendo que na validação do *MATLAB*, esse ponto é o próprio ponto central do sistema de coordenadas, neste caso sendo  $(0, 0, 0)$ , porém pode ser facilmente alterado para a posição desejada, conforme iremos ver mais adiante com a aplicação no CoppeliaSim.

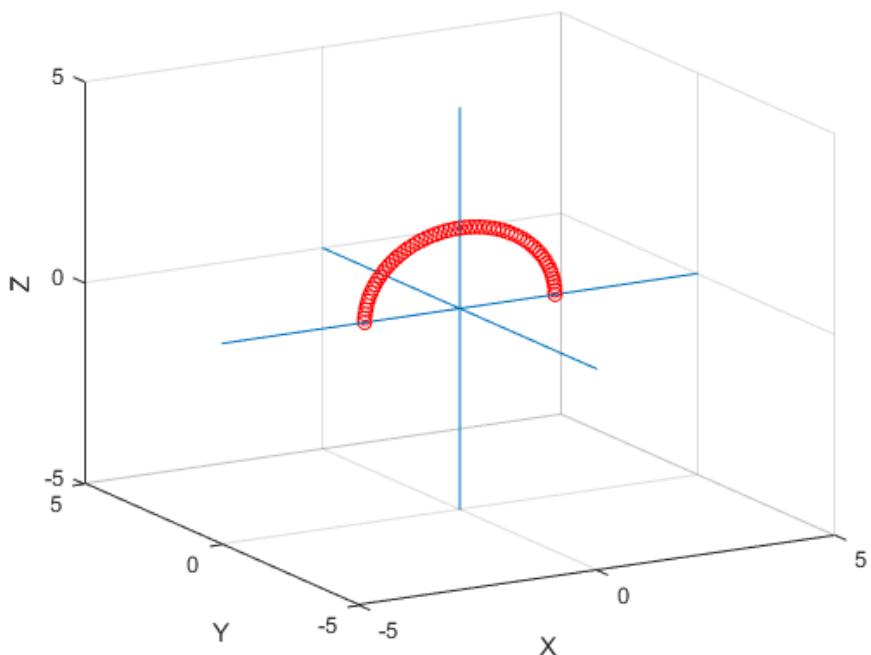


Figura 28 – Resultado da matriz homogênea em Y.

Os mesmos conceitos utilizados para a matriz homogênea em  $y$  são utilizados para a matriz em torno do eixo  $z$ , sendo que neste caso será usada a Eq. 2.4. O resultado por sua vez, é semelhante com os obtidos na rotação sobre o eixo  $x$  no caso anterior, a única diferença é que os pontos agora foram construídos ao redor do eixo  $z$  conforme exibido na Figura 29

Para o objetivo de uma nova trajetória que desvie completamente do operador, é realizado o somatório das duas matrizes, resultando em uma matriz mista, onde é expressada pela Equação 3.9. Este somatório é realizado quando se deseja que a trajetória resultante seja uma trajetória simétrica tanto no plano  $xy$  quanto no plano  $xz$ . Importante salientar que o fator de escala ficou duplicado, entretanto, este valor pode ser descartado pois não iremos utilizar neste trabalho.

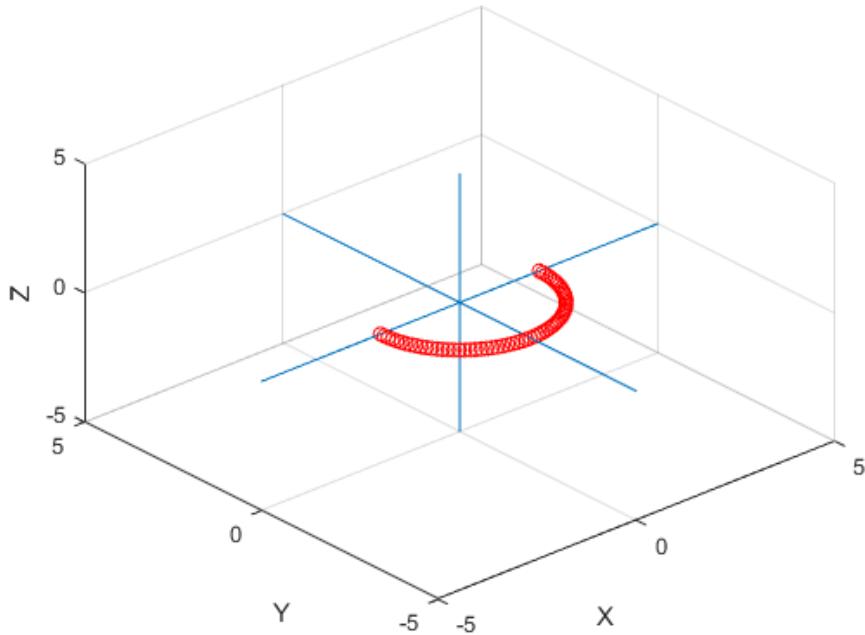


Figura 29 – Resultado da matriz homogênea em Z.

$$R_H = \begin{bmatrix} \cos \theta & 0 & -\sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 * \cos \theta & -\sin \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta + 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta + 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad (3.9)$$

Posterior ao cálculo matricial, devemos efetuar uma multiplicação entre a matriz  $R_H$  e um vetor de posição mostrado na Eq. 3.10, o que resulta em um novo vetor de posição  $\vec{NewP}$ . As componentes  $t_x$ ,  $t_y$  e  $t_z$  são valores que representam o ponto inicial de rotação, para o caso prático no simulador, estes elementos são os pontos atuais do manipulador.

$$\vec{NewP} = \begin{bmatrix} 2 * \cos \theta & -\sin \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta + 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta + 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \times \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 2 * t_x * \cos \theta - t_y * \sin \theta - t_z * \sin \theta \\ t_y * (\cos \theta + 1) + t_x * \sin \theta \\ t_z * (\cos \theta + 1) + t_x * \sin \theta \\ 2 \end{bmatrix} \quad (3.10)$$

Os cálculos mencionados acima são aplicados no *Matlab* que por sua vez resulta na Figura 30. Os pontos apresentados na cor azul são referentes aos resultados da matriz  $y$  e  $z$ , os pontos destacados pela cor vermelha são resultados do vetor posição  $\vec{NewP}$ .

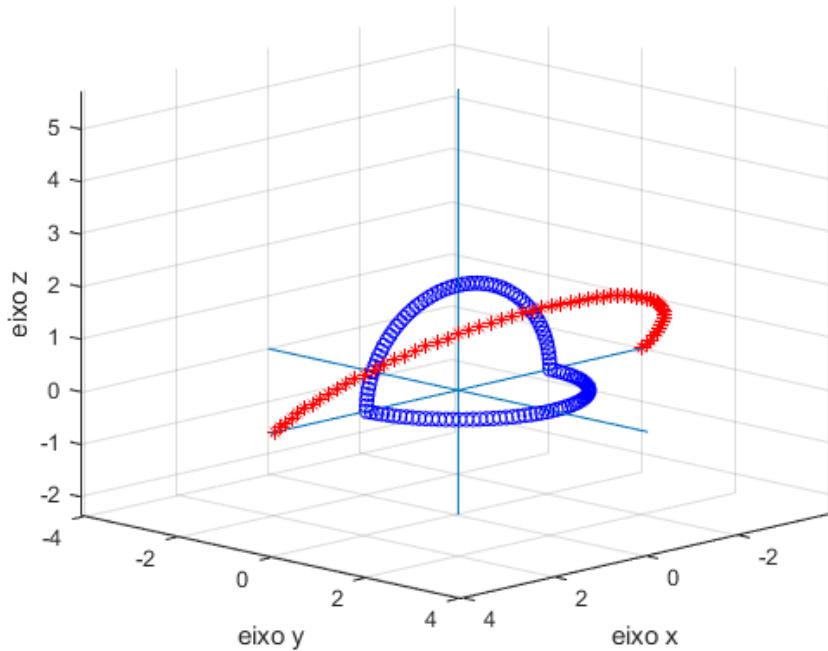


Figura 30 – Resultado da soma de matrizes.

Quando utilizada a matriz resultante da Eq. 3.10 no simulador CoppeliaSim, a posição de referência para gerar a rotação será o próprio obstáculo. O cálculo para que isso seja possível é a soma entre o vetor  $\vec{Ob}$  com o vetor  $\vec{NewP}$ , resultando na Eq. 3.11.

$$\vec{NewP}_{ob} = \begin{bmatrix} 2 * t_x * \cos \theta - t_y * \sin \theta - t_z * \sin \theta + Obx \\ t_y * (\cos \theta + 1) + t_x * \sin \theta + Oby \\ t_z * (\cos \theta + 1) + t_x * \sin \theta + Obz \\ 2 \end{bmatrix} \quad (3.11)$$

Ao realizar testes experimentais dois parâmetros foram adicionados a Equação 3.11, sendo eles o *Kappa*  $\kappa$  e o *Epsilon*  $\varepsilon$ .

O parâmetro  $\kappa$  serve para fazer um recuo do manipulador através do eixo  $x$  quando o manipulador é identificado próximo do obstáculo, esse recuo garante um afastamento da esfera de segurança, garantindo assim uma proteção maior para que não haja colisões entre o manipulador e o operário, ou seja,  $\kappa$  é um fator de segurança. Através de testes experimentais o valor adequado de  $\kappa$  é de 0.35.

Para o parâmetro  $\varepsilon$  os valores são definidos como -1 ou +1, devido este parâmetro ter como objetivo alterar o sentido de rotação no eixo  $y$ ,  $\varepsilon$  permite trajetórias por cima do obstáculo quando é fixado com +1, ou por baixo do obstáculo quando definido como -1.

A Equação 3.12 apresenta a matriz homogênea final com todos os parâmetros necessários, a qual é utilizada para criação de novos pontos de rotação, os quais são passados

para a cinemática inversa fazendo com que o manipulador siga estes pontos criando assim uma nova trajetória. Através de testes experimentais essa matriz também sofreu alguns ajustes nos sinais de suas componentes para que seja possível rotacionar corretamente em torno do obstáculo.

$$\vec{NewP_{ob}} = \begin{bmatrix} t_x * \cos \theta + t_x * \cos \theta * \kappa - t_y * \sin \theta - t_z * \sin \theta + Ob_x \\ t_y * (\cos \theta) + t_x * \sin \theta * \varepsilon + Ob_y \\ t_z * (\cos \theta) - t_x * \sin \theta + Ob_z \end{bmatrix} \quad (3.12)$$

Quando identificado que o manipulador está próximo do operador, a matriz homogênea é aplicada com seus parâmetros pré-definidos, entretanto, algumas informações precisam ser conhecidas para que a ação de evasão do manipulador possa rotacionar com o melhor caminho ao redor do obstáculo detectado.

Através do resultado da Eq. 3.8 é fornecido o valor da posição das componentes no eixo  $x$ ,  $y$  e  $z$  do vetor  $\vec{S}$ . Para que a posição dessas componentes sejam identificadas no espaço, fixamos um sistema de coordenadas no ponto de obstáculo, este sistema tem o sentido positivo de  $x$  apontando para o lado esquerdo, sentido positivo de  $y$  apontando para fora do plano e o sentido positivo de  $z$  apontando para cima, como apresentado na Figura 31.

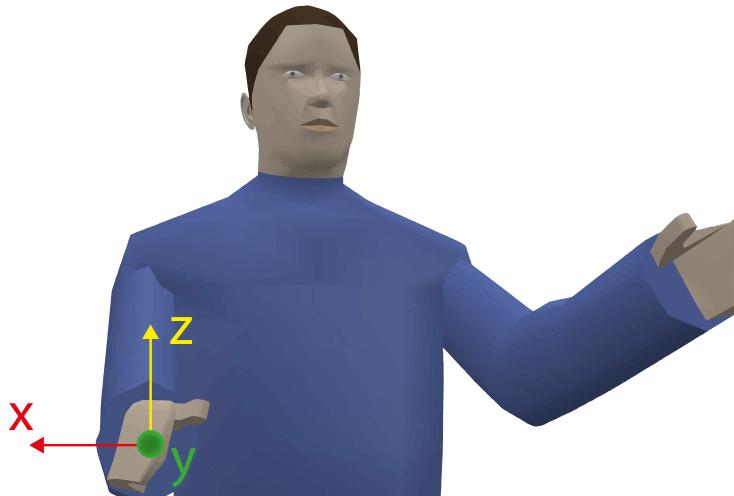


Figura 31 – Sentido positivo do Sistema de coordenadas fixado no obstáculo.

Para a componente  $S_x$  existem valores positivos e valores negativos. Através destes valores conseguimos identificar se o manipulador se aproxima pela direita ou pela esquerda do obstáculo. Quando a componente  $S_x$  é positiva, o manipulador se aproxima pelo lado esquerdo, quando a componente  $S_x$  é negativa, o manipulador se aproxima pelo lado direito, a Figura 32 apresenta os dois casos citados.

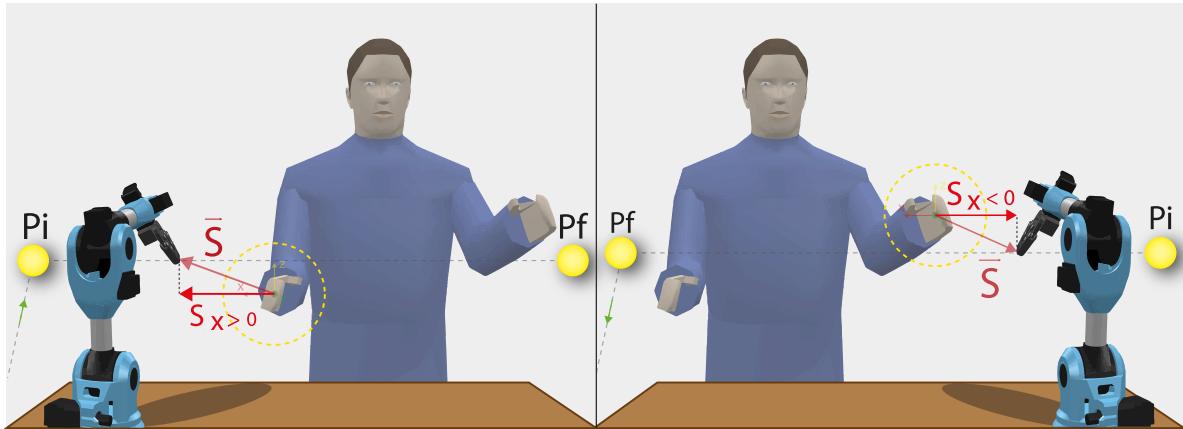


Figura 32 – Sentido  $x$  da aproximação do manipulador em relação ao obstáculo.

A componente  $S_z$  tem um comportamento parecido com a componente  $S_x$ , obtendo valor tanto positivo quanto negativo. Com estes valores podemos identificar se o manipulador esta se aproximando com uma altura maior do que onde o obstáculo está localizado, ou se o manipulador se aproxima com uma altura menor do que o obstáculo. Quando a componente  $S_z$  é positiva o manipulador é mais alto que o obstáculo, entretanto, quando o  $S_z$  é negativo o manipulador está se aproximando com uma altura menor que o obstáculo. A Figura 33 apresenta as abordagens citadas.

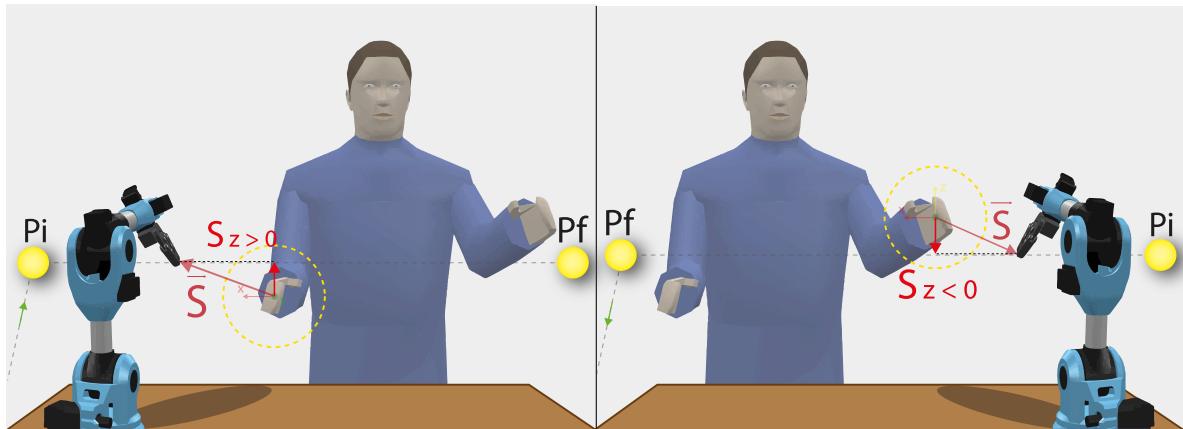


Figura 33 – Sentido  $z$  da aproximação do manipulador em relação ao obstáculo.

No CoppeliaSim os cálculos são realizados em radianos de maneira que conseguimos variar o ângulo  $\theta$  entre  $0$  e  $+\pi$ , ou entre  $0$  e  $-\pi$ . Ao conhecer por onde o manipulador se aproxima do obstáculo através das componentes  $S_x$  e  $S_z$  uma ação pode ser tomada para gerar uma nova trajetória. Essa trajetória é definida em 4 diferentes métodos de rotação.

O primeiro método, é quando o manipulador se aproxima pelo lado esquerdo ( $S_x > 0$ ) com sua altura mais alta ( $S_z > 0$ ) do que o obstáculo. A ação a ser realizada é de uma rotação no sentido horário por cima do obstáculo (Figura 34.a) Para realizar essa ação, a variação de  $\theta$  deve ser de  $0$  até  $-\pi$ , e o valor de  $\varepsilon$  é de  $-1$ .

O segundo método, é quando o manipulador se aproxima pelo lado direito ( $S_x < 0$ )

do obstáculo, porém agora com sua altura mais baixa ( $S_z < 0$ ) do que a do obstáculo. A ação a ser realizada é de uma rotação no sentido horário por baixo (Figura 34.b). Por sua vez, para realizar essa ação, a variação de  $\theta$  deve ser de 0 até  $-\pi$ , e o valor de  $\varepsilon$  é de +1.

O terceiro método, é quando o manipulador se aproxima novamente pelo lado esquerdo ( $S_x > 0$ ) do obstáculo, porém agora com sua altura mais baixa ( $S_z < 0$ ) do que a do obstáculo. A ação a ser realizada é de uma rotação no sentido anti-horário por baixo (Figura 34.c). Por sua vez, para realizar essa ação, a variação de  $\theta$  deve ser de 0 até  $+\pi$ , e o valor de  $\varepsilon$  é de +1.

Por último mas não menos importante, o quarto método, é quando o manipulador se aproxima novamente pelo lado direito ( $S_x < 0$ ) do obstáculo, porém agora com sua altura mais alta ( $S_z > 0$ ) do que a do obstáculo. A ação a ser realizada é de uma rotação no sentido anti-horário por cima (Figura 34.d). Por sua vez, para realizar essa ação, a variação de  $\theta$  deve ser de 0 até  $+\pi$ , e o valor de  $\varepsilon$  é de -1.

Através da Tabela 4 é gerado uma simplificação e melhor visualização dos 4 diferentes métodos citados, podendo facilmente ser aplicado a simulação.

Tabela 4 – Sentido de giro através dos parâmetros de ângulo e  $\varepsilon$ .

Ação de rotação	$\hat{\text{Ângulo}}$	$\varepsilon$	$S_x$	$S_z$
Sentido horário por cima	$-\pi$	-1	$S_x > 0$	$S_z > 0$
Sentido horário por baixo	$-\pi$	1	$S_x < 0$	$S_z < 0$
Sentido anti-horário por baixo	$\pi$	1	$S_x > 0$	$S_z < 0$
Sentido anti-horário por cima	$\pi$	-1	$S_x < 0$	$S_z > 0$

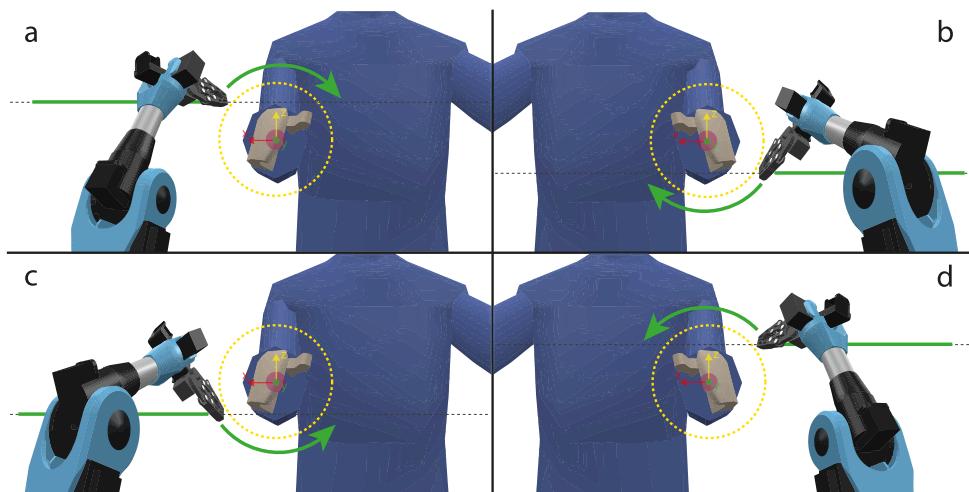


Figura 34 – Ações de rotação através dos parâmetros de ângulo e  $\varepsilon$ .

Por fim a vantagem da matriz homogênea apresentada na Eq. 3.12 é dada pela facilidade de mudar o sentido dos novos pontos no espaço. Na validação apresentada acima, é perceptível que através do  $\varepsilon$  e também dos valores de variação de  $\theta$  conseguimos fazer

com que os pontos rotacionem tanto por baixo quanto por cima do obstáculo, possuindo como referência a posição  $S_z$  de aproximação do manipulador.

Outra vantagem a ser destacada é que os pontos são criados em torno de um referencial, este referencial na prática é dado pelo próprio obstáculo (mão ou/e antebraço do operador), quando o obstáculo realiza movimentos independente da direção ou sentido, os pontos que são criados através da matriz homogênea acompanham o ponto de referência que é atualizado a cada ciclo do código, fazendo com que o manipulador acompanhe estes pontos e não se aproxime do trabalhador.

# 4 Resultados

Na presente seção, são apresentados os principais resultados obtidos após a implementação da metodologia proposta para a geração automática de trajetórias para um manipulador industrial em presença de obstáculos. Nas simulações realizadas, desejamos avaliar principalmente se a aplicação da matriz homogênea é capaz de desviar do operador com uma distância segura. Além de efetuar o desvio dinâmicamente ao mesmo tempo que o obstáculo se move, ou quando encontra mais pontos de obstáculos enquanto o manipulador está desviando do primeiro obstáculo.

Para melhor visualização, o efetuador final cria uma linha na cor verde por onde realiza sua trajetória, de maneira que deixa seu rastro e conseguimos analisar todos os pontos de passagem durante seu percurso.

## 4.1 Simulação com o manipulador Kuka LBR iiwa 7

### 4.1.1 Geração de trajetórias com obstáculos estáticos

Esta simulação consiste em deixar os dois braços do operador de modo estático no trajeto em que o manipulador está realizando, os braços ficam entre os dois pontos intermediários de passagem. O manipulador inicia sua trajetória de modo linear entre os pontos, e ao encontrar o primeiro obstáculo, identificado como a mão do operador, o manipulador é capaz de realizar a rotação no sentido horário passando assim por cima da mão do operador e ao mesmo tempo se afastando, como pode ser observado através da Figura 35.

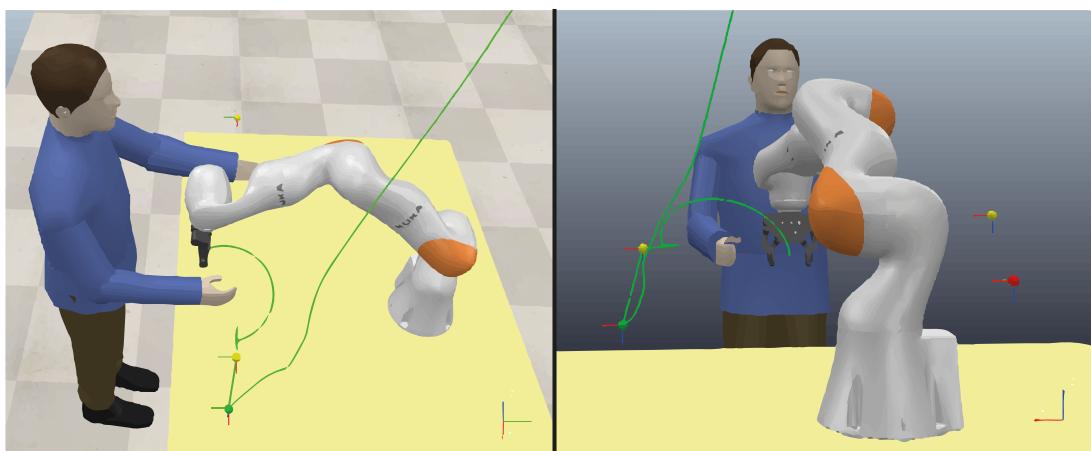


Figura 35 – Nova trajetória com braço do operador estático.

Entretanto, existem dois obstáculos sendo o segundo a mão esquerda do operador. Quando o manipulador conclui o primeiro desvio, a ponta do manipulador fica mais baixo

que o segundo obstáculo de forma que sua rotação é gerada no sentido anti-horário por baixo. O resultado relacionado aos dois desvios são apresentadas na Figura 36.

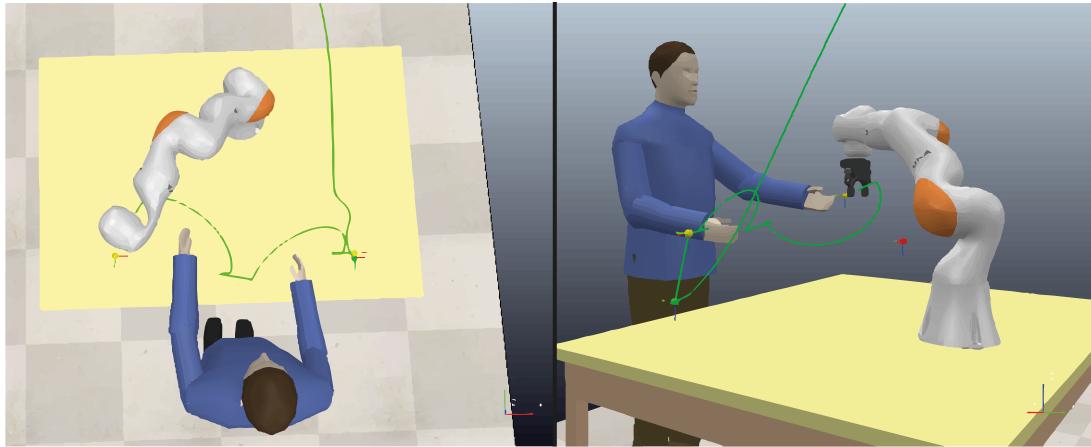


Figura 36 – Realização de novas trajetórias com obstáculos em posições diferentes.

#### 4.1.2 Manipulador desviando de dois pontos estáticos com esferas de segurança concatenadas

A Figura 37 exibe o antebraço direito do operador no caminho da trajetória do manipulador, o qual é identificado e gerado uma nova trajetória para não haver colisões, entretanto, a trajetória em forma de rotação acontece em torno do ponto localizado no antebraço, de maneira que o manipulador começa a se aproximar da mão do operador, o que não é o ideal, contudo, para que não exista a colisão com qualquer parte do braço, o manipulador inicia uma nova rotação agora referente à mão, conseguindo assim desviar de dois pontos sequencialmente. Ou seja, o braço do operador fica livre de colisões.

#### 4.1.3 Manipulador desviando dinâmicamente do operador

As duas abordagens anteriores comprovam que o manipulador é capaz de criar uma nova trajetória e desviar do operador, contudo, as simulações foram realizadas com o operador estático, sem nenhum movimento de seus braços.

Para validar que a metodologia proposta tem êxito com o operador em movimento, realizamos a translação do braço direito enquanto o manipulador estava desviando (como pode ser visto através da Figura 38), percebendo assim que a distância que o manipulador percorre em sua nova trajetória é muito maior do que as apresentadas na Figura 35. Através de sua nova trajetória, o manipulador é capaz de seguir os movimentos do trabalhador em todos os sentidos, evitando que haja colisões no ambiente de trabalho.

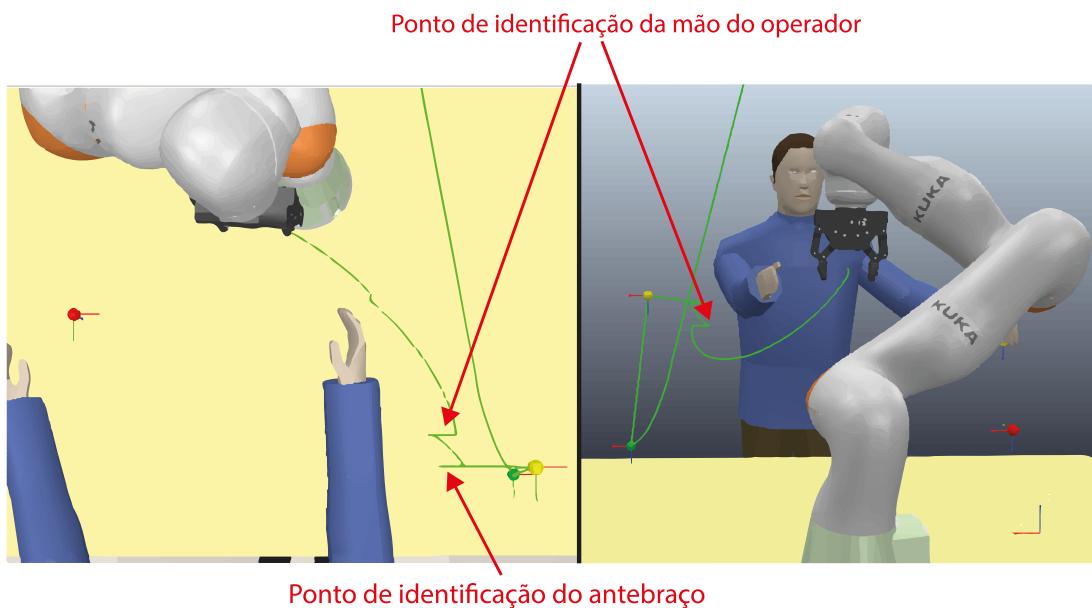


Figura 37 – Nova trajetória com identificação de dois obstáculos simultâneos.

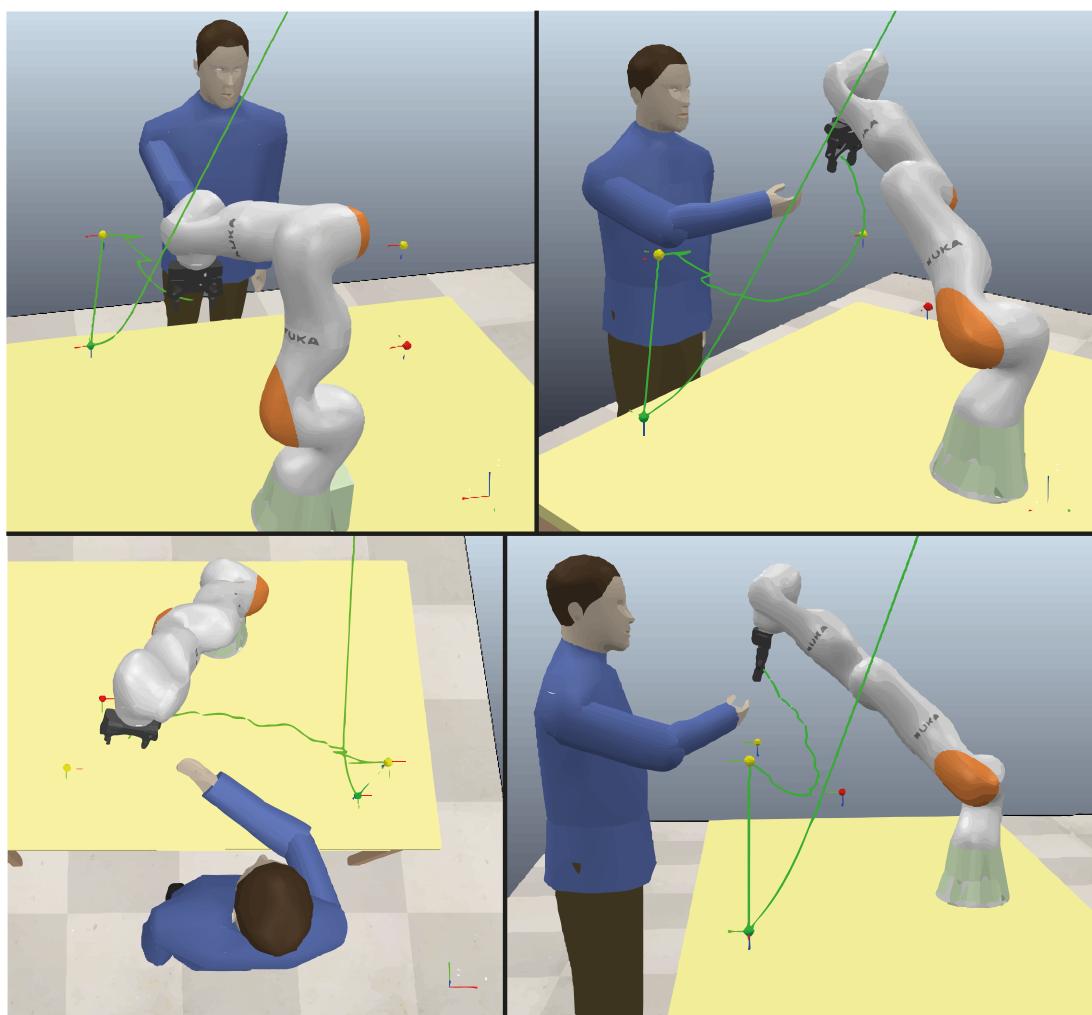


Figura 38 – Criação de trajetória com obstáculo dinâmico.

## 4.2 Simulação com o manipulador ABB IRB 4600

A metodologia proposta funcionou perfeitamente para o manipulador Kuka LBR iiwa 7, entretanto, analisando o desenvolvimento da metodologia é notável que a aplicação pode ser implementada para outros tipos de manipuladores com alguns pequenos ajustes.

Nessa seção do trabalho as simulações foram expandidas para utilização do manipulador ABB IRB 4600 o qual tem 6 juntas, uma a menos que o manipulador anterior. O parâmetro da quantidade de juntas no simulador é ajustado para se adequar a cada um dos manipuladores. O ponto de referência do obstáculo para este caso também é modificado e realocado na altura da cintura do operador. Através de testes experimentais os valores de raio da esfera de segurança são ajustados para um valor de 70 cm que é capaz de envolver todo o corpo do operador. Outro parâmetro ajustado foi a variável  $\kappa$ , essa variável foi ajustado para um valor de 1,3, o qual foi escolhido após realizar testes experimentais.

A Figura 39 apresenta o manipulador desviando do operador através da nova trajetória. O desvio aconteceu no sentido horário e por cima.

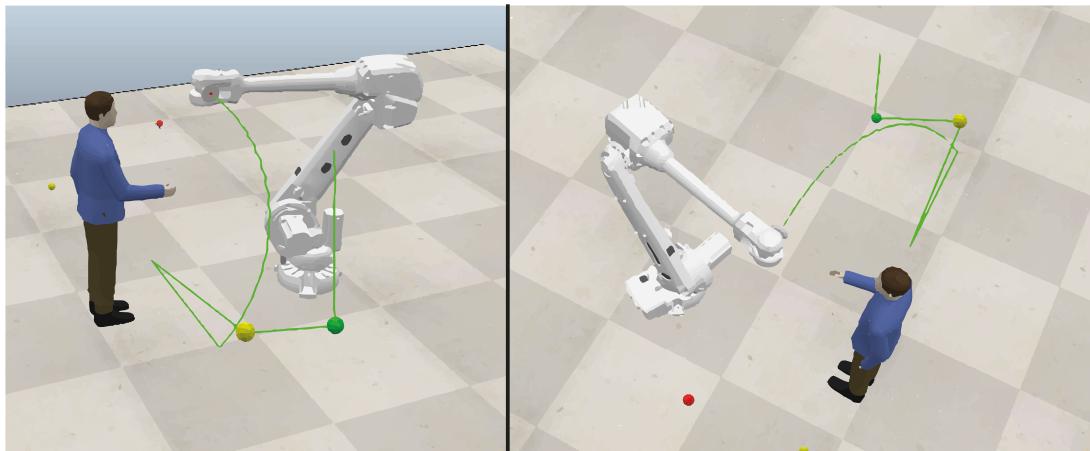


Figura 39 – Aplicação da metodologia usando um robô ABB IRB 4600.

O resultado anterior apresenta o manipulador se aproximando da esquerda para a direita olhando do ponto de vista do robô para o operador. A Figura 40, mostra o manipulador se aproximando pelo outro lado e realizando a nova trajetória. Essa nova trajetória é realizada no sentido horário porém por baixo, devido o manipulador estar mais baixo do que o ponto de referência do obstáculo.

A geração de novas trajetórias permite que todas as aplicações apresentadas para o robô Kuka possam ser utilizadas também neste robô industrial (ABB IRB 4600). A característica do manipulador de conseguir acompanhar dinamicamente o operador também é validada. Na Figura 41 percebe-se que a trajetória criada pelo manipulador foi diferente das vistas anteriormente devido que o operador se moveu enquanto o manipulador

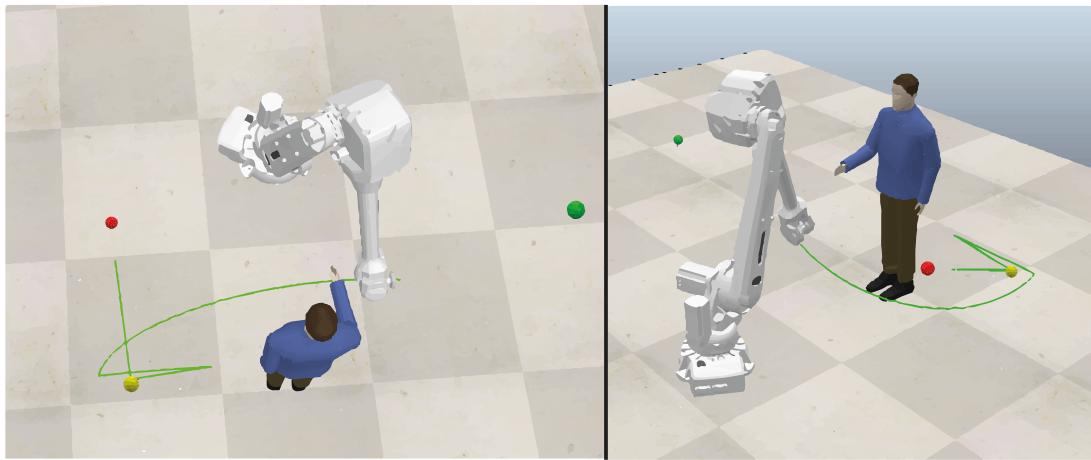


Figura 40 – Comprovação da aplicação da metodologia por outro sentido de identificação.

realizava sua nova trajetória de desvio, assim o manipulador acompanhou a dinâmica do sistema evitando a colisão com o trabalhador.

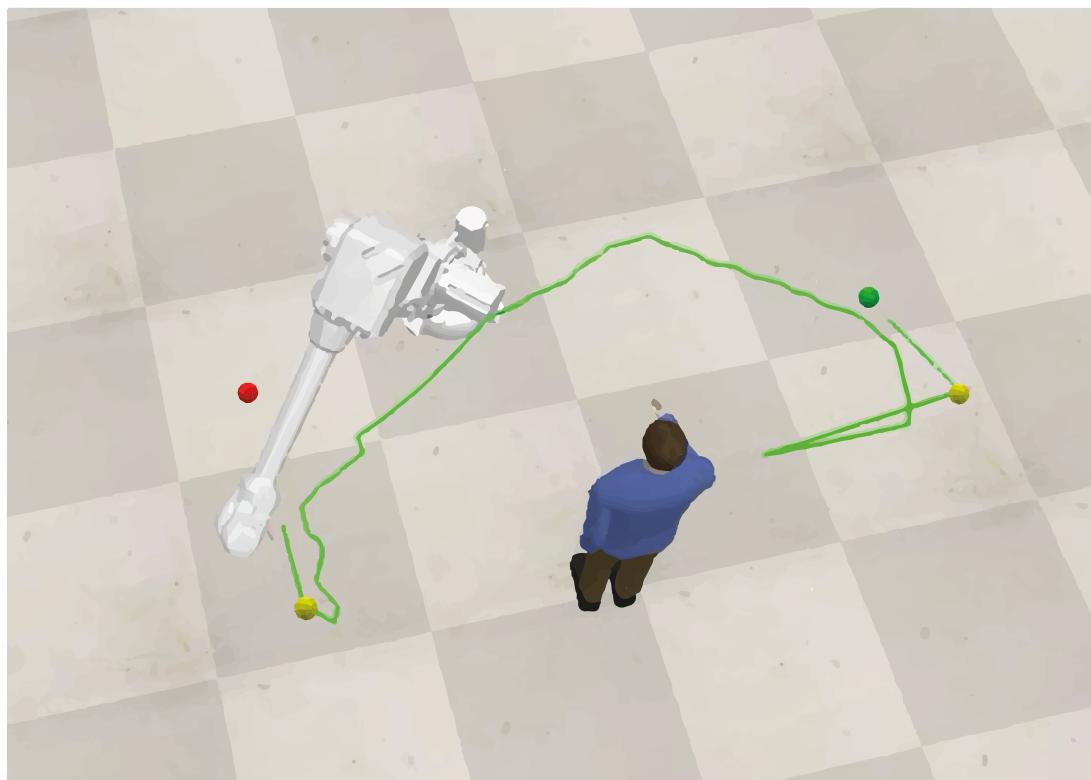


Figura 41 – Manipulador ABB IRB 4600 com obstáculo dinâmico.

### 4.3 Simulação com o manipulador Sawyer

A expansão da metodologia agora é aplicada para o robô Sawyer. Em relação a parte física do robô como, juntas, elos e dimensões, este robô é muito parecido com o robô Kuka sendo possível utilizar os mesmos parâmetros mencionados em seções anteriores.

A Figura 42 apresenta a nova trajetória gerada para que o robô Sawyer possa desviar do operador evitando a colisão. Um ponto importante a ser destacado para este robô, e a funcionalidade de ele ter uma câmera e *display* acoplado ao próprio robô, o que permite na prática expandir as funcionalidades dessa aplicação para detecção de pessoas como obstáculos através da própria câmera do robô.

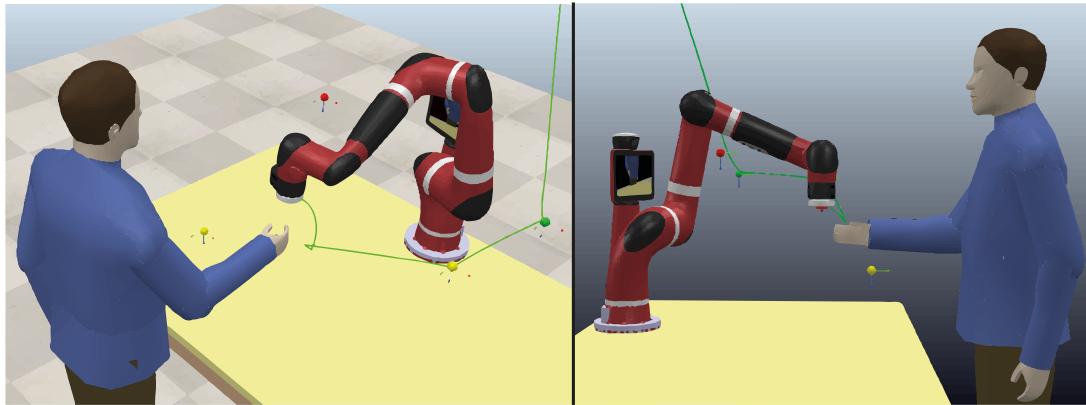


Figura 42 – Criação de nova trajetória aplicada para robô Sawyer.

Através de todas as simulações apresentadas e destacadas nessa Seção 4, podemos confirmar que a geração de novas trajetórias funcionam teoricamente para qualquer tipo de robô, apenas realizando alguns ajustes em seus parâmetros conforme a aplicação. A fim de realizar uma validação completa, testes físicos com os robôs mencionados neste trabalho e outros modelos de robôs podem ser implementados.

## 5 Conclusões

O presente trabalho apresentou uma nova abordagem metodológica para a geração automática de trajetórias em ambientes com obstáculos. A geração de novas trajetórias é realizada através da combinação de matrizes de transformação homogênea que rotacionam nos eixos  $y$  e  $z$  e de vetores de posição que também são utilizados para que a trajetória seja criada de forma correta.

Com a revisão teórica exibida neste documento, foram compilados fundamentos teóricos suficientes para a concepção do método proposto pelo autor do presente trabalho. O método inicialmente foi implementado em um simulador que contempla a estrutura base dos manipuladores usados para o desenvolvimento e a validação do algoritmo.

As três etapas do método proposto são referentes aos cálculos e algoritmos utilizados. Na primeira etapa, o componente *dummy* fornece informações de posição e orientação de um ponto no espaço, fazendo com que pontos de início e destino da trajetória possam ser conhecidos. Cálculos vetoriais oferecem informações para que o manipulador consiga realizar a trajetória de forma alternativa passando por alguns pontos intermediários pré-definidos. Na segunda etapa, identificam-se os obstáculos no espaço, criando um raio de segurança que permite a identificação de potenciais colisões contra o manipulador ao se aproximar do operador. A terceira etapa do método, refere-se a geração de uma nova trajetória através da combinação das matrizes de transformação homogênea com vetores de posição. Com essa abordagem é possível determinar se o manipulador deve criar sua nova trajetória se afastando por cima ou por baixo do obstáculo e qual o sentido de rotação adequado que deve ser executado.

Como visto na seção de resultados, a matriz de transformação homogênea funcionou corretamente na geração de novas trajetórias, evitando colisões com o trabalhador ao se aproximar do obstáculo. Esta abordagem também permitiu que a trajetória se adapte dinamicamente enquanto o operador se movimenta.

As principais conclusões que podem ser abstraídas da execução da presente monografia encontram-se listadas a seguir:

- O conhecimento de simuladores de robótica tem um grande diferencial na realização de testes, pois possibilita criar projetos e modificá-los para encontrar a melhor eficiência sem se preocupar com danos materiais.
- A matriz homogênea mista utilizada no trabalho tem um grande potencial na geração de novas trajetórias conseguindo definir o ângulo de giro da trajetória, distância de afastamento do obstáculo juntamente com o sentido de rotação.

- A aplicação de vetores de posição é um tópico essencial na robótica, possibilitando conhecer a posição de cada ponto no espaço, desde obstáculos, pontos de destino e a posição atual do próprio manipulador.
- A geração de trajetórias pode ser implementada para vários tipos de robôs com alguns ajustes de parâmetros, o que torna uma aplicação de grande relevância.
- A metodologia proposta garante uma maior segurança para que operadores possam compartilhar o mesmo local de trabalho com robôs sem correrem riscos de acidentes.

## 5.1 Considerações finais

O atual projeto permitiu que o autor dessa monografia pudesse aperfeiçoar os conhecimentos relacionados com a área da robótica, aplicando conceitos muito relevantes nessa área como os de geração de trajetórias, robótica colaborativa, e matrizes de transformação homogênea. Além dos conhecimentos adquiridos com o desenvolvimento da metodologia, o projeto fez com que as habilidades em trabalhar com simuladores fossem melhoradas, permitindo aprender uma nova linguagem de programação chamado LUA. Todos os fundamentos apresentados no presente documento permitem a expansão para novas aplicabilidades e melhorias na geração de trajetórias.

## 5.2 Trabalhos futuros

O método proposto no presente documento permite a um manipulador gerar novas trajetórias ao identificar um obstáculo no entorno. Entretanto, para garantir ainda mais a segurança dos trabalhadores, novas pesquisas e trabalhos futuros podem ser desenvolvidos relacionados aos tópicos apresentados a seguir:

- Implementação de um algoritmo para detecção da pose de pessoas através de visão computacional e câmeras de segurança, fazendo com que partes do corpo humano sejam detectadas como obstáculos dentro da área industrial. Como apresentado na Figura ?? essa aplicação já foi iniciada pelo autor do presente trabalho, entretanto, encontra-se em fase de desenvolvimento e em trabalhos futuros pode ser dado continuidade.
- Implementação dos métodos apresentados neste trabalho em um robô físico para a validação de resultados práticos.
- Compartilhamento de informações da trajetória de vários robôs em um ambiente industrial para que possam realizar o serviço em sincronia e de maneira colaborativa.

# Referências Bibliográficas

- 1 MULLER, C. Robots and humans can work together with new iso guidance. In: *International Federation of Robotics IFR*. [S.l.: s.n.], 2020. 14
- 2 LAZARTE, M. *World Robotics 2020 Industrial Robots and Service Robots*. 2016. Disponível em: <<https://www.iso.org/news/2016/03/Ref2057.html>>. 14
- 3 VICENTINI, F. Terminology in safety of collaborative robotics. *Robotics and Computer-Integrated Manufacturing*, v. 63, p. 101921, 2020. ISSN 0736-5845. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0736584519300614>>. 14
- 4 MARVEL, J. A. Performance metrics of speed and separation monitoring in shared workspaces. *IEEE Transactions on automation Science and Engineering*, IEEE, v. 10, n. 2, p. 405–414, 2013. 14
- 5 MARVEL, J. A.; FALCO, J.; MARSTIO, I. Characterizing task-based human–robot collaboration safety in manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. 45, n. 2, p. 260–275, 2015. 14
- 6 CHINNIAH, Y. Robot safety: Overview of risk assessment and reduction. In: *ICRA 2016*. [S.l.: s.n.], 2016. 14
- 7 GAMERO, I. *O que faz um robô colaborativo? Todas as aplicações possíveis*. 2018. Disponível em: <<https://www.pollux.com.br/blog/o-que-faz-um-robo-colaborativo-todas-as-aplicacoes-possiveis>>. 14
- 8 GUALTIERI, L. et al. An evaluation methodology for the conversion of manual assembly systems into human-robot collaborative workcells. *Procedia Manufacturing*, v. 38, p. 358–366, 2019. ISSN 2351-9789. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2351978920300470>>. 14
- 9 GUALTIERI, L. et al. Safety, ergonomics and efficiency in human-robot collaborative assembly: Design guidelines and requirements. *Procedia CIRP*, v. 91, p. 367–372, 2020. ISSN 2212-8271. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2212827120308350>>. 14
- 10 EUROBOTS. *Kuka LBR IIWA 14 R820*. 2016. Disponível em: <<https://www.eurobots.net/rob--usado-lbr-iiwa-pt.html>>. 14
- 11 FLACCO, F. et al. A depth space approach to human-robot collision avoidance. In: *2012 IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2012. p. 338–345. 15
- 12 KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, v. 5, n. 1, p. 90–98, 1986. Disponível em: <<https://doi.org/10.1177/027836498600500106>>. 15
- 13 FLACCO, F.; LUCA, A. D.; KHATIB, O. Control of redundant robots under hard joint constraints: Saturation in the null space. *IEEE Transactions on Robotics*, v. 31, n. 3, p. 637–654, 2015. 15

- 14 MEZIANE, R.; OTIS, M. J.-D.; EZZAIDI, H. Human-robot collaboration while sharing production activities in dynamic environment: Spader system. *Robotics and Computer-Integrated Manufacturing*, v. 48, p. 243–253, 2017. ISSN 0736-5845. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0736584515301447>>. 15
- 15 NIKU, S. B. *Introdução à Robótica - Análise, Controle, Aplicações*. 2<sup>a</sup>. ed. Rio de Janeiro: LTC, 2013. ISBN 9788521622376. 17, 18, 19, 22, 25
- 16 CRAIG, J. J. *Robótica*. 3<sup>a</sup>. ed. São Paulo: Editora Pearson, 2012. ISBN 9788581431284. 17, 19, 24, 25
- 17 CORKE, P. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. [S.l.]: Springer Berlin Heidelberg, 2011. ISBN 9783642201431. 18
- 18 FANUC. *Como saber se o Robô SCARA é a escolha certa para a sua aplicação*. 2021. Disponível em: <<https://www.fanuc.eu/pt/pt/robôs/página-filtro-robôs/scara-series/selection-support>>. 18
- 19 FANUC. *Robô SCARA SR-3iA*. 2021. Disponível em: <<https://www.fanuc.eu/pt/pt/robôs/página-filtro-robôs/scara-series/scara-sr-3ia>>. 19
- 20 SICILIANO, B. et al. *Robotics: Modelling, Planning and Control*. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 1846286417. 19, 25
- 21 KUKA. Kuka sensitive robotics lbr iiwa. 2016. 20, 28
- 22 WINTERLE, P. *Vetores e Geometria Analítica*. 2<sup>a</sup>. ed. São Paulo: Pearson, 2014. ISBN 9788543002392. 20
- 23 BOLDRINI, J. L. et al. *Álgebra linear*. 3<sup>a</sup>. ed. São Paulo: Harbra, 1980. ISBN 9788529402024. 20
- 24 HALLIDAY, D.; RESNICK, R.; WALKER, J. *Fundamentos de física: mecânica*. 9<sup>a</sup>. ed. Rio de Janeiro: GEN, 2012. ISBN 9788521619031. 21
- 25 TSAI, L.-W. *Robot Analysis: The Mechanics of Serial and parallel Manipulators*. Estados Unidos da América: Wiley, 1999. ISBN 0471325937. 24
- 26 MARTINS, N. A. et al. Desempenho de seguimento de trajetória no espaço da tarefa de robôs manipuladores: Um projeto de controlador neural adaptativo. *SBIC*, 2005. 25
- 27 DIANATFAR, M.; LATOKARTANO, J.; LANZ, M. Review on existing vr/ar solutions in human–robot collaboration. *Procedia CIRP*, v. 97, p. 407–411, 2021. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2212827120314815>>. 26
- 28 INDUSTRIAIS, A.-. C. de Estudo Especial de Sistemas Integrados para R. *Robôs e dispositivos robóticos — Requisitos de segurança para robôs industriais Parte 1: Robôs*. 2018. Disponível em: <<https://www.abntcatalogo.com.br/norma.aspx?ID=398253>>. 26
- 29 ZHANG, R. et al. A reinforcement learning method for human-robot collaboration in assembly tasks. *Robotics and Computer-Integrated Manufacturing*, v. 73, p. 102227, 2022. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S07>>. 26

- 30 BUXBAUM, H.-J.; SEN, S.; HäUSLER, R. A roadmap for the future design of human-robot collaboration. *IFAC-PapersOnLine*, v. 53, n. 2, p. 10196–10201, 2020. 21th IFAC World Congress. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2405896320335114>>. 26
- 31 KUKA. *LBR iiwa*. 2021. Disponível em: <<https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa>>. 26
- 32 HUANG, J.; TODO, I. Chapter 31 - a human-safe control for collision avoidance by a redundant robot using visual information. In: ARAI, T.; YAMAMOTO, S.; MAKINO, K. (Ed.). *Systems and Human Science*. Amsterdam: Elsevier Science, 2005. p. 425–437. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B9780444518132500336>>. 27
- 33 LERUSALIMSCHY, R.; CELES, W.; FIGUEIREDO, L. H. de. *Lua: About*. 2021. Disponível em: <<https://www.lua.org/about.html>>. 30