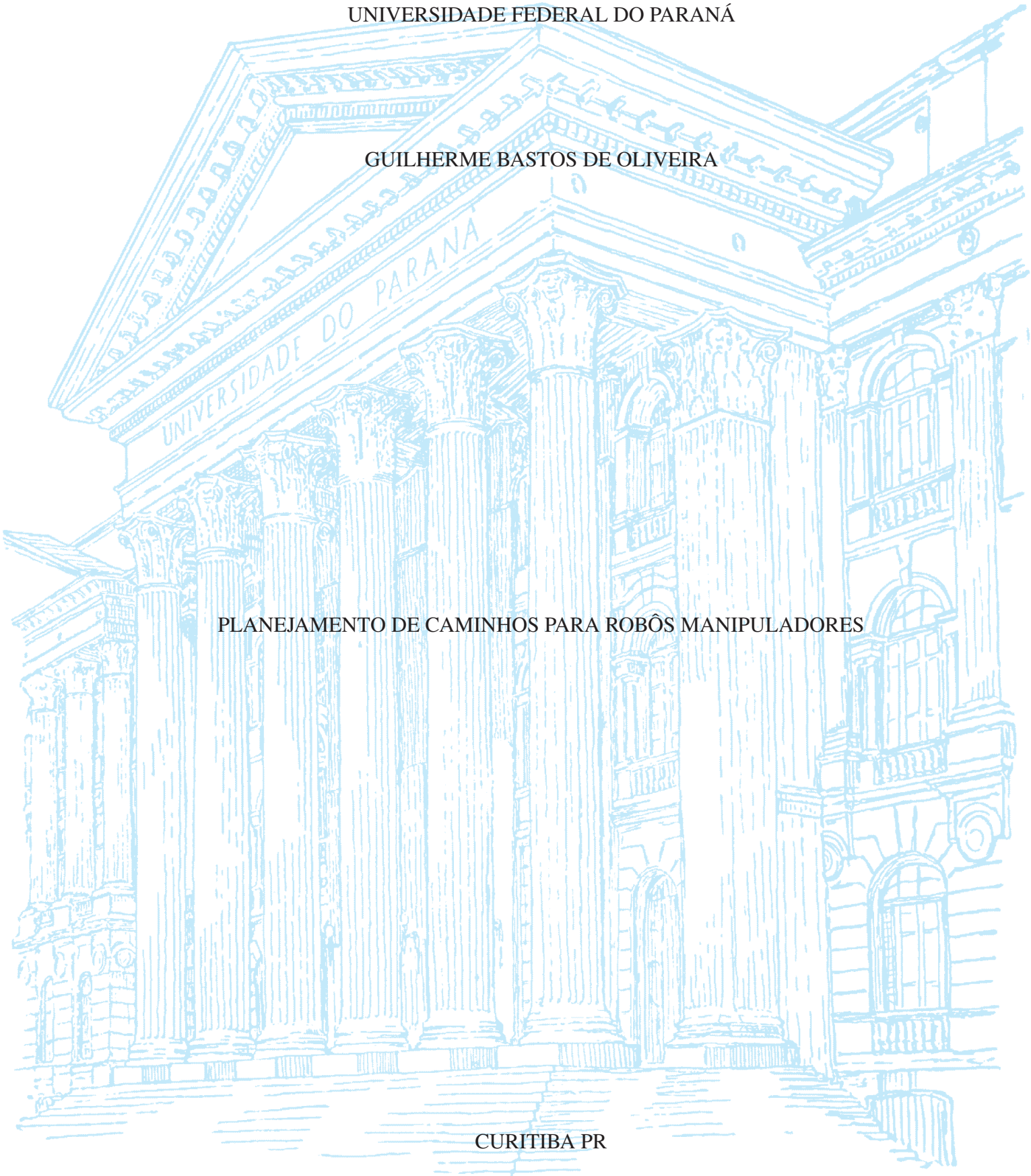
UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME BASTOS DE OLIVEIRA



PLANEJAMENTO DE CAMINHOS PARA ROBÔS MANIPULADORES

CURITIBA PR

2022

GUILHERME BASTOS DE OLIVEIRA

PLANEJAMENTO DE CAMINHOS PARA ROBÔS MANIPULADORES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: André Luiz Pires Guedes.

CURITIBA PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Oliveira, Guilherme Bastos de
Planejamento de caminhos para robôs manipuladores / Guilherme Bastos de Oliveira. – Curitiba, 2022.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.

Orientador: André Luiz Pires Guedes

1. Robôs – Programação. 2. Algoritmos. 3. Robôs móveis – Planejamento de caminhos. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Informática. III. Guedes, André Luiz Pires. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **GUILHERME BASTOS DE OLIVEIRA** intitulada: **Planejamento de Caminhos Para Robôs Manipuladores**, sob orientação do Prof. Dr. ANDRÉ LUIZ PIRES GUEDES, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 16 de Dezembro de 2022.

Assinatura Eletrônica
25/12/2022 10:52:05.0
ANDRÉ LUIZ PIRES GUEDES
Presidente da Banca Examinadora

Assinatura Eletrônica
28/12/2022 13:24:24.0
RICARDO DUTRA DA SILVA
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
24/12/2022 19:10:44.0
RENATO JOSÉ DA SILVA CARMO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Dedico este trabalho àqueles que vieram antes de mim e tornaram isto possível

AGRADECIMENTOS

Agradeço à minha mãe, Renata, por todo o suporte, incentivo e confiança, essenciais durante esta jornada. Ao prof. Dr. André Guedes, pelos incontáveis conselhos, inúmeras horas de atenção e grande apoio, fundamentais para o desenvolvimento deste trabalho. À minha família, pelo carinho e preocupação com o meu desenvolvimento e sucesso. E aos meus amigos, por todos os momentos de companheirismo e amparo, que fizeram o caminho ser mais agradável.

RESUMO

O problema de planejamento de caminhos pode ser descrito para diversos tipos de robôs, neste trabalho serão tratados os robôs manipuladores (aqueles que possuem uma base fixa e tem como objetivo mover sua ponta manipuladora). Existem grandes dificuldades na abordagem analítica para manipuladores complexos, portanto, foram estudadas abordagens numéricas para o problema. Tais dificuldades surgem da complexidade em codificar as interações entre objetos que podem rotacionar no espaço. O trabalho discute algoritmos baseados em campos potenciais, *sampling*, aleatorizados, decomposição de células e um exemplo simples que pode ser resolvido de maneira analítica a fim de demonstração. Neste trabalho uma abordagem foi implementada como dissertação de mestrado, resultando numa solução para o problema de planejamento de caminhos para um manipulador com 3 graus de liberdade, por meio de juntas rotativas. A solução fez proveito do conceito de espaço de configurações (*C-Space*) e algoritmos de busca clássicos. Inicialmente ocorre uma etapa de pré-processamento que gera um espaço de configurações discretizado, esse espaço é então explorado por uma busca em largura modificada.

Palavras-chave: planejamento de caminhos, solução numérica, robô manipulador

ABSTRACT

The path planning problem can be described for different types of robots, in this work manipulator robots (those that have a fixed base and aim to move their manipulator end) will be treated. There are great difficulties in the analytical approach to complex manipulators, therefore, numerical approaches to the problem were studied. Such difficulties arise from the complexity of encoding the interactions between objects that can rotate in space. The work discusses algorithms based on potential fields, sampling, randomization, cell decomposition and a simple example that can be solved analytically, for comparison porpuses. In this work, an approach was implemented as a master's thesis, resulting in a solution to the path planning problem for a manipulator with 3 degrees of freedom, through rotating joints. The solution took advantage of the configuration space concept (C-Space) and classical search algorithms. Initially, there is a pre-processing step that generates a discretized configuration space, this space is then explored by a modified breadth first search.

Keywords: path planning, numeric solution, manipulator robot

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
ROS	Robotic Operating System
IK	Inverse Kinematics
C-Space	Espaço de configuração
PRM	Probabilistic Roadmap Method
RRT	Rapidly-exploring Random Trees

LISTA DE SÍMBOLOS

α	ângulo do eixo do primeiro grau de liberdade
β	ângulo do eixo do segundo grau de liberdade
γ	ângulo do eixo do terceiro grau de liberdade
$\mathbb{R}^2$	espaço cartesiano bidimensional
$\mathbb{R}^3$	espaço cartesiano tridimensional
$\mathcal{O}$	conjunto de obstáculos
$\mathcal{W}$	espaço de trabalho
$\mathcal{A}$	ator, robô que se move pelo espaço

SUMÁRIO

1 INTRODUÇÃO

O planejamento de caminhos é um problema que possui diversas formas e abordagens. É possível definir o problema para um robô móvel, um conjunto de robôs móveis, desvio de obstáculos estáticos ou dinâmicos, robôs de base fixa para manipulação de objetos e planejamento único ou múltiplos planejamentos.

Esse problema não é trivial de ser resolvido de maneira genérica e possui poucos casos específicos onde é conhecido um algoritmo analítico para obter a solução ótima em tempo viável.

Neste trabalho será abordado primariamente o problema de encontrar caminhos para um robô manipulador de base fixa e 3 graus de liberdade, a partir de uma configuração inicial até uma configuração objetivo, navegando por um espaço com obstáculos estáticos.

Existem diferentes abordagens numéricas para encontrar soluções viáveis e aproximações para o problema de planejamento de caminhos. Serão destacados neste trabalho os algoritmos que utilizam *sampling*, campos potenciais, decomposição de células e, para um problema exemplo, uma solução *ad hoc* a partir da construção de um espaço de configurações (*C-Space*).

Esse problema se encontra em diversas situações, como o uso de braços robóticos para construção e manutenção na Estação Espacial Internacional (Rybus, 2018), automações na indústria (Gao et al., 2015) e manipulação de tubos de ensaio (Afrisal et al., 2020).

O objetivo deste trabalho é apresentar a abordagem que foi implementada para o problema de planejamento de caminhos para um robô manipulador com 3 graus de liberdade em um ambiente com obstáculos. A solução se baseia na construção de um espaço de configurações discreto do ambiente de trabalho do manipulador, atráves de uma modelagem por enumeração e o algoritmo de busca em largura com uma pequena modificação para armazenar a informação do vértice pai.

A solução possui duas etapas principais, o pré-processamento, para a construção de um espaço de configurações, e a busca, onde é utilizado um algoritmo de busca em grafos para encontrar uma sequência de movimentos que permita o manipulador sair de uma posição inicial e chegar em uma posição final, evitando obstáculos.

2 FUNDAMENTOS

Neste capítulo serão apresentadas as definições necessárias para tratar do problema abordado no trabalho. Além disso o conceito de espaço de configurações será explorado, assim como uma breve explicação sobre a soma de Minkowski.

2.1 DEFINIÇÕES

Para entender melhor o problema é necessário definir seus elementos. Nesta seção serão definidos ambiente/espaço de trabalho, robô/ator, ações, obstáculos, objetivos, estados e espaço de configurações. As definições aqui descritas foram inicialmente baseadas em LaValle (2006) e adaptadas para facilitar o entendimento da solução proposta.

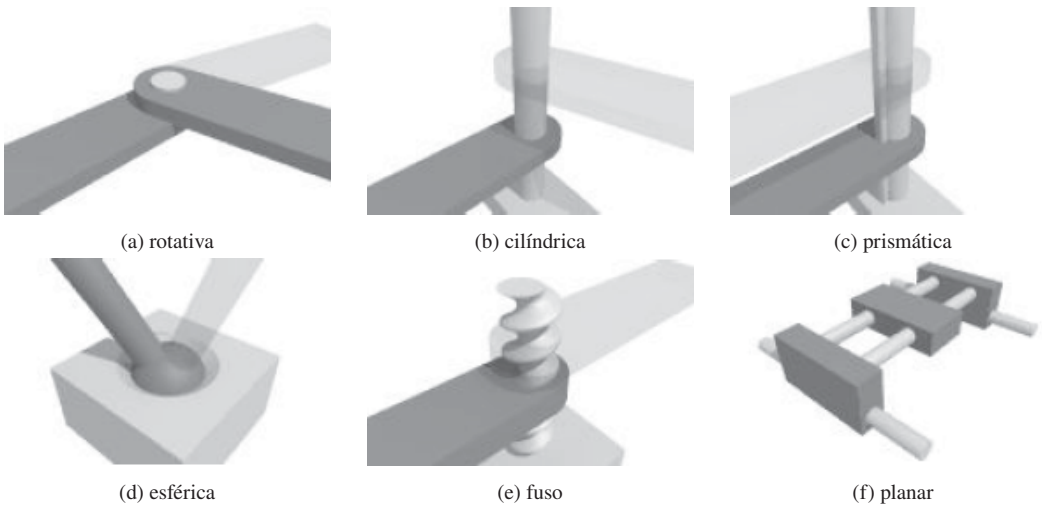
2.1.1 Ambiente

O ambiente ou espaço de trabalho $\mathcal{W}$ é o espaço onde estão inseridos o robô e obstáculos. Grande parte dos trabalhos de planejamento utilizam $\mathcal{W} = \mathbb{R}^2$ ou $\mathcal{W} = \mathbb{R}^3$. É possível estudar diversos espaços de trabalho, como espaços com mais dimensões ou a superfície de uma esfera. Neste trabalho serão tratados cenários com $\mathcal{W} = \mathbb{R}^3$.

A região de obstáculos $\mathcal{O} \subseteq \mathcal{W}$ representa partes do espaço de trabalho que estão permanentemente ocupadas e não podem ser ocupadas pelo ator.

2.1.2 Ator ou Robô

Um ator ou robô $\mathcal{A}$ é um corpo geometricamente modelado, existente em $\mathcal{W}$ e controlado por um plano de ação, que será definido na sequência. O ator pode assumir diversas formas de movimento e características, destacando-se principalmente os tipos de juntas possíveis, sendo as principais: rotativa, cilíndrica, prismática, esférica, fuso e planar. A Figura 2.1 apresenta uma ilustração de tais juntas.



(a) rotativa (b) cilíndrica (c) prismática

(d) esférica (e) fuso (f) planar

Figura 2.1: 6 tipos comuns de juntas para robôs. Adaptado de Groover (1986).

Neste trabalho serão considerados robôs de base fixa com partes móveis baseadas em juntas rotativas, planares ou prismáticas, que em conjunto controlam a posições de uma ponta manipuladora.

2.1.3 Estado

Um estado x é uma representação da posição e orientação das partes móveis de $\mathcal{A}$ e consequentemente seu corpo como um todo e sua ponta manipuladora.

Um ator possui um estado inicial x_I, o estado em que se encontra no início do problema e um estado objetivo x_O, o estado que deve ser atingido para solucionar o problema.

O conjunto de todos os estados possíveis de um ator é chamando de espaço de estados e denominado por X. O espaço de estados pode ser discreto ou contínuo, dependendo das transformações possíveis sobre o ator. Em ambas as situações o espaço de estados geralmente é representado de maneira implícita, uma vez que na maioria dos casos é extenso demais para ser computado em tempo hábil.

2.1.4 Ação

Uma ação a é uma transformação em $\mathcal{W}$ que, ao ser aplicada sobre um estado x gera um novo estado x', de acordo com uma função de transição de estado f. Ou seja, $x' = f(x, a)$.

Um espaço de ações $U(x)$ representa todo o conjunto de ações possíveis a partir de um estado x. O conjunto de todas as ações possíveis sobre todos os estados é definida como $U = \bigcup_{x \in X} U(x)$. Neste trabalho, de acordo com o manipulador utilizado e escolhas feitas durantes a implementação, as ações serão modeladas de maneira discreta.

2.1.5 Plano de Ação

Um plano de ação é uma sequência de ações que, quando aplicadas em ordem através de uma função de transição de estado, transforma um estado x_1 em um estado x_n através de $n - 1$ ações.

Além de satisfazer a condição do estado objetivo existe um possível critério adicional para o plano de ação, que é a otimalidade, ou seja, o plano de ação viável deve otimizar a performance de alguma maneira (como distância percorrida e/ou tempo gasto).

2.1.6 Estados e Configuração

No caso que será analisado ao longo deste trabalho, podemos especificar mais ainda essas definições. As definições a seguir são para o robô manipulador com 3 graus de liberdade por juntas de rotação.

Um estado pode ser descrito por um vetor (α, β, γ), onde $0 \leq \alpha, \beta, \gamma \leq 360$ representam a posição de cada uma das juntas.

O espaço de estados é um espaço no $\mathbb{N}^3$ onde todos os componentes possuem um valor entre 0 e 360, e cada um de seus pontos corresponde a um estado do manipulador.

Uma ação $a = (a_1, a_2, a_3)$, com $a_1, a_2, a_3 \in \{-1, 0, 1\}$, corresponde a uma movimentação em uma ou mais juntas.

O ambiente será o $\mathbb{R}^3$ e os obstáculos serão dados por um conjunto de paralelepípedos.

O espaço de configurações é um remapeamento do espaço de estados, que possui configurações válidas, que são aquelas livres de colisão, e as inválidas, onde existe colisão. Uma

exploração sobre o conceito de espaço de configurações e suas aplicações será apresentada na Seção 2.2.

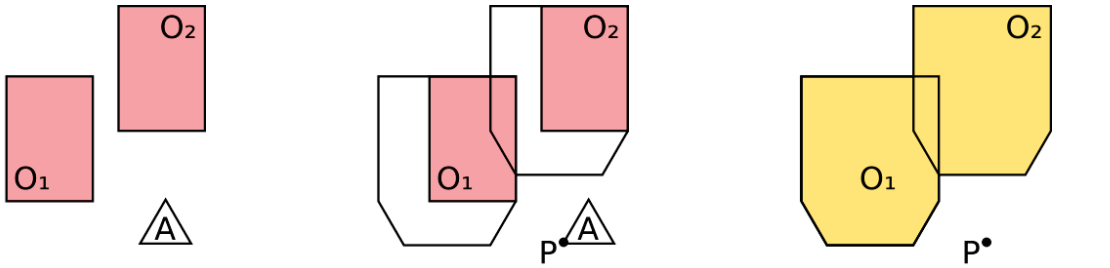
Sobre as definições acima é importante destacar que para os fins deste trabalho foram realizadas simplificações por discretização uma vez que o espaço de estados representa uma posição no mundo real e portanto o mapeamento mais realista seria através de números reais.

Pelo modo que o problema será tratado é mais interessante utilizar tal definição, além disso, para fins práticos, existe uma relação de balanceamento entre o nível de granularidade da discretização, o tempo de execução e qualidade do caminho, uma vez que uma granularidade menor permite garantir que não existem obstáculos menores entre passos, porém o espaço de estados aumenta, o que requer maior tempo computacional. Para este trabalho foi escolhido 1° como o passo da discretização, pois, para o tamanho do manipulador escolhido permite a detecção dos obstáculos em todos os cenários de testes.

2.2 C-SPACE

O espaço de configurações (também conhecido como *C-Space*) é uma transformação do espaço de trabalho para um espaço que possibilita tratar o ator de maneira mais simples, através da codificação da complexidade da interação dos obstáculos com o ator e seus movimentos para que seja possível utilizar algoritmos de busca para encontrar um caminho.

Uma forma simples de visualizar e entender o processo pode ser vista em um exemplo no plano, com um robô móvel de formato rígido. Pode-se imaginar que essa transformação "encolhe" o robô para ser representado como um ponto e "aumenta" os obstáculos pelo tamanho do robô, como demonstrado na Figura 2.2.



(a) Obstáculos O_1 e O_2 com um ator A

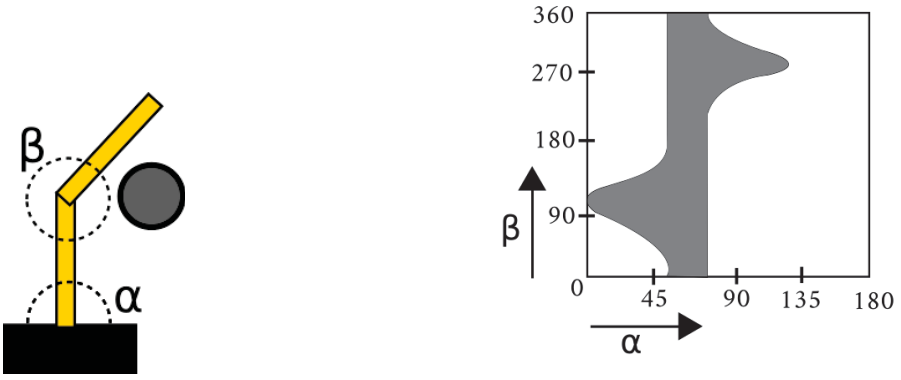
(b) Transformação dos obstáculos a partir de um ponto de referência P

(c) espaço de configurações com obstáculos O_1 e O_2 e ator representado por um ponto P

Figura 2.2: Transformação de obstáculos no espaço de trabalho para o espaço de configurações.

Para robôs que possuem juntas que permitem rotação ou possuem formato dinâmico é necessário considerar também os diferentes formatos que o robô pode assumir, como por exemplo na figura 2.3. Para o caso de um robô em um espaço tridimensional com juntas rotativas a aplicação do conceito necessita de uma grande escalada na complexidade analítica, principal na modelagem dos obstáculos e incrementos nas dimensões das configurações, e portanto não será utilizada.

Uma configuração de um robô está associada a um ponto no espaço de configurações, porém nem todas as configurações são válidas no *C-Space*, uma vez que o espaço de configurações inclui as informações sobre obstáculos e as limitações do manipulador. Neste caso o espaço de configurações é dado por um subespaço no $\mathbb{R}^2$ que representa estados que o manipulador pode atingir sem colidir com obstáculos.



(a) ator e obstáculo

(b) espaço de configurações correspondente

Figura 2.3: Ator bidimensional com 2 eixos de rotação e um obstáculo circular e espaço de configurações correspondente.

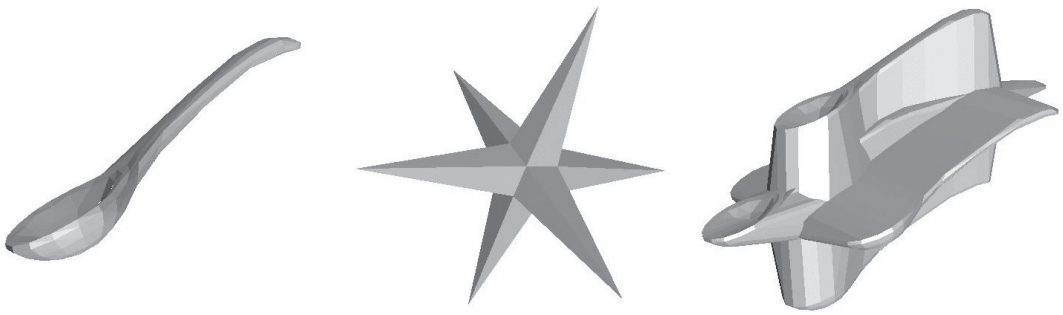
2.3 SOMA DE MINKOWSKI

Uma das ferramentas que podem ser aplicadas na ideia de realizar transformações no espaço e obstáculos para representar o robô como um ponto no espaço é a soma de Minkowski.

Esta operação geralmente é realizada sobre os obstáculos, a fim de inflar eles o suficiente para garantir que o robô, ao ser representado por um ponto, colida com os obstáculos de modo equivalente ao espaço real.

A soma de Minkowski de dois conjuntos de vetores A e B de posição no espaço euclidiano é formada pela adição de cada vetor em A a cada vetor em B. Aplicando a soma de Minkowski em polígonos ou poliedros podemos imaginar que a operação "engorda" um polígono ou poliedro a partir de outro.

Podemos aplicar a soma de Minkowski entre os obstáculos e o ator, de modo que o formato e tamanho do obstáculo seja tal que não é necessário verificar a colisão entre os objetos em si, apenas se o ponto que representa se o ator está dentro do obstáculo. A Figura 2.4 contém o exemplo de uma soma de Minkowski.



(a) Poliedro A

(b) Poliedro B

(c) Poliedro C

Figura 2.4: Representação da soma de Minkowski entre dois poliedros A e B, resultando em um terceiro poliedro C. Adaptado de Hachenberger (2013).

3 O PROBLEMA

Dado um espaço de trabalho, um conjunto de obstáculos e, um ator capaz de se mover, definir um plano de ações para que esse ator se mova pelo espaço evitando colisão com os obstáculos, a partir de sua configuração inicial a fim de chegar a uma configuração objetivo (que pode ser um configuração completa do ator ou apenas de sua ponta manipuladora) é um problema que pode ter diferentes níveis de complexidade, dado pelas características de cada espaço e sistema em si.

A configuração objetivo geralmente é dada principalmente pela posição da ponta manipuladora do ator, não importando para o resultado final a configuração do resto do ator, desde que a rota seja viável. Outra maneira de definir uma configuração objetivo é através de um estado do manipulador, que será utilizado neste trabalho.

Por um lado, qualquer configuração que faça a ponta manipuladora atingir o objetivo é viável, o que permite mais flexibilidade quando utilizados algoritmos de busca aleatorizados e *sampling*, por outro lado, encontrar uma ou mais configurações que fazem a ponta atingir o objetivo é, por si só, um desafio, a depender do manipulador e suas características. Além disso, com mais de uma configuração que atende o objetivo, surge mais um desafio, escolher a melhor configuração baseado em algum critério, como a quantidade de energia necessária, distância percorrida, tempo gasto pela movimentação, etc.

Se o problema de decidir a configuração objetivo pode ser simplificado, seja por meio implícito, como no algoritmo de busca, ou de maneira explícita, é vantajoso transformar o espaço de trabalho para um espaço de configurações (ou *C-Space*) que será gerado a partir das limitações do ator e obstáculos, tornando possível a utilização de algoritmos de planejamento já existentes para computar o caminho.

O objetivo é utilizar algoritmos conhecidos de planejamento de caminhos no espaço de trabalho, fazendo com que não seja necessário levar em conta os movimentos mais complexos do ator, como rotações. Com isso o espaço de configurações não possui necessariamente as mesmas dimensões que o espaço de trabalho, dado que cada dimensão corresponde a um dos graus de liberdade do ator, portanto o tamanho dimensional do espaço é alterado.

Podemos ter espaços com dimensões diferentes ($\mathbb{R}^2$, $\mathbb{R}^3$), movimentação com ou sem rotação, diferentes graus de liberdade, obstáculos com formas previsíveis (como quadrados, círculos, caixas, esferas, poliedros convexos) ou não.

A ideia mais simples para o problema do planejamento é, a partir de uma rota fornecida para a ponta manipuladora, encontrar de maneira analítica os movimentos que o ator deve realizar para que sua ponta siga a rota e chegue ao destino. Está abordagem porém ignora os obstáculos que o resto do ator pode colidir e o problema de encontrar uma configuração do ator que satisfaça a condição de posicionamento de sua ponta, portanto, necessita de refinamento.

Outra consideração crucial ao lidar com atores de maior complexidade envolve a abordagem do problema de cinemática inversa ou, em inglês, *Inverse Kinematics* (IK). O desafio inerente a este problema consiste em determinar a configuração que satisfaça um objetivo específico, definido como a posição desejada no espaço para uma determinada parte do ator.

Em muitos casos, para atores altamente complexos, a determinação da configuração desejada não é trivial e pode levar a múltiplas ou infinitas soluções. Em particular, atores com graus de liberdade redundantes podem apresentar uma infinidade de configurações que atendem ao objetivo estabelecido, e, em tais cenários, uma solução analítica direta pode não estar disponível.

A Figura 3.1 ilustra quatro possíveis soluções para um problema de cinemática inversa, todas as configurações satisfazem a condição do local que a área da ponta manipuladora deve abranger.

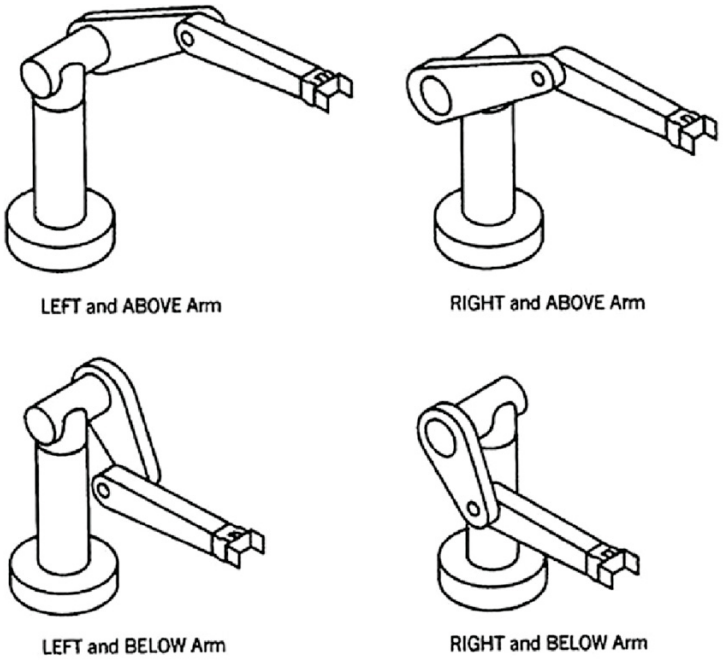


Figura 3.1: 4 diferentes solução de IK para uma posição da ponta, de Patel et al. (2014).

Para exemplificar as possíveis abordagens nos diferentes contextos dois casos serão apresentados a seguir. Primeiro um simples, que pode contar com uma solução analítica de forma relativamente direta e um exemplo complexo, onde a solução analítica enfrenta mais desafios, devido à juntas de rotação e formato não rígido do manipulador, e como pode ser montada uma solução numérica.

3.1 EXEMPLO SIMPLES - ROBÔ CARTESIANO

Dados um espaço 3D e um ator baseado em um robô cartesiano, ou seja, que possui 3 graus de liberdade, através de movimentação em dois eixos na base e um corpo telescópico, e obstáculos no espaço podemos definir o problema de encontrar uma rota ótima para a cabeça entre dois pontos. A Figura 3.2 mostra uma possível manifestação de um ator com tais características.

Neste caso existe apenas um estado (e uma configuração) para cada posição da ponta, ou seja, a descrição da configuração objetivo, mesmo que apenas pela posição da ponta implica em um estado específico.

3.1.1 Solução Analítica

Dado que o ator possui uma base estática que permite movimento nos eixos x e y, enquanto sua extremidade manipuladora está equipada com uma junta telescópica, é factível simplificar o tratamento de obstáculos do espaço de trabalho para o espaço de configuração. Essa simplificação é alcançada considerando apenas o contorno mais amplo dos obstáculos em cada intervalo ao longo do eixo z, mapeando o manipulador como um ponto no plano e utilizando obstáculos bidimensionais no espaço de configuração.

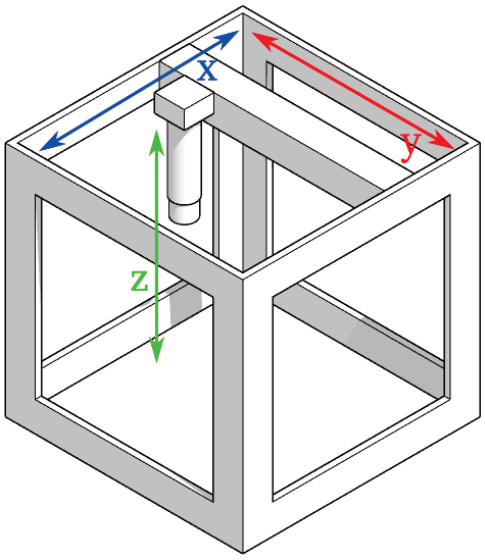


Figura 3.2: Representação de um possível ator com 3 graus de liberdade através da translação, com dois eixos em uma base fixa e uma ponta manipuladora em um cilindro telescópico, respectivamente x, y e z na imagem.

Para isso é preciso transformar os obstáculos presentes no espaço de trabalho em obstáculos do espaço de configuração, obtendo uma projeção no plano dos objetos e, em seguida, aplicando a soma de Minkowski de um disco com o mesmo raio da extremidade manipuladora. Isso nos permite utilizar esses obstáculos no algoritmo de planejamento de caminhos, a fim de evitar colisões com o corpo do manipulador quando a extremidade manipuladora se encontra abaixo do nível dos obstáculos.

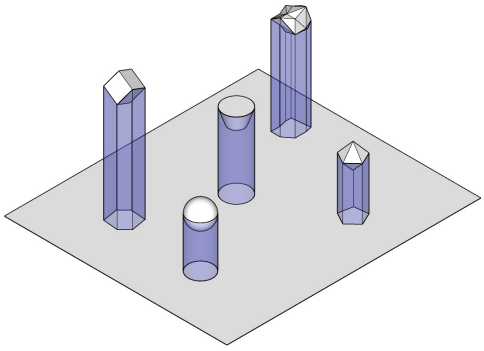


Figura 3.3: Representação de obstáculos em um espaço tridimensional e suas projeções (em azul) para inserção no espaço de configurações a fim de evitar colisões com o corpo telescópico do ator.

A Figura 3.3 oferece uma representação visual das projeções obtidas a partir de um conjunto de obstáculos de exemplo. O algoritmo de planejamento, que resolve o problema de planejamento de caminho para um ponto no plano, pode ser implementado construindo um grafo de visibilidade e, em seguida, empregando um algoritmo de busca de caminho mínimo neste grafo, com uma complexidade temporal de $O(n^2 \log n)$, como descrito em Schwartz e Sharir (1988).

Para representar a solução de maneira esquematizada, a Figura 3.4 contém um diagrama que representa o algoritmo utilizado para resolver o problema de planejamento de caminhos para um manipulador simples. Esse diagrama associa etapas específicas do algoritmo a exemplos visuais, fornecendo uma visão geral clara do funcionamento do algoritmo.

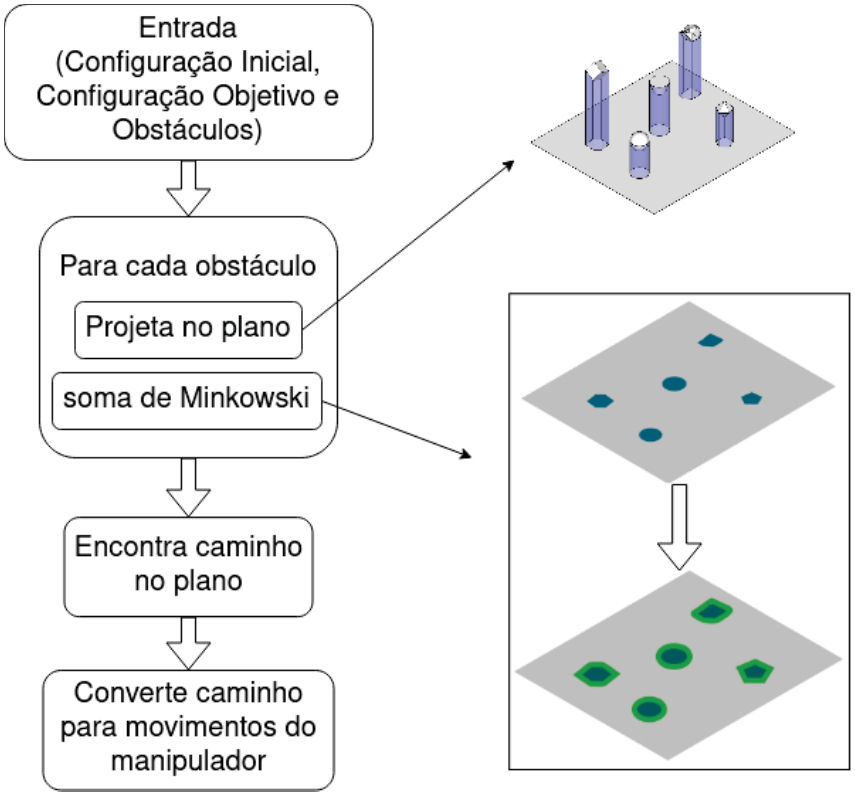


Figura 3.4: Diagrama que representa o algoritmo para resolver o exemplo simples.

3.2 EXEMPLO COMPLEXO - BRAÇO ROBÓTICO

Dado um ator baseado em um braço robótico, que possui uma base fixa rotativa e um braço com 2 juntas em um espaço 3D podemos definir outro problema, similar ao anterior, onde o ator possui também 3 graus de liberdade. É evidente porém, que utilizar a mesma abordagem não é possível, uma vez que o processo da soma de Minkowski e projeção dos obstáculos não é suficiente para codificar os movimentos de rotação do ator. A Figura 3.5 mostra uma possível manifestação de um ator com tais características.

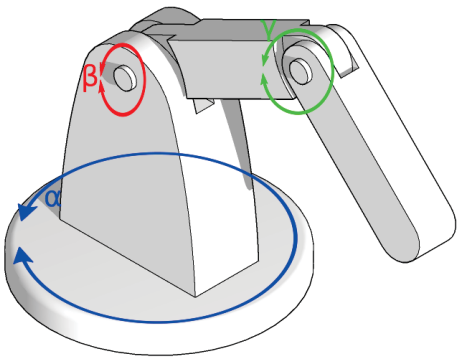


Figura 3.5: Representação de um possível ator com 3 graus de liberdade através de rotações, com um eixo em uma base fixa e um braço com duas juntas tendo a ponta manipuladora em sua extremidade, respectivamente α, β e γ na imagem.

3.2.1 Dificuldades da Solução Analítica

O mapeamento realizado no exemplo anterior não é possível, além disso, o movimento baseado em rotação dificulta uma possível abordagem analítica para o problema. O espaço de configurações teria que contar com dimensões envolvendo coordenadas polares devido aos eixos de rotação, portanto a conversão dos obstáculos é muito complexa e custosa.

Existem técnicas que permitem fazer a representação explícita de obstáculos no espaço de configurações com rotação quando são tratados robôs ou objetos que se movem de maneira independente pelo espaço de trabalho, como em Lozano-Pérez e Wesley (1979); Lozano-Perez (1983) onde são aplicados processos de "crescimento" semelhantes a soma de Minkowski mas que levam em conta a possibilidade de rotação. Além disso outra maneira de fazer isso seria focar no espaço livre e a partir dele fazer a representação explícita dos obstáculos, como em Brooks (1983).

Nessas abordagens porém, não são considerados objetos que possuem movimentos que alteram sua forma, como braços robóticos manipuladores de base fixa. Sendo assim, uma abordagem numérica para o problema se torna atrativa.

3.2.2 Possibilidade da Solução Numérica

Tendo em vista os problemas de uma solução analítica pode se ter como alternativa uma solução numérica.

Uma possível solução numérica ingênua seria, dada uma modelagem que permita teste de colisão, dos obstáculos e manipulador, inicializa um espaço de configurações sem obstáculos, gera um caminho considerando o espaço de configurações e então testa o caminho proposto, caso haja uma colisão cria um obstáculo representativo no espaço configuração e recalcula a rota considerando também o novo obstáculo, repetindo o processo até que um planejamento viável seja encontrado.

Embora seja viável encontrar uma solução por meio de um algoritmo de tentativa e erro simples, sua aplicação generalizada é limitada, devido a diversas razões. Os caminhos gerados por esse método frequentemente não apresentam otimização, e enfrentam obstáculos técnicos relacionados aos custos associados aos testes de colisão. Além disso, a obtenção de informações a partir das colisões é complexa, uma vez que tais informações são locais e se referem apenas a partes específicas do robô ao longo do trajeto, como apresentado em Faverjon (1984).

3.3 DEFINIÇÃO DO PROBLEMA

O objetivo do trabalho é obter um algoritmo que possa ser aplicado a um robô manipulador com 3 graus de liberdade, composto por 3 juntas rotativas, na mesma estrutura do manipulador do Exemplo Complexo, apresentado na Seção 3.2, como ilustrado na Figura 3.5, a fim de obter uma rota de movimentação que evita obstáculos.

O problema que será resolvido é o seguinte: dados duas configurações (inicial e objetivo) e um conjunto de obstáculos, determinar um plano de ações para o robô manipulador de modo que, a aplicação das ações do plano, a partir da configuração inicial, obtenha-se a configuração destino.

4 ABORDAGENS DA LITERATURA

Quando considerada a opção de não utilizar uma solução analítica para o planejamento de caminho de robôs manipuladores, são encontradas algumas alternativas. Neste capítulo serão discutidas as principais abordagens atuais para o problema de planejamento: campos potenciais, algoritmos de amostragem e aleatorizados e decomposição celular.

4.1 CAMPOS POTENCIAIS ARTIFICIAIS

A abordagem por campos potenciais se assemelha a uma simulação física, onde o objeto para qual se deseja obter o planejamento de caminho é tratado como uma partícula afetada por um campo potencial artificial. O campo potencial artificial é construído de modo que a partícula seja atraída para a configuração objetivo ao mesmo tempo que é repelida pelos obstáculos, a Figura 4.1 contém um exemplo simplificado.

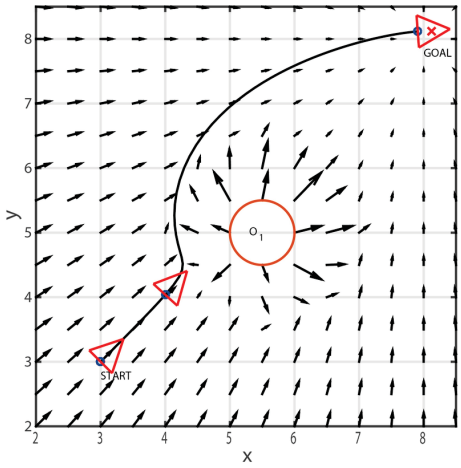


Figura 4.1: Representação de uso simples de campos potenciais para planejamento de caminho de um ponto inicial (*START*) até um objetivo (*GOAL*) evitando um obstáculo (O_1) (Fedele et al., 2017).

Um dos desafios encontrados ao trabalhar com campos potenciais artificiais é a possibilidade de existirem mínimos locais que, sem alterções ao algoritmo, como a inclusão de elementos aleatorizados, resulta no objetivo nunca ser atingido pela partícula.

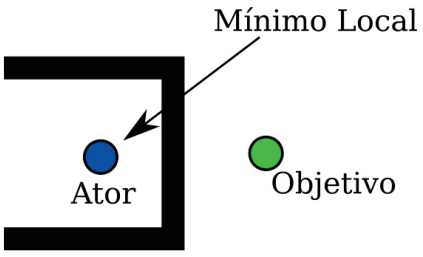


Figura 4.2: Representação de uma configuração onde a execução de uma abordagem através de campos potenciais encontra um mínimo local.

Na Figura 4.2, podemos observar que a partícula está posicionada em um mínimo local. Isso ocorre porque o obstáculo exerce uma força de repulsão sobre a partícula, enquanto o

objetivo exerce uma força de atração. Devido à configuração côncava do obstáculo, a partícula fica impedida de contorná-lo, pois é constantemente atraída em direção ao objetivo e, ao mesmo tempo, repelida pelas paredes do obstáculo, resultando em sua retenção nessa posição mínima.

Algumas alternativas para lidar com o problema de mínimos locais existem, porém suas aplicações fogem do escopo deste trabalho.

A utilização de campos potenciais para planejar rotas para robôs foi proposta por diversos pesquisadores, destacando-se: Barraquand e Latombe (1990) que aplicou um algoritmo de Monte-Carlo junto com a estratégia e Hwang e Ahuja (1992) que faz uma separação do algoritmo entre um planejador a nível global que planeja o caminho inicial e um nível local, que utiliza heurísticas para refinar o trajeto.

Foi proposto recentemente em Wang et al. (2018) uma melhoria do método de campos potenciais artificiais para o planejamento de um robô com 9 graus de liberdade.

4.2 ALGORITMOS BASEADOS EM *SAMPLING*

Outra estratégia para o problema de planejamento é a estratégia baseada em amostragem (do inglês *sampling*), os algoritmos que utilizam tal estratégia também são considerados como aleatorizados. Os algoritmos de *sampling* para o problema de planejamento de movimento podem ser caracterizados como algoritmos aleatorizados pois exploram o espaço de estados através de amostras obtidas de maneira aleatória de acordo com a estratégia de exploração.

O objetivo desta estratégia é evitar a construção explícita dos obstáculos no espaço de configurações, explorando o espaço através de esquemas de sondagem que consideram módulos de detecção de colisão como "caixa-preta".

Os algoritmos baseados em amostragem conseguem obter soluções para certos problemas de robótica, manufatura e aplicações biológicas, que seriam praticamente impossíveis usando técnicas que representam explicitamente os obstáculos no espaço de configurações, segundo LaValle (2006).

Uma das primeiras propostas de utilização de algoritmos aleatorizados ocorreu em Barraquand e Latombe (1991) que impulsionou o desenvolvimento e aplicação de diversos algoritmos para este problema. A solução proposta em Barraquand e Latombe (1991) consiste em construir um grafo a partir dos mínimos locais de uma função de campos potenciais definida sobre o espaço de configurações do robô e então buscar neste grafo um caminho que será utilizado como a rota do robô. A vantagem surge a partir das propriedades da função potencial e a eficiência da técnica de Monte-Carlo para sair de mínimos locais.

Além do algoritmo *Randomized Path Planning* apresentado em Barraquand e Latombe (1991), alguns outros principais algoritmos de planejamento baseado em *sampling* são *Probabilistic Roadmap* (Kavraki e Latombe, 1994; Kavraki et al., 1996; Amato e Wu, 1996; Švestka e Overmars, 1997), *Rapidly-exploring Random Tree* (LaValle, 1998; LaValle e James J. Kuffner, 2001), *Ariadne's Clew* (Ahuactzin et al., 1998) e *Expansive Space Tree* (Hsu et al., 1997).

Dos algoritmos citados acima, destacam-se *Probabilistic Roadmap Method* (PRM) e *Rapidly-exploring Random Trees* (RRT), que foram considerados os algoritmos aleatorizados mais popularizados para o problema de planejamento de rota de robôs de acordo com Elbanhawi e Simic (2014).

Considerando o problema da cinemática inversa, discutido na Seção 3.2.2, um uso diferente de um RRT para planejar o caminho de um manipulador é apresentada em Bertram et al. (2006). A mudança principal é resolver o problema da cinemática inversa de maneira integrada ao do problema de planejamento, explorando métricas de distância e heurísticas para aumentar a probabilidade de encontrar um objetivo a partir de uma configuração do manipulador.

A estratégia apresentada em Bertram et al. (2006) é necessária pois, além do problema de cinemática inversa não possuir solução analítica conhecida para robôs com mais de 6 graus de liberdade, de acordo com Craig (2005), pode possuir mais de uma solução (inclusive um espaço contínuo) e é possível que enquanto uma solução é alcançável outra não seja, se estiverem em componentes desconexos do espaço de configurações.

Sendo assim, não é viável tentar computar um caminho para todos os objetivos gerados pela solução do problema de cinemática inversa e o método apresentado se torna uma alternativa mais simples, uma vez que apenas uma árvore é explorada e é possível verificar se o objetivo foi atingido durante a expansão, sem ter que previamente calcular uma configuração objetivo no espaço de configurações.

4.3 DECOMPOSIÇÃO CELULAR

Decomposição celular (do inglês *cell decomposition*) é uma estratégia que tem como objetivo representar o espaço de trabalho de maneira subdivida em células livres ou ocupadas, células vizinhas são subespaços adjacentes. Geralmente essas células são exploradas como um grafo, utilizando algoritmos de busca clássicos.

Uma das principais estruturas para representar um espaço tridimensional por células para a finalidade de planejamento de caminho é a *octree*, uma árvore em que todos os vértices não folha possuem 8 vértices filhos. Na maioria dos algoritmos de planejamento cada vértice representa uma área cúbica do espaço e possui um de três possíveis estados: vazio, misto ou ocupado. Diversos algoritmos de planejamento exploram *octrees* e suas diversas aplicações. A Figura 4.3 contém uma representação gráfica de um exemplo de uma *octree*.

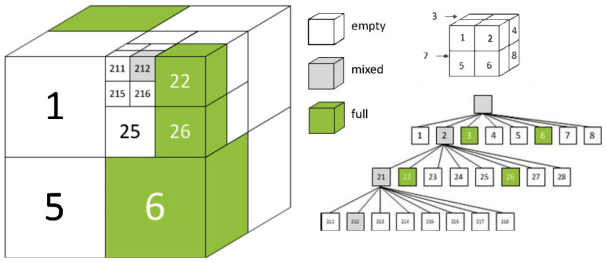


Figura 4.3: Representação hierárquica de uma *octree* (Ramer et al., 2013).

Um dos primeiros usos da estratégia de *cell decomposition* foi descrito em Lozano-Perez (1981), utilizando células para representar o espaço obtido através de aproximações poliédricas dos obstáculos e do manipulador, a computação do volume percorrido pelo robô e simplificação do *C-Space* através de projeções para espaços com menos dimensões.

Outra proposta foi de Faverjon (1984), utilizando uma *octree* no espaço de configurações (que no robô utilizado é descrito pelas juntas) do manipulador. Ou seja, o espaço utilizado na busca do caminho é dado por um vetor de 3 posições onde cada posição representa um ângulo de uma junta do manipulador. Em Faverjon (1986) é discutido um aprimoramento focado em manipulação de objetos para processos industriais, em Faverjon (1989) é apresentada uma modificação que faz uso de modelos hierárquicos para representar os objetos. Como o espaço de juntas é representado por uma *octree*, o algoritmo é limitado, a princípio, para manipuladores com 3 juntas.

Em Hamada e Hori (1996) é apresentada uma abordagem que modela o manipulador como uma cadeia de cubos e o espaço como uma *octree*, a estratégia pode ser usada de

maneira independente da quantidade de graus de liberdade do manipulador, estrutura e diferentes manipuladores para o planejamento de caminhos.

Uma proposta para acelerar o processo de planejamento de caminhos foi dada em Ando (2003), verificando se uma célula é livre ou não apenas quando essa é considerada pelo planejador. Além disso, em Ando (2003) o processo de busca é separado entre uma etapa local e uma etapa global que utilizam o algoritmo A* (A-Estrela) para realizar a busca em grades de diferentes resoluções a fim de diminuir o espaço explorado.

5 SOLUÇÃO

A solução proposta e implementada começa com a modelagem do espaço de estados para o espaço de configurações através da discretização das possíveis configurações das juntas. Neste caso, o espaço de configurações, modelado como um espaço de estados discreto, é enumerado explicitamente, para cada uma de suas configurações, e para cada configuração é computada sua viabilidade, para que seja executado um algoritmo de busca a fim de encontrar um caminho entre configurações.

É importante ressaltar duas limitações claras com esta abordagem, primeiro, caso um obstáculo seja menor que o espaço existente entre dois estados que um caminho inválido seja computado, isso pode causar uma colisão não dectada devido, portanto para contemplar obstáculos arbitrariamente pequenos a abordagem necessita de outra estratégia. A abordagem, por questões de simplicidade, considera apenas o estado inteiro como objetivo, e não a posição da ponta manipuladora. Como é possível que em casos que não exista caminho entre dois estados, mas é possível que entre outros estados, que atinjam a mesma posição da ponta exista, utilizar o estado como destino gera uma perda de flexibilidade nos possíveis caminhos.

O espaço de configurações formado pode ser tratado como um grafo e diversos algoritmos de busca clássicos podem ser utilizados. O grafo onde as buscas serão realizadas é um grafo implicito da estrutura de dados gerada durante a etapa de construção do espaço de configurações.

Inicialmente foi implementando o algoritmo de Dijkstra, porém, após a definição sobre a vizinhança e custo, não foi mais necessário um algoritmo que considerasse pesos nas arestas. Portanto foi implementada uma busca em largura, modificada para manter o registro do vértice pai, a fim de reconstruir o caminho até o estado objetivo a partir do estado inicial, que atinge o mesmo resultado com um menor custo computacional. Essa abordagem implica que a computação inicial para montar o espaço de configurações é um pré-processamento, se é desejado obter diversos caminhos, no mesmo cenário com a mesmo configuração inicial, alterando apenas a configuração objetivo, todos caminhos (se existirem) pode ser computados de maneira muito mais rápida, uma vez que a maior parte do tempo consumido pelo algoritmo é a construção do espaço e da árvore através da busca em largura modificada.

5.1 FERRAMENTA

Os experimentos realizados foram montados com o auxílio da plataforma *Unity* da Unity Technologies (2021) um motor de jogos (em inglês conhecido como *game engine*) que possui diversas funções que facilitaram o desenvolvimento dos testes, como por exemplo, visualização tridimensional, simulação e funções auxiliares para colisão. A ferramenta possui licença gratuita para projetos não lucrativos. A linguagem de programação utilizada na plataforma é C Sharp. Os testes foram executados utilizando uma máquina com o sistema operacional Linux Mint 20.3 sobre um processador AMD Ryzen™5 5600X e 32GB de RAM.

5.2 MANIPULADOR

O modelo de manipulador utilizado é o *Universal Robot 5*, obtido através do pacote Open Robotics (2022). O pacote fornece a possibilidade de gerar um arquivo com a descrição do robô em URDF (unified robot description format), formato que pode ser importado para o

motor de jogos Unity através do pacote URDF-Importer, fornecido pelo próprio motor de jogos. É possível encontrar uma visualização do manipulador na Figura 5.1.

O manipulador completo possui 6 graus de liberdade, porém apenas os três primeiros serão utilizados no trabalho pois os 3 eixos de movimento mais extremos tem como função a orientação da ponta manipuladora, o que não faz parte do escopo do trabalho.

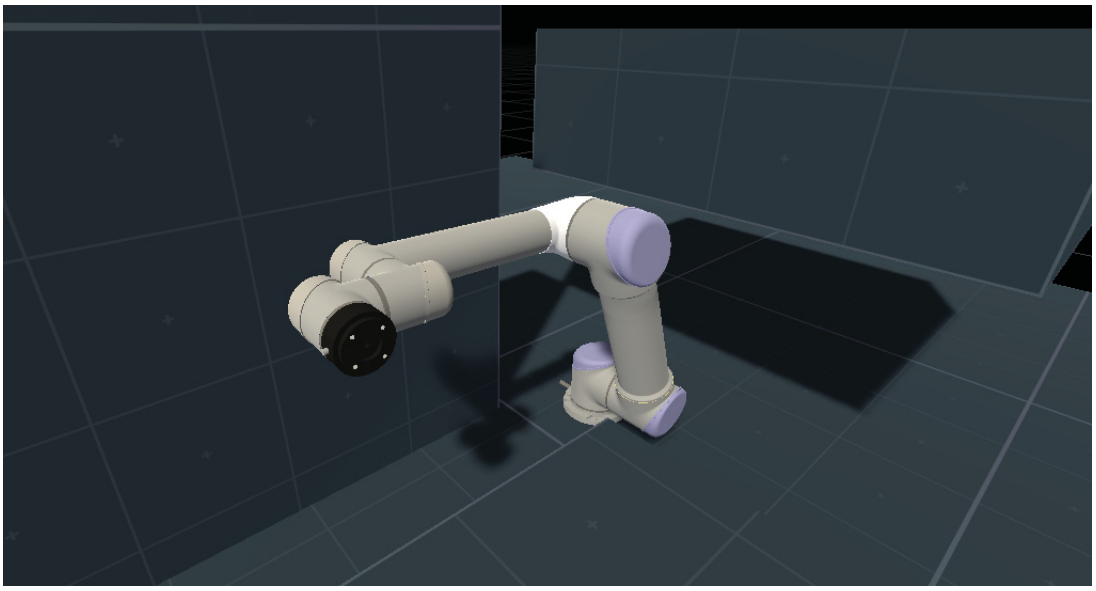


Figura 5.1: renderização do manipulador no motor Unity.

5.3 REPRESENTAÇÃO

O robô manipulador e os obstáculos são representados através de objetos da Engine Unity, que utiliza *meshs* para armazenar informações geométricas dos modelos. A escolha de utilizar essa abordagem para representar computacionalmente os elementos se deu por conta da facilidade de explorar a detecção de colisão da própria Unity.

Existem diversas maneiras de detectar uma colisão entre dois sólidos geométricos, o foco do trabalho é o algoritmo de planejamento e utilizar os métodos do motor para detecção de colisão e visualização do resultado acelerou o processo de desenvolvimento. Outras alternativas podem ser estudados e testadas em trabalhos futuros. Uma vez que, mesmo sendo altamente otimizada para a execução de jogos e suas necessidades computacionais a Unity pode não ser o melhor ambiente para a computação desse algoritmo e outras implementações, ao custo de mais tempo de desenvolvimento e refinamento de elementos que fogem do escopo deste trabalho, resultem em um menor tempo de computação, permitindo não só uma execução mais rápida como também o uso de uma granularidade menor na divisão do espaço de configurações.

5.4 ESCOLHAS DA IMPLEMENTAÇÃO

Para o teste da solução proposta, foram realizadas escolhas cruciais durante a implementação do núcleo do algoritmo deste estudo. Tais decisões desempenham um papel fundamental na determinação do desempenho e dos resultados alcançados, abrangendo desde a seleção de estruturas de dados até características do algoritmo, como custo e granularidade.

5.4.1 Estrutura de Dados

Conforme abordado na Seção 4.3, uma das estruturas amplamente empregadas na representação de espaços tridimensionais, onde áreas livres e ocupadas são delimitadas, é a *octree*. Dado que, neste estudo, o espaço de configurações está definido no domínio $\mathbb{R}^3$, poderia parecer natural optar por essa estrutura. No entanto, neste trabalho, duas razões principais levaram à escolha de outra abordagem.

Uma das vantagens da *octree* reside na capacidade de representar o espaço de forma que várias regiões contíguas do mesmo tipo sejam agrupadas em uma região maior, geralmente durante a fase de construção. Isso resulta na economia do custo computacional associado a cada região na menor granularidade possível. No entanto, dado que a abordagem proposta envolve a enumeração completa do espaço de configurações, a economia de tempo computacional na construção não seria significativa, pois o algoritmo teria que criar a *octree* após a enumeração completa, utilizando uma estratégia *bottom-up*.

Além disso, a estrutura da *octree* é também útil para reduzir o custo de operações de busca. No entanto, no contexto da implementação deste trabalho, o tempo computacional consumido pela fase de busca não possui um impacto tão substancial quanto o pré-processamento. Por essas razões, optou-se por empregar uma estrutura mais simples, que elimina a necessidade de uma fase de construção da *octree*, que é explorada como um grafo no algoritmo de busca em largura.

Na estrutura utilizada, as configurações são armazenadas em um vetor e cada configuração possui um identificador único, o estado correspondente do manipulador, e a estrutura auxiliar necessária para executar o algoritmo de busca em largura.

O identificador de uma configuração também é o seu índice no vetor, para facilitar os acessos e a execução da busca. Seja $c = (c_1, c_2, c_3)$ uma configuração em um *C-Space* de granularidade G, o indentificador da configuração c é dado por:

$$id(c) = c_1 \times G^2 + c_2 \times G + c_3$$

Cada configuração possui 6 vizinhos, que diferem por $+-1$ em exatamente um dos componentes da configuração. É possível computar o identificar dos vizinhos a partir do identificador de uma configuração. Sejam os vizinhos:

$$\begin{aligned} v_1 &= (c_1 - 1, c_2, c3) \\ v_2 &= (c_1 + 1, c_2, c3) \\ v_3 &= (c_1, c_2 - 1, c3) \\ v_4 &= (c_1, c_2 + 1, c3) \\ v_5 &= (c_1, c_2, c3 - 1) \\ v_6 &= (c_1, c_2, c3 + 1) \end{aligned}$$

Podemos computar então, o identificador de cada vizinho da seguinte forma:

$$id(v_1) = (id(c) - G^2) \bmod G^3$$
$$id(v_2) = (id(c) + G^2) \bmod G^3$$
$$id(v_3) = (id(c) - G) \bmod G^3$$
$$id(v_4) = (id(c) + G) \bmod G^3$$
$$id(v_5) = (id(c) - 1) \bmod G^3$$
$$id(v_6) = (id(c) + 1) \bmod G^3$$

5.4.2 Granularidade

Como neste trabalho foi utilizado um manipulador com juntas rotativas de capacidade de giro de 360°, o passo de 1° foi escolhido para definir a granularidade. A granularidade para a discretização do espaço de estados, como apresentada na Seção 2.1.6 será utilizada, vale ressaltar que essa escolha influência diretamente no refinamento do caminho obtido e no custo computacional do algoritmo, uma vez que a etapa de computação do espaço de configurações possui custo $\theta(G^3)$ onde G é o número de estados possíveis para cada eixo do manipulador.

Além disso, o tamanho mínimo de um obstáculo para ser detectado com uma granularidade muda de acordo com a posição do manipulador. Um exemplo de como isso pode ocorrer, com um manipulador simplificado no plano pode ser visto na Figura 5.2, onde o espaço possível para um obstáculo ser "saltado" pelo teste de colisão do algoritmo possui áreas (demarcardas em rosa) diferentes para difrentes configurações do manipulador.

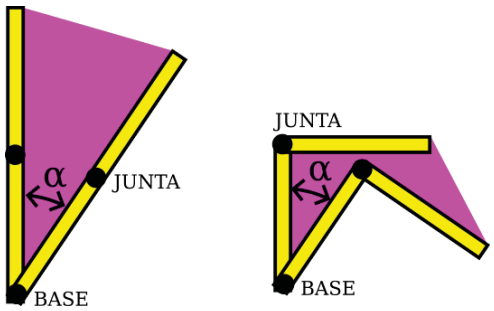


Figura 5.2: um mesmo manipulador, com uma mesma rotação no eixo base, produz áreas percorridas diferentes para posições diferentes da junta.

5.4.3 Custo

Também foi necessário tomar uma decisão em relação ao custo do movimento, o que é considerado na movimentação do manipulador a fim de poder comparar caminhos e definir um critério de otimização para o algoritmo. Como o trabalho não possuiu um intuito direto de ser aplicado com um manipulador real não, mas sim estudar e testar a modelagem proposta, não é útil considerar o custo de energia para movimentação do manipulador (uma vez que numa aplicação real, eixos diferentes podem possuir motores e portanto custos energéticos distintos).

Outro possível critério, o volume total percorrido (do mesmo modo que a área percorrida, como exemplificado na Figura 5.2, mas no espaço e não no plano). Tal critério pode ser interessante para aplicações que desejam minimizar o espaço necessário para que braço realize uma tarefa, como em linhas de montagem em indústrias, mas não é algo necessário para o escopo deste

trabalho. Um critério mais simples foi implementado, o movimento total do manipulador, dado por quantos graus cada eixo do manipulador se moveu durante a execução do caminho.

O custo de movimentopode ser calculado a partir das ações que compõe o plano de ações, o custo total é a soma do custo das ações, e o custo de uma ação (a_1, a_2, a_3) é definido como $\|a\| = |a_1| + |a_2| + |a_3|$. É possível ações mais complexas sejam decompostas em uma sequência de ações mais simples, que terá o mesmo custo total.

Para observer um exemplo de como diferentes ações podem possuir diferentes custos e serem transformadas em uma sequência de ações de custo um, considere as ações abaixo:

$$a = (0, 0, 1)$$
$$b = (0, 1, 0)$$
$$c = (1, 0, 0)$$
$$d = (1, 0, 1)$$
$$e = (1, 1, 1)$$

Temos que $\|a\| + \|b\| = \|d\|$ e $\|a\| + \|b\| + \|c\| = \|e\|$, Para que o algoritmo de busca obtenha o melhor caminho possível é necessário que o custo para os estados vizinhos atingidos através das ações d e e, assim como todas as possíveis ações com custo diferente de 1, seja diferente dos demais vizinhos, ou então não considerar essa vizinhança durante busca, e manter o peso igual para todas as ações.

Como é possível converter qualquer ação como uma sequência de ações de custo um, para fins de simplicidade da implementação, para a vizinhança, serão considerados vizinhos apenas estados que diferem em apenas um eixo e neste eixo, por uma unidade.

A opção de manter apenas ações com custo 1 na vizinhança foi escolhida pois permite utilizar um algoritmo de busca menos custoso e ainda mantém o custo final igual, uma vez que tanto em custo quanto em movimentação dos manipuladores, a execução em sequência das ações a, b, c é equivalente a ação e, pelo exemplo.

Uma maneira de visualizar a relação de vizinha entre os estados é modelar o conjunto de configurações como um reticulado de cubos. Com essa visualização em mente, é possível perceber que existem diferentes maneiras de definir um cubo como vizinho, como todos que compartilham arestas, vértices ou faces, a maneira utilizada pode ser definida como vizinhança apenas por cubos que compartilham faces, como ilustrado na Figura 5.3.

5.5 ALGORITMO

O primeiro passo da solução, considerando que todas as informações necessárias foram fornecidas no formato esperado (local e posição inicial do manipulador, obstáculos) é a etapa de construção do *C-Space*, que pode ser considerada uma etapa de pré-processamento.

A construção do *C-Space* consiste em posicionar virtualmente o manipulador em todas as suas possíveis configurações, discretizadas de acordo com a granularidade definindo um reticulado de 360^3, como mencionado na Seção 5.4. Então cada configuração é testada para colisões com os obstáculos ou entre as partes do manipulador, coletando o resultado do teste para cada configuração. O fluxo do pré-processamento e suas etapas são descritos pelo diagrama da Figura 5.4.

As informações obtidas nesse processo são associadas de forma a permitir consultas com base em vizinhança. Portanto, a estrutura resultante do pré-processamento inclui as

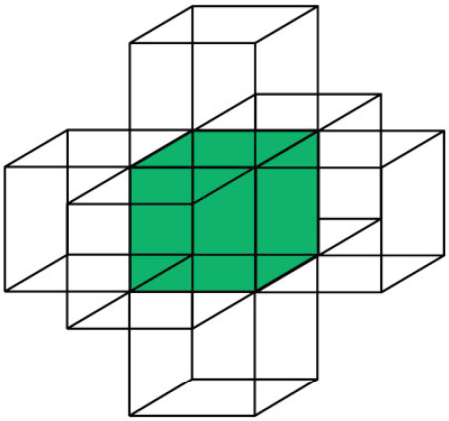


Figura 5.3: Relação de vizinhanças na visualização de estados por cubos. Adaptado de Zhang et al. (2022).

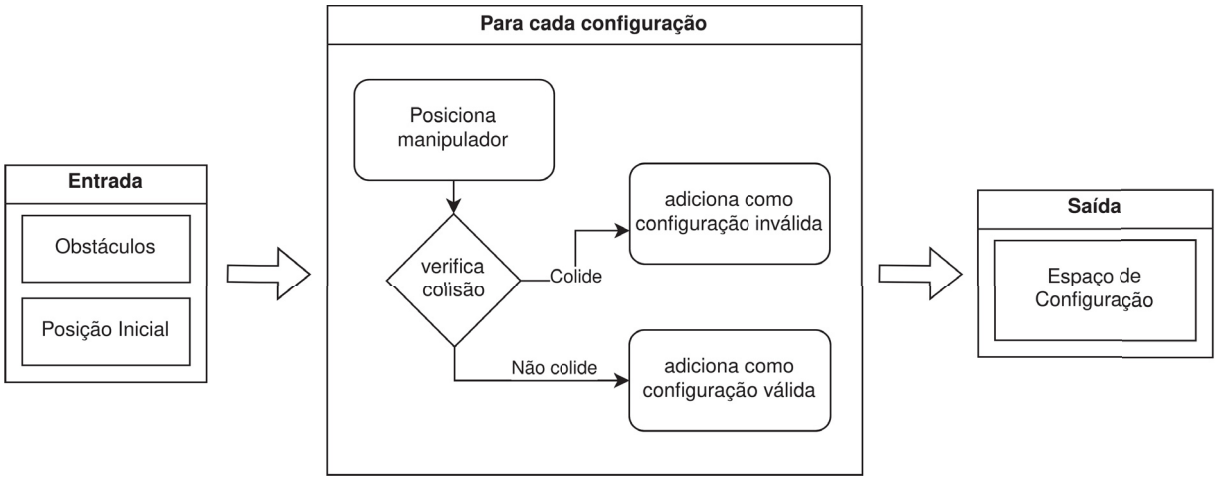


Figura 5.4: Diagrama que representa a etapa de pré-processamento do algoritmo implementado na solução.

configurações viáveis, e o grafo implícito através dos dados necessários para estabelecer relações entre configurações vizinhas no espaço de configuração, tratando-as como vértices vizinhos.

Após o pré-processamento, a próxima etapa envolve localizar o vértice que representa a posição final do manipulador por meio de um processo de busca. Um diagrama que descreve essa etapa pode ser visto na Figura 5.5. Nessa fase, utilizamos uma versão adaptada da busca em largura, que atribui a cada vértice explorado a informação de qual vértice foi seu antecessor na busca. Isso nos permite, ao final, percorrer essa estrutura de vértices antecessores até retornar à configuração inicial.

Uma vez que o caminho tenha sido encontrado, resta apenas a conversão das configurações em instruções que orientarão as mudanças de estado no manipulador, culminando na conclusão do processo de planejamento.

Ao unir ambas as etapas obtém-se uma implementação funcional que utiliza algoritmos clássicos para a solução do problema de planejamento de caminhos através de modelagem do espaço de configuração, busca em grafos e detecção de colisão. Expandir a implementação para outros tipos de robôs (formatos e graus de liberdade diferentes) assim como testar outros algoritmos para detecção de colisão podem ser considerados em trabalhos futuros.

Sobre a etapa de pré-processamento, é interessante destacar que é possível evitar a sua computação em alguns casos. Quando temos uma instância do problema em que apenas o estado

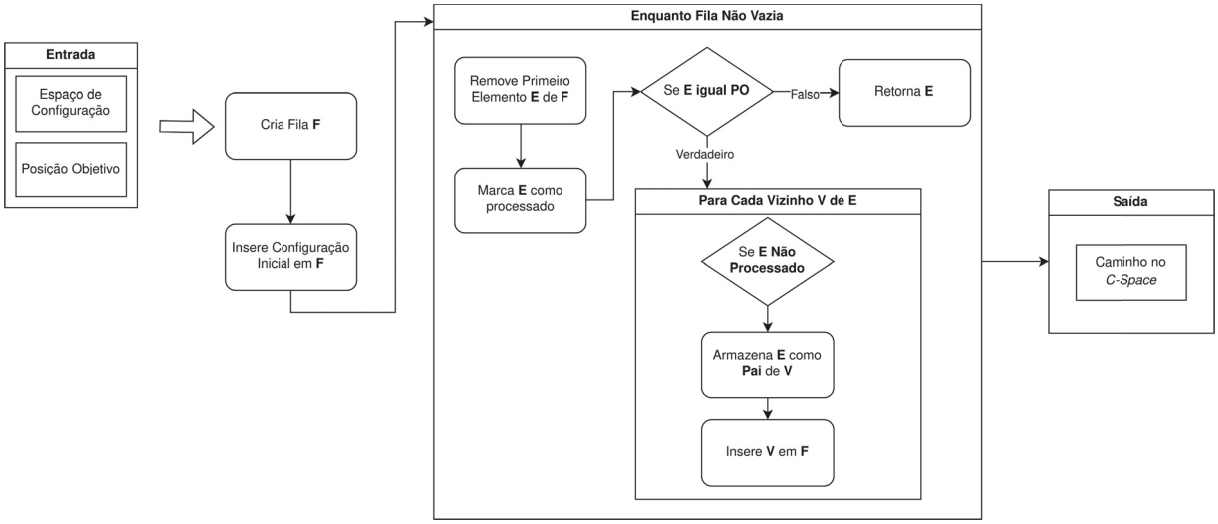


Figura 5.5: Diagrama que representa a etapa de busca do algoritmo implementado na solução.

inicial ou final é alterado, é evidente que uma nova busca precisa ser realizada, e caso sejam alterados é evidente que é necessário gerar o espaço de configuração novamente. No entanto, se mantivermos os obstáculos inalterados e o estado inicial utilizado na construção do espaço de configuração, é viável executar somente a fase de busca, explorando a mesma estrutura de dados e informações previamente obtidas na etapa de pré-processamento.

Isso significa que, se desejarmos realizar diferentes consultas com base na mesma configuração do ambiente (mantendo obstáculos inalterados e estado inicial), podemos economizar tempo e recursos, já que podemos explorar o grafo implícito gerado na fase de pré-processamento em uma única execução. Isso permite otimizar o custo computacional e agilizar a resolução de diferentes consultas.

5.6 INSTÂNCIAS

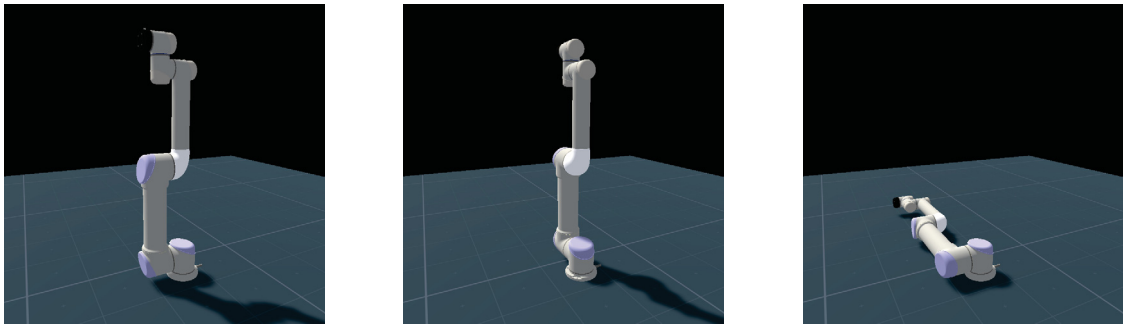
Para realizar os experimentos foram criadas instâncias, a fim de encontrar possíveis casos que fizessem com o que o planejador falhasse, seja por passar obstáculos devido a falha apresentada na Seção 5.4, não foram consideradas instâncias em que não existe algum caminho viável entre dois pontos.

Visualizando o manipulador na representação da Figura 3.5 e usando o mesmo esquema de nomenclatura das juntas (α, β, γ), podemos perceber e associar a relação entra o a posição do robô e o estado que descreve a posição, como mostrado na Figura 5.6. Sendo assim, a rotação da junta na base é descrita por α, a rotação da junta que conecta o braço a base é descrita por β e a rotação da junta que permite a dobra do braço é descrita por γ.

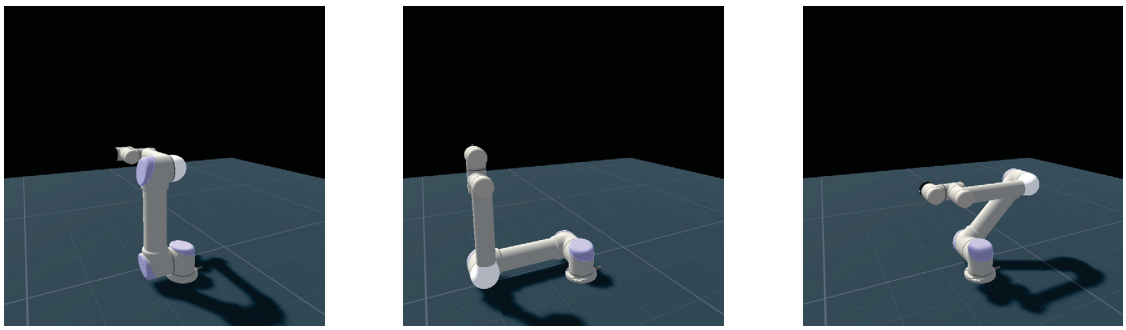
É necessário destacar que o estado inicial do manipulador nas instâncias não é necessariamente o estado padrão (com todas as juntas posicionadas em 0). A Figura 5.6 contém imagens do robô em diferentes estados e as posições de suas juntas.

A Figura 5.6(a) contém o manipulador com todas as juntas em 0. As Figuras 5.6(b), 5.6(c) e 5.6(d) representam uma mudança de 90 graus nos eixos α, β, γ, respectivamente. Dois estados de exemplo mais complexos são apresentados nas Figuras 5.6(e) e 5.6(f).

A **Instância A** (ver Figura 5.7 para uma representação gráfica dos obstáculos e estados do manipulador) possui 4 obstáculos e foi a primeira instância montada para testar a detecção de colisão, uma vez que o caminho mais curto, ignorando colisão seria mais simples e fácil de perceber.

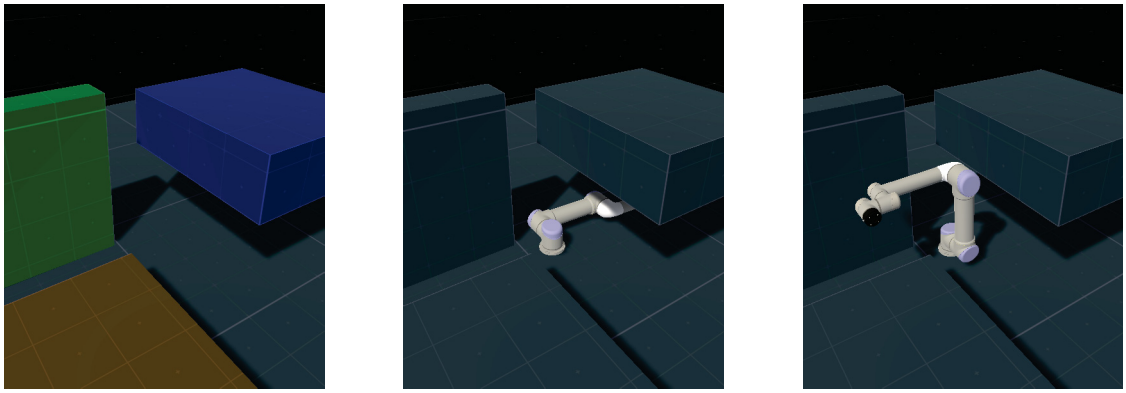


(a) estado: (0, 0, 0) (b) estado: (90, 0, 0) (c) estado: (0, 90, 0)



(d) estado: (0, 0, 90) (e) estado: (90, 270, 90) (f) estado: (90, 45, 225)

Figura 5.6: Estados do manipulador e posições das juntas.



(a) Quantidade de obstáculos: 4. (b) Estado inicial: (0, 0, 90). (c) Estado objetivo: (180, 90, 0).

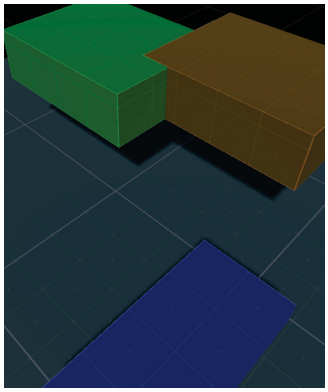
Figura 5.7: Estados do manipulador e obstáculos para a instância A.

A **Instância B** (ver Figura 5.8 para uma representação gráfica dos obstáculos e estados do manipulador) possui 4 obstáculos e foi planejada com o mesmo intuito da **Instância A**, uma vez que o caminho mais curto, ignorando colisão seria mais simples e fácil de perceber.

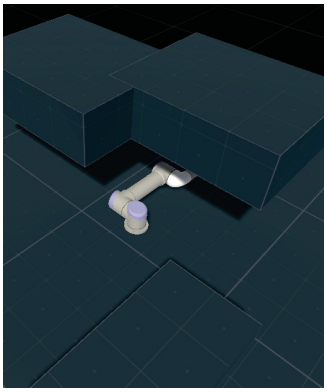
A **Instância C** (ver Figura 5.9 para uma representação gráfica dos obstáculos e estados do manipulador) possui 5 obstáculos e foi planejada com o intuito de testar se o algoritmo implementado seria capaz de calcular um caminho que exige diversos passos, uma vez que é necessário fazer vários movimentos para manobrar o robô entre os obstáculos.

A **Instância D** (ver Figura 5.10 para uma representação gráfica dos obstáculos e estados do manipulador) possui 4 obstáculos e foi planejada com o intuito de testar o efeito da granularidade no algoritmo implementado, com os obstáculos mais finos seria mais próvavel gerar caminhos que saltassem por obstáculos entre passos.

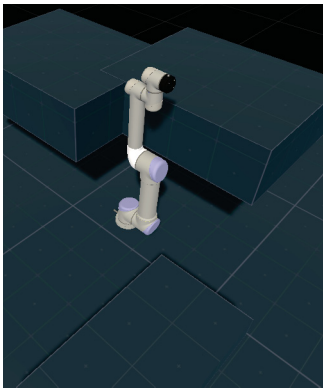
A **Instância E** (ver Figura 5.11 para uma representação gráfica dos obstáculos e estados do manipulador) possui 5 obstáculos e foi planejada por último, para demonstrar como seria



(a) Quantidade de obstáculos: 4.

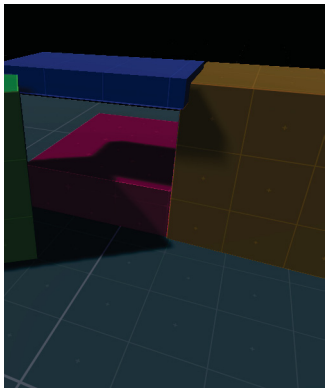


(b) Estado inicial: (0, 0, 0).

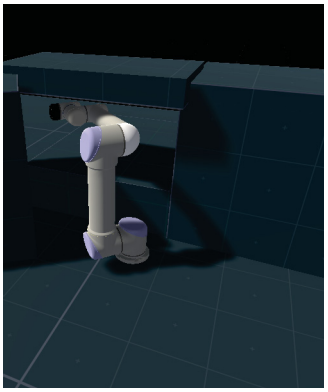


(c) Estado objetivo: (180, 90, 0).

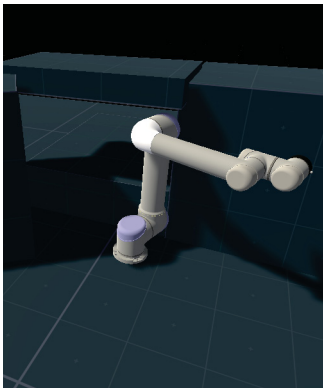
Figura 5.8: Estados do manipulador e obstáculos para a instância B.



(a) Quantidade de obstáculos: 5.

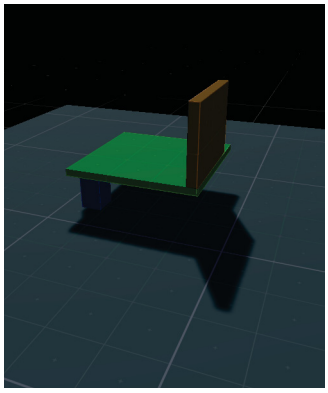


(b) Estado inicial: (200, 0, 90).

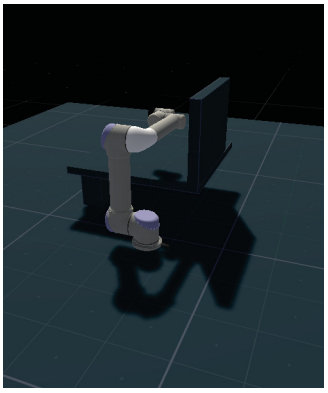


(c) Estado objetivo: (45, 0, 90).

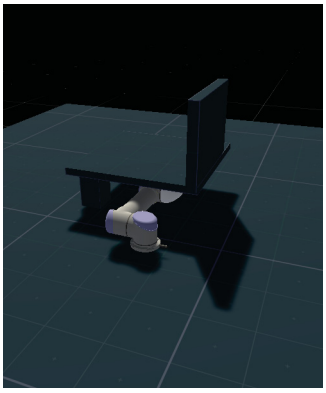
Figura 5.9: Estados do manipulador e obstáculos para a instância C.



(a) Quantidade de obstáculos: 4.



(b) Estado inicial: (0, 90, 0).



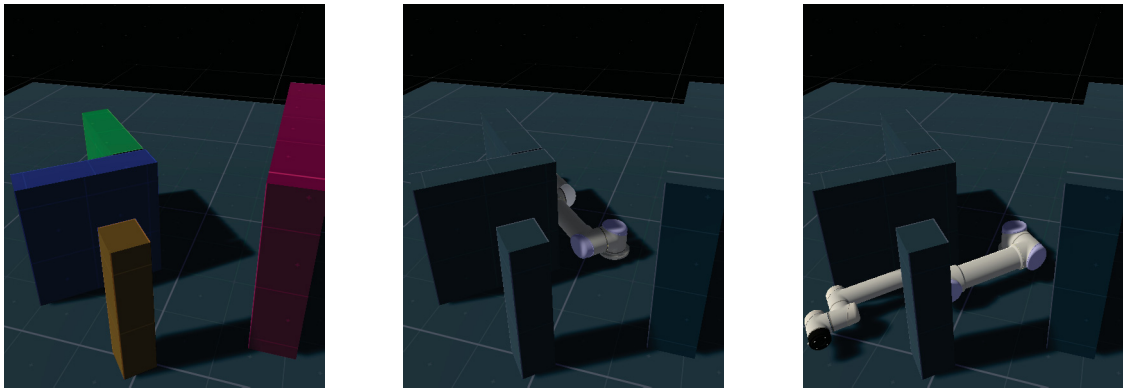
(c) Estado objetivo: (0, 0, 90).

Figura 5.10: Estados do manipulador e obstáculos para a instância D.

um caminho gerado pelo algoritmo em um espaço apertado e obstáculos que exigem longa movimentação para serem evitados.

5.7 RESULTADOS

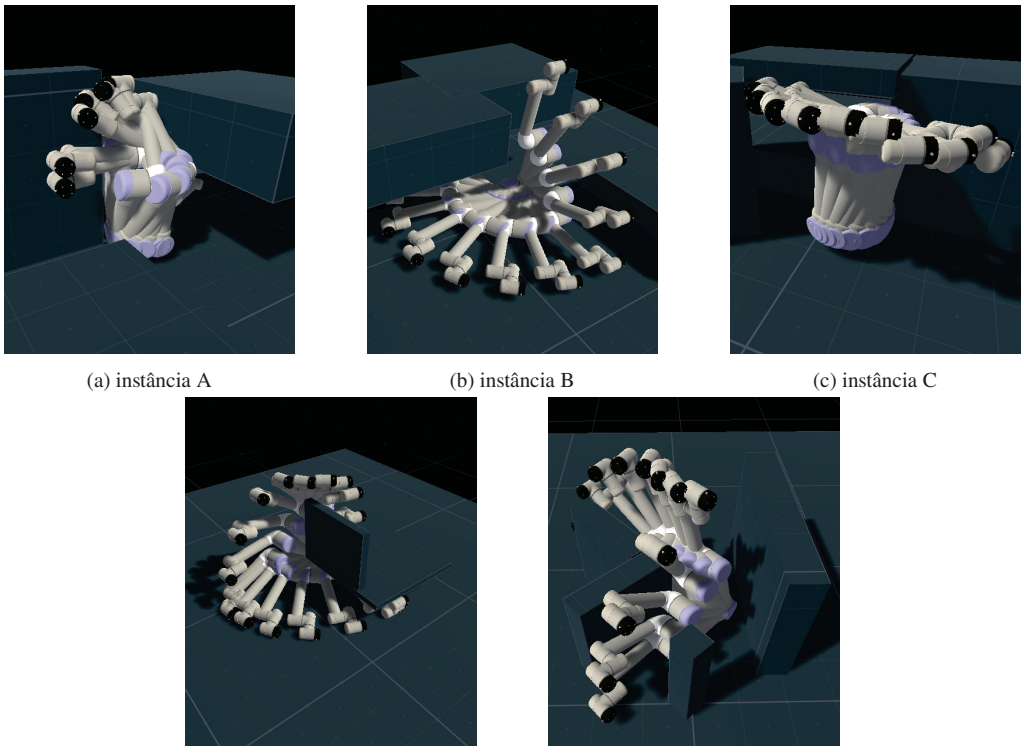
Após a execução da solução para as instâncias apresentadas na Seção 5.6, foi possível constatar que, para a granularidade definida (360 passos em cada um dos 3 eixos) obteve-se um



(a) Quantidade de obstáculos: 5. (b) Estado inicial: (315, 90, 0). (c) Estado objetivo: (45, 0, 90).

Figura 5.11: Estados do manipulador e obstáculos para a instância E.

caminho válido, como demonstrado na Figura 5.12, através da sobreposição da imagem do robô em certos intervalos de tempo da execução do caminho encontrado.



(a) instância A (b) instância B (c) instância C

(d) instância D (e) instância E

Figura 5.12: Caminhos obtidos pelo algoritmo proposto, para todas as instâncias apresentadas.

A Tabela 5.1 apresenta dados sobre o tempo de execução da solução proposta, para cada umas das instâncias apresentadas, para três granularidades diferentes. É possível perceber, através dos dados informados, a direta relação entre a o tempo de execução da solução em relação a granularidade. Isso é relevante uma vez que é necessário fazer o balanço entre a confiança do caminho encontrado e o tempo de execução. Os tempos em vermelho, destacados na tabela, representam combinações de instâncias e granularidades em que o caminho encontrado, a depender da interpolação utilizada para calcular o movimento entre passos, pode resultar em colisões não calculadas.

	90 passos		180 passos		360 passos	
Instâncias	C-Space	Busca	C-Space	Busca	C-Space	Busca
A	10,99s	0,02s	84,14s	0,20s	665,47s	1,36s
B	12,8s	0,03s	100,9s	0,25s	776,0s	1,86s
C	11,4s	0,03s	88,58s	0,26s	683,8s	2,3s
D	11,4s	0,05s	89,3s	0,39s	689,0s	3,17s
E	9,99s	0,03s	76,6s	0,27s	588,9s	1,90s

Tabela 5.1: Tempo de execução, separado por parte da computação, para todas as instâncias, comparando três granularidades de passos. Em vermelho combinações de instância e granularidade que podem não resultar um caminho livre de colisões.

6 CONSIDERAÇÕES FINAIS

Como demonstrado na Seção 5.7 a solução proposta conseguiu, para os cenários de teste apresentados, ou computar uma rota para satisfazer o problema de mover o manipulador de uma configuração inicial até um configuração final ou determinar que tal caminho não é viável.

Após a implementação da solução proposta e a execução nos cenários de teste é possível perceber que a abordagem necessita de refinamento para uso em ambientes reais, uma vez que o tempo de espera necessário para obter a resposta, devido ao custo computacional dos algoritmos utilizados, deixa a desejar para um uso em tempo real.

Além disso, sem uma maneira de verificar se as configurações intermediárias (oriundas da granularidade da discretização para o espaço de configurações) colidem com os obstáculos, faz com que a solução não seja completamente confiável sem conhecimento prévio do tamanho dos obstáculos e os volumes percorridos entre cada passo das configurações.

6.1 TRABALHOS FUTUROS

Existem diversas modificações que podem ser feitas para alterar e possivelmente aprimorar o comportamento da implementação em outras situações, como por exemplo implementação direta dos testes de colisão, sem depender da ferramenta Unity, utilizar outros formatos de robôs manipuladores e testar estratégias para contornar a limitação da granularidade e tamanho dos obstáculos, como granularidade dinâmica e hierárquica, e teste de colisão com a interpolação do movimento entre configurações.

A computação do espaço de configurações durante o processo de busca, e não antes, poderia permitir um ganho de performance uma vez que em certos casos seria possível encontrar um caminho sem explorar todo o espaço de configurações. Outra opção para diminuir o custo computacional para cada consulta de caminho seria manter uma estrutura com o espaço de configurações e executar novamente apenas o algoritmo de busca.

Com a estrutura atual, seria possível, caso o ponto inicial seja o mesmo, apenas escolher outra configuração objetivo e reconstruir o caminho a partir da estrutura obtida com a busca em largura modificada.

REFERÊNCIAS

Afrisal, H., Setiyono, B., Yusuf, M. F., Suin, R. M. e Toirov, O. (2020). Trajectory planning with obstacle avoidance of 3 DoF robotic arm for test tube handling system. Em *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. IEEE.

Ahuactzin, J. M., Gupta, K. e Mazer, E. (1998). Manipulation planning for redundant robots: A practical approach. *The International Journal of Robotics Research*, 17(7):731–747.

Amato, N. e Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. Em *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, páginas 113–120 vol.1.

Ando, S. (2003). A fast collision-free path planning method for a general robot manipulator. Em *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, páginas 2871–2877 vol.2.

Barraquand, J. e Latombe, J.-C. (1990). A monte-carlo algorithm for path planning with many degrees of freedom. Em *Proceedings. IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press.

Barraquand, J. e Latombe, J.-C. (1991). Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649.

Bertram, D., Kuffner, J., Dillmann, R. e Asfour, T. (2006). An integrated approach to inverse kinematics and path planning for redundant manipulators. Em *Proceedings. International Conference on Robotics and Automation*, páginas 1874–1879. IEEE.

Brooks, R. A. (1983). Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(2):190–197.

Craig, J. (2005). *Introduction to robotics : mechanics and control*. Pearson/Prentice Hall, Upper Saddle River, N.J.

Elbanhawi, M. e Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77.

Faverjon, B. (1984). Obstacle avoidance using an octree in the configuration space of a manipulator. Em *Proceedings. 1984 IEEE International Conference on Robotics and Automation*, volume 1, páginas 504–512.

Faverjon, B. (1986). Object level programming of industrial robots. Em *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, páginas 1406–1412.

Faverjon, B. (1989). Hierarchical object models for efficient anti-collision algorithms. Em *Proceedings, 1989 International Conference on Robotics and Automation*, páginas 333–340 vol.1.

Fedele, G., D'Alfonso, L., Chiaravalloti, F. e D'Aquila, G. (2017). Obstacles avoidance based on switching potential functions. *Journal of Intelligent & Robotic Systems*, 90(3-4):387–405.

Gao, M., Chen, D., Yang, Y. e He, Z. (2015). A fixed-distance planning algorithm for 6-DOF manipulators. *Industrial Robot: An International Journal*, 42(6):586–599.

Groover, M. (1986). *Industrial robotics : technology, programming, and applications*. McGraw-Hill, New York, NY.

Hachenberger, P. (2013). 3d Minkowski sum of polyhedra. *CGAL User and Reference Manual. CGAL Editorial Board*, 4(1):569–578.

Hamada, K. e Hori, Y. (1996). Octree-based approach to real-time collision-free path planning for robot manipulator. Em *Proceedings of 4th IEEE International Workshop on Advanced Motion Control - AMC '96 - MIE*, volume 2, páginas 705–710 vol.2.

Hsu, D., Latombe, J.-C. e Motwani, R. (1997). Path planning in expansive configuration spaces. Em *Proceedings of International Conference on Robotics and Automation*, volume 3, páginas 2719–2726 vol.3.

Hwang, Y. e Ahuja, N. (1992). A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32.

Kavraki, L. e Latombe, J.-C. (1994). Randomized preprocessing of configuration space for path planning: articulated robots. Em *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, volume 3, páginas 1764–1771 vol.3.

Kavraki, L., Svestka, P., Latombe, J.-C. e Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580.

LaValle, S. M. (1998). Rapidly-exploring random trees : a new tool for path planning. *The annual research report*.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

LaValle, S. M. e James J. Kuffner, J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400.

Lozano-Perez (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120.

Lozano-Perez, T. (1981). Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(10):681–698.

Lozano-Pérez, T. e Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570.

Open Robotics (2022). Robotic operating system - industrial. `http://wiki.ros.org/Industrial`.

Patel, S., Sobh, T. e Mahmood, A. (2014). Goal directed synthesis of serial manipulators based on task descriptions. Em *Chaos Modeling and Control Systems Design*, páginas 319–350. Springer International Publishing.

Ramer, C., Reitelshofer, S. e Franke, J. (2013). A robot motion planner for 6-DOF industrial robots based on the cell decomposition of the workspace. Em *IEEE ISR 2013*. IEEE.

Rybus, T. (2018). Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. *Progress in Aerospace Sciences*, 101:31–48.

Schwartz, J. e Sharir, M. (1988). A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37(1-3):157–169.

Unity Technologies (2021). Unity. `https://unity.com`.

Wang, W., Zhu, M., Wang, X., He, S., He, J. e Xu, Z. (2018). An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators. *International Journal of Advanced Robotic Systems*, 15(5).

Zhang, H., Zhuang, Q. e Li, G. (2022). Robot path planning method based on indoor spacetime grid model. *Remote Sensing*, 14(10):2357.

Švestka, P. e Overmars, M. H. (1997). Motion planning for carlike robots using a probabilistic learning approach. *The International Journal of Robotics Research*, 16(2):119–143.