João Roberto S. Benevides

Planejamento de Rota para Manipulador Planar de Base Livre Flutuante utilizando o Algoritmo RRT

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Ciências, Programa de Engenharia Elétrica.

Área de Concentração: Sistemas Dinâmicos
Orientador: Prof. Dr. Valdir Grassi Junior

São Carlos
2015

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

B461p Benevides, João Roberto Soares
Planejamento de Rota para Manipulador Espacial
Planar de Base Livre Flutuante utilizando o Algoritmo
RRT / João Roberto Soares Benevides; orientador Prof.
Dr. Valdir Grassi Junior. São Carlos, 2015.

Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia Elétrica e Área de Concentração em
Sistemas Dinâmicos -- Escola de Engenharia de São
Carlos da Universidade de São Paulo, 2015.

1. Planejamento de Rota. 2. RRT. 3. Manipulador
Espacial de Base Livre Flutuante. 4. Manipulador
Dinamicamente Equivalente. 5. Robótica Espacial. I.
Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **JOÃO ROBERTO SOARES BENEVIDES.**

Título da dissertação: "Planejamento de rota para manipulador espacial planar de base livre flutuante utilizando o algoritmo RRT".

Data da defesa: 27/02/2015

Comissão Julgadora:

Resultado:

Prof. Dr. **Valdir Grassi Junior (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Associado **Adriano Almeida Gonçalves Siqueira**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Bruno Vilhena Adorno**
(Universidade Federal de Minas Gerais/UFMG)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **Luis Fernando Costa Alberto**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo César Lima Segantine**

Dedicatória

Dedico esta dissertação aos meus pais, Luciana e Cláudio e minhas irmãs, Ana Cláudia e Ana Beatriz.

Agradecimentos

Agradeço a Deus por sempre me guiar em todos os momentos difíceis ao longo dessa jornada.

Agradeço a minha família, por sempre apoiar minhas decisões e me dar forças para seguir em frente.

À minha companheira de todas as horas, Rayza Araújo, pelo apoio e carinho incondicionais.

Ao meu orientador, Valdir Grassi Junior, pela paciência, pela colaboração, pelas valiosas discussões e por ter acreditado em mim.

Aos grandes amigos que sempre estiveram ao meu lado, mesmo que distantes geograficamente: Maciel Gomes, Henrique Ferreira, Kim Landel, Jales Monteiro, Tancredo Augusto, Davi Castro e Igor Prado.

Aos meus amigos de mestrado, que muito me ajudaram a chegar nesse momento: Maurício Nakai, Mauricio Escalante, Elmer Gamboa, Solander Agostinho, Christoph Mitschka, Hiparco Lins e todos os membros do LASI.

Finalmente, agradeço à Universidade de São Paulo, pela concessão da estrutura e dos meios necessários para a realização desse trabalho, bem como a Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) pela concessão da bolsa de mestrado (Edital RH-Mestrado 003/2013).

*“A história da ciência mostra que
teorias são perecíveis. A cada nova
verdade revelada, adquirimos uma
melhor compreensão da natureza e
nossas concepções e visões são
modificadas.”*

Nikola Tesla

Resumo

Benevides, J. R. S. Planejamento de Rota para Manipulador Planar de Base Livre Flutuante utilizando o Algoritmo RRT. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

Como tópico de fundamental importância na robótica, o planejamento de rotas tem encontrado excelentes resultados nos últimos anos através da utilização de algoritmos baseados no conceito de árvore de exploração rápida, RRT. No entanto, a aplicação desses métodos em sistemas robóticos espaciais revela um cenário ainda a ser explorado. O comportamento não-holonômico e a presença de singularidades dinâmicas são alguns fatores que dificultam a consideração de obstáculos no planejamento de rotas desses sistemas. Além disso, os trabalhos relacionados ao planejamento de movimento para manipuladores espaciais mostram-se concentrados na estratégia ponto-a-ponto, com interesse especial nos aspectos particulares da dinâmica desses sistemas. De modo geral, para manipuladores espaciais, o planejamento de trajetória envolvendo o desvio de obstáculos depende de uma rota previamente computada. Contudo, essa tarefa carece de formulações ou técnicas solidificadas, sobretudo para manipuladores espaciais de base livre flutuante. Com esta motivação, o trabalho proposto nesta dissertação de mestrado cria um planejador de rotas com suporte a desvio de obstáculos para um manipulador espacial planar de base livre flutuante. O modelo dinâmico utilizado é baseado no conceito de manipulador dinamicamente equivalente e incorporado a um algoritmo baseado no conceito de RRT.

Palavras-Chave: *Planejamento de Rotas, RRT, Robótica Espacial, Manipulador de Base Livre Flutuante, Manipulador Dinamicamente Equivalente.*

Abstract

Benevides, J. R. S. Path Planning for a Free-Floating Planar Space Manipulator using the RRT algorithm. Dissertation (Master's Degree) - Engineering School of Sao Carlos, University of Sao Paulo, Sao Carlos, 2015.

As major challenge in the field of robotics, path planning has experienced successful results in recent years by means of the RRT algorithm. However, the application of such algorithms in space manipulators reveals a scenario yet to be explored. The non-holonomic behavior, added to the presence of dynamic singularities are only a few factors that make collision-avoidance path planning of these systems such a hard task. Besides, works in the field of motion planning of space manipulators often concentrate in the strategy point-to-point, with particular interest in the complex dynamics of such systems. As a rule of thumb, collision-avoidance for space manipulators depends on a previous computed path. However, this task still lacks robust formulations, specially in the case of free-floating manipulators. With this motivation, the proposed work creates a collision-avoiding path planning for a free-floating planar manipulator. The dynamic model is based on the Dynamically Equivalent Manipulator and the concept of Rapidly-Exploring Random Trees serves as a frame for the developed algorithm.

Keywords: *Path Planning, RRT, Space Robotics, Free-Floating Manipulator, Dynamically Equivalent Manipulator.*

Lista de Figuras

FIGURA 1.1	Manipulador espacial em uma tarefa de manutenção do telescópio Hubble.	4
FIGURA 3.1	Manipulador UARME.	19
FIGURA 3.2	Manipulador de três elos.	21
FIGURA 3.3	Ilustração de um problema de cinemática inversa.	23
FIGURA 3.4	Representação do i -ésimo elo do ME (Pazelli; 2011)	24
FIGURA 3.5	Sistemas de coordenadas fixas nos elos do ME (Liang et al.; 1996).	27
FIGURA 3.6	Representação do i -ésimo elo do MDE (Pazelli; 2011)	28
FIGURA 4.1	Em 4.1(a), os estados inicial e final são definidos como x_{init} e x_{goal} , respectivamente. Uma área de tolerância é definida ao redor de x_{goal} . Em 4.1(b), uma configuração aleatória x_{rand} é amostrada no espaço de configuração livre e então é selecionado o nó mais próximo em T , x_{near} . Em 4.1(c), x_{near} gera primitivas de movimento para então avaliar a expansão que possui menor custo a x_{rand} . Finalmente, em 4.1(d), x_2 é adicionado como novo estado de T , tendo o ramo e_1 conectando x_1 a x_2 . O processo é repetido iterativamente até que uma solução seja encontrada ou o número máximo de tentativas seja ultrapassado.	32
FIGURA 4.2	Crescimento de duas RRTs. As árvores representadas em azul e vermelho encontram um nó suficientemente próximo, representado em verde, para então encontrar uma solução para o problema de planejamento.	40

FIGURA 5.1	A representação ilustrativa da aplicação da soma e diferença de Minkowski em um cenário composto de um obstáculo triangular A na presença de um robô de raio R é mostrado na figura 5.1(a). O resultado da aplicação do método pode ser verificado na figura 5.1(b).	43
FIGURA 5.2	Representação ilustrativa do espaço de trabalho de um manipulador de três elos. O valor de z depende dos valores de W_1 , W_2 e W_3	44
FIGURA 5.3	Representação ilustrativa da diferença geométrica entre o MDE e o ME.	45
FIGURA 5.4	Representação ilustrativa do teorema dos eixos separadores . . .	46
FIGURA 5.5	Fluxograma resumido do planejador proposto	53
FIGURA 6.1	Resposta típica encontrada pelo planejador, destacada em vermelho.	58
FIGURA 6.2	Resposta do planejador de trajetória	62
FIGURA 6.3	A geometria do UARM-E é destacada em vermelho. O MDE aparece como um segmento de reta azul. O obstáculo segue representado na cor preto. Finalmente, uma região de tolerância é definida ao redor do ponto de destino e representada na cor verde.	63
FIGURA 6.4	Limitações na expansão da árvore RRT, indicando a presença de mínimos locais.	63
FIGURA 6.5	Em 6.5(a), a árvore possui somente 1000 nós. A árvore mostrada em 6.5(b) é composta por 5000 nós. Em 6.5(c), 15000 nós estão densamente distribuídos. Em 6.5(d), 25000 nós constituem a árvore melhor expandida.	65
FIGURA 6.6	Crescimento do custo computacional com o número de iterações	66
FIGURA 6.7	A figura 6.7(a) mostra uma expansão com 1000 nós, em que não há $dist(e_1, e_2) < \delta$. A figura 6.7(b) mostra a mesma expansão após 10000 nós.	67

FIGURA 6.8 Na figura 6.8(a), é aplicado um viés $p = 100\%$ por 1000 iterações.

Em 6.8(b), 2000 iterações são utilizadas e nesse caso $p = 5\%$. Para a figura

6.8(c), $p = 10\%$ e a solução da cinemática inversa é fixa. Finalmente, a

figura 6.8(d) une duas posições dentro de 10000 iterações, com $p = 10\%$ e

solução de cinemática inversa aleatória. 68

FIGURA 6.9 Expansão após 50000 nós. 69

FIGURA 6.10 As figuras 6.10(a),6.10(c) e 6.10(e) representam expansões com

16445, 7280 e 4916 nós, respectivamente. As rotas associadas a essas figu-

ras foram conectadas pelo planejador a partir de 90, 89 e 111 nós, respec-

tivamente. A rota computada é representada pelos nós verdes nas figuras

6.10(b),6.10(d) e 6.10(f) 71

Lista de Tabelas

TABELA 3.1	Parâmetros DH para um manipulador planar de três elos	21
TABELA 3.2	Parâmetros do ME(UARM-E)	29
TABELA 3.3	Parâmetros do MDE(UARM-E)	29
TABELA 4.1	Comparação entre algoritmos num espaço livre binário. (adap- tada de (Urmson e Simmons; 2003))	37
TABELA 5.1	Organização das entradas discretas	50

Lista de Abreviaturas e Siglas

DH - Denavit-**H**artenberg

EVA - do inglês, ***E**xtra-**V**ehicular **A**ctivity*

LASI - **L**aboratório **S**istemas **I**nteligentes

MBF - Manipulador de **B**ase **F**ixa

MBLC - Manipulador Espacial de **B**ase **L**ivre **C**ontrolada

MBLF - Manipulador Espacial de **B**ase **L**ivre **F**lutuante

MDE - Manipulador **D**inamicamente **E**quivalente

ME - Manipulador **E**spacial

MV - Manipulador **V**irtual

PRM - do inglês, ***P**robabilistic **R**oad**M**aps*

RRT - do inglês, ***R**apidly-exploring **R**andom **T**ree*

SAT - do inglês, ***S**eparating **A**xis **T**heorem*

UARM-E - do inglês, ***U**nder**A**ctuated **R**obot **A**r**M***, acrescido do sufixo ***E***, que remete a **E**spacial

ZRM - do inglês, ***Z**ero **R**eaction **M**aneuver*

Lista de Símbolos

X_{free}	espaço de configurações livre do robô
X_{obs}	região de obstáculos fixos
q	variável de junta
q^d	variável desejada de junta
θ_i	variável de junta rotacional
θ_0	posição angular de origem
θ_f	posição angular de fim
a_i	comprimento do i -ésimo elo
α_i	ângulo entre os eixos das juntas $i - 1$ e i
d_i	distância no eixo i entre dois eixos comuns de conexão
$M(q)$	matriz de inércia
$C(q, \dot{q})$	matriz de forças centrífugas e de Coriolis
$g(q)$	matriz de forças gravitacionais
C_i	centro de massa do i -ésimo elo para o MBF
C_0	centro de massa total do MDE
S_i	vetor entre C_0 e C_i
P_i	eixo de rotação da i -ésima junta
J_i	i -ésima junta do MBF
J'_i	i -ésima junta do MDE
L_i	vetor que conecta J_i e C_i
R_i	vetor que conecta C_i e J_{i+1}
W_i	vetor que conecta as juntas J'_i e J'_{i+1}

m'_i	massa do i -ésimo elo do MDE
I'_i	tensor inercial do i -ésimo elo do MDE
l_{ci}	vetor da i -ésima junta do MDE
τ	torque
x_{init}	estado inicial na RRT
x_{rand}	estado amostrado aleatoriamente na RRT
x_{near}	estado mais próximo de x_{rand} na RRT
x_{goal}	estado desejado na RRT
x_{new}	estado novo na RRT
ρ	função métrica empregada
u	vetor de entradas aleatórias
U	conjunto de entradas possíveis para um estado
p	viés
N	número de entradas em U
h	número de entradas que levam a uma colisão
FVR	frequência de violação de restrição
Γ	limiar de iterações
ε	mínimo valor de proximidade
X_{goal}	região de interesse contendo o estado final x_{goal}
Δ_i	incremento de τ_i
k	grau de discretização
U_s	conjunto de entradas avaliadas para o mesmo estado
x_i	conjunto de coordenadas das juntas e do efetuador final referente à configuração i
D_i	posição cartesiana da junta i do manipulador
σ_i	distância euclidiana entre a posição da junta i para dois estados distintos
v	velocidade angular da junta
ΔT	passo de integração

Sumário

1	Introdução	3
2	Planejamento de Movimento para Robôs Manipuladores	9
2.1	Planejamento de Movimento para Manipuladores Fixos	10
2.2	Planejamento de Movimento para Manipuladores Espaciais	11
2.2.1	As Singularidades Dinâmicas	13
2.2.2	O Desvio de Obstáculos	14
3	Manipulador Espacial de Base Livre Flutuante	17
3.1	Robô Espacial UARM-E	18
3.2	Modelagem para um manipulador de base fixa	19
3.2.1	Cinemática direta	20
3.2.2	Cinemática inversa	22
3.2.3	Modelo dinâmico	24
3.3	Modelagem para um manipulador de base livre	24
3.4	Manipulador Dinamicamente Equivalente	26
3.4.1	Modelo dinâmico de um manipulador espacial de base livre flutuante	26
4	Árvore Aleatória de Exploração Rápida (RRT)	31
4.1	O Algoritmo RRT básico	31
4.2	Propriedades e Vantagens do Algoritmo RRT	33
4.3	Aperfeiçoamento do algoritmo RRT	35
4.3.1	O problema de convergência	36
4.3.2	Sobrecarregamento computacional	37
4.3.3	Redução da sensibilidade métrica	38
4.3.4	RRTs bidirecionais	39
4.3.5	Considerações finais	40

5	Desenvolvimento de um planejador de trajetórias RRT	41
5.1	Descrição do ambiente de simulação	41
5.1.1	O ambiente de testes inicial	42
5.1.2	O ambiente de testes para o MDE	44
5.1.3	O ambiente com suporte à detecção de colisão	45
5.2	Implementação e avaliação do “método das caixas”	47
5.3	Implementação do método de redução de sensibilidade da métrica	49
5.4	Implementação do método das árvores bidirecionais	51
5.5	Funcionamento do planejador	53
6	Resultados	57
6.1	Convergência para o robô pontual	57
6.2	Convergência para o MDE	58
6.3	Avaliação do método da redução de sensibilidade métrica	62
6.4	Avaliação do método das árvores bidirecionais	66
7	Conclusão	73
	Conclusões	73
7.1	Conclusões Gerais	73
7.2	Trabalhos Futuros	74
8	Anexos	75
8.1	Anexo 1	75
	Referências	77

Capítulo 1

Introdução

Missões espaciais envolvem por diversas vezes ambientes hostis, onde temperaturas extremas, radiação e gravidade tornam perigoso e complicado o acesso humano em atividades extra-veiculares - EVAs (do inglês, *Extra-Vehicular Activities*). Com o objetivo de auxiliar em serviços de construção e manutenção de estruturas e módulos espaciais, robôs manipuladores têm desempenhado um papel fundamental em tais operações. Tarefas como substituição de componentes, reabastecimento de combustível e reparo em satélites são fundamentais no sentido de tornar as operações em órbita mais flexíveis e proporcionar uma maior vida útil à missão (Yoshida; 2009). Essa motivação tem levado pesquisadores a explorar mais profundamente o potencial dos manipuladores espaciais.

Um dos maiores desafios no campo da robótica espacial está no planejamento de movimento (Choset et al.; 2005). Além do acoplamento dinâmico, um dos principais obstáculos encontrados para o planejamento e controle destes robôs está em seu comportamento não-holonômico, resultante da não-integrabilidade do momento angular (Li et al.; 2006). Diversos métodos e considerações foram propostos no intuito de contribuir nessa área (Sakawa; 1999; Feihu; 2013; Nanos e Papadopoulos; 2012; Dahl et al.; 2011). Sistemas atuantes em robôs manipuladores que planejam trajetórias livres de colisão não se encontram disponíveis comercialmente (Craig; 2005). De maneira geral, seu objetivo consiste em especificar uma tarefa através de uma linguagem de alto nível, de tal maneira que o robô automaticamente compile esta especificação em um conjunto de primitivas de baixo nível ou controles de retroalimentação (Choset et al.; 2005).

Figura 1.1: Manipulador espacial em uma tarefa de manutenção do telescópio Hubble.

No âmbito do planejamento de movimentos para manipuladores espaciais, alguns trabalhos propuseram soluções a fim de minimizar os distúrbios dinâmicos gerados pelo movimento do braço. Yoshida et al. (2001) propuseram o conceito de Manobra de Reação Zero - ZRM (do inglês, *Zero Reaction Maneuver*). Segundo essa formulação, teoricamente não existirá efeito sobre a base durante o movimento do manipulador espacial e isto foi verificado experimentalmente em órbita, com auxílio de testes realizados no manipulador espacial japonês ETS-VII. Entretanto, uma particularidade da ZRM está em sua aplicação bastante limitada ao lidar com manipuladores não-reduntantes. Wu et al. (2009) exploram esse ponto e parametrizam as equações do manipulador levando em conta as restrições cinemáticas e dinâmicas. Os parâmetros desconhecidos são calculados através de algoritmos genéticos que objetivam minimizar a reação da base.

De modo a facilitar a descrição do espaço de configurações livre do robô, X_{free} , diferentes planejadores de movimento foram propostos até meados da década de 90. Uma vez definido o ambiente de trabalho, o planejador estaria apto a construir mapas de rotas, isto é, caminhos que eventualmente levariam à solução de um problema de planejamento. Contudo, esses métodos dependiam da representação explícita de X_{free} e dessa forma, à medida que a dimensão do espaço de configuração crescia, os modelos de planejadores tornavam-se impraticáveis devido ao alto custo computacional (Choset et al.; 2005). Foi então que o enorme potencial dos métodos baseados em amostragem começou a ser investigado. Tais métodos baseiam-se no fato de que é mais vantajoso, em termos com-

putacionais, verificar se uma única configuração do robô encontra-se em X_{free} ou não.

Pesquisas relacionadas ao tema levaram à abordagem dos mapas de rota probabilísticos, PRM (do inglês, *Probabilistic Roadmaps*), para isso, um grafo é construído no espaço de configuração por meio da geração de configurações aleatórias. Posteriormente, é feita uma tentativa de conectar pares de configurações vizinhas por meio de um planejador local. Apesar de existir garantias formais de exploração global em termos de completude probabilística, o problema de conexão pode ser tão complexo quanto o projeto de um controlador não-linear, especialmente em sistemas dinâmicos e não-holonômicos (LaValle; 1998). Numa outra categoria figuram os campos potenciais artificiais, descritos em (Barraquand e Latombe; 1991). Entretanto, o método depende altamente da escolha de uma boa função potencial heurística, o que torna uma tarefa desafiadora quando combinada a obstáculos e restrições de cinemática e dinâmica diferenciais. Vale ressaltar que os dois métodos mencionados aqui possuem, em geral, uma visão local do ambiente e portanto o planejador pode, eventualmente, conduzir a exploração à mínimos locais (Craig; 2005).

Seguindo essa tendência, trabalhos significantes foram propostos no campo dos algoritmos baseados em amostragem (Bohlin e Kavraki; 2000; Hsu et al.; 1997; LaValle; 1998). O conceito de Árvores Aleatórias de Exploração Rápida - RRT (do inglês, *Rapidly-Exploring Random Tree*) foi apresentado em (LaValle; 1998) e destaca-se por conceber várias propriedades presentes nas técnicas de planejamento aleatórias já existentes. Ademais, esta técnica foi idealizada para lidar restrições não holonômicas e leva em conta maiores graus de liberdade.

O algoritmo RRT foi idealizado a fim de suplantiar os déficits apresentados pelos métodos recorrentes de planejamento utilizados até a época de sua publicação. O conceito-chave por trás do método RRT está em direcionar a exploração rumo a seções do ambiente ainda não exploradas, através de um algoritmo simples e ao mesmo tempo eficiente. Basicamente, essa técnica consiste em expandir uma árvore de exploração a partir do estado de origem do sistema robótico até o estado destino desejado. Para atingir esse objetivo, amostra-se sucessivos estados aleatórios em X_{free} . Busca-se então o estado (também chamado de nó) existente mais próximo para então expandi-lo rumo àquele estado amostrado. O processo é repetido até que a condição final seja satisfeita ou um tempo pré-definido

seja atingido (LaValle; 1998; LaValle e Kuffner Jr.; 1999; Kuffner Jr. e LaValle; 2000). Da mesma forma que os mapas de rota probabilísticos, o algoritmo RRT foi demonstrado ser probabilisticamente completo, isto é, a probabilidade de encontrar uma solução (se existir) converge para 1 na medida em que o tempo cresce ao infinito Kuffner Jr. e LaValle (2000),.

No que tange às classes de manipuladores espaciais, Dubowsky e Papadopoulos (1993) categorizam o manipulador espacial de base livre flutuante (MBLF) como aquele que não possui ou não opera com jatos propulsores e/ou não utiliza rodas de reação na base. Essa característica permite, por sua vez, conservar combustível e energia elétrica, possibilitando assim missões espaciais mais auto-suficientes. O protótipo de um manipulador espacial planar de base livre flutuante foi desenvolvido em (Pazelli; 2011) e encontra-se no Laboratório de Sistemas Inteligentes (LASI) da USP em São Carlos. O sistema de flutuação é baseado na técnica de colchões de ar e o manipulador distingue-se por possuir uma estrutura mecânica modular que possibilita a configuração de uma a seis juntas ativas, com um ou dois braços. Além disso, o tipo de motor selecionado possibilita o estudo dos manipuladores com juntas passivas. Essa característica é especialmente interessante para robôs espaciais por possibilitar aplicações em procedimentos envolvendo economia de energia, ou ainda, tratamento de falhas. A eletrônica embarcada no robô permite o controle de posição e velocidade das juntas. Um computador remoto é responsável pelo gerenciamento dos algoritmos para controle de posição e trajetória do sistema robótico. Esse terminal é responsável pela aquisição dos dados e envio de instruções através de um protocolo de comunicação sem fio.

Com base nos tópicos mencionados anteriormente, o objetivo deste trabalho consiste essencialmente no desenvolvimento de um planejador de rota autônomo para um manipulador espacial de base livre flutuante. Por autônomo, entende-se um planejador capaz de retornar uma solução livre de colisão entre duas configurações no espaço, usando para isso somente o conhecimento do mapa e das características do robô. Para alcançar o objetivo, será utilizado o método RRT e suas variantes. O desempenho, assim como os principais aspectos desse método, são avaliados em um manipulador planar de base livre flutuante, simulado em ambiente MATLAB. O sistema proposto é configurado e avaliado para um espaço finito e leva em conta a presença de obstáculos fixos.

O trabalho segue conforme a seguinte organização:

- O **Capítulo 2** faz uma revisão sobre as principais referências a respeito de planejamento de movimentos para manipuladores espaciais e por fim contextualiza a importância deste trabalho.
- O **Capítulo 3** apresenta o robô UARM-E e descreve os modelos cinemáticos e dinâmicos para manipuladores de base fixa. O modelo dinamicamente equivalente é introduzido como uma alternativa aos complexos equacionamentos do manipulador espacial.
- O **Capítulo 4** introduz o algoritmo RRT básico e seu funcionamento. Além disso, trabalhos relacionados ao aperfeiçoamento do algoritmo base são abordados.
- O **Capítulo 5** descreve a metodologia empregada no desenvolvimento do planejador. Os ambientes de simulação são descritos, bem como as técnicas e heurísticas utilizadas para integrar todas as modificações propostas.
- O **Capítulo 6** avalia as etapas fundamentais do projeto e promove uma análise dos resultados.
- O **Capítulo 7** finaliza o trabalho com as conclusões referentes a essa dissertação e sugestões de trabalhos futuros.

Capítulo 2

Planejamento de Movimento para Robôs Manipuladores

A utilização do termo “planejamento de movimento” no título deste capítulo tem por objetivo englobar as contribuições feitas no planejamento de rota e trajetória de manipuladores. Estes tipos de planejamento se diferenciam em termos de conceito da seguinte forma. O planejamento de trajetória especifica detalhadamente a rota juntamente com as especificações temporais ao longo de um percurso. Em outras palavras, tem por objetivo calcular uma função $q^d(t)$ que percorra o caminho desejado e satisfaça completamente o movimento do robô (Spong et al.; 2005). Em contraste, o planejador de rota tem por objetivo encontrar um percurso necessário para se chegar em um destino a partir de uma configuração inicial, levando em consideração apenas as especificações espaciais.

Com o objetivo de introduzir os conceitos fundamentais utilizados nos trabalhos relacionados a planejamento de movimento de manipuladores espaciais, o capítulo inicia com uma breve revisão sobre as abordagens tradicionais para manipuladores fixos. Posteriormente, o planejamento para manipuladores espaciais é abordado, mostrando as contribuições propostas por trabalhos relacionados e as principais dificuldades encontradas na resolução desta tarefa. Finalmente, conclui-se através da contextualização dos objetivos e contribuições do trabalho proposto nesta dissertação.

2.1 Planejamento de Movimento para Manipuladores Fixos

De maneira geral, o objetivo do planejador de trajetórias é capacitar o sistema a interpretar as necessidades do usuário a partir de instruções e parâmetros. Por exemplo, ao fornecer a posição e orientação desejadas do efetuador final, um planejador de trajetórias deve ser capaz de gerar comandos para cada junta, a fim de obedecer as especificações almejadas dentro de um intervalo de tempo definido.

Entretanto, frequentemente surge a necessidade de um maior detalhamento na especificação do movimento. Para desempenhar esse papel, atribui-se uma sequência de pontos intermediários entre as configurações inicial e final. Portanto, a ferramenta do robô deverá passar por cada um desses pontos intermediários a fim de completar o movimento. Na verdade, cada ponto representa um sistema de coordenadas especificando a posição e orientação da ferramenta em relação à base. A adição do fator tempo nesse planejamento implica encontrar uma função que conduza o manipulador, preferencialmente de forma suave, entre cada intervalo. Este método, chamado de ponto-a-ponto, é a maneira mais simples de se especificar o movimento de um manipulador (Kelly et al.; 2006).

O estudo de planejamento pode ser executado no espaço das juntas ou no espaço da tarefa. Entretanto, para que um robô seja controlado no espaço das juntas, faz-se necessário a conversão desde o espaço cartesiano através da cinemática inversa, tarefa que pode tornar-se extremamente complexa quando na presença de vários graus de liberdade e/ou obstáculos nas proximidades do robô. Por outro lado, o planejamento exclusivo no espaço cartesiano encontra empecilho por conta das singularidades do mecanismo, decorrentes da não-inversibilidade da matriz jacobiana. Esta matriz representa a transformação linear relacionando a velocidade das juntas com a velocidade do efetuador.

Quando o planejamento é executado no espaço das juntas, ao considerar um problema básico de movimentação da ferramenta entre duas configurações específicas dentro de um determinado intervalo de tempo, surgem restrições em $q(t)$:

$$\begin{aligned} q(0) &= \theta_0 & \dot{q}(0) &= 0 \\ q(t_f) &= \theta_f & \dot{q}(t_f) &= 0. \end{aligned} \tag{2.1}$$

As quatro restrições apresentadas na equação 2.1 podem sustentar uma função polinomial $q(t)$ de grau no mínimo igual a três:

$$q(t) = a_0 + a_1t + a_2t^2 + a_3t^3. \quad (2.2)$$

As derivações da equação 2.2 resultam nas funções velocidade e aceleração. Estas são combinadas à equação 2.1 e assim são encontrados os parâmetros desconhecidos para a construção da função polinomial cúbica. Maiores detalhes sobre a formulação de cinemática direta podem ser encontradas Spong et al. (2005).

Por outro lado, a tarefa de planejamento ponto-a-ponto, sem a necessidade de parar, exige que novos parâmetros sejam calculados a fim de obter uma expressão que conecte estes pontos de forma suave ao longo da trajetória. Nesse caso, os valores de velocidades intermediários são atribuídos através de alguma heurística ou de forma que resulte numa aceleração contínua nesses pontos. Nesse caso, um polinômio de ordem maior deve descrever a função de movimento.

2.2 Planejamento de Movimento para Manipuladores Espaciais

Os estudos conduzidos em manipuladores estão concentrados na categoria de base fixa (Dubowsky e Papadopoulos; 1993). Devido ao complexo equacionamento dinâmico dos manipuladores espaciais, os trabalhos relacionados ao planejamento de trajetória para esse tipo de robô focam em otimizações de determinados aspectos do movimento, deixando assim o campo bastante aberto para investigações e introdução de novas técnicas. Seja a minimização dos esforços de reação na base ou melhoria da estabilidade do efetuador final, desde o fim dos anos 1980, projetos de controle e técnicas de planejamento vêm sendo desenvolvidos com o objetivo de contornar as dificuldades da simulação de ambientes espaciais. As contribuições dos trabalhos incipientes, bem como as alternativas de abordagem dos planejadores que visam minimizar os esforços na base são tratados nessa seção.

De acordo com Ellery (2000), o planejamento de trajetória de manipuladores espaciais

envolve quatro etapas principais, a saber:

1. Calcular uma série de pontos no espaço cartesiano;
2. Converter estes pontos para o espaço das juntas através da cinemática inversa;
3. Ajustar uma curva quadrática de modo a unir os pontos no espaço das juntas;
4. Verificar limitações dos atuadores de entrada através de formulações dinâmicas.

A partir daí é então aplicado um algoritmo de geração de trajetória a fim de movimentar o efetuador final em uma linha reta, passando por pontos de interpolação. Essa abordagem obviamente ignora a presença de obstáculos no caminho, porém propõe uma fundamentação matemática uma vez que o caminho encontra-se planejado.

Dentre os trabalhos que iniciaram o estudo de planejamento de movimento para manipuladores espaciais, no geral, os seguintes merecem destaque. Apoiado no estudo da dinâmica de tais sistemas, métodos de planejamento de rota foram empregados em (Dubowsky e Torres; 1990) a fim de reduzir o consumo de combustível. Vafa (1990) propôs um algoritmo para eliminar as perturbações de atitude do satélite baseando-se no método do Manipulador Virtual. Um controle adaptativo foi projetado em (Walker e Wee; 1991) na intenção de obter uma trajetória globalmente estável do efetuador final. O emprego de técnicas de controle para manipuladores terrestres, analisado por Dubowsky e Papadopoulos (1993), teve por objetivo encontrar um planejamento de trajetória ótimo visando evitar a saturação dos propulsores, contudo o alto consumo de combustível limitou a aplicabilidade desta abordagem.

Objetivando a minimização do consumo de combustível, Sakawa (1999) desenvolveu um planejador de trajetória para um manipulador espacial de base livre controlada (MBLC) utilizando a teoria de controle ótimo nas equações de movimento não-lineares, de forma que uma dada condição terminal fosse obedecida no fim do trajeto. De maneira similar, o conceito de Manobra de Reação Zero (ZRM), apresentado em (Yoshida et al.; 2001), mostrou-se eficiente e válido para graus de liberdade superiores a seis. O ZRM tem por objetivo anular o deslocamento da base em MBLCs. Para este objetivo, os movimen-

tos das juntas são equacionados de forma a garantir essa propriedade, cuja validação foi executada posteriormente no manipulador espacial japonês ETS-VII.

Inspirado no ZRM, Wu et al. (2009) contribuíram com a extensão do conceito a robôs com menos graus de liberdade. Para isso, a função de movimento da junta é parametrizada de forma um pouco mais complexa que a equação 2.2, e é dada por:

$$\theta_i(t) = \Delta_{i1} \sin(a_{i0} + a_{i1}t + \dots + a_{i7}t^7) + \Delta_{i2}, \quad (2.3)$$

onde $0 < t < t_f$ e Δ_{i1} e Δ_{i2} são definidos por:

$$\Delta_{i1} = \frac{\theta_{i_max} - \theta_{i_min}}{2}, \Delta_{i2} = \frac{\theta_{i_max} + \theta_{i_min}}{2}, \quad (2.4)$$

no qual θ_{i_max} e θ_{i_min} representam os ângulos máximo e mínimo de cada junta i , respectivamente. Similarmente ao manipulador de base fixa (MBF), as derivações da equação 2.3 são combinadas a restrições de movimento consideradas, com o intuito de parametrizar os coeficientes da equação 2.3. Neste caso em especial, cinco coeficientes são parametrizados e somente dois permanecem como incógnitas. O trabalho utiliza algoritmos genéticos para encontrar esses parâmetros de modo a minimizar a reação da base.

2.2.1 As Singularidades Dinâmicas

Equivalentemente às singularidades de mecanismo encontradas em MFs, Papadopoulos e Dubowsky demonstraram a presença de singularidades dinâmicas em manipuladores de base livre flutuante (MBLFs) (Papadopoulos e Dubowsky; 1991, 1993). Contudo, a existência e localização das singularidades dinâmicas não podem ser previstas simplesmente pela estrutura do manipulador cinemático, isso acontece pois essas singularidades são agora produtos das propriedades dinâmicas do robô e dependentes do caminho tomado durante o trajeto.

Dada a importância das singularidades dinâmicas no projeto, planejamento e controle de MBLFs, diversos trabalhos relacionados ao planejamento de rota/trajetória para

MBLFs propuseram alternativas para contornar esse problema. Papadopoulos (1993) propõe um método que consiste em executar pequenos movimentos cíclicos de forma a alterar a atitude do robô para escapar de configurações onde sabidamente há singularidades dinâmicas. As contribuições apresentadas em (Tortopidis e Papadopoulos; 2006) permitem o controle cartesiano ponto-a-ponto do efetuador final juntamente com o controle de atitude do sistema utilizando apenas os atuadores do manipulador. A ideia básica para esta abordagem consiste em representar os ângulos das juntas por polinômios de ordem elevada como argumentos numa função cosseno. Já a técnica de planejamento de rota proposta em (Nanos e Papadopoulos; 2012) propõe uma abordagem no espaço da tarefa, na qual o efetuador final é capaz de evitar uma singularidade dinâmica qualquer. Para isso, parte-se de um caminho pré-computado para então avaliar as configurações iniciais do sistema que são apropriadas para evitar as singularidades dinâmicas durante o movimento. Dessa maneira, essa técnica permite o uso do espaço de trabalho do robô em sua plenitude.

2.2.2 O Desvio de Obstáculos

Para validar a eficiência de qualquer planejador de rota/trajetória, este deve lidar de maneira automática com o problema de obstáculos em seu ambiente de trabalho, desviar dos mesmos para então obter uma trajetória livre de colisão e, além disso, atingir um ponto de aproximação seguro para o efetuador final (Ellery; 2000). No entanto, em todos os trabalhos nos quais se propõem alternativas de modo a contornar as singularidades dinâmicas e evitar obstáculos, faz-se necessário o conhecimento do caminho a ser percorrido previamente ao planejamento da trajetória.

Nesse contexto, os métodos de mapas de rota probabilísticos (PRM), campos potenciais artificiais e árvores de exploração rápida (RRT) figuram entre os mais recorrentes no que se refere à aplicação em manipuladores robóticos de base fixa, MBFs. Como exemplo de trabalhos recentes envolvendo esses algoritmos em MBFs, Kim et al. (2014) apresentam um método baseado em árvores RRT para o planejamento de rota livre de colisão e oclusão, para sistemas com sete graus de liberdade. Já o trabalho proposto em (Malone et al.; 2013) propõe e avalia o método *Safety*-PRM, projetado para a construção de mapas de rota para o uso em espaço de trabalho sujeito a incertezas no modelo.

Esta incorporação das imprecisões é tratada por meio de modificações nos algoritmos de verificação de colisão, oferecendo caminhos com bom grau de confiabilidade ainda que na presença de ambientes imprecisos.

Para concluir, os trabalhos relacionados ao planejamento de trajetória e rota para manipuladores espaciais mostram-se concentrados na estratégia ponto-a-ponto, de modo a evitar as singularidades dinâmicas ou focando em aspectos particulares do movimento desses sistemas. Nos trabalhos mais gerais, o planejamento de trajetória envolvendo desvio de obstáculos depende de uma rota previamente computada, técnica que carece de formulações ou técnicas solidificadas para manipuladores espaciais. Com esta motivação, o trabalho proposto nesta dissertação de mestrado cria um planejador de rota com suporte a desvio de obstáculos para um manipulador espacial planar de base livre flutuante. Resumidamente, o modelo dinâmico será baseado no manipulador dinamicamente equivalente (MDE) que será utilizado como intermediário do manipulador espacial em um algoritmo baseado no conceito de RRT. Os conceitos utilizados, bem como o desenvolvimento e avaliação são descritos nos capítulos a seguir.

Capítulo 3

Manipulador Espacial de Base Livre Flutuante

Manipuladores espaciais, MEs, tem por característica principal o acoplamento dinâmico entre a base do sistema (satélite) e o braço robótico. Dessa maneira, o movimento gerado pelo braço acarreta movimentos coordenados na base espacial e vice versa. Há um certo número de problemas envolvendo o controle e dinâmica de sistemas robóticos espaciais que são inerentes a essa área, isto se dá graças à dinâmica complexa e única encontrada em várias aplicações robóticas no espaço. Dubowsky e Papadopoulos (1993) classificam tipos representativos de sistemas robóticos espaciais e identificam seus problemas característicos de planejamento e controle, sobretudo com enfoque nos manipuladores de base livre controlada e manipuladores de base livre flutuante.

Os manipuladores espaciais de base livre controlada, MBLCs, contam com controle ativo de posição e atitude através de jatos propulsores ou rodas de reação, os quais tem por função compensar o deslocamento gerado. Desta forma, a base mantém-se estacionária no espaço de trabalho e por conseguinte, a maioria das leis de controle para manipuladores terrestres são válidas (Dubowsky e Papadopoulos; 1993). Um dos desafios para o controle deste tipo de manipulador está, no entanto, nas perturbações causadas pelo movimento do mesmo. Estas perturbações afetam diretamente no controle de posição e atitude do robô e levam a um consumo excessivo de combustível dos propulsores responsáveis pelo controle de atitude. O consumo desenfreado deste recurso dispendioso e não-renovável, por sua vez, compromete a duração do tempo útil do sistema em órbita.

Em outra categoria, sistemas manipuladores de base livre flutuante, MBLFs, permitem

que o veículo espacial do manipulador se mova livremente em resposta aos movimentos do braço (Papadopoulos e Dubowsky; 1991). Uma vez que não há auxílio de jatos propulsores ou rodas de reação na base, o sistema é capaz de conservar energia elétrica e combustível. Esses sistemas possuem uma contrapartida de possuírem uma maior complexidade dos sistemas de controle e planejamento de movimento. Por essas razões, MBLFs têm recebido maior atenção da comunidade científica nos últimos tempos.

3.1 Robô Espacial UARM-E

O Laboratório de Sistemas Inteligentes (LASI-USP-São Carlos) conta atualmente com um protótipo de um manipulador de base livre flutuante, o UARM-E (Pazelli; 2011). Essa plataforma de testes consiste de um sistema mecânico, um sistema eletrônico e um terminal remoto responsável pelo ambiente de simulação e controle. A estrutura do sistema mecânico é planar e composta por uma base, juntas conectadas por elos e um efetuador. O módulo físico do robô flutua sobre um colchão de ar formado entre as bases de suas juntas e uma plataforma de apoio.

Os atuadores presentes no robô são responsáveis pelos movimentos dos elos do braço robótico. Esses atuadores possuem capacidade de torque em baixa velocidade de rotação, mínimo acoplamento físico entre o rotor e o estator e inexistência de sistema de redução, a fim de viabilizar a realização de experimentos de subatuação, graças à possibilidade de configuração das juntas como ativas ou passivas. Contudo, a base não possui qualquer sistema de atuação, ficando livre para se mover sobre a mesa do tipo *Ealing*. Esta mesa cria uma fina película de ar pressurizado sobre a estrutura do braço robótico de forma a minimizar os efeitos gravitacionais, permitindo que o robô se desloque com mínimo atrito e perturbação. As juntas do robô possuem o formato de *puck* com base larga e distribuição de massa uniforme para possibilitar sua flutuação adequada sobre a mesa. A estrutura do robô está projetada para possibilitar diversas configurações do manipulador com um braço ou dois. O sensoramento, efetuado pelo robô por meio de encoders ópticos incrementais localizados em cada junta, realimenta o módulo de controle ao transmitir leituras de posicionamento das juntas e da base.

A Figura 3.1 mostra o manipulador espacial de base livre flutuante desenvolvido em (Pazelli; 2011). Na figura, o robô é mostrado na configuração com um braço e duas juntas.

Figura 3.1: Manipulador UARME.

As especificações de projeto enfatizam a necessidade da ausência total de uma conexão física com o meio externo. Dessa forma, cabos ou fios não devem existir entre o robô configurado e o ambiente, evitando assim uma interferência mecânica indesejada. Nesse sentido, toda a eletrônica de comando e a alimentação dos componentes do robô são alocadas em sua base flutuante, onde a comunicação da plataforma com o computador remoto é baseada no padrão de comunicação sem fio ZigBee. O circuito embarcado na base do robô conta com um microcontrolador Rabbit[®]4000, o qual é responsável por receber os sinais enviados pelo computador através do *modem* ZigBee, tratar os pacotes de instruções e transmiti-los a amplificadores que então acionarão os motores. Da mesma forma, recebe as leituras dos encoders através de decodificadores de quadratura e transmite esse pacote ao terminal remoto. Para o sistema de alimentação, são utilizadas baterias de polímero de lítio (LiPo) devido a sua facilidade de aquisição e possibilidade de recarga controlada.

3.2 Modelagem para um manipulador de base fixa

Neste trabalho, pretende-se utilizar um robô espacial de base livre flutuante configurado com um braço contendo dois ou três elos. A seguir apresenta-se uma introdução aos princípios da modelagem, que contempla os modelos cinemático e dinâmico de ma-

nipuladores robóticos com base fixa. Posteriormente, é apresentada uma breve discussão sobre a modelagem para robôs manipuladores de base livre flutuante e suas dificuldades. Finalmente, é apresentado o conceito de Modelo Dinamicamente Equivalente, que propõe determinar um manipulador fixo a uma junta esférica e passiva, dinamicamente equivalente a um manipulador de base livre flutuante.

3.2.1 Cinemática direta

A cinemática direta de um robô manipulador está relacionada com suas juntas individuais e a posição/orientação da ferramenta ou efetuador final. Contudo, a análise cinemática de um manipulador composto de n -elos pode tornar-se bastante complexa. Com o objetivo de sistematizar um procedimento para tal análise, a convenção de Denavit-Hartenberg (DH) reúne aspectos específicos da geometria do manipulador e as representa em transformações homogêneas A_i , dadas pelo produto de quatro transformações básicas.

$$\begin{aligned}
 A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\
 &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned} \tag{3.1}$$

As variáveis θ_i , d_i , a_i e α_i na equação 3.1 representam, respectivamente, o ângulo de junta, o *offset* de elo, o comprimento do elo e a torção do elo. Todos esses são parâmetros associados ao elo i e à junta i . Para a matriz de transformação A_i , a única variável é θ_i para juntas de revolução e d_i para juntas prismáticas. Os outros parâmetros são constantes e dependentes exclusivamente da geometria do robô. A figura 3.2 ilustra um robô manipulador planar de três elos.

Figura 3.2: Manipulador de três elos.

Na figura 3.2, os eixos z apontam todos para fora da página. Para este caso, onde todas as juntas são de revolução, a única variável é θ_i , conforme mostra a tabela 3.1.

Tabela 3.1: Parâmetros DH para um manipulador planar de três elos

Elo	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3

Dispondo dos parâmetros DH, as matrizes de transformação A_1 , A_2 e A_3 podem ser calculadas de acordo com a equação 3.1. Para mapear o sistema de coordenadas do efetuador em relação à base, a matriz T é computada conforme descrito na equação 3.2.

$$T_3^0 = A_1 A_2 A_3 = \begin{bmatrix} c_{123} & -s_{123} & 0 & a_1 c_1 + a_2 c_{12} + a_3 c_{123} \\ s_{123} & c_{123} & 0 & a_1 s_1 + a_2 s_{12} + a_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

Através da inspeção da equação 3.2, conclui-se que a matriz de rotação retorna a orientação do plano de coordenadas do efetuador com respeito ao plano de coordenadas da base. Da mesma maneira, as posições (x, y) do plano de coordenadas da base são descritas pela equação 3.3:

$$\begin{aligned}x &= a_1 c_1 + a_2 c_{12} + a_3 c_{123} \\y &= a_1 s_1 + a_2 s_{12} + a_3 s_{123}.\end{aligned}\tag{3.3}$$

De maneira análoga, a expansão do problema para o caso de n juntas é conseguida sem maiores contratempos através da utilização da convenção de DH.

3.2.2 Cinemática inversa

Ao contrário do problema de cinemática direta, que possui uma solução única, derivada através de multiplicações de matrizes, o cálculo da cinemática inversa é consideravelmente mais complexo. Essa tarefa consiste em encontrar as variáveis de junta a partir de uma posição e orientação do efetuador final. As principais dificuldades na obtenção de uma solução para esse problema são (Siciliano et al.; 2009):

- Pode haver múltiplas soluções;
- Pode haver infinitas soluções;
- Pode não haver soluções admissíveis;
- Normalmente, as equações em questão são não-lineares, logo, nem sempre é possível encontrar uma solução na forma fechada;
- O grau de complexidade cresce conforme aumenta o número de graus de liberdade.

Não há um método genérico para a resolução do problema. As abordagens algébricas ou geométricas são utilizadas como recurso para computar uma solução na forma fechada. Entretanto, quando uma solução desse tipo não existe ou é difícil de computar, uma alternativa é recorrer a técnicas de solução numérica. Essas técnicas destacam-se pela capacidade de se adaptar a qualquer estrutura cinemática. Porém, apresentam o revés de não serem capazes de considerar todas as soluções possíveis. Quando levado para o problema de planejamento de trajetória, essa desvantagem representa uma perda no

controle da qualidade da solução. Como uma configuração deve ser escolhida sem o conhecimento do espaço de trabalho, pode ocorrer que a solução não esteja livre de colisão. A figura 3.3 ilustra o problema descrito. Na figura 3.3(a) são mostradas as soluções de cinemática inversa para o robô manipulador de três elos com posição e atitude de efetuador final fixas. No caso da presença de um obstáculo semelhante ao da figura 3.3(b), a segunda solução aparece como única alternativa livre de colisão.

Figura 3.3: Ilustração de um problema de cinemática inversa.

Outro problema a ser levado em conta é a consideração de uma solução que esteja desconectada do ponto de partida, isto é, o algoritmo de planejamento de rota não pode encontrar um caminho desde o ponto de partida até o objetivo especificado, embora este último pertença ao espaço de configuração livre do robô.

Por outro lado, o problema de verificar se determinada solução é factível exige um extenso pré-processamento computacional. Uma vez que a representação explícita da conectividade do espaço de configuração é um problema tão complexo quanto o planejamento de rota em si, serão considerados doravante somente caminhos factíveis no desenvolvimento desse trabalho.

3.2.3 Modelo dinâmico

Figura 3.4: Representação do i -ésimo elo do ME (Pazelli; 2011)

Seja q um vetor coluna contendo as n variáveis de entrada do manipulador de base fixa (MBF), de tal forma que os elementos de $q = [q_1 \ q_2 \ \dots \ q_n]^T$ sejam coordenadas generalizadas. A formulação de Euler-Lagrange para um manipulador robótico com base fixa, ou MBF, é determinado pela seguinte equação dinâmica (Spong et al.; 2005):

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau. \quad (3.4)$$

Na equação 3.4, a matriz $M(q) \in \mathbb{R}^{n \times n}$ denota a matriz de inércia. O vetor definido por $C(q, \dot{q})\dot{q} \in \mathbb{R}^n$ é composto pelas forças centrífugas e de Coriolis. O vetor gravidade $g(q) \in \mathbb{R}^n$ é dado por $g(q) = [g_1(q), \dots, g_n(q)]^T$. As forças generalizadas, τ_i , representam os torques e forças aplicadas nas juntas. Considere a representação do i -ésimo elo do MBF na Figura 3.4, C_i denota o centro de massa do i -ésimo elo do MBF; P_i representa o eixo de rotação da i -ésima junta J_i . L_i é o vetor que conecta J_i e C_i , enquanto R_i é o vetor que conecta C_i e J_{i+1} .

3.3 Modelagem para um manipulador de base livre

No caso de robôs manipuladores montados em satélites, que sejam livres para translação e rotação, é mostrado que a posição do efetuador final não é mais função somente

dos ângulos de junta, como mostra a equação 3.3, mas sim uma função do histórico dos ângulos de junta aplicados até então (Kanade e Xu; 1993). Isso leva os problemas de cinemática direta e inversa a assumirem um viés completamente distinto. Um dos primeiros trabalhos a considerar esse tipo de análise é descrito detalhadamente em (Longman; 1993), no qual os termos cinética direta e inversa são cunhados exclusivamente para esse problema. Nessa abordagem é especificado o movimento relativo dos elos sucessivos do robô como uma função do tempo, essas são utilizadas para determinar o movimento dinâmico do satélite, e por conseguinte, do efetuador final. No geral, o problema descrito em (Longman; 1993) requer encontrar um histórico dos ângulos de junta que satisfaça o problema de equação diferencial com valores limites.

Apesar dos esforços e contribuições trazidos pelas abordagens de cinética direta e inversa, tratam-se de formulações muito complexas e de difícil análise e aplicabilidade. A abordagem revela-se complicada pois, ao envolver o problema de cinética inversa, especifica-se não somente onde se pretende posicionar o efetuador final no espaço inercial, como também a atitude desejada do satélite.

Tendo como principal motivação a simplificação da modelagem e análise matemática de manipuladores espaciais, Vafa e Dubowsky (1987) propuseram o conceito de Manipulador Virtual (MV). O MV consiste em um manipulador de base fixa com a primeira junta esférica e passiva, responsável por emular o comportamento flutuante da base do robô. Este MV, por sua vez, será cinematicamente equivalente ao manipulador espacial. A utilização desta técnica reduz em três o número de equações dinâmicas necessárias, uma vez que sua definição incorpora as restrições impostas pelo princípio da conservação de momento linear. Entretanto, o MV é por definição uma cadeia cinemática sem massa e, portanto, abrange somente o âmbito de simulação, excluindo assim a possibilidade de construção física para testes reais. Com base nisso, o Manipulador Dinamicamente Equivalente (MDE) (Liang et al.; 1996) foi desenvolvido a fim de estender os conceitos do MV, representando o manipulador espacial tanto cinemático quanto dinamicamente através de um manipulador de base fixa equivalente.

3.4 Manipulador Dinamicamente Equivalente

Como alternativa para as complexas abordagens cinemático-dinâmicas em manipuladores espaciais, foi apresentado em (Liang et al.; 1996) o Manipulador Dinamicamente Equivalente. O método consiste em mapear um manipulador espacial de base livre flutuante em um manipulador de base fixa, preservando suas propriedades dinâmicas e cinemáticas. O sistema proposto permite não somente modelar um manipulador espacial de base livre com métodos tradicionais, como também possibilita o estudo experimental de plataformas espaciais em ambientes convencionais sem a necessidade de recorrer a estruturas complexas para simular o ambiente espacial.

O MDE pode ser construído fisicamente e adotado em estudos experimentais do comportamento dinâmico de manipuladores espaciais. Além disso, pode ser utilizado como ferramenta para o desenvolvimento do modelo dinâmico do robô. O método é aplicável não somente para manipuladores de base livre flutuante, como também para aqueles de base livre controlada.

3.4.1 Modelo dinâmico de um manipulador espacial de base livre flutuante

Considera-se um manipulador rígido de n -elos montado em uma base livre flutuante. Denota-se a base do Manipulador Espacial (ME) como elo 1 e os elos seguintes como elo 2 até $n + 1$. Assumindo que forças e torques externos não agem no ME, seu centro de massa C_0 permanece fixo no espaço inercial e é escolhido como origem do sistema de coordenadas (sistema 0 na figura 3.5). Na figura 3.5, S_i denota o vetor entre C_0 e C_i . Os sistemas de coordenadas 1 até $n + 1$ são fixados aos centros de massa de cada elo.

Figura 3.5: Sistemas de coordenadas fixas nos elos do ME (Liang et al.; 1996).

A derivação do método MDE utiliza a equação de Lagrange no ME e obtém ainda o modelo dinâmico de um manipulador fixo de base representada por uma junta esférica e passiva. Considerando o MDE trabalhando na ausência de gravidade e com sua base localizada no centro de massa do manipulador espacial, as condições sob as quais estes modelos são equivalentes satisfazem:

$$\begin{aligned}
 m'_i &= \frac{M_t^2 m_i}{\sum_{k=1}^{i-1} m_k \sum_{k=1}^i m_k}, \quad i = 2, \dots, n+1 \\
 I'_i &= I_i, \quad i = 1, \dots, n+1 \\
 W_1 &= R_1, \\
 W_i &= R_i + L_i, \quad i = 2, \dots, n+1 \\
 l_{c1} &= 0, \\
 l_{ci} &= \frac{\sum_{k=1}^{i-1} m_k}{M_t} L_i, \quad i = 2, \dots, n+1.
 \end{aligned} \tag{3.5}$$

Os vetores W_i conectam as juntas J'_i e J'_{i+1} no MDE; m'_i corresponde à massa do i -ésimo elo do MDE; $I'_i \in \mathbb{R}^{3 \times 3}$ denota o tensor inercial correspondente ao i -ésimo elo do MDE; l_{ci} representa o vetor da i -ésima junta do MDE até o centro de massa do i -ésimo elo. Liang et al. (1996) demonstram ainda que o valor de m'_1 não influencia na equivalência dinâmica entre o MDE e o ME e portanto a massa do primeiro elo do MDE pode ser atribuída arbitrariamente com um valor positivo não nulo. A figura 3.6 ilustra

os parâmetros do manipulador dinamicamente equivalente. De maneira análoga ao robô de base fixa, o vetor P'_i representa o eixo de rotação da i -ésima junta J'_i .

Figura 3.6: Representação do i -ésimo elo do MDE (Pazelli; 2011)

Seja o vetor de coordenadas generalizadas $q = [\phi \ \theta \ \psi \ \theta_2 \ \dots \ \theta_{n+1}]^T \in \mathbb{R}^{(n+3)}$ decomposto segundo $q = [q_b^T \ q_m^T]^T$, onde b e m representam os componentes da base e do manipulador, respectivamente. De maneira análoga à equação 3.4, o sistema é aplicado às equações de Euler-Lagrange e assumem a forma especial:

$$M(q_m)\ddot{q} + C(q_m, \dot{q})\dot{q} = \tau. \quad (3.6)$$

Uma vez que a aceleração da gravidade é assumida nula no ambiente espacial, o vetor gravidade é também inexistente. Na equação 3.6, $M(q_m) \in \mathbb{R}^{(n+3) \times (n+3)}$ denota a matriz de inércia, simétrica e definida positiva, dependente apenas das coordenadas do manipulador, $C(q_m, \dot{q}) \in \mathbb{R}^{(n+3) \times (n+3)}$ a matriz de Coriolis e forças centrífugas e finalmente $\tau = [0 \ \tau_2 \ \dots \ \tau_{n+1}]^T \in \mathbb{R}^{(n+3)}$ representa o vetor de torques aplicados sobre as juntas do MDE.

Partindo das equivalências apresentadas na equação 3.5, o MDE é determinado para o robô UARM-E configurado com um braço e duas juntas ativas. A tabela 3.2 revela os parâmetros físicos do robô espacial enquanto a tabela 3.3 relaciona os parâmetros calculados para o MDE desse robô. Nas tabelas 3.2 e 3.3, Robô 1 refere-se ao sistema de flutuação baseado na mesa de ar, enquanto Robô 2 especifica os parâmetros para o sistema alternativo de flutuação, no qual tubos de ar-comprimido auxiliam no processo.

Os dados foram extraídos de (Pazelli; 2011).

Tabela 3.2: Parâmetros do ME(UARM-E)

Robô 1				
	$m_i(kg)$	$I_i(kgm^2)$	$L_i(m)$	$R_i(m)$
Base	4,780	0,0404	0	0,150
Elo 2	1,380	0,0182	0,144	0,111
Elo 3	1,011	0,0115	0,103	0,112
Robô 2				
	$m_i(kg)$	$I_i(kgm^2)$	$L_i(m)$	$R_i(m)$
Base	5,616	0,0503	0	0,155
Elo 2	1,516	0,0194	0,152	0,103
Elo 3	1,144	0,0130	0,116	0,099

Tabela 3.3: Parâmetros do MDE(UARM-E)

Robô 1				
	$m_i(kg)$	$I_i(kgm^2)$	$W_i(m)$	$l_{ci}(m)$
Base	4,780	0,0404	0,100	0
Elo 2	2,410	0,0182	0,191	0,096
Elo 3	1,177	0,0115	0,201	0,089
Robô 2				
	$m_i(kg)$	$I_i(kgm^2)$	$W_i(m)$	$l_{ci}(m)$
Base	5,616	0,0503	0,105	0
Elo 2	2,592	0,0194	0,192	0,103
Elo 3	1,328	0,0130	0,199	0,100

Capítulo 4

Árvore Aleatória de Exploração Rápida (RRT)

A fim de contribuir com os avanços no campo dos algoritmos baseados em amostragem, o algoritmo RRT básico, proposto por (LaValle; 1998), apresenta como principal distinção o seu suporte natural a sistemas não-holonômicos e com vários graus de liberdade. Resumidamente, uma árvore RRT cresce incrementalmente a partir de um estado inicial rumo a um estado final. Para isso, conta com o emprego de entradas de controle, de modo a conduzir a árvore a estados amostrados aleatoriamente no espaço de configuração livre do robô. O algoritmo de construção de uma RRT funciona de acordo com a metodologia descrita a seguir.

4.1 O Algoritmo RRT básico

Seja T uma árvore com raiz em um estado inicial, x_{init} . A cada iteração, um estado aleatório x_{rand} é amostrado uniformemente no espaço livre de configuração, X_{free} . O algoritmo emprega então um método de busca a fim de encontrar o estado em T mais próximo de x_{rand} de acordo com uma métrica ρ . Esse estado é então denominado x_{near} . Em seguida, amostra-se para x_{near} um vetor de entradas aleatórias u , dentre o conjunto de entradas U possível para esse estado. Cada uma das entradas em u é utilizada então para avaliar a expansão de x_{near} dentro de um intervalo de integração fixo. Dentre aqueles estados gerados após a expansão que estão livres de colisão, escolhe-se o que possui o menor custo à x_{rand} , observado mais uma vez a métrica ρ em vigor. A esse estado selecionado

denomina-se x_{new} , que é então adicionado aos vértices da árvore T , bem como o ramo (x_{near}, x_{new}) é adicionado aos ramos de T . O processo é repetido até que se encontre um estado x_{new} que se aproxime suficientemente do objetivo x_{goal} ou então que seja atingido um certo número pré-definido de ramos em T . Após o sucesso na expansão da árvore, a melhor solução pode ser escolhida a partir de uma função custo que premie o trajeto que possui o menor tempo de execução, por exemplo. A figura 4.1 ilustra o processo descrito em um espaço de trabalho euclidiano bidimensional.

Figura 4.1: Em 4.1(a), os estados inicial e final são definidos como x_{init} e x_{goal} , respectivamente. Uma área de tolerância é definida ao redor de x_{goal} . Em 4.1(b), uma configuração aleatória x_{rand} é amostrada no espaço de configuração livre e então é selecionado o nó mais próximo em T , x_{near} . Em 4.1(c), x_{near} gera primitivas de movimento para então avaliar a expansão que possui menor custo a x_{rand} . Finalmente, em 4.1(d), x_2 é adicionado como novo estado de T , tendo o ramo e_1 conectando x_1 a x_2 . O processo é repetido iterativamente até que uma solução seja encontrada ou o número máximo de tentativas seja ultrapassado.

A estrutura de um código RRT básico pode ser vista no algoritmo 1. No caso deste pseudo-algoritmo em especial, um caminho é retornado assim que o estado final esteja incluído no intervalo de tolerância definido pelo algoritmo.

Algoritmo 1 RRT básico

RRTmain()

```

1:  $Tree \leftarrow x_{start}$ ;
2:  $x_{new} \leftarrow x_{start}$ ;
3: while  $Distancia(x_{new}, x_{goal}) > ErroTolerancia$  do
4:    $x_{target} = \text{SampleTarget}()$ ;
5:    $x_{near} = \text{NearestVertex}(Tree, x_{target})$ ;
6:    $x_{new} = \text{ExtendTowards}(x_{nearest}, x_{target})$ ;
7: end while
8: return  $\text{Trajectory}(Tree, x_{new})$ ;

```

4.2 Propriedades e Vantagens do Algoritmo RRT

Com base nos estudos e resultados apresentados em (LaValle; 1998) e (Kuffner Jr. e LaValle; 2000), as principais características e vantagens do algoritmo RRT, além das já mencionadas, são:

1. Forte tendência de crescimento para regiões ainda não exploradas;
2. Sob condições gerais, o algoritmo é probabilisticamente completo;
3. Simplicidade estrutural, permitindo análise de desempenho mais inteligíveis;
4. Não depende de uma representação explícita da região de configuração livre do robô.
5. Os estados sempre permanecem conectados, não importando o tamanho da árvore;
6. Depende de poucas heurísticas ou parâmetros arbitrários;
7. Pode ser adaptado para resolver o problema do planejamento de rota e também incorporado a inúmeros sistemas de planejamento;
8. Ao fim de uma busca bem-sucedida, retorna o conjunto de entradas necessárias para a realização da trajetória.

Via de regra, a métrica ρ empregada baseia-se em uma função custo que deverá traduzir o custo de levar o robô de um estado a outro. A métrica ideal é representada pelo custo ótimo de deslocamento entre dois estados. Entretanto, a prática revela que calcular esse custo ótimo é uma tarefa no mínimo tão complexa quanto o próprio problema de projetar a trajetória em si (LaValle e Kuffner Jr.; 1999; Cheng e LaValle; 2001). Em uma situação hipotética onde dois estados estão bem próximos em termos de uma métrica baseada somente na distância euclidiana, o emprego de uma métrica adequada deveria levar em conta o menor trajeto entre esses dois estados. Dessa maneira, o algoritmo deve selecionar o estado que está de fato mais próximo e gerar portanto uma expansão adequada da árvore RRT.

Uma árvore T é construída de tal forma que todos os seus vértices e ramos estão em X_{free} , isto é, o espaço de configuração livre do robô. Um planejador RRT parte do pressuposto que uma região de obstáculo fixa, X_{obs} , deve ser evitada e que não se dispõe de uma representação matemática explícita dessa região. Consequentemente, é possível apenas verificar se um ponto está situado em X_{obs} ou não, anulando assim o tempo necessário para pré-processamento do ambiente. Este fato faz do algoritmo RRT um planejador ideal para o problema de consulta-única em planejamento de trajetória, ou seja, o mesmo ambiente é usado para responder a um único problema.

Uma equação de transição de estado da forma $\dot{x} = f(x, u)$ é definida com o objetivo de representar o modelo dinâmico e as restrições não-holonômicas. O vetor u é selecionado a partir de um conjunto de entradas válidas, U . Ao integrar f por um intervalo de tempo fixo, Δt , o próximo estado x_{new} pode ser determinado para um dado estado inicial x combinado a uma entrada u . O sistema é resolvido preferivelmente com técnicas que possuem alto grau de integração, como o método Runge-Kutta ou o método de Euler (LaValle; 1998).

Ao término da busca pela trajetória, é conhecido o conjunto de entradas que levará o robô do estado inicial ao desejado. Dessa forma, desconsiderando os erros associados à modelagem, a simples passagem de parâmetros do planejador para as referências do controlador finaliza o problema do planejamento de trajetória.

Observações realizadas através de experimentos e simulações, conduzidos em (Lavalle et al.; 2000), revelam que os caminhos gerados por uma RRT, apesar de não serem suavizados, não se distanciam muito do caminho ótimo. Resultados apontaram uma proporção entre o caminho gerado e o ótimo oscilando consistentemente entre 1,3 e 1,7.

Paralelamente à abordagem de árvores de exploração aleatória, os métodos de mapas de rota probabilísticos e campos potenciais artificiais propõem alternativas para o problema de planejamento de rota e trajetória. Uma breve descrição dos métodos e comparações com a abordagem de RRTs são realizadas a seguir.

No método de mapas de rota probabilísticos, o algoritmo planejador trabalha de modo a tentar unir configurações aleatórias, amostradas no espaço de configuração, sendo este representado por um grafo, expandido iterativamente à medida que as configurações são conectadas ou desconectadas. Quando utilizada para resolver problemas gerais, esta abordagem apresenta o revés da complexidade demasiada envolvendo a conexão entre duas configurações amostradas (LaValle; 1998). Contrariamente ao algoritmo RRT, este método realiza sucessivos pré-processamentos do ambiente antes do planejamento, fato justificado por este último ser criado para múltiplas-consultas, ou seja, o mesmo ambiente é usado para resolver vários problemas de planejamento.

Embora trabalhos venham sendo feitos no intuito de contornar o tradicional problema dos mínimos locais para a abordagem de campos potenciais artificiais, o próprio conceito deste método colabora para o surgimento de tais dificuldades e portanto, a escolha de uma boa função potencial heurística é fundamental para o sucesso do planejamento. Além disso, quando levada para um ambiente com obstáculos e restrições cinemático-dinâmicas, a tarefa da escolha desta função torna-se um desafio incomensurável na área de planejamento de rota.

4.3 Aperfeiçoamento do algoritmo RRT

Desde a proposta inicial apresentada em (LaValle; 1998), inúmeros trabalhos foram publicados com o objetivo de melhorar o desempenho do algoritmo RRT. Para os problemas mais críticos, citamos a colaboração de cada nova abordagem do problema de

construção de árvores RRT.

4.3.1 O problema de convergência

Dependendo do ambiente e dos métodos empregados para a confecção do algoritmo, a expansão da árvore RRT pode levar um tempo maior que o esperado até convergir para uma solução dentro de uma faixa de tolerância aceitável. Dentre as várias questões envolvendo a amostragem de x_{rand} , foi observado que uma sutil preferência de amostragem a x_{goal} resulta numa melhoria significativa na taxa de sucesso das buscas de trajetória. Em outras palavras, esse viés atua de modo a fazer o algoritmo ter a probabilidade p de escolher o próprio estado final como “aleatório” e, por conseguinte, probabilidade $1 - p$ de sortear ao acaso um estado qualquer no espaço livre de configuração. Por outro lado, um viés muito elevado da probabilidade p , referente ao sorteio de x_{goal} , leva o algoritmo a cair facilmente nos mesmos pontos fracos que a abordagem de campos potenciais, isto é, o problema de mínimos locais. Experimentalmente observou-se que um viés por volta de $p = 5\%$ produz os efeitos esperados na melhoria de convergência (Urmson e Simmons; 2003).

O emprego de diferentes modificações no algoritmo RRT básico é investigado em (Urmson e Simmons; 2003). Dentre estas análises está uma comparação entre um viés de $p = 5\%$ e outro de $p = 10\%$ com relação à configuração final. Os resultados indicam que, para o programa livre de viés, é necessária uma quantidade muito maior de nós na árvore até que uma solução seja encontrada. Os resultados não sugerem uma regra absoluta para o valor ideal de p , pois ora o custo médio para um planejamento favorece 5% de viés e, para outros ambientes e condições, um viés de 10% apresenta uma solução menos custosa. A tabela 4.1 mostra um comparativo entre custos necessários para um mesmo problema utilizando abordagens distintas. Em ambientes mais complexos, a utilização de viés se mostrou ainda mais vantajosa.

Tabela 4.1: Comparação entre algoritmos num espaço livre binário. (adaptada de (Urmson e Simmons; 2003))

Algoritmo	Custo médio	Média # de nós em T	Tempo médio(s)
RRT	17,67	396	0,377
Viés de 5 %	17,63	167	0,148
Viés de 10 %	17,29	140	0,159

4.3.2 Sobrecarregamento computacional

O algoritmo RRT necessita de uma função que determine o vértice mais próximo da configuração amostrada. Portanto, à medida que o algoritmo RRT cresce e o número de vértices torna-se significativo, observa-se uma perda de rendimento considerável no desempenho do programa. O trabalho exposto em (Svenstrup et al.; 2011) propõe reduzir o tempo computacional, de forma que mais vértices sejam adicionados no mesmo intervalo de tempo a fim de gerar trajetórias com melhor qualidade.

Basicamente, a ideia empregada nessa abordagem fundamenta-se no particionamento do espaço de configuração d -dimensional em um número de caixas d -dimensionais. Então, partindo do princípio que nós muito afastados não devem e nem podem ser conectados na maioria dos casos, é calculado o custo de trajeto somente para nós localizados em caixas ao redor do estado amostrado, ao invés de todos os nós em todo o espaço de configuração. Entretanto, para o método representar uma melhoria no custo computacional, o mesmo deve ser introduzido somente após um valor limiar Γ , referente a uma iteração do algoritmo RRT básico. Isso ocorre pois a árvore com poucas iterações cobre uma pequena região no espaço de trabalho do robô. Consequentemente, o algoritmo de busca através de caixas verifica muitas áreas não-promissoras, atrasando assim o crescimento da árvore RRT. (Svenstrup et al.; 2011) não chegam a uma conclusão sobre um valor ideal para o limiar Γ . Conjecturas apresentadas no próprio trabalho sugerem que um valor de Γ igual a 50% do total de iterações resulta em um valor razoável para os fins propostos.

Os resultados aferidos em (Svenstrup et al.; 2011) revelam um comportamento quase linear no crescimento do custo computacional quando empregado o método das caixas, em comparação com um crescimento exponencial no caso do método RRT básico. Os testes foram conduzidos para sistemas com duas e quatro dimensões. A curva avaliada

relaciona o crescimento do tempo de execução com o crescimento da árvore RRT.

4.3.3 Redução da sensibilidade métrica

RRTs utilizam heurísticas baseadas em métricas com o propósito de levar a uma exploração ambiciosa do espaço de estado. Segundo Cheng e LaValle (2001), o desempenho degrada substancialmente quando a métrica escolhida não reflete o real custo para mover-se de uma configuração a outra. De fato, a métrica ideal consiste no custo ótimo de deslocamento entre dois estados do robô. Entretanto, calcular esse custo ótimo é no mínimo tão complexo quanto resolver o problema do planejamento de trajetória em si.

Para remediar esta situação, foi proposta em (Cheng e LaValle; 2001) uma modificação no algoritmo RRT básico a fim de refinar sua estratégia de exploração na ausência de uma métrica adequada. A contribuição dessa abordagem não consiste em desenvolver uma métrica específica, mas em coletar informações ao longo da exploração e expansão de T de acordo com a métrica adotada e as informações selecionadas.

Os procedimentos utilizados para a coleta de informações úteis ao planejador consistem em:

1. Manter registro das entradas aplicadas em cada nó da RRT. Caso a entrada u já tenha sido aplicada e avaliada para um determinado estado x , u não será posteriormente considerada parte do conjunto de entradas disponíveis àquela configuração específica, x , em uma próxima ocasião. O motivo desse descarte se deve a u conduzir x a um estado já existente em T ou que resulte em colisão. Caso não haja mais entradas passíveis de aplicação em x o estado é completamente descartado de futuras extensões.
2. A Frequência de Violação de Restrição (FVR) é utilizada para estimar a probabilidade de colisão do caminho gerado através da aplicação de uma sequência de entradas aleatórias. Inicialmente, cada estado possui uma FVR nula. Após esse estado ser selecionado como x_{near} , sua FVR é incrementada de um valor h/N , onde N é o número de entradas em U e h a quantidade de entradas que levam a uma colisão ou restrição de movimento. Adicionalmente, o nó pai daquele estado terá

sua FVR incrementada de h/N^2 . De modo análogo, o m -ésimo pai terá sua FVR acrescida de h/N^{m+1} , com exceção do nó inicial x_{init} ;

A obtenção das FVRs, bem como das informações de exploração, auxiliam o planejador RRT a escolher um melhor x_{near} . Nesse processo, o planejador verifica primeiramente se todas as entradas de um determinado estado x_i estão marcadas como *expandida*. Em caso afirmativo, x_i será dispensado das buscas, caso contrário será considerada uma probabilidade FVR(x_i) de não escolher este estado como x_{near} . Como resultado do emprego dos procedimentos descritos, percebe-se uma significativa melhora na qualidade das soluções e na exploração do ambiente (Cheng e LaValle; 2001).

4.3.4 RRTs bidirecionais

No intuito de contribuir na solução do problema de convergência da árvore RRT, LaValle e Kuffner Jr. (1999) introduzem a abordagem da busca bidirecional, no qual duas RRTs são expandidas incrementalmente de forma independente. Esta abordagem possui seus principais aspectos e funcionamento básico descritos a seguir.

A primeira árvore possui, assim como no método RRT básico, sua raiz no estado inicial x_{init} . Uma segunda árvore é então crescida a partir do estado final x_{goal} . Assim sendo, um estado aleatório x_{rand} é amostrado e uma tentativa de expansão é feita a partir de x_{init} . Posteriormente é feita uma tentativa de crescimento a partir de x_{goal} tomando o mesmo estado x_{rand} como parâmetro. O processo se repete até que dois estados estejam suficientemente próximos. Considerando dois estados x_1 e x_2 , pertencentes a árvores distintas. A proximidade entre esses estados é verificada se $\rho(x_1, x_2) < \varepsilon$, para um valor pequeno de $\varepsilon > 0$. Ou ainda, utiliza-se uma função para conexão desses estados, como sugerido em (Kuffner Jr. e LaValle; 2000). Uma observação deve ser feita no momento da integração da equação de transição de estado: Uma vez que uma das árvores cresce a partir de x_{goal} , a integração deve ser calculada de trás pra frente em relação ao tempo quando considerada esta árvore.

Figura 4.2: Crescimento de duas RRTs. As árvores representadas em azul e vermelho encontram um nó suficientemente próximo, representado em verde, para então encontrar uma solução para o problema de planejamento.

Quando comparado a tentativas de crescer uma única árvore RRT, o método proposto para a expansão de árvores bidirecionais mostrou uma melhoria na eficiência da busca, sobretudo para espaço de estado com altas dimensões. Entretanto, a verificação da conexão de pares entre as árvores gera um custo computacional adicional que representa talvez o maior revés do emprego desta abordagem.

4.3.5 Considerações finais

De forma resumida, os aperfeiçoamentos aqui considerados tiveram o propósito de não somente melhorar o desempenho do algoritmo proposto neste trabalho, como também solucionar o problema do planejamento de rota em si.

Foi verificado posteriormente que o emprego do algoritmo RRT básico não era suficiente para obter uma rota, no caso de um manipulador espacial de base livre flutuante (MBLF). Portanto, os algoritmos de aperfeiçoamento foram modificados e/ou adaptados ao contexto de MBLFs. Em seguida, foram implementados e tiveram seu desempenho avaliado. Esses procedimentos são descritos a seguir nos capítulos 5 e 6.

Capítulo 5

Desenvolvimento de um planejador de trajetórias RRT

O presente capítulo está concentrado em descrever os elementos integrantes do planejador de trajetória para um manipulador espacial planar de base livre flutuante e apresentar suas contribuições na resolução do problema geral. O capítulo está organizado da seguinte forma. A seção 5.1 apresenta os ambientes de simulação e ferramentas utilizados durante as etapas principais do desenvolvimento. A seção 5.2 avalia a primeira grande alteração no algoritmo RRT básico, relativa à introdução do método das caixas. A seção 5.3 dá continuidade a essa sequência através da análise do algoritmo que propõe reduzir a sensibilidade da métrica. A seção 5.4 faz uma investigação a respeito da última modificação significativa incorporada ao algoritmo, as árvores bidirecionais. A seção 5.5 descreve o funcionamento básico do planejador proposto nesse trabalho. Ademais, foram feitos ajustes e modificações nas propostas utilizadas, de modo a obter êxito no planejamento. Essas alterações são apresentadas ao longo do capítulo.

5.1 Descrição do ambiente de simulação

Com o objetivo de avaliar os distintos elementos que compõem o algoritmo RRT básico e seus métodos, foram projetados cenários e objetivos para as principais fases de teste. Essa abordagem teve por finalidade garantir uma evolução gradual na complexidade do

problema e consequentemente facilitar uma análise de desempenho dos elementos constituintes do planejador. Esses ambientes e métodos são descritos nas subseções 5.1.1 a 5.1.3.

5.1.1 O ambiente de testes inicial

A abordagem inicial teve por objetivo principal a implementação e avaliação dos seguintes elementos fundamentais do algoritmo RRT básico:

1. Construção do mapa;
2. Amostragem da configuração aleatória;
3. Verificação do nó mais próximo;
4. Geração de primitivas de movimento;
5. Geração da trajetória após a convergência;
6. Verificação de colisão simplificada.

Para atingir o propósito de avaliar cada um dos aspectos mencionados, diversas simplificações foram feitas ao ambiente e modelos utilizados no robô. Para a construção do espaço de trabalho, foi considerado um mapa bidimensional de tamanho $30 \times 30(m)$. Esse mapa foi constituído de modo a contar com dois obstáculos fixos. O primeiro possui forma circular, com raio e posição definidos. O segundo é formado a partir de dois círculos concatenados, também com raios e posições conhecidos, no intuito de formar uma zona de mínimo local. Ademais, foi criada uma região de interesse X_{goal} , na qual o robô encontra a solução do problema de planejamento.

A aplicação da soma e diferença de Minkowski em um problema de planejamento, geralmente permite que um robô possa ser representado de maneira mais simples no espaço cartesiano. Basicamente, o processo é feito através da expansão geométrica dos obstáculos de uma quantidade igual à subtraída do robô. A figura 5.1 mostra de maneira ilustrativa o princípio básico explorado por essa técnica.

Figura 5.1: A representação ilustrativa da aplicação da soma e diferença de Minkowski em um cenário composto de um obstáculo triangular A na presença de um robô de raio R é mostrado na figura 5.1(a). O resultado da aplicação do método pode ser verificado na figura 5.1(b).

Essa simplificação é utilizada nessa etapa no intuito de reduzir o algoritmo de detecção de colisão a uma simples verificação baseada em distâncias euclidianas. Para um robô localizado no ponto $P(x_p, y_p)$, não haverá colisão com um obstáculo circular de raio R , centrado em $D(x_d, y_d)$, se e somente se:

$$\sqrt{(x_p - x_d)^2 + (y_p - y_d)^2} > R \quad (5.1)$$

A simplificação do ambiente de testes permitiu a verificação dos elementos fundamentais propostos. Os resultados são verificados posteriormente no capítulo 6. Entretanto, a geometria complexa dos manipuladores robóticos desfavorece a aplicação de soma e diferença de Minkowski no planejamento de rota para manipuladores espaciais, no geral. Portanto, uma técnica para detecção de colisão mais eficiente será apresentada mais a frente, na subseção 5.1.3.

5.1.2 O ambiente de testes para o MDE

Posteriormente à verificação dos elementos fundamentais do algoritmo RRT básico, decidiu-se representar o robô espacial em sua forma equivalente MDE.

O conceito de MDE, visto em detalhes na seção 3.4, mapeia o manipulador espacial em um sistema robótico equivalente. Os parâmetros de equivalência do robô UARM-E, caracterizados como robô 1 na tabela 3.3, são incorporados ao modelo equivalente utilizado durante esta etapa do projeto. A forma adotada para a representação do manipulador foi a de uma cadeia cinemática aberta, representada por segmentos de reta. Adicionalmente, foi introduzida apenas uma verificação simples de intersecção entre segmentos de reta para averiguar colisão com o próprio robô próprio UARM-E. Além disso, foi implementada uma verificação de limitação angular entre elos subsequentes, baseada nas características do ME. O ambiente, por sua vez, é definido livre de obstáculos fixos e o espaço de configuração livre do robô é portanto representado por um círculo de raio W , onde W corresponde à soma do comprimento de todos os elos do MDE, conforme ilustra a figura 5.2.

Figura 5.2: Representação ilustrativa do espaço de trabalho de um manipulador de três elos. O valor de z depende dos valores de W_1 , W_2 e W_3 .

A justificativa para essa abordagem se deve à ausência de equivalência geométrica entre o manipulador espacial e o MDE. Embora as variáveis de junta e posição do efetuador final sejam coincidentes a todo instante, os modelos ocupam posições diferentes no espaço de

trabalho. Por conseguinte, a introdução de um algoritmo para detecção de colisão nesse momento não teria finalidade até que o manipulador fosse construído apropriadamente a partir do modelo espacial. Essa diferença é ilustrada na figura 5.3.

Figura 5.3: Representação ilustrativa da diferença geométrica entre o MDE e o ME.

5.1.3 O ambiente com suporte à detecção de colisão

Para integrar o ambiente de testes definitivo, o robô espacial foi finalmente caracterizado e para ele foi implementada uma verificação de colisão robusta, baseada no teorema dos eixos separadores.

O teorema dos eixos separadores - SAT (do inglês, *Separating Axis Theorem*), proposto por Gottschalk (1996), foi implementado nesse trabalho a fim de realizar uma rápida detecção de colisão entre os obstáculos e o robô. O teorema afirma que dois objetos convexos não se intersectam se existir pelo menos uma reta (chamada aqui de eixo) na qual as projeções dos objetos não se sobrepõem. Trata-se de um caso particular do teorema dos hiper-planos separadores de Minkowski, aplicado ao problema de detecção de colisão. A figura 5.4 apresenta de forma ilustrativa o princípio básico do teorema dos eixos separadores.

Tendo em vista a aplicabilidade do teorema restrita a polígonos convexos, cada elo do robô, incluindo a base foi descrito apropriadamente como um retângulo. O manipulador espacial planar de base livre flutuante foi portanto considerado como uma cadeia cinemática aberta, composta por objetos poligonais conectados entre si.

Figura 5.4: Representação ilustrativa do teorema dos eixos separadores

O algoritmo 2 apresenta o pseudo-código do teorema dos eixos separadores para dois objetos bidimensionais, O_1 e O_2 . Neste algoritmo é feito um “varrimento” desde 0 a 360° para o ângulo de inclinação β do eixo separador, com um incremento de ξ entre cada verificação de sobreposição de projeção.

Algoritmo 2 Algoritmo SAT

SAT(O_1, O_2)

```

1: estado  $\leftarrow$  COLIDE;
2:  $\beta \leftarrow 0$ ;
3: while ( $\beta < 360$ ) AND (estado = COLIDE) do
4:   eixo  $\leftarrow$  Amostra_Eixo( $\beta$ );
5:   if Projeta( $O_1, O_2, eixo$ ) = separado then
6:     estado  $\leftarrow$  NAO_COLIDE
7:     break;
8:   else
9:      $\beta = \beta + \xi$ 
10:  end if
11: end while
12: return estado;

```

A respeito da execução do SAT básico, apresentado no algoritmo 2, algumas particularidades foram observadas e implementações foram feitas visando reduzir o custo computacional da verificação ou simplesmente adaptar o teorema à natureza do problema. Primeiramente, os eixos considerados serão sempre perpendiculares a retas paralelas aos lados dos objetos analisados. Ora, para que as projeções não se sobreponham, há pelo menos uma reta que passa entre os objetos sem tocá-los. Na figura 5.4, essa reta é chamada de reta separadora. Por conseguinte, há pelo menos uma reta, paralela ao lado de um dos objetos em questão, que também passa livremente entre os dois objetos. A segunda simplificação é uma decorrência desse fato. Como um eixo é perpendicular a dois lados paralelos quaisquer, o algoritmo implementado desconsidera uma nova verificação para o outro lado paralelo de um determinado objeto. Em outras palavras, no caso de um retângulo, verifica-se a sobreposição em apenas, no máximo, dois eixos. Por fim, não é realizada uma verificação de colisão entre dois objetos subsequentes. A razão para esse fato está na verificação de limitação angular ser realizada anteriormente à execução do algoritmo SAT. Logo, se os ângulos de junta em determinado momento são factíveis, os elos subsequentes seguramente não colidirão.

5.2 Implementação e avaliação do “método das caixas”

Uma das preocupações iniciais na confecção do algoritmo foi que o mesmo sofresse uma degradação no desempenho devido a elevado carregamento computacional. Para essa finalidade, foi implementado o “método das caixas”, apresentado por Svenstrup et al. (2011). A seção 4.3.2 cobre os principais aspectos desse algoritmo. Resumidamente, essa técnica visa diminuir consideravelmente o custo computacional através do emprego de uma heurística baseada na divisão do ambiente em caixas uniformemente espaçadas.

Com o objetivo de avaliar o desempenho do método, foi empregado o ambiente descrito na subseção 5.1.1. A metodologia empregada para a avaliação desta abordagem foi realizada conforme a seguinte sequência de etapas:

1. Armazena-se o conjunto de amostras necessárias para chegar a uma solução;
2. Resolve-se o mesmo problema várias vezes, aplicando o conjunto de amostras salvo

e iniciando o algoritmo a partir de um valor limiar diferente Γ ;

3. Compara-se os resultados através do tempo despendido e do número necessário de iterações.

Este procedimento foi realizado por 100 vezes. Foi avaliado, em cada uma dessas vezes, o tempo necessário e o número de amostras. As observações mais importantes extraídas desse experimento são:

1. Para um valor de Γ inferior a 40%, o tempo necessário para atingir o objetivo é de 1,2 a 6 vezes superior ao tempo comparado (sem influência do método);
2. Para diferentes problemas são encontradas faixas distintas de valores Γ nas quais o método passa a apresentar melhoria.
3. A redução de custo computacional total foi, para o melhor caso avaliado, igual a 6,77%;
4. O método é baseado em uma métrica exclusivamente dependente da distância euclidiana.

Para computar a pequena faixa na qual o método passa a ser eficiente é necessário empregar um procedimento de investigação exaustivo. Esse fator, quando confrontado ao conceito de busca única do planejador RRT, apresenta uma severa desvantagem em termos de calibração e autonomia do método.

A aplicação do método é imediata em ambientes onde o robô pode ser reduzido a um simples ponto. Entretanto, manipuladores robóticos não podem valer-se de tal simplificação. Em consequência disso, os algoritmos-base necessitam de adaptações no intuito de integrar os vários pontos do manipulador a uma única configuração e interpretá-los de modo a simplificar as pesquisas de vizinhança do algoritmo RRT.

A tarefa de encontrar um valor ideal Γ ou uma região que reduza o tempo computacional para um determinado problema de planejamento provou não ser uma tarefa simples. Em vista dos fatores acima observados, o método não se revelou viável para integrar o planejador de trajetórias e por esse motivo não foi considerado posteriormente nesse trabalho.

5.3 Implementação do método de redução de sensibilidade da métrica

O problema que envolve a dependência da qualidade da métrica escolhida na convergência para uma solução foi abordado na seção 4.3.3. A tentativa de crescer uma árvore baseada simplesmente no algoritmo RRT básico para um manipulador espacial, levando em conta o ambiente proposto e descrito em 5.1.3, fracassou em encontrar uma solução para o problema. As métricas avaliadas não expandiam a árvore adequadamente e conduziam a busca invariavelmente a mínimos locais. Na tentativa de contornar essa questão, foi implementado e adaptado o trabalho proposto em (Cheng e LaValle; 2001), cujo funcionamento é explicado em detalhes na seção 4.3.3.

Basicamente, o algoritmo proposto muda o RRT básico em dois procedimentos fundamentais. Primeiramente, o procedimento responsável por encontrar o nó mais próximo sofre uma alteração considerável, mostrada em detalhes no algoritmo 3 (vide Anexo 1). Já o procedimento responsável por encontrar uma entrada adequada e fazer a integração numérica tem sua forma alterada mostrada no algoritmo 4 (vide Anexo 2). Dessa maneira é possível, ao menos teoricamente, não somente poupar tempo computacional como também premiar outras possibilidades de expansão. Uma vez que são acumuladas informações sobre a taxa de sucesso para cada nó, bem como mantido o registro sobre a expansão ou não de determinada entrada, o algoritmo mostrou ser uma alternativa promissora.

Entretanto, o algoritmo necessitava adaptar-se à realidade do problema em questão. Como princípio fundamental para o sucesso na implementação do mesmo, deveria ser mantido registro de todas as entradas possíveis e suas eventuais expansões, resultantes em sucessos ou fracassos. Contudo, as primitivas de movimento eram até então geradas através do sorteio de uma entrada, esta por sua vez realizada segundo uma distribuição uniforme. Essa entrada sorteada consistia de um pequeno acréscimo ou decréscimo na entrada (torque) do sistema. Consequentemente havia ali uma infinidade de entradas possíveis e portanto a discretização das mesmas era necessária para que o método proposto fosse empregado.

Na realidade, a não-discretização da entrada na implantação desse método acarreta

um problema antagônico ao próprio conceito proposto por ele. Como não há o descarte de estados infrutíferos na consideração para estados posteriores, o algoritmo insiste em explorar locais não-promissores e cria uma nuvem de estados novos que permanentemente expandem-se para o mesmo local, com parâmetros de torque e velocidade praticamente idênticos. Esse comportamento aberrante foi verificado experimentalmente e logo em seguida rejeitado.

O procedimento adotado para a discretização das entradas é explicado em detalhes a seguir. Seja um estado qualquer $x \in X$, cujo nó pai é x_{pai} . Considere o conjunto de entradas necessárias para levar x_{pai} a x , dentro de um determinado instante Δt , representado por um vetor de torques $\tau = \begin{pmatrix} \tau_1 & \tau_2 & \cdots & \tau_n \end{pmatrix}^T$, onde n denota o número de membros do manipulador espacial. Impondo um limiar igual a $\pm \Delta_i$ no acréscimo/decréscimo de τ_i , e lembrando que o torque $\tau_1 = 0$ devido às condições definidas para a equivalência do MDE, resulta no conjunto de entradas discretizadas mostrado na tabela 5.1.

Tabela 5.1: Organização das entradas discretas

0	...	0	...	0
$\tau_2 - \Delta_2$	...	τ_2	...	$\tau_2 + \Delta_2$
$\tau_3 - \Delta_3$	...	τ_3	...	$\tau_3 + \Delta_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\tau_{n-1} - \Delta_{n-1}$	...	τ_{n-1}	...	$\tau_{n-1} + \Delta_{n-1}$
$\tau_n - \Delta_n$	...	τ_n	...	$\tau_n + \Delta_n$

 k colunas

Dessa maneira, sendo k o grau de discretização de cada uma das entradas em τ , o número total possível de combinações de entrada é igual a k^{n-1} . Daí decorre que cada nó da árvore necessitará de, no mínimo, um vetor V de informações de tamanho k^{n-1} . Esse vetor V deve manter registros de todas as expansões, referentes a todas as combinações de entrada possíveis para aquele nó, indicando se já foram avaliadas ou não. Entretanto, como associar um respectivo conjunto de entradas a um elemento específico no vetor V ?

No intuito de solucionar esse problema e ainda utilizar a memória computacional de forma eficiente, o vetor V é construído da seguinte forma. São armazenados, para uma determinada entrada amostrada, os índices $\begin{pmatrix} i_2 & i_3 & \cdots & i_n \end{pmatrix}$ das colunas referentes à matriz de torques apresentada na tabela 5.1. Esse conjunto de entradas será portanto

atrelado à linha L do vetor V conforme computado na equação 5.2.

$$L = 1 + \sum_{j=2}^n (i_j - 1)k^{n-j}. \quad (5.2)$$

Outra adaptação feita para que um desempenho satisfatório fosse atingido refere-se à atualização de informações. Considere U o conjunto total de entradas possíveis para um estado x e $U_s \subset U$ o conjunto de entradas sorteadas e avaliadas para o mesmo estado. Uma heurística foi empregada para descartar estados com alta probabilidade de condução à colisão. A seguinte simplificação foi feita: “Caso toda entrada em U_s seja avaliada como resultante em colisão, o estado x é desconsiderado de futuras extensões e as FVRs dos nós-pai associados a ele são punidos de acordo”. Ora, considere que o conjunto de entradas é sorteado aleatoriamente. Se todas as entradas em U_s resultam em colisão e U_s representa uma parcela significativa de todo o conjunto U , há uma alta probabilidade de todas as entradas em U também levarem a colisão ou a um estado improdutivo.

Essa simplificação foi utilizada pois observou-se que nós infrutíferos permaneciam sendo selecionados, mesmo quando a maioria das entradas em U já tinha sido avaliada e descartada. O nó x não era eliminado das buscas pois havia entradas em U ainda não avaliadas. Ademais, novos nós gerados possuem uma FVR nula. Logo, avaliar cada entrada em U referente a um nó infrutífero consome um tempo precioso. Esse tempo poderia ser gasto na computação de um caminho que de fato levasse o robô ao destino.

O incorporamento do algoritmo proposto no código-base tem seus resultados apresentados e discutidos posteriormente na seção 6.3.

5.4 Implementação do método das árvores bidirecionais

A utilização de árvores bidirecionais na busca por uma solução no algoritmo RRT foi proposta inicialmente por LaValle e Kuffner Jr. (1999) e discutida em detalhes na seção 4.3.4. Embora esse método teoricamente proponha melhorar substancialmente a convergência da busca, diversos são os fatores que desencorajam a recorrência a essa abordagem. São eles: a necessidade de uma descrição da cinemática inversa; a mudança

no sentido de integração quando se trabalha com a árvore crescente no sentido inverso e o consequente acréscimo de custo computacional; a recorrência a um algoritmo para conexão entre os nós das árvores expandidas.

Apesar de todas as complicações associadas a essa abordagem, é praticamente unânime a utilização desse recurso nos trabalhos envolvendo planejamento de rota RRT em manipuladores fixos. A razão para esse fato está na dificuldade encontrada na convergência para um determinado estado. Uma vez que na abordagem bidirecional, uma das árvores “nasce” justamente no estado desejado, o problema de convergência é deslocado para a questão da conexão entre estados suficientemente próximos.

Considere Δt o tempo de integração utilizado durante o crescimento da árvore principal. O problema da inversão do sentido de integração foi solucionado através da utilização de um tempo de integração $\Delta t' = -\Delta t$, cada vez que uma expansão no sentido contrário era exigida. O motivo para essa consideração reside na equivalência $\int_a^b f(x)dx = -\int_b^a f(x)dx$.

Para solucionar a questão da cinemática inversa, foi implementado um algoritmo que calcula iterativamente a posição das juntas anteriores ao efetuador final e considera as limitações físicas do robô. Ao fim desse processo, é construído o manipulador espacial a partir do MDE gerado. Finalmente é realizada a verificação de colisão através do algoritmo SAT, em que este é responsável por verificar colisão com obstáculos e com o próprio robô.

Através da disposição da cinemática inversa para a configuração final, a localização das outras posições de junta é agora conhecida. A adição dessas informações na computação da métrica é fundamental no sentido de considerar mais parâmetros e portanto traduzir mais fielmente o custo entre dois estados.

No intuito de unir dois nós pertencentes a duas árvores distintas, o critério adotado para a conexão consistiu em encontrar um valor de $x = \sum dist(x_a - x_b) < \varepsilon$, para um valor pequeno de ε . Nesse caso, x_a e x_b representam o conjunto de coordenadas das juntas e do efetuador final para duas configurações a e b , respectivamente. Uma vez encontrado um $x < \varepsilon$, o algoritmo conecta todos os nós-pai de x_a e x_b . O caminho retornado pelo planejador é então capaz de conduzir um manipulador espacial planar de base livre flutuante desde a configuração inicial até a configuração final.

5.5 Funcionamento do planejador

Com o objetivo de fornecer detalhes sobre a estrutura do algoritmo proposto nesta dissertação, a figura 5.5 mostra os principais blocos de construção e como estão inseridos no contexto geral. Dependendo da complexidade e tamanho de cada bloco, alguns aspectos do funcionamento são omitidos, porém sem intenção de prejudicar o entendimento do todo. Da mesma forma, as passagens de parâmetros entre os blocos podem revelar somente ideia das entradas e saídas. Dentre os blocos principais, destacam-se:

Figura 5.5: Fluxograma resumido do planejador proposto

- **Add to Tree:** Adiciona novas configurações e ramos à árvore.
- **Build Path:** Constrói o caminho entre as configurações inicial e final, baseada na conexão das duas árvores.
- **Build SM:** Constrói o manipulador espacial planar de base livre flutuante a partir do MDE gerado. Essa construção é feita aplicando cinemática direta a partir do efetuador final.

- **Check Connection:** Aplica o algoritmo de verificação de conexão entre duas árvores.
- **Closest Neighbor:** Verifica o nó na árvore mais próximo. O pseudo-algoritmo é mostrado no algoritmo 3 (vide Anexo 1).
- **Collision Detection:** Verifica todas as entradas sorteadas e identifica aquelas que estão livres de colisão. Compreende os blocos unificados de verificação de ângulo limite e algoritmo SAT.
- **IK Solver:** Computa a cinemática inversa para a configuração final, conforme descrito na seção 5.4.
- **Init Map:** Carrega os obstáculos presentes no mapa. Estes são descritos como objetos poligonais.
- **Init Parameters:** Carrega todos os parâmetros constantes do manipulador e do algoritmo. Inicializa também os parâmetros variáveis definidos pelo usuário, tais como descrição da tarefa e viés desejado.
- **Input Sampler:** Amostra um conjunto de entradas a partir do conjunto total de entradas (U_s a partir de U). Considere a tabela 5.1 representando as entradas discretizadas. Uma quantidade $m < k$ de amostras é sorteada para cada junta, sendo k o grau de discretização. Dessa maneira m^{n-1} conjunto de entradas são sorteados.
- **Model Integration:** Integra o modelo dinâmico do MDE para todas as entradas amostradas.
- **Random Sampler:** Sorteia uma configuração aleatória válida. Para essa finalidade, uma sequência de ângulos é sorteada para a construção do MDE. O MBLF é então construído a partir do MDE. Caso seja detectada alguma colisão ou limitação física do robô, uma nova amostra é computada. A mesma configuração amostrada é utilizada para a árvore que cresce no sentido oposto, chamada aqui de árvore 2.
- **Reset:** Reinicia o planejador caso não haja uma aproximação considerável nas primeiras 1000 iterações.

- **Select Best Set:** Verifica dentre todas as expansões computadas, aquela que está livre de colisão e possui o menor custo de movimento para a configuração desejada.
- **Update Info:** Atualiza os valores de FVR, marca as entradas que já foram avaliadas e aplica a heurística proposta na seção 5.3.

Capítulo 6

Resultados

O presente capítulo apresenta os resultados obtidos durante as etapas críticas de desenvolvimento desse trabalho, apresentadas no capítulo 5, utilizando os métodos e conceitos propostos nos capítulos 3 e 4. A seção 6.1, explana a metodologia empregada para a avaliação do caso do robô pontual e expõe os resultados obtidos. A seção 6.2 propõe um planejamento de trajetória baseado no MDE. Mostra-se, além dos resultados, a abordagem utilizada para chegar em uma solução. A seção 6.3 promove uma análise da influência do emprego do método de redução de sensibilidade métrica no contexto do trabalho. Finalmente, a seção 6.4 mostra os resultados do algoritmo definitivo e discute as soluções encontradas.

Todos os resultados aqui mostrados foram obtidos à partir de simulações, executadas em MATLAB - versão R2012a, através de códigos *.m*. Todos os experimentos foram realizados em um computador com processador *Intel® Core i7*, 3.40GHz e 12Gb de memória RAM.

6.1 Convergência para o robô pontual

Para alcançar a convergência do planejador no caso do robô pontual, foi utilizado o ambiente de testes inicial, descrito na subseção 5.1.1. No que tange a geração de primitivas, foi selecionado um tempo de integração $\Delta T = 0.002s$ e utilizado o método de

Euler para a integração numérica. A métrica ρ utilizada para a busca do nó mais próximo e da melhor primitiva gerada foi baseada simplesmente na distância euclidiana entre dois estados.

Com base na metodologia adotada, o algoritmo proposto foi executado por um total de 100 vezes. Visando limitar a quantidade de buscas em cada execução do algoritmo, um total de 10000 iterações foi definido como limite máximo. Tendo esses parâmetros e condições em vista, 96% dos casos retornaram uma solução válida. A figura 6.1 ilustra uma resposta típica encontrada pelo planejador.

Figura 6.1: Resposta típica encontrada pelo planejador, destacada em vermelho.

6.2 Convergência para o MDE

Os testes realizados nesta etapa basearam-se no uso do manipulador dinamicamente equivalente ao UARM-E, no intuito de planejar uma trajetória válida. Para esse propósito, o ambiente definido na seção 5.1.2 foi utilizado.

Uma vez definido os aspectos gerais do ambiente e parâmetros do robô, foi designada

a tarefa de planejar a seguinte trajetória, para o robô configurado com três elos: partir desde um estado inicial, com velocidades e torques nulos em todas as juntas e chegar em um estado final, com velocidades e torques também nulos. A configuração inicial adotada foi uma em que todos os ângulos das juntas são nulos, isto é, o robô possui todas as juntas localizadas no plano horizontal. Como configuração desejada, o efetuador final deve encontrar um ponto arbitrário $Z = (0.45, 0.08)$, com uma orientação de $\theta = 33^\circ$. A motivação de atribuir uma simples posição no espaço para o efetuador final reside em duas razões principais: 1) Especificar um objeto/alvo a ser alcançado pelo efetuador final, independentemente das configurações de junta; 2) Evitar o problema de cinemática inversa.

A fim de planejar a trajetória referente a esse problema, uma sintonia da métrica ρ deveria ser previamente realizada. Como decidiu-se escapar dos problemas relacionados à cinemática inversa, uma descrição explícita dos valores de cada junta para a configuração final não estava disponível. Contudo, a posição D_{n-1} da última junta pôde ser calculada a partir da posição do efetuador final através da equação 6.1. Essa equação é parte da solução do problema de cinemática inversa para esse robô.

$$D_{n-1} = D_n - W_{n-1} \begin{pmatrix} \cos \theta_{n-1} \\ \sin \theta_{n-1} \end{pmatrix}. \quad (6.1)$$

Na equação 6.1, D é a configuração do manipulador, D_n denota a localização $\begin{pmatrix} x & y \end{pmatrix}^T$ do efetuador final, W_{n-1} representa o comprimento do último elo do MDE e θ_{n-1} corresponde à orientação do efetuador final.

Via de regra, a métrica ρ envolve uma ponderação de distâncias euclidianas. Nesse sentido, o conhecimento da localização de duas juntas no espaço cartesiano permite parametrizar duas variáveis, σ_n e σ_{n-1} , relativas à distância euclidiana entre a posição dos efetuadores finais e das últimas juntas em dois estados distintos, respectivamente. A equação 6.2 mostra os valores dessas variáveis para uma comparação entre duas configurações D e D' .

$$\begin{aligned}\sigma_n^2 &= (D_n - D'_n)^T (D_n - D'_n), \\ \sigma_{n-1}^2 &= (D_{n-1} - D'_{n-1})^T (D_{n-1} - D'_{n-1}).\end{aligned}\tag{6.2}$$

Entretanto, uma métrica exclusivamente dependente de σ_n e σ_{n-1} mostrou não ser capaz de retornar uma solução quando consideramos os parâmetros de velocidade e torque. O fato do algoritmo RRT ser probabilisticamente completo não assegura que o programa encontre uma solução dentro de um tempo factível. A experiência revelou que considerar somente as distâncias euclidianas resulta em uma trajetória onde a velocidade e torque aumentam sem limites à medida que o manipulador se aproxima da configuração final. Esse fato é compreensível, já que quanto mais próximas duas configuração forem uma da outra, menor será o custo, independentemente de outros fatores.

Portanto, fez-se necessária a introdução dos parâmetros de velocidade e torque na computação da métrica em vigor. De maneira geral, a função custo adotada deveria dar cada vez mais importância aos valores de velocidade e torque à medida que a configuração fosse se aproximando daquela desejada. Em outras palavras, a tarefa consistia em encontrar uma métrica ρ que traduzisse adequadamente o valor do custo de movimento entre dois estados. A equação 6.3 considera os principais fatores deste movimento.

$$\rho(x) = k_1(x)x + k_2(x)\tau + k_3(x)v.\tag{6.3}$$

Na equação 6.3, a variável τ denota a diferença total entre torques correspondentes de dois estados a e b , ou seja, $\tau = \sum |\tau_a - \tau_b|$. De maneira análoga, v representa as diferenças de velocidade, onde $v = \sum |v_a - v_b|$. A variável principal x foi considerada como $x = -(d_{n-1} + d_n)$. Dessa forma, ao passo que uma configuração se aproxima da outra, x tende a 0. As variáveis k_1 , k_2 e k_3 correspondem às ponderações dos custos de distância euclidiana, torque e velocidade, respectivamente. Essas variáveis k foram definidas então como funções de x no intuito de balancear as diferentes grandezas. Foram avaliadas inicialmente duas possibilidades principais para estimar uma função para k_i . Nomeadamente, as abordagens linear e exponencial foram consideradas.

Como resultado de observações no crescimento da árvore, as abordagens caíam invaria-

velmente no problema de mínimos locais. À medida que a métrica favorecia os parâmetros de velocidade e torque em detrimento da distância euclidiana, o custo computado evitava explorações no rumo de uma possível solução. A principal razão para a ocorrência desse fato estava na diferença de magnitudes entre os diferentes parâmetros computados aliada a uma seleção inadequada de função modelo. Como contra-medida, a função sigmoide foi escolhida para representar k tendo em vista sua natureza de transição suave. Após uma calibração das magnitudes, as funções genéricas apresentada na equação 6.4 foram capazes de gerar uma solução viável, dentro de uma tolerância estipulada, após somente 688 iterações. A resposta desse planejamento pode ser verificada na figura 6.2.

$$\begin{aligned} k_1 &= 0.8 - \frac{0.1}{1 + e^{(-5-100x)}}, \\ k_2 &= \frac{1.5 \cdot 10^{-3}}{1 + e^{(-6-50x)}}, \\ k_3 &= \frac{5 \cdot 10^{-6}}{1 + e^{(-5-120x)}}. \end{aligned} \tag{6.4}$$

A figura 6.2 mostra que o planejador consegue boas aproximações em termos de posição e velocidade. Apesar da variável torque terminar o planejamento com um valor pequeno, uma calibração mais refinada de k_2 poderia resultar em um valor mais próximo de zero. Entretanto, este objetivo não foi perseguido pela seguinte razão. Como o torque é o parâmetro controlável, o erro relacionado a ele pode ser solucionado através do desligamento dos motores ao fim do planejamento. Em seguida, o emprego de uma técnica de controle poderia ser aplicada de modo a estabilizar na solução almejada.

O planejamento de trajetória foi apresentado nesta etapa com o objetivo de mostrar as dificuldades relacionadas à convergência nesse tipo de abordagem, mesmo quando na presença de um ambiente simples e contando com um manipulador de geometria simplificada. A partir da próxima seção todos os esforços serão concentrados no sentido de resolver o planejamento de rota.

Figura 6.2: Resposta do planejador de trajetória

6.3 Avaliação do método da redução de sensibilidade métrica

Após os resultados obtidos exclusivamente com o MDE, uma nova tarefa foi proposta, desta vez considerando a geometria do UARM-E. A tarefa consistia em partir de uma configuração inicial, onde todos os ângulos de juntas são nulos, e mover o efetuador a um ponto arbitrário $Z = (0.30, 0.15)$, sob uma orientação $\theta = 45^\circ$. Além disso, o ambiente passou a contar com um obstáculo retangular fixo, localizado dentro do espaço de trabalho

do robô. A figura 6.3 mostra o UARM-E na configuração inicial proposta.

Figura 6.3: A geometria do UARM-E é destacada em vermelho. O MDE aparece como um segmento de reta azul. O obstáculo segue representado na cor preto. Finalmente, uma região de tolerância é definida ao redor do ponto de destino e representada na cor verde.

Os resultados observados com a utilização do algoritmo desenvolvido até aquele momento indicaram um crescimento insatisfatório da árvore. Verificou-se a presença de zonas de mínimo local durante a busca, localizadas em torno de duas configurações específicas, representadas nas figuras 6.4(a) e 6.4(b).

Figura 6.4: Limitações na expansão da árvore RRT, indicando a presença de mínimos locais.

A razão para esse comportamento está no fato de que os parâmetros de massa e momento de inércia da base são consideravelmente maiores que os referentes a outros

elos do manipulador. Para executar um movimento de desvio de obstáculo, pode ser necessário um deslocamento considerável da base, caso o robô não possua muitos graus de liberdade. Como não se dispõe da posição da primeira junta do manipulador espacial, o deslocamento da base não pode ser equacionado na função custo. Como consequência, o algoritmo prefere premiar uma rápida aproximação da última junta e efetuator final a explorar maiores possibilidades de expansão da árvore. Uma exploração mais refinada está associada a um maior deslocamento da base, e como esse parâmetro não está sendo levado em conta, o algoritmo é conduzido a zonas de mínimo local, como mostra a figura 6.4. A configuração mostrada na figura 6.4(a) está a ponto de colidir com o obstáculo. Já a figura 6.4(b) mostra o manipulador em uma configuração onde os últimos elos estão em seus limites de abertura.

No intuito de tornar o planejador mais robusto à métrica empregada, foi implementada a metodologia descrita na seção 5.3. A geração de primitivas foi baseada em um passo de integração $\Delta T = 0.002s$. O método de Euler foi utilizado para resolver a integração numérica. Os testes conduzidos utilizaram entre 1000 e 25000 iterações.

Para o caso de um manipulador, uma possível representação ilustrativa da árvore RRT é através da plotagem de todas as suas configurações. Entretanto, é complicado visualizar essa expansão pois rapidamente é formada uma nuvem de representações geométricas do robô, impossibilitando assim a distinção da árvore. Para melhor visualizar o crescimento da árvore RRT, decidiu-se plotar todos os pontos referentes às posições do efetuator final. A figura 6.5 mostra diferentes expansões, obtidas para quatro limites distintos de iteração.

Figura 6.5: Em 6.5(a), a árvore possui somente 1000 nós. A árvore mostrada em 6.5(b) é composta por 5000 nós. Em 6.5(c), 15000 nós estão densamente distribuídos. Em 6.5(d), 25000 nós constituem a árvore melhor expandida.

Os resultados mostraram que o planejador conseguiu escapar da zona mínimos locais. Isto foi possível graças às informações adquiridas ao longo do planejamento contribuírem com uma melhor exploração do ambiente. Entretanto, verificou-se também que não foi possível crescer uma árvore entre as configurações especificadas dentro do período avaliado. Não foi observada alguma expansão no sentido de explorar a região de tolerância.

Embora haja uma tendência do algoritmo a explorar cada vez mais o espaço de trabalho livre do robô, como observado na figura 6.5, o fator custo computacional deve ser considerado para algoritmos dessa complexidade. Quanto maior a expansão da árvore,

mais demorada será a computação de um única iteração. A figura 6.6 retrata o aumento do custo computacional à medida que o algoritmo vai realizando cada vez mais iterações. Esse gráfico foi obtido através do algoritmo com suporte de métrica, após 50000 iterações.

Figura 6.6: Crescimento do custo computacional com o número de iterações

Observa-se que o tempo necessário para adicionar um estado evolui praticamente de forma linear com o tempo. A natureza ruidosa do gráfico apresentado se deve a dois motivos principais. Primeiramente, configurações aleatórias são geradas em condições de colisão ou ineficazes. Dessa maneira, o algoritmo de sorteio deve ser repetido até que uma configuração válida seja sorteada. O segundo motivo deve-se a um possível fracasso no encontro de uma configuração livre de colisão, o algoritmo então repete a busca, considerando a expansão de outro nó.

6.4 Avaliação do método das árvores bidirecionais

As principais dificuldades encontradas na implementação do algoritmo de crescimento de uma árvore RRT bidirecional tiveram soluções propostas e explicadas em detalhes na seção 5.4. Para a geração de primitivas foi determinado um tempo de integração $\Delta T = 0.002s$ e utilizado o método de Euler para a integração numérica.

Para esta etapa de implementação, a representação da expansão da árvore deu-se através do método descrito na seção 6.3. Nesse caso, a árvore que cresce a partir do destino foi representada na cor vermelho. Dessa maneira, é possível saber se duas configurações estão pelo menos próximas e, com base nisso, utilizar heurísticas para realizar uma conexão

entre os nós.

Uma das primeiras observações sobre o crescimento da árvore bidirecional estava em sua expansão acelerada durante as primeiras iterações. Foi verificado que essas iterações iniciais eram responsáveis pela criação do “tronco” principal da árvore. É natural que as iterações seguintes contribuam para a expansão da RRT, mas essa constatação encorajou à seguinte simplificação no algoritmo de busca: “Caso não exista duas posições de efetuador final e_1 e e_2 suficientemente próximas, isto é, $dist(e_1, e_2) < \delta$ para um δ pequeno, durante as primeiras K iterações, a busca é reiniciada”. Os valores de $\delta = 0.05$ e $K = 1000$ foram utilizados. A justificativa para essa simplificação pode ser vista na figura 6.7.

Figura 6.7: A figura 6.7(a) mostra uma expansão com 1000 nós, em que não há $dist(e_1, e_2) < \delta$. A figura 6.7(b) mostra a mesma expansão após 10000 nós.

Os resultados foram obtidos a partir da manipulação da função custo e do viés para o estado final. O emprego de funções sigmóides foi mantido e a sintonia dos parâmetros resumiu-se à equação 6.3, com o diferencial de incluir também a posição da primeira junta do UARM-E. Assim sendo, o custo é agora função de uma nova variável $x = -\sum |dist(a, b)|$, em que $dist(a, b)$ representa as distâncias euclidianas entre todas as posições de junta do manipulador, relativas a duas configurações a e b . Foram atribuídos pesos diferentes na computação do custo de cada junta. Os deslocamentos associados à

base, por exemplo, foram equacionados de modo a impactar no custo de movimento de maneira mais incisiva. Vale notar que a cinemática inversa possui duas soluções para a posição da segunda junta. O algoritmo pode então manter essa escolha fixa ou aleatória. A figura 6.8 mostra o resultado de diferentes configurações para a mesma métrica.

Figura 6.8: Na figura 6.8(a), é aplicado um viés $p = 100\%$ por 1000 iterações. Em 6.8(b), 2000 iterações são utilizadas e nesse caso $p = 5\%$. Para a figura 6.8(c), $p = 10\%$ e a solução da cinemática inversa é fixa. Finalmente, a figura 6.8(d) une duas posições dentro de 10000 iterações, com $p = 10\%$ e solução de cinemática inversa aleatória.

Percebe-se a sensibilidade do viés empregado na qualidade da solução. O algoritmo não encontra solução por diversas vezes em condições desfavoráveis àquela situação específica. Inclusive, para o caso mostrado na figura 6.8(d), o encontro de posições é passível de

não acontecer, mesmo após milhares de nós serem criados. Como observado, as iterações iniciais tendem a definir se um crescimento será promissor ou não.

O algoritmo aumenta consideravelmente o número de pontos de interesse ao passo que a árvore expande de modo significativo. Por pontos de interesse, entende-se um par de configurações em que os efetuidores finais estejam suficientemente próximos, nesse caso específico $dist(e_1, e_2) < 0.05$. A figura 6.9 representa uma expansão satisfatória contando com 40768079 pontos de interesse.

Figura 6.9: Expansão após 50000 nós.

Para finalizar o problema do planejamento de rota, um valor $|x| < \varepsilon$ deve ser encontrado para um pequeno valor de ζ . Lembrando que definiu-se $x = -\sum |dist(a, b)|$, a procura por um valor pequeno de $|x|$ representa justamente o ponto de encontro entre as duas árvores. Dessa forma, suponhamos que os nós x_1 e x_2 , pertencentes às árvores T_1 e T_2 , respectivamente, satisfaçam $|x| < \varepsilon$. O algoritmo então verifica o conjunto de nós desde a raiz de ambas as árvores. Todos esses nós são então conectados a partir da origem até o destino especificado de uma maneira unidirecional. Atribuindo um valor de $\varepsilon = 10^{-5}$, o algoritmo foi capaz de planejar um caminho para o problema proposto em diversas situações.

Com o objetivo de avaliar o desempenho do planejador, foram executados 100 testes de avaliação. Em média, o algoritmo necessitou de 3827 iterações para encontrar uma solução. Houve 11 casos em que o planejador não conseguiu encontrar uma solução, mesmo após 25000 iterações serem avaliadas. Dentre as soluções encontradas, a que

obteve o melhor desempenho apresentou uma solução após 1604 iterações. Já o caso que mostrou o pior desempenho, retornou uma solução após 16445 iterações. A figura 6.10 exemplifica algumas dessas expansões e seus respectivos caminhos computados.

(a)

(b)

(c)

(d)

(e)

(f)

Figura 6.10: As figuras 6.10(a), 6.10(c) e 6.10(e) representam expansões com 16445, 7280 e 4916 nós, respectivamente. As rotas associadas a essas figuras foram conectadas pelo planejador a partir de 90, 89 e 111 nós, respectivamente. A rota computada é representada pelos nós verdes nas figuras 6.10(b), 6.10(d) e 6.10(f)

Capítulo 7

Conclusão

7.1 Conclusões Gerais

Esta dissertação de mestrado propôs uma nova abordagem integrada para o problema de planejamento de rota para um manipulador espacial de base livre flutuante (MBLF). O algoritmo geral foi moldado a partir dos conceitos básicos de RRTs, no intuito de planejar rotas livres de colisão de forma automática, conforme apresentado no capítulo 4. Com o objetivo de contornar a dinâmica complexa e os problemas intrínsecos ao planejamento de movimento de MBLFs, foi utilizado o conceito de modelo dinamicamente equivalente (MDE). Contudo, a simples associação do MDE ao planejador de rota RRT básico mostrou não ser capaz de retornar uma solução quando obstáculos são considerados no planejamento.

Os resultados apresentados no capítulo 6 compreenderam diversas etapas do desenvolvimento do trabalho. Um planejador simples foi inicialmente construído no intuito de avaliar os blocos integrantes principais do algoritmo RRT. A segunda fase de testes mostrou o raciocínio utilizado para resolver o problema do planejamento de trajetória com base exclusivamente no MDE ao robô UARM-E em um ambiente desprovido de obstáculos. A terceira fase adicionou um obstáculo no planejamento e considerou a descrição geométrica do robô UARM-E. Foi mostrada a necessidade de inclusão do algoritmo de redução de métrica. Paralelamente, heurísticas foram criadas para melhorar o desempe-

nho do algoritmo geral. Somente após a inclusão da abordagem de RRTs bidirecionais, associada a uma heurística exploradora, é que foi possível encontrar uma solução para o problema de planejamento de rota. Os resultados basearam-se em simulações envolvendo o robô UARM-E configurado com um único braço, sendo uma base e dois elos.

Através do ambiente de desenvolvimento proposto foi possível solucionar o planejamento de rota para um manipulador espacial planar de base livre flutuante. O ambiente permite a configuração de um MBLF genérico com n -graus de liberdade em um ambiente também configurável. O planejador proposto destaca-se por considerar a dinâmica, cinemática e as restrições associadas ao MBLF na computação de uma rota factível. O método proposto pode ser facilmente incorporado a protótipos de manipuladores espaciais com o objetivo de auxiliar na computação de uma trajetória válida. Portanto, pode-se afirmar que os resultados obtidos neste trabalho foram satisfatórios e os objetivos alcançados com sucesso.

7.2 Trabalhos Futuros

Os seguintes trabalhos são sugeridos:

1. **Troca do ambiente de programação.** Optar por uma programação orientada a objetos, visando uma melhor modulação e desempenho do algoritmo. As linguagens C++ e Python figuram entre as mais utilizadas para esse fim.
2. **Extensão do planejador para dois braços.** Estender o ambiente de planejamento de modo a garantir suporte a um manipulador configurado com dois braços.
3. **Testes de planejamento.** Realizar testes experimentais no robô UARM-E. Para esse fim, sugere-se utilizar um planejador de trajetórias para validar a rota planejada pelo método proposto neste trabalho.

Capítulo 8

Anexos

8.1 Anexo 1

Algoritmo 3 NearestNeighbor modificado

NearestNeighbor($x_{rand}, Tree$)

```

1:  $d_{min} \leftarrow \infty$ ;
2:  $d'_{min} \leftarrow \infty$ ;
3: for  $i = 1$  to  $N$  do
4:   if  $\exists u(x_i) \neq EXPANDIDO$  then
5:      $d \leftarrow \rho(x_i, x_{rand})$ ;
6:     if  $d < d'_{min}$  then
7:        $d'_{min} \leftarrow d$ ;
8:        $x_{best} \leftarrow x_i$ ;
9:        $k \leftarrow rand(0, 1)$ 
10:      if  $k > FRV(x_i)$  then
11:         $d_{min} \leftarrow d$ ;
12:         $x_{best} \leftarrow x_i$ ;
13:      end if
14:    end if
15:  end if
16: end for
17: if  $d_{min} \neq \infty$  then
18:  return  $x_{best}$ ;
19: else
20:  return  $x'_{best}$ ;
21: end if

```

Algoritmo 4 SelectInput modificado

SelectInput($x_{near}, x_{rand}, x_{new}, Tree, sucesso$)

```
1:  $d_{min} \leftarrow \infty$ ;
2:  $sucesso \leftarrow falso$ ;
3: for  $i = 1$  to  $N$  do
4:   if  $u(x_i) \neq EXPANDIDO$  then
5:      $x' \leftarrow Integra(x_{near}, u(x_i))$ ;
6:     if  $Dx'$  then
7:        $d \leftarrow \rho(x', x_{rand})$ ;
8:       if  $d < d_{min}$  then
9:          $d_{min} \leftarrow d$ ;
10:       $sucesso \leftarrow verdadeiro$ ;
11:       $u_{best} \leftarrow u(x_i)$ ;
12:    end if
13:  else
14:    marcar  $u(x_i)$  como  $EXPANDIDO$ ;
15:    AtualizaInfo( $x_{near}, Tree$ );
16:  end if
17: end if
18: end for
19: marcar  $u_{best}$  como  $EXPANDIDO$ ;
20: return  $u_{best}$ ;
```

AtualizaInfo($x_{near}, Tree$)

```
1:  $p \leftarrow 1/M$ ;
2:  $FRV(x_{near}) \leftarrow FRV(x_{near}) + p$ ;
3:  $p \leftarrow 1/M^2$ ;
4:  $x_1 \leftarrow x_{near}$ ;
5: while  $x_1 \neq raiz$  do
6:    $x_2 \leftarrow N\acute{o}Pai(x_1)$ ;
7:    $FRV(x_2) \leftarrow FRV(x_2) + p$ ;
8:    $p \leftarrow p/M$ ;
9:    $x_1 \leftarrow x_2$ ;
10: end while
```

Referências

- Barraquand, J. e J.-C. Latombe (1991). Robot motion planning: A distributed representation approach. *Int. J. Rob. Res.* 10(6), 628–649.
- Bohlin, R. e E. Kavraki (2000). Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, Volume 1, pp. 521–528 vol.1.
- Cheng, P. e S. LaValle (2001). Reducing metric sensitivity in randomized trajectory design. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, Volume 1, pp. 43–48 vol.1.
- Choset, H., K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, e S. Thrun (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Boston: MIT Press.
- Craig, J. (2005). *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson Education, Incorporated.
- Dahl, K., D. Thompson, D. McLaren, Y. Chao, e S. Chien (2011). Current-sensitive path planning for an underactuated free-floating ocean sensorweb. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3140–3146.
- Dubowsky, S. e E. Papadopoulos (1993). The kinematics, dynamics, and control of free-flying and free-floating space robotic systems. *Robotics and Automation, IEEE Transactions on* 9(5), 531–543.

- Dubowsky, S. e M. A. Torres (1990). Minimizing attitude control fuel in space manipulator systems.
- Ellery, A. (2000). *An Introduction to Space Robotics*. Springer Praxis Books / Astronomy and Planetary Sciences. Springer.
- Feihu, J. (2013). Path planning of free-flying space robot using memetic algorithm. In *Strategic Technology (IFOST), 2013 8th International Forum on*, Volume 2, pp. 19–22.
- Gottschalk, S. (1996). Separating axis theorem. Technical Report TR96-024, UNC Chapel Hill, Chapel Hill, NC.
- Hsu, D., J.-C. Latombe, e R. Motwani (1997). Path planning in expansive configuration spaces. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, Volume 3, pp. 2719–2726 vol.3.
- Kanade, T. e Y. Xu (1993). *Space robotics : dynamics and control / edited by Yangsheng Xu, Takeo Kanade*. Boston : Kluwer Academic Publishers.
- Kelly, R., V. Davila, e J. Perez (2006). *Control of Robot Manipulators in Joint Space*. Advanced Textbooks in Control and Signal Processing. Springer.
- Kim, Y.-J., S.-Y. Park, J.-J. Kim, J.-H. Wang, J.-Y. Lee, e J.-J. Lee (2014). A rrt-based collision-free and occlusion-free path planning method for a 7dof manipulator. In *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*, pp. 1017–1021.
- Kuffner Jr., J. J. e S. M. LaValle (2000). RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Volume 2, pp. 995 –1001 vol.2.
- LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.
- Lavalle, S. M., J. J. Kuffner, e Jr. (2000). Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pp. 293–308.

-
- LaValle, S. M. e J. J. Kuffner Jr. (1999). Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 1, pp. 473–479 vol.1.
- Li, C., B. Liang, e W. Xu (2006). Autonomous trajectory planning of free-floating robot for capturing space target. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 1008–1013.
- Liang, B., Y. Xu, e M. Bergerman (1996). Mapping a space manipulator to a dynamically equivalent manipulator. Technical Report CMU-RI-TR-96-33, Robotics Institute, Pittsburgh, PA.
- Longman, R. (1993). The kinetics and workspace of a satellite-mounted robot. In Y. Xu e T. Kanade (Eds.), *Space Robotics: Dynamics and Control*, Volume 188 of *The Kluwer International Series in Engineering and Computer Science*, pp. 27–44. Springer US.
- Malone, N., K. Manavi, J. Wood, e L. Tapia (2013). Construction and use of roadmaps that incorporate workspace modeling errors. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1264–1271.
- Nanos, K. e E. Papadopoulos (2012). On cartesian motions with singularities avoidance for free-floating space robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 5398–5403.
- Papadopoulos, E. e S. Dubowsky (1991). On the nature of control algorithms for free-floating space manipulators. *Robotics and Automation, IEEE Transactions on* 7(6), 750–758.
- Papadopoulos, E. e S. Dubowsky (1993). Dynamic singularities in free-floating space manipulators. *ASME J. Dynamic Systems, Measurement and Control*, 115:1 115, 44–52.
- Papadopoulos, E. G. (1993). Nonholonomic behavior in free-floating space manipulators and its utilization. In *Nonholonomic Motion Planning*, pp. 423–445. Kluwer Academic Publishers.

- Pazelli, T. (2011). *Montagem e controle H -Infinito não linear de manipuladores espaciais com base flutuante*. Tese de Doutorado, EESC-USP.
- Sakawa, Y. (1999). Trajectory planning of a free-flying robot by using the optimal control. *Optimal Control Applications and Methods* 20(5), 235–248.
- Siciliano, B., L. Sciavicco, L. Villani, e G. Oriolo (2009). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer.
- Spong, M., S. Hutchinson, e M. Vidyasagar (2005). *Robot Modeling and Control*. Wiley.
- Svenstrup, M., T. Bak, e H. J. Andersen (2011). Minimising computational complexity of the RRT algorithm a practical approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5602 –5607.
- Tortopidis, I. e E. Papadopoulos (2006). Point-to-point planning: methodologies for underactuated space robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 3861–3866.
- Urmson, C. e R. Simmons (2003). Approaches for heuristically biasing RRT growth. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Volume 2, pp. 1178 – 1183 vol.2.
- Vafa, Z. (1990). Space manipulator motions with no satellite attitude disturbances. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 1770–1775 vol.3.
- Vafa, Z. e S. Dubowsky (1987). On the dynamics of manipulators in space using the virtual manipulator approach. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, Volume 4, pp. 579–585.
- Walker, M. e L.-B. Wee (1991). An adaptive control strategy for space based robot manipulators. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 1673–1680 vol.2.

- Wu, J., S. Shi, B. Wang, L. Jiang, e H. Liu (2009). Path planning for minimizing base reaction of space robot and its ground experimental study. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pp. 4627–4632.
- Yoshida, K. (2009). Achievements in space robotics. *Robotics Automation Magazine, IEEE* 16(4), 20–28.
- Yoshida, K., K. Hashizume, e S. Abiko (2001). Zero reaction maneuver: flight validation with ets-vii space robot and extension to kinematically redundant arm. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, Volume 1, pp. 441–446 vol.1.