Daniel Alves Barbosa de Oliveira Vaz

Planejamento de movimento cinemático-dinâmico para robôs móveis com rodas deslizantes

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo para obtenção do Título de Mestre em Ciências pelo Programa de Pós-Graduação de Engenharia Elétrica.

Área de Concentração:
Sistemas Dinâmicos

Orientador:
Prof. Dr. Valdir Grassi Junior

São Carlos
2011

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

Ficha catalográfica preparada pela Seção de
Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Vaz, Daniel Alves Barbosa de Oliveira

Planejamento de movimento cinemático-dinâmico para robôs
móveis com rodas deslizantes/ Daniel Alves Barbosa de Oliveira
Vaz ; orientador Valdir Grassi Junior. – São Carlos, 2011.
66 p.

Dissertação (Mestrado - Programa de Pós-Graduação em En-
genharia Elétrica e Área de Concentração em Sistemas Dinâmi-
cos) – Escola de Engenharia de São Carlos – Universidade de São
Paulo, 2011.

1. Robôs móveis. 2. Planejamento de trajetória. 3. Acom-
panhamento de trajetória. I. Planejamento de movimento
cinemático-dinâmico para robôs móveis com rodas deslizantes.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **DANIEL ALVES BARBOSA DE OLIVEIRA VAZ**

Título da dissertação: "Planejamento de movimento cinemático-dinâmico para robôs móveis com rodas deslizantes."

Data da defesa: 30/11/2011

Comissão Julgadora:

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

Prof. Associado **Marco Henrique Terra**
(Escola de Engenharia de São Carlos/EESC)

Prof. Dr. **Denis Fernando Wolf**
(Instituto de Ciências Matemáticas e de Computação/ICMC)

Resultado:

Aprovado

Aprovado

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Titular **Denis Vinicius Coury**

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

*“Aos meus pais Dlcio Vaz de Oliveira Machado e Zlia Barbosa de Oliveira com amor e
gratido.”*

Agradecimentos

A Deus por ter me dado força, paciência e inteligência, virtudes indispensáveis para a conclusão deste trabalho.

A todos meus familiares que sempre me apoiaram ao longo da realização deste trabalho.

Ao Prof Dr. Valdir Grassi Junior pela oportunidade, orientação, paciência, sabedoria e pelo tempo dedicado a este trabalho.

Aos meus amigos da Igreja Presbiteriana de São Carlos, obrigado pelo carinho e hospitalidade.

Aos meus amigos de república, Igor e Sérgio, pela amizade e o convívio ao longo destes anos.

Aos meus amigos que me acolheram em sua residência: Miguel e Giovana.

Aos meus amigos da pós-graduação: Tatiana, Amanda, Aline, Moussa, Roberto, Raphael, Fabíolo, João, Samuel, André e Gildson pela amizade, companheirismo e colaborações durante a realização das disciplinas e deste trabalho.

Ao Prof Dr. Luís Fernando Costa Alberto pela dedicação, carisma, didática e sabedoria; virtudes transmitidas ao longo do curso de Sistemas Não-Lineares.

Aos demais professores por compartilharem a sabedoria.

Ao corpo de funcionários do Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela concessão da bolsa de mestrado.

*“Algumas pessoas acham que foco significa dizer sim para a coisa em que você irá se focar.
Mas não é nada disso. Significa dizer não às centenas de outras boas idéias que existem.”*
(Steve Jobs)

Resumo

O planejamento de movimento é um dos problemas fundamentais em navegação autônoma para robôs móveis. Uma vez planejado o caminho, o robô executa o acompanhamento da trajetória, frequentemente, com o auxílio de um controlador em malha fechada. Este controlador tem o objetivo de minimizar os erros de acompanhamento, a fim de que a trajetória executada se aproxime da trajetória planejada. Entretanto a maioria dos planejadores de movimento não levam em consideração o modelo dinâmico do robô, dificultando assim o trabalho do controlador que executa o acompanhamento da trajetória. Incluindo as restrições cinemáticas e dinâmicas do modelo do robô, o custo computacional durante a fase de planejamento de trajetória será mais alto. Isto ocorre pois são necessárias mais variáveis para representar o espaço de estados do robô. No entanto ao levar em consideração tais restrições durante a fase de planejamento, as trajetórias geradas serão factíveis de serem acompanhadas. Os planejadores probabilísticos de movimento podem ser usados para minimizar o impacto do alto custo computacional, devido ao aumento de variáveis que representam o espaço de estados. Tais planejadores também são chamados de planejadores de movimento baseados em amostragem. A busca por um caminho livre de colisões entre dois estados é feito de maneira aleatória. Caso exista uma solução, a probabilidade do algoritmo encontrá-la tende para 1 quanto o tempo de busca tende a infinito, isto é, quanto mais tempo o algoritmo possui para realizar a busca será mais provável que ele encontre a solução. Neste trabalho é proposto um planejador de movimentos baseado em amostragem que leva em consideração os aspectos cinemáticos e dinâmicos do robô. Além disto esta abordagem de planejamento desenvolvida permite conhecer e levar em consideração os efeitos do controlador que faz o acompanhamento da trajetória, ainda na fase de planejamento de movimento. As trajetórias planejadas foram executadas no robô Pioneer 3AT. Foram levantados os dados relacionados ao desempenho do algoritmo em termos de custo computacional. E na sequência são apresentados os resultados experimentais tanto na parte de planejamento de trajetórias quanto na fase de acompanhamento.

Abstract

Motion planning is one of the fundamental problems in autonomous navigation for mobile robots. Once the path is planned, the robot performs the trajectory tracking, often with the aid of a closed loop controller. This controller is designed to minimize tracking errors, in order that tracked trajectory get closer to planned path. However, the most motion planners do not take into account the dynamic model of the robot, thus hindering the work of closed loop controller. When including the kinematic constraints and dynamic model of the robot, the computational cost during the planning phase trajectory will be increased. This is because more variables are needed to represent the state space of the robot. But when taking into account these constraints during the planning phase, the trajectories generated are feasible to be followed. The probabilistic motion planners can be used to minimize the impact of high computational cost due to the increase of variables that represent the state space. These planners are also called sampling based motion planners. The search for a collision-free path between two states is done randomly. If a solution exists, the probability of the algorithm to find it tends to one while the search time tends to infinity, that is, the longer time the algorithm has to perform the search will be more likely to find the solution. This paper proposes a sampling based motion planner that takes into account the kinematic and dynamic aspects of the robot. Furthermore this approach allows one to know and take into account the effects of the controller that perform the trajectory tracking, still in the motion planning phase. The planned trajectories were performed on the robot Pioneer 3AT. Data related to the computational cost of the algorithm were analyzed. Following the experimental results are presented both in the planning of trajectories and in the tracking phase.

Lista de Figuras

1.1	Robô enviado a marte e braços robóticos industriais	17
1.2	Robô Pioneer 3AT – P3-AT	20
2.1	(a) Espaço de trabalho, (b) espaço de configurações, (c) robô representado como um ponto.	24
2.2	Expansão de uma RRT	29
3.1	Geometria do robô móvel.	32
3.2	Forças de tração e resistivas do robô móvel.	34
4.1	Abordagem de Planejamento Cinemático-Dinâmico de Movimento baseado em Acompanhamento – Exemplo de expansão.	41
4.2	Diagrama de blocos da fase de expansão	41
4.3	Diagrama de blocos da fase de simulação de acompanhamento	41
5.1	Diagrama simplificado de classes – Software rrt-motion-planner	45
5.2	Diagrama da classe RRT	46
5.3	Diagrama de classes – RobotGeometryModel, EnvModel e World	49
5.4	Diagrama de classes – RobotModel	50
5.5	Diagrama de classes - Tracking	53
5.6	Diagrama de classes - ProxyRobot e ProxyPosition	54
6.1	Passagem estreita - Dinâmico	57
6.2	Passagem estreita - Baseado em acompanhamento	58
6.3	Estacionamento paralelo - Dinâmico	59
6.4	Estacionamento paralelo - Baseado em acompanhamento	59
6.5	Passagem estreita - Resultado experimental	60

6.6	Estacionamento paralelo - Resultado experimental	60
6.7	Passagem estreita - Resultado experimental	61
6.8	Passagem estreita - Resultado experimental	61

Lista de Tabelas

6.1	Tabela de desempenho - Passagem estreita	56
6.2	Tabela de desempenho - Estacionamento paralelo	58
6.3	Parâmetros do Pioneer 3-AT.	62

Sumário

1	Introdução	17
2	Planejamento de movimento	23
2.1	Espaço de configurações	23
2.2	Planejadores de movimento baseado em amostragem	25
2.3	Planejamento Cinemático-Dinâmico	26
2.4	Rapidly-exploring Random Tree	26
2.4.1	Restrições diferenciais no espaço de estados	26
2.4.2	Algoritmo RRT	27
2.4.3	Planejadores baseados em RRT	28
3	Modelagem do robô móvel	31
3.1	Modelo cinemático	31
3.2	Modelo dinâmico	34
3.3	Seguimento de trajetória	36
4	Planejamento de movimento cinemático-dinâmico baseado em acompanhamento	39
4.1	Metodologia	39
5	Implementação	43
5.1	Ferramentas e bibliotecas	43
5.1.1	Player/Stage	43
5.1.2	PQP – Proximity Query Package	44

5.1.3	LEMON	44
5.2	Arquitetura do software	44
5.2.1	Classe RRT	45
5.2.2	Classe StateSampler	47
5.2.3	Classes DistanceMeter e BiasedMeterP3AT	47
5.2.4	Classes TrackingControl, ProportionalControl e Pioneer3ATState . . .	48
5.2.5	Classes RobotGeometryModel, EnvModel e World	48
5.2.6	Classe RobotModel	49
5.3	Geração de torques	51
5.4	Acompanhamento de trajetória	53
5.4.1	Implementação do acompanhamento de trajetória para o Player/Stage .	54
6	Resultados	55
6.1	Planejamento de movimento	55
6.2	Acompanhamento das trajetórias planejadas	59
6.2.1	Planejador de movimento cinemático-dinâmico	59
6.2.2	Planejador de movimento cinemático-dinâmico baseado em acompa- nhamento	61
7	Conclusão	63
	Referências Bibliográficas	65

1 Introdução

Os robôs ou sistemas robóticos foram concebidos com o propósito de auxiliar e até mesmo substituir a ação humana em diversos tipos de tarefas. É habitual observar a atuação de robôs em tarefas repetitivas, ou mesmo perigosas que trazem riscos a vida. Como exemplo de tarefas que hoje são realizadas por robôs pode-se citar: linha de montagem e pintura de veículos, exploração submarina, exploração espacial, exploração e manutenção em ambientes inóspitos, combate a incêndios, desarmamento de bombas, dentre outras.

Figura 1.1: Robô enviado a marte e braços robóticos industriais

O planejamento de movimento faz-se necessário para a realização de tarefas de locomoção do robô e tem como objetivo determinar quais ações o robô deve executar de forma que alcance seu objetivo final, evitando colisões ao longo da trajetória. No processo de planejamento de movimento utiliza-se informações tais como o mapa do ambiente e o modelo do robô. O modo como o robô se locomove e suas restrições de movimento, são fatores importantes na etapa de planejamento de movimento.

Quanto a forma de locomoção, os robôs podem ser classificados em omnidirecionais sendo capazes de se locomover em qualquer direção, e não holonômicos que possuem determinadas restrições ao se movimentar. Os veículos de rodas esterçantes, incapazes de se movimentar lateralmente, são um exemplo de modelo não holonômico. Devido a essa restrição são neces-

sárias algumas manobras até que se atinja um determinado estado final, como por exemplo na manobra de estacionamento paralelo ou baliza.

É importante ressaltar que o planejamento de movimento não se restringe apenas a área de robótica. Os métodos de planejamento de movimento podem ser utilizados na área de animação de personagens digitais, planejamento cirúrgico, desenho assistido por computador – CAD, desenvolvimento de novos medicamentos, etc.

A maioria dos planejadores básicos de movimento ignoram completamente a dinâmica do modelo do robô, levando em consideração somente a cinemática. Isto pode resultar em caminhos incapazes de serem executados pelo robô, devido a limitação das forças e torques que acionam o robô real. A solução clássica em pesquisas robóticas tem sido desacoplar o problema geral em duas partes: resolver o planejamento básico de movimento na busca de uma trajetória e logo em seguida projetar um controlador que satisfaça a dinâmica do modelo e siga o caminho planejado (LAVALLE; KUFFNER, 2001). Donald et al. (1993) segue outra vertente e propõe o termo planejamento cinemático-dinâmico de movimento, cujo objetivo é incluir na fase de planejamento de movimento, as restrições impostas pelo modelo dinâmico do robô, possibilitando a geração de caminhos factíveis e facilitando o seguimento da trajetória planejada.

O *Rapidly-exploring Random Tree*, RRT, é um método de planejamento de movimento baseado em amostragem e foi proposto com a intenção de resolver problemas de planejamento cinemático-dinâmico (LAVALLE; KUFFNER, 2001). Quando leva-se em conta a dinâmica do robô o número de graus de liberdade aumenta, isto é, o número mínimo de parâmetros que representam completamente o estado do robô cresce. Os métodos de planejamento de movimento baseados em amostragem são apropriados quando o número de graus de liberdade é grande, uma vez que estes métodos amostram estados aleatórios e aplicam as ações de controle no modelo do sistema robótico para propagar o movimento do robô realizando testes de colisão ao longo desta propagação. Este é o funcionamento básico do RRT que realiza duas tarefas simultaneamente: expande uma árvore aleatória que possui ótima capacidade de exploração do ambiente e guia o crescimento dessa árvore em direção ao estado final desejado.

Há várias modificações no método básico de RRT que podem ser aplicadas para alcançar um melhor desempenho na busca de soluções. Abaixo são destacadas algumas extensões feitas ao método básico que permitem ser aplicadas em problemas de planejamento de movimento do tipo cinemático-dinâmico, tais como: melhorar a qualidade da solução de forma incremental, trabalhar com planejamento de movimento em tempo real e trabalhar em ambientes com passagens estreitas dentre outras.

Embora o algoritmo básico de RRT mostre-se efetivo na geração de trajetória factíveis,

ele não fornece controle sobre a qualidade das soluções produzidas. Com o fim de atender os requisitos de tempo real e aumentar a qualidade da trajetórias geradas, uma nova extensão para o RRT foi introduzido em (FERGUSON; STENTZ, 2006). Esta nova extensão é capaz de produzir rapidamente uma solução inicial e aprimorá-la enquanto houver tempo ocioso de processamento. O aperfeiçoamento é incremental, isto é, a qualidade das soluções subsequentes é melhorada com base nas informações das trajetórias geradas anteriormente.

Em problemas de planejamento de movimento que envolve corredores estreitos, o algoritmo básico RRT apresenta dificuldades em encontrar uma trajetória. A abordagem chamada *Obstacle-based RRT* (RODRIGUEZ et al., 2006) usa a informação sobre a forma dos obstáculos para guiar o crescimento da árvore. O planejador de movimentos resultante desta mudança é mais rápido e consegue resolver um problema com corredores estreitos em um número menor de iterações.

No campo experimental há uma grande preocupação com a capacidade dos métodos de planejamento de movimento de atender os requisitos de tempo real. O *Execution extended RRT* (ERRT) (BRUCE; VELOSO, 2002) intercala planejamento de movimento e execução, utilizando uma fase de simulação e então aplicando os comandos em um robô real. Este método introduz duas novas extensões encontradas em trabalhos anteriores sobre RRT que são: *way-point cache* e busca penalizada por custo adaptativo. Juntos eles aumentam a qualidade dos caminhos obtidos e a eficiência no replanejamento de trajetórias. O método ERRT foi aplicado com sucesso na RoboCup (BRUCE; VELOSO, 2002).

Percebeu-se que poucos trabalhos de planejamento de movimento foram realizados para a classe de robôs móveis com rodas deslizantes. Essa classe de robôs utiliza o deslizamento das rodas, que causa o aparecimento de uma velocidade lateral, para mudar a orientação do robô. Além disso grande parte dos trabalhos de planejamento cinemático-dinâmico de movimento focam em resultados de simulação, não apresentando resultados experimentais de seguimento das trajetórias planejadas.

Duas abordagens são apresentadas neste trabalho. Uma delas é o algoritmo básico de RRT para planejamento de movimento cinemático-dinâmico de um robô móvel de rodas deslizantes utilizando um modelo que possui como entradas de controle os torques aplicados nos motores de roda do robô móvel (VAZ; INOUE; GRASSI, 2010). A outra é uma nova abordagem de planejamento cinemático-dinâmico, proposta neste trabalho, que aqui denominamos de Planejamento cinemático-dinâmico de movimento baseado em acompanhamento. Esta proposta pode ser considerada uma extensão de planejadores de movimentos baseados em RRT. O diferencial desta extensão é aplicar o mesmo controlador usado na execução da trajetória ainda na fase de

planejamento de movimento.

Portanto, as principais contribuições deste trabalho foram: aplicar o planejamento cinemático-dinâmico baseado em RRT para a classe de robôs móveis com rodas deslizantes; e propor uma nova extensão para o método básico de planejamento cinemático-dinâmico baseado em RRT, aqui denominada de planejamento de movimento baseado em acompanhamento.

Neste trabalho as trajetórias obtidas pelo planejador de movimentos são simuladas utilizando-se os softwares Player/Stage (GERKEY; VAUGHAN; HOWARD, 2003) e então são feitos experimentos reais com o robô Pioneer-3AT visto na Figura 1.2, que é um robô móvel não holonômico com rodas deslizantes. Os resultados deste trabalho concentram-se em comparar os resultados de um planejador de movimentos cinemático-dinâmico básico (VAZ; INOUE; GRASSI, 2010), com um planejador de movimentos cinemático-dinâmico baseado em acompanhamento apresentado no Capítulo 4. O modelo cinemático-dinâmico utilizado é apresentado por Caracciolo, Luca e Iannitti (1999), que mostra o seguimento de trajetória para um robô com sistema de direção semelhante.

Figura 1.2: Robô Pioneer 3AT – P3-AT

O texto é dividido da seguinte forma: o Capítulo 2 apresenta conceitos fundamentais sobre planejamento de movimento, discute as abordagens de planejamento baseado em amostragem e planejamento cinemático-dinâmico, além de apresentar de maneira detalhada o método de planejamento de movimento RRT. Os modelos cinemático e dinâmico do robô Pioneer 3-AT, utilizados pelo planejador são apresentados no Capítulo 3, que também propõe um controlador proporcional usado no planejamento de movimento e no acompanhamento das trajetórias planejadas. A abordagem proposta nessa dissertação que leva em consideração o mesmo controlador

em fases distintas, planejamento e acompanhamento, é apresentada no Capítulo 4. O Capítulo 5 mostra a implementação do planejamento baseado em acompanhamento usando como base o algoritmo RRT. Os resultados experimentais são mostrados no Capítulo 6, as conclusões são apresentadas no Capítulo 7.

2 Planejamento de movimento

O objetivo central do planejamento de movimento é levar o robô de um estado atual a um estado final desejado, evitando colisões com os obstáculos presentes no ambiente durante o percurso. Esta é uma tarefa desafiadora e considerada um dos problemas fundamentais da robótica autônoma (CHOSSET et al., 2005).

Esse capítulo apresenta a definição dos conceitos básicos utilizados no planejamento de movimento, em seguida é apresentado a abordagem de planejamento baseado em amostragem, planejamento cinemático-dinâmico, e é discutido com maior profundidade o algoritmo do *Rapidly-exploring Random Tree*, planejador de movimentos utilizado ao longo desse trabalho.

2.1 Espaço de configurações

O espaço de configurações é um dos conceitos mais importantes no planejamento de movimento. Antes de definir o conceito de configuração e espaço de configurações, é interessante comentar sobre o espaço de trabalho do robô. O espaço de trabalho, $\mathcal{W}$, é o espaço físico por onde o robô se desloca. No caso de robôs móveis que se locomovem no plano, o espaço de trabalho é o $\mathbb{R}^2$.

Seja $\mathcal{F}_{\mathcal{W}}$ o sistema de coordenadas inercial e $\mathcal{F}_{\mathcal{R}}$ o sistema de coordenadas fixo no corpo do robô. A configuração, q , de um robô é a representação completa da posição de todos os pontos do robô, relativa ao sistema de coordenadas fixo do ambiente, $\mathcal{F}_{\mathcal{W}}$. O número mínimo de parâmetros que especificam completamente a configuração de um robô é definido como sendo o número de graus de liberdade do robô e também representa a dimensão do espaço de configurações (CHOSSET et al., 2005).

O espaço de configurações $\mathcal{C}$ é o conjunto de todas as possíveis configurações que podem ser assumidas pelo robô. Dentro do contexto de espaço de configurações, pode-se definir dois conjuntos que auxiliam na formulação de problemas de planejamento de movimento: o espaço

de configurações ocupado por obstáculos, $\mathcal{C}_{obs}$, e o espaço de configurações livre, $\mathcal{C}_{free}$.

A Figura 2.1, apresentada em Choset et al. (2005), mostra a diferença entre espaço de trabalho e espaço de configurações para um dos casos mais simples de se obter $\mathcal{C}_{obs}$, caso em que se considera um robô móvel de geometria circular se movendo no plano. O estado do robô é representado pela posição do centro geométrico. Deslizando o robô pelo contorno do obstáculo, a posição do centro geométrico define uma linha, Figura 2.1 (b), que representa a fronteira de colisão entre robô e obstáculo. Após essa operação o robô pode ser representado por um ponto. Realizando essa mesma operação para todos os obstáculos do ambiente, tem-se a representação completa de $\mathcal{C}_{obs}$ e consequentemente, $\mathcal{C}_{free}$.

Figura 2.1: (a) Espaço de trabalho, (b) espaço de configurações, (c) robô representado como um ponto.

O objetivo do planejamento de movimento é encontrar um caminho em $\mathcal{C}_{free}$ que leve o robô de uma configuração inicial q_{init} a uma configuração final q_{goal} . Alguns métodos de planejamento de movimento são ditos completos. Isto significa que se existe uma solução, o método consegue encontrá-la em tempo finito ou então simplesmente retorna uma mensagem relatando que falhou na busca do caminho (CHOSSET et al., 2005). Os métodos completos fazem a construção explícita do espaço de configurações. Como dito anteriormente, um dos casos mais simples de construção explícita do espaço de configurações é para um robô de geometria circular movendo-se no plano. O custo computacional utilizado na construção explícita do espaço de configurações de robôs com geometria retangular e cuja orientação é levada em consideração é caro. Para problemas de planejamento de movimento cinemático-dinâmico, isto é, quando a dinâmica do robô é levada em consideração, além da postura do robô também são levados em consideração durante a etapa de planejamento as velocidades linear e angular, resultando em um estado (x, y, ψ, v, ω) tornando a solução do problema de planejamento de movimento ainda mais custosa.

2.2 Planejadores de movimento baseado em amostragem

Para problemas de planejamento de movimento cuja dimensão de $\mathcal{C}$ é grande, a busca por soluções completas se torna um problema computacional custoso de ser tratado. Tendo isso em mente, os planejadores de movimento baseados em amostragem evitam a construção explícita de $\mathcal{C}_{obs}$ e ao invés disso realizam uma busca por caminhos livres no espaço de configurações utilizando técnicas de amostragem. Essa busca é possibilitada pelo uso de detectores de colisões que permitem distinguir se a amostra encontra-se em $\mathcal{C}_{free}$ ou não. Esta abordagem de planejamento de movimento tem possibilitado a solução de problemas em que o número de graus de liberdade é bastante grande (CHOSSET et al., 2005; LAVALLE, 2006).

Contudo por trabalharem com uma amostragem do espaço de configurações, estes métodos não podem ser considerados métodos completos de planejamento. Ao invés disso, utiliza-se um conceito mais fraco que este. Considera-se que os métodos de planejamento de movimento baseados em amostragem são probabilisticamente completos. Ou seja, quanto maior o número de amostras a probabilidade do método encontrar uma solução existente tende a 1. Um dado de grande importância mas de difícil estimação é a taxa de convergência do algoritmo baseado em amostragem (LAVALLE, 2006).

Os métodos baseados em amostragem podem ser classificados em métodos de busca única (*single query*) ou busca múltipla (*multiple query*). Dado o único par (q_{init}, q_{goal}) , que representa respectivamente a configuração inicial e a configuração final desejada, o algoritmo de busca única retorna uma solução em tempo finito, ou não havendo ou não encontrando uma solução, reporta falha na busca. Já os métodos de busca múltipla investem um esforço na exploração do espaço de configurações para contruírem uma estrutura de dados que armazena informações sobre o ambiente. Essa estrutura é então usada para tornar eficiente a busca de soluções para diferentes pares (q_{init}, q_{goal}) (CHOSSET et al., 2005).

A maioria dos planejadores básicos de movimento ignoram completamente a dinâmica do robô, levando em consideração somente a cinemática. Isto pode resultar em caminhos incapazes de serem executados pelo robô real, devido a limitação das forças e torques que acionam o robô (LAVALLE; KUFFNER, 2001). A abordagem de planejamento cinemático-dinâmico de movimento propõe uma solução alternativa que leva em consideração tais restrições.

2.3 Planejamento Cinemático-Dinâmico

O planejamento cinemático-dinâmico de movimento de robôs móveis tem como objetivo planejar caminhos factíveis, isto é, que obedecem as restrições impostas pelo modelo dinâmico do robô, além de evitar colisões com os obstáculos presentes no ambiente. Uma vez que o modelo dinâmico do robô é utilizado na fase de planejamento, as trajetórias planejada e executada se assemelham, quanto maior for a fidelidade entre modelo e o robô real.

Quando a dinâmica é levada em consideração, a dimensão do espaço de estados que representa o robô cresce, uma vez que é necessário representar não apenas a posição e orientação, mas também as velocidades ao longo do caminho planejado. Como discutido na Seção anterior, os planejadores de movimento baseados em amostragem são eficientes na resolução de problemas em que a dimensão do espaço de estados é grande.

Em LaValle e Kuffner (2001) é proposto um método de planejamento de movimento baseado em amostragem, classificado como um algoritmo de busca única para planejamento cinemático-dinâmico de movimento, que utiliza como base a construção de árvores aleatórias que exploram o espaço de estados rapidamente. Esse algoritmo é denominado *Rapidly-exploring Random Tree*.

2.4 Rapidly-exploring Random Tree

O método de planejamento de movimento baseado em RRT, foi desenvolvido com o intuito de solucionar problemas de planejamento cinemático-dinâmico em um espaço de estados que tenha restrições diferenciais de primeira ordem. O conceito de espaço de configurações utilizado nos algoritmos clássicos de planejamento é substituído pelo conceito de espaço de estados para o planejamento cinemático-dinâmico. A seguinte analogia é apresentada em LaValle e Kuffner (2001): seja, $q \in \mathcal{C}$, uma configuração do robô no espaço de configurações e representando o espaço de estados por $\mathcal{X}$, o estado do robô, $x \in \mathcal{X}$, é definido como $x = (q, \dot{q})$.

2.4.1 Restrições diferenciais no espaço de estados

Utilizando mecânica Lagrangiana, a dinâmica do modelo pode ser representada por um conjunto de equações diferenciais de segunda ordem, da forma $h(\ddot{q}, \dot{q}, q) = 0$. O conjunto de equações diferenciais de segunda ordem, pode ser representado no espaço de estados por um conjunto de m equações implícitas, de primeira ordem, da forma $g(\dot{x}, x) = 0$, $m < 2n$, sendo n

a dimensão do espaço de configurações, $\mathcal{C}$. Sobre determinadas condições, o teorema da função implícita pode ser aplicado para permitir que as restrições diferenciais de primeira ordem sejam expressas como

$$\dot{x} = f(x, u) \quad (2.1)$$

tal que $u \in U$, e U representa o conjunto das entradas, ou controles, permitido (LAVALLE; KUFFNER, 2001).

2.4.2 Algoritmo RRT

O algoritmo RRT, descrito pelos Algoritmos 1 e 2, utiliza o conjunto de equações diferenciais de primeira ordem, representada pela equação 2.1, para estimar o estado futuro do robô através de integração numérica. Os dados de entrada do algoritmo RRT são: estado inicial do robô, x_{init} , estado final desejado x_{goal} ou região final desejada r_{goal} , número máximo de tentativas de expansão, modelo cinemático-dinâmico do robô e o mapa do ambiente. De posse dessas informações é criado um grafo que irá representar a árvore de expansões.

Algoritmo 1 Construção - RRT

Entrada:

x_{init} : estado inicial.
 n : número de tentativas de expansão da árvore.
 r_{goal} : região final desejada.

Saída:

Uma árvore $T = (V, E)$ que tem como raiz x_{init} e possui n nós.

```

1:  $V \leftarrow q_0$ 
2:  $E \leftarrow \emptyset$ 
3: for  $i = 1$  até  $n$  do
4:    $x_{rand} \leftarrow$  estado no espaço de estados livre
5:    $x_{chosen} \leftarrow$  expande RRT (Algoritmo 2)
6:   if região_do_objetivo_foi_alcançada( $x_{chosen}$ ) then
7:     break
8:   end if
9: end for
10: return  $T$ 
```

O grafo é inicializado com um único nó que representa o estado inicial do robô. Um estado x_{rand} é amostrado do conjunto de todos os possíveis estados, $\mathcal{X}$. É realizada uma busca no grafo, pelo nó mais próximo de x_{rand} , uma vez encontrado, esse nó é denominado x_{near} . Considerando x_{near} como o estado atual do robô, um conjunto de entradas de controle permitidas são aplicadas no modelo cinemático-dinâmico para calcular as possíveis expansões desse estado,

Algoritmo 2 Expansão - RRT

Entrada:

$T = (V, E)$: uma RRT
 x : um estado em que a árvore T deverá crescer rumo a ele.

Saída:

Um novo estado x_{chosen} em direção a x , ou $\emptyset$, caso falha.

```

1:  $x_{near} \leftarrow$  vizinho mais próximo de  $x$  na árvore  $T$ 
2:  $x_{chosen} \leftarrow \emptyset$ 
3:  $d_{min} \leftarrow \infty$ 
4: for  $u$  in  $U$  do
5:    $x_{temp} \leftarrow$  EstimaNovoEstado( $x_{aux}, u$ )
6:   if RoboEmPosiçãoSegura( $x_{temp}$ ) then
7:      $d \leftarrow \rho(x, x_{temp})$ 
8:     if  $d < d_{min}$  then
9:        $d_{min} \leftarrow d$ 
10:     $x_{chosen} \leftarrow x_{temp}$ 
11:   end if
12: end if
13: end for
14:  $V \leftarrow V \cup \{x_{chosen}\}$ 
15:  $E \leftarrow E \cup \{(x_{near}, x_{chosen})\}$ 
16: return  $x_{chosen}$ 

```

isto é, estima-se os possíveis estados futuros do robô. Durante o processo de expansão é verificado se o robô encontra-se a uma distância segura dos obstáculos. Essa verificação é realizada para todos os estados intermediários entre o estado atual e o estado estimado futuro. Dentre os possíveis estados futuros estimados a partir do nó x_{near} , o estado, x_{chosen} , que se encontrar mais próximo de x_{rand} é escolhido para ser inserido no grafo, que também armazena a aresta (x_{near}, x_{chosen}) .

O processo é repetido até que o número máximo de tentativas de expansões seja excedido ou que o robô alcance o objetivo final, x_{goal} ou r_{goal} . A figura 2.2 mostra esse processo, sendo que as linhas pontilhadas representam as possibilidades de expansão e a linha destacada em vermelho representa a expansão escolhida de acordo com uma métrica ρ .

2.4.3 Planejadores baseados em RRT

O algoritmo básico do RRT, discutido acima, tem como objetivo promover a exploração rápida do espaço de estados enquanto verifica se o objetivo final foi alcançado. Dessa maneira o algoritmo não possui uma heurística que busque encontrar um caminho para x_{goal} , isto é, o objetivo final é alcançado aleatoriamente devido a boa capacidade de exploração do espaço de estados intrínseca ao algoritmo RRT (LAVALLE; KUFFNER, 2000).

Figura 2.2: Expansão de uma RRT

Para que o objetivo seja alcançado mais rapidamente, a variável q_{rand} não é mais amostrada aleatoriamente de uma distribuição uniforme. Ao invés disso uma heurística de planejamento utilizada em conjunto com o algoritmo básico do RRT, guia a amostragem e atribui à variável q_{rand} o estado final desejado com uma certa probabilidade p . Com probabilidade $(1 - p)$ é retornado um estado aleatório, amostrado de uma distribuição uniforme.

Para pequenos valores de p , tal como 0.05, a árvore converge mais rapidamente para o objetivo do que quando é utilizado apenas uma amostragem aleatória uniforme. Se o privilégio a q_{goal} for muito grande o algoritmo RRT acabará funcionando como um planejador baseado em campos potenciais, o que poderia ocasionar o aprisionamento do robô em mínimos locais (LAVALLE; KUFFNER, 2000; CHOSET et al., 2005).

Existem outras técnicas de planejamento de movimento baseados em RRT. LaValle e Kuffner (2000) propõe também o privilégio de uma região ao redor de x_{goal} e apresentam a possibilidade de criar duas árvores que exploram o espaço de estados, sendo que a raiz de uma árvore seria o estado inicial e a outra teria como raiz o estado final desejado. Dessa maneira as duas árvores seriam alternadamente expandidas até que elas se cruzem.

O RRT também pode ser utilizado como planejador local. Em Nourani-vatani et al. (2006) foi desenvolvido um cortador de gramas automatizado que usa um planejador global para explorar completamente o ambiente. Quando o planejador global não consegue encontrar um caminho livre de obstáculos, o método RRT é utilizado para tentar buscar uma trajetória factí-

vel. Em Vallejo, Jones e Amato (2001) o planejamento global verifica se as regiões do espaço de configurações que englobam o estado inicial e final do robô são interconectadas, isto é, se existe um caminho que ligue o estado inicial ao estado final. Caso exista o algoritmo RRT é utilizado para gerar uma trajetória factível, baseado nos *waypoints* calculados pelo planejamento global.

3 Modelagem do robô móvel

Nesse capítulo, são desenvolvidos os modelos cinemático e dinâmico para o robô Pioneer 3-AT. Este robô é bastante conhecido e empregado em pesquisas na área de robótica móvel, podendo ser modelado como um veículo de rodas deslizantes. É dotado de quatro rodas atuadas por dois motores, sendo que um motor é responsável por acionar as rodas do lado direito e outro motor aciona as rodas do lado esquerdo. A mudança de orientação do robô é feita ao se aplicar torques com valores distintos para cada motor. A modelagem aqui apresentada está baseada em (CARACCILO; LUCA; IANNITTI, 1999; KOZLOWSKI; PAZDESKI, 2004). Estes modelos são utilizados pelo método de planejamento de movimento, no Algoritmo 2 Linha 5, para calcular o estado futuro do robô, dado o estado atual e as entradas de controle. O capítulo ainda discute a estratégia de controle utilizada para realizar o seguimento das trajetórias planejadas.

3.1 Modelo cinemático

A geometria do robô é mostrada na Figura 3.1, sendo que (X, Y) é o sistema de coordenadas inercial, (X', Y') o sistema de coordenadas fixo no corpo tendo como origem o centro de gravidade do robô CG . A coordenada do centro de gravidade do robô no sistema de coordenadas inerciais é (x, y) , e ψ é o ângulo entre o eixo X' e o eixo X . As coordenadas do centro instantâneo de rotação (CIR) são (x'_{CIR}, y'_{CIR}) . O parâmetro a corresponde à distância entre o centro de gravidade e o eixo de simetria da roda atuada frontal paralelo ao eixo Y' , b a distância entre centro de gravidade e o eixo de simetria paralelo ao eixo Y' da roda atuada traseira, c é a metade da distância entre as rodas ao longo do eixo Y' e r o raio das rodas.

No sistema de coordenada do corpo a velocidade linear do robô é dada por $\mathbf{v} = [v_{x'} \ v_{y'} \ 0]^T$, sendo $v_{x'}$ e $v_{y'}$ as componentes de $\mathbf{v}$ no eixo X' e Y' , e a velocidade angular é dada por $\boldsymbol{\omega} = [0 \ 0 \ \omega]$, sendo $\omega = \dot{\psi}$. As velocidades lineares e angulares de cada roda atuada são dadas por v_i e ω_i , sendo $i = 1, 2, \dots, 4$ a i -ésima roda atuada.

Figura 3.1: Geometria do robô móvel.

As velocidades absolutas $\dot{x}$, $\dot{y}$ e $\dot{\psi}$ no sistema de coordenadas inercial são dadas por

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{x'} \\ v_{y'} \\ \omega \end{bmatrix} \quad (3.1)$$

$$= R(\psi) \begin{bmatrix} v_{x'} \\ v_{y'} \\ \omega \end{bmatrix}.$$

Diferenciando a equação acima em relação ao tempo tem-se

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\psi} \end{bmatrix} = R(\psi) \begin{bmatrix} \dot{v}_{x'} - \dot{\psi} v_{y'} \\ \dot{v}_{y'} + \dot{\psi} v_{x'} \\ \dot{\omega} \end{bmatrix} \quad (3.2)$$

$$= R(\psi) \begin{bmatrix} a_{x'} \\ a_{y'} \\ \dot{\omega} \end{bmatrix}.$$

Diferentemente da maioria dos robôs móveis, a velocidade lateral do RMRD é geralmente diferente de zero, isso vem da estrutura mecânica do RMRD que faz o deslizamento lateral necessário para que o veículo mude de orientação, e quando a velocidade do robô $v_{y'} = 0$ não há deslocamento lateral. A velocidade angular ω e a velocidade lateral $v_{y'}$ ambas desaparecem quando o robô movimenta-se em linha reta, e o CIR vai para infinito ao longo do eixo Y' . Já

em uma trajetória curva o CIR desloca x'_{CIR} ao longo do eixo X' . Se x'_{CIR} ultrapassar a base das rodas do robô, o veículo desliza drasticamente com perda de estabilidade de movimento. Assim, para completar o modelo cinemático do RMRD, a seguinte restrição não-holonômica foi introduzida em (CARACCIOLO; LUCA; IANNITTI, 1999)

$$v_{y'} = x'_{CIR} \dot{\psi}, \text{ com } x'_{CIR} \in (-b, a). \quad (3.3)$$

No entanto, de (3.1) obtém-se

$$v_{y'} = -\sin \psi \dot{x} + \cos \psi \dot{y}. \quad (3.4)$$

Seja $\mathbf{q} = [x \ y \ \psi]^T$, o estado do robô, substituindo a equação (3.4) em (3.3) e escrevendo na forma Pfaffian, tem-se

$$\begin{bmatrix} -\sin \psi & \cos \psi & -x'_{CIR} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = A(\mathbf{q})\dot{\mathbf{q}} = 0$$

A partir da Figura 3.1, as velocidades $\dot{\mathbf{q}}$ podem ser expressas como:

$$\dot{\mathbf{q}} = S(\mathbf{q})\boldsymbol{\eta}, \quad (3.5)$$

onde:

$$S(\mathbf{q}) = \begin{bmatrix} \cos \psi & -x'_{CIR} \sin \psi \\ \sin \psi & x'_{CIR} \cos \psi \\ 0 & 1 \end{bmatrix},$$

$$\boldsymbol{\eta} = \begin{bmatrix} v_{x'} \\ \omega \end{bmatrix}.$$

$S(\mathbf{q})$ é uma matriz de posto completo, cujas colunas estão no espaço nulo de $A(\mathbf{q})$, ou seja

$$S^T(\mathbf{q})A^T(\mathbf{q}) = 0$$

É interessante observar que como a $\dim(\boldsymbol{\eta}) = 2 < \dim(\mathbf{q}) = 3$, a equação (3.5) descreve a cinemática de um robô subatuado, que é um sistema não-holonômico por causa da restrição (3.3).

3.2 Modelo dinâmico

A Figura 3.2 mostra as forças de tração $F_{x'i}$ que são sujeitas a forças resistivas longitudinais $R_{x'i}$. O Pioneer 3-AT possui dois motores, cada um responsável por acionar as rodas de um dos lados do robô. Então assume-se que a atuação das rodas é igual em cada lado, ou seja, $F_{x'1} = F_{x'3}$ e $F_{x'2} = F_{x'4}$, diminuindo assim o deslizamento longitudinal. As forças laterais $R_{y'i}$ que atuam sobre as rodas são consequências do deslizamento lateral. O momento resistivo M_r em volta do centro de massa, que se opõe ao momento M é induzido pelas forças $R_{y'i}$ e $R_{x'i}$.

Figura 3.2: Forças de tração e resistivas do robô móvel.

Para um veículo, como mostrado na Figura 3.2, de massa m e de inércia I no centro de massa, as equações de movimento no sistema de coordenada do corpo são

$$\begin{aligned} ma_{x'} &= 2F_{x'1} + 2F_{x'2} - R_{x'}, \\ ma_{y'} &= -R_{y'}, \\ I\ddot{\psi} &= 2c(F_{x'1} - F_{x'2}) - M_r. \end{aligned} \quad (3.6)$$

Para representar as forças longitudinais $R_{x'}$, as forças laterais $R_{y'}$, e o momento resistivo M_r , deve-se considerar mg como a carga gravitacional, sendo m a massa do robô e g a aceleração da gravidade, que é dividida sobre as rodas do robô, e introduzir o modelo Coulomb de atrito para o contato das rodas com a superfície. Desse modo, desde que o robô tenha uma linha

mediana longitudinal de simetria, obtém-se

$$R_{z'1} = R_{z'2} = \frac{b}{2(a+b)}mg,$$

$$R_{z'3} = R_{z'4} = \frac{a}{2(a+b)}mg.$$

Em velocidades baixas, a carga lateral transferida devido as forças centrífugas sobre percursos curvados pode ser desconsiderada. No caso de superfícies duras, pode-se assumir que o contato entre as rodas e o solo é retangular e que a carga vertical produz uma pressão uniformemente distribuída. Assim $R_{x'i} = f_r R_{z'i} \text{sgn}(v_{x'i})$ sendo que: f_r é o coeficiente resistivo de rolagem independente da velocidade e $\text{sgn}(\cdot)$ a função sinal. A força resistiva longitudinal total é

$$\begin{aligned} R_{x'} &= \sum_{i=1}^4 R_{x'i} \\ &= \frac{f_r mg}{2} (\text{sgn}(v_{x'1}) + \text{sgn}(v_{x'2})), \end{aligned}$$

Introduzindo o coeficiente de atrito lateral μ , temos que $R_{y'i} = \mu R_{z'i} \text{sgn}(v_{y'i})$. Portanto a força lateral total é dada por

$$\begin{aligned} R_{y'} &= \sum_{i=1}^4 R_{y'i} \\ &= \mu \frac{mg}{a+b} (b \text{sgn}(v_{y'1}) + a \text{sgn}(v_{y'3})), \end{aligned}$$

enquanto o momento resistivo será

$$\begin{aligned} M_r &= \mu \frac{abmg}{a+b} (\text{sgn}(v_{y'1}) - \text{sgn}(v_{y'3})) + \\ &\quad \frac{f_r cmg}{2} (\text{sgn}(v_{x'1}) - \text{sgn}(v_{x'2})). \end{aligned}$$

Segundo Caracciolo, Luca e Iannitti (1999), utilizando-se notação matricial, o modelo dinâmico reescrito no sistema de coordenadas inerciais é

$$M\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) = E(\mathbf{q})\tau, \quad (3.7)$$

sendo

$$\begin{aligned}
 M &= \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}, \\
 C(\mathbf{q}, \dot{\mathbf{q}}) &= \begin{bmatrix} \cos \psi R_{x'} - \sin \psi R_{y'} \\ \sin \psi R_{x'} + \cos \psi R_{y'} \\ M_r \end{bmatrix}, \\
 E(\mathbf{q}) &= \begin{bmatrix} \frac{\cos \psi}{r} & \frac{\cos \psi}{r} \\ \frac{\sin \psi}{r} & \frac{\sin \psi}{r} \\ \frac{c}{r} & -\frac{c}{r} \end{bmatrix}, \\
 \tau &= \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_3 \\ \tau_2 + \tau_4 \end{bmatrix}, \tau_i = r F_{x'i}.
 \end{aligned}$$

Incluindo a restrição não-holonômica no modelo dinâmico (3.7), como introduzido em (CARACCIOLO; LUCA; IANNITTI, 1999), tem-se

$$M\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) = E(\mathbf{q})\tau + A^T(\mathbf{q})\lambda, \quad (3.8)$$

sendo λ o vetor de multiplicadores de Lagrange correspondente a equação (3.3). Para estimar as acelerações a partir do torque aplicado, devemos diferenciar a equação (3.5), substituir em (3.8) e eliminar λ , então teremos o seguinte resultado (CARACCIOLO; LUCA; IANNITTI, 1999):

$$\dot{\boldsymbol{\eta}} = (S^T M S)^{-1} S^T (E\tau - M\dot{S}\boldsymbol{\eta} - C) \quad (3.9)$$

O planejador de movimentos apresentado no Capítulo 2 utiliza as equações (3.5) e (3.9) para incluir respectivamente as restrições cinemáticas e dinâmicas na fase de planejamento.

3.3 Seguimento de trajetória

Uma vez que a trajetória planejada leva em consideração o modelo cinemático-dinâmico do robô, ela será factível. Os parâmetros do modelo matemático, que representa um sistema físico, devem ser ajustados para que os resultados obtidos a partir do modelo sejam válidos para o sistema físico real. Levando em consideração que os parâmetros do modelo representam bem o robô P3-AT e que o planejamento cinemático-dinâmico baseado em RRT gera caminhos possíveis de serem realizados, as trajetórias planejada e executada serão bastante semelhantes. Portanto o erro de deslocamento do robô, isto é, o erro entre posição de referência e posição atual do robô, ao longo da trajetória tende a ser pequeno e por isso propomos uma estratégia de

controle simples para realizar o seguimento do caminho planejado.

Seja o estado de referência $q_r = [x_r \ y_r \ \psi_r]^T$, o estado atual do robô $q_c = [x_c \ y_c \ \psi]^T$ e o estado que representa o erro $q_e = q_r - q_c$. Da equação (3.5), o seguinte controlador proporcional foi desenvolvido:

$$\boldsymbol{\eta}_e = S^+ \mathbf{q}_e$$

$$\begin{bmatrix} v_e \\ w_e \end{bmatrix} = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\frac{x_{CIR} \sin \psi}{x_{CIR}^2 + 1} & \frac{x_{CIR} \cos \psi}{x_{CIR}^2 + 1} & \frac{1}{x_{icr}^2 + 1} \end{bmatrix} \begin{bmatrix} x_r - x_c \\ y_r - y_c \\ \psi_r - \psi \end{bmatrix} \quad (3.10)$$

sendo que S^+ é a pseudo-inversa da matriz S e $\boldsymbol{\eta}_e = [v_e \ \omega_e]^T$. A partir das velocidades de erro apresentadas acima, temos a seguinte lei de controle:

$$\begin{aligned} v_{track} &= v_r + k_v v_e \\ w_{track} &= w_r + k_\omega w_e \end{aligned} \quad (3.11)$$

sendo que k_v, k_ω são constantes e v_r, ω_r as velocidades de referência, linear e angular respectivamente. Os resultados do seguimento de trajetória são mostrados no Capítulo 6.

4 Planejamento de movimento cinemático-dinâmico baseado em acompanhamento

Os planejadores de movimento básicos desconsideram completamente a dinâmica do robô, levando em conta apenas a cinemática. Ao fazer isto os caminhos gerados podem ser impraticáveis, devido a limitações de força e torque do robô real. Grande parte das pesquisas conduzidas em robótica na área de planejamento de movimento, separam o problema de planejamento em duas fases. Na primeira fase utilizam um planejador de movimentos básico e na segunda fase projetam um controlador com o objetivo de acompanhar o caminho planejado obedecendo as restrições dinâmicas do modelo do robô.

A abordagem aqui proposta denominada Planejamento de Movimento Cinemático-Dinâmico Baseado em Acompanhamento aplica o controlador que será utilizado no acompanhamento da trajetória planejada na fase de planejamento de movimento. Esta abordagem é implementada a partir do planejador de movimento cinemático-dinâmico básico, apresentado no capítulo 2, que possui como algoritmo base o RRT.

4.1 Metodologia

O comportamento do algoritmo RRT básico é inteiramente aproveitado nesta abordagem de planejamento de movimento. Isto significa que todas as características inerentes ao método RRT, tais como: habilidade de exploração do ambiente e aleatoriedade nas trajetórias planejadas, são compartilhadas pela abordagem proposta.

A seguir descreveremos, em detalhes, o método de planejamento de movimento cinemático-dinâmico baseado em acompanhamento. A fase inicial do algoritmo implica em definir o estado inicial e o estado final desejado do robô. O estado do robô é definido pelo vetor $x = (x, y, \psi, v, \omega)$ sendo (x, y, ψ) a posição e orientação do robô e (v, ω) as velocidades li-

near e angular. Logo em seguida define-se um número máximo de tentativas para atingir o estado final desejado. O processo de buscar uma trajetória entre o estado inicial e o estado final é semelhante ao proposto no Capítulo 2 Seção 2.4. Um estado aleatório x_{rand} é amostrado, e é feita uma busca na árvore para encontrar o estado mais próximo de x_{rand} , a este estado é dado o nome de x_{near} . A partir de x_{near} , que já se encontra na árvore, são aplicadas as entradas de controle permitidas gerando um conjunto de possíveis estados futuros. Ao contrário do algoritmo básico do RRT que apenas escolheria, dentre os possíveis estados futuros, o estado que mais se aproxima de x_{rand} , agora há uma nova fase que chamamos de fase de simulação de acompanhamento.

Seja q_{near} o estado a partir do qual é feita a expansão para os estados futuros estimados, $q_{estimated}$ o estado futuro estimado e $u_{control}$ a entrada de controle que transforma q_{near} em $q_{estimated}$. A fase de acompanhamento tem como objetivo utilizar os dados acima apresentados como entrada de um controlador e estimar a influência do controle na execução da trajetória. Desta maneira saberemos ainda na fase de planejamento de movimento o que acontecerá ao utilizarmos o controle na fase de execução da trajetória planejada. Teremos o seguinte resultado:

$$q_{tracked} = control_law(q_{near}, q_{estimated}, u_{control}) \quad (4.1)$$

Sendo $q_{tracked}$ o estado acompanhado esperado ao aplicarmos o controlador no robô real. Após aplicarmos a Equação 4.1 a todos os estados estimados ($q_{estimated}$) teremos um conjunto de estados acompanhados. Dentre os estados acompanhados, o que mais se aproxima do estado aleatório q_{rand} é escolhido e adicionado na árvore.

Na Figura 4.1 tem-se um exemplo de expansão da árvore RRT utilizando a abordagem proposta. As linhas tracejadas pretas indicam as possíveis trajetórias estimadas ao aplicar as entradas de controle tendo como estado inicial x_{near} . As linhas pontilhadas mostram os estados acompanhados, isto é, mostram as trajetórias obtidas aplicando-se a Equação 4.1. E a linha pontilhada verde mostra a trajetória final escolhida.

O diagrama de blocos mostrado na Figura 4.2 sumariza o processo de expansão usado para obter os possíveis estados futuros. As entradas de controle aplicadas no modelo dinâmico são um conjunto de acelerações linear e angular. Desta maneira para cada par de aceleração $(\dot{v}, \dot{\omega})$ é calculado o torque que o motor deve alcançar. Caso o torque calculado esteja dentro dos limites físicos do motor do robô, a entrada de controle é validada. O modelo cinemático é propagado através de integração numérica e obtemos os possíveis estados futuros.

De posse dos possíveis estados futuros, o diagrama de blocos da Figura 4.3 realiza a abor-

dagem proposta, alimentando a lei de controle 3.11 conforme descrito pela equação 4.1. Depois que todos os estados futuros estimados passarem pela fase de simulação de acompanhamento, o estado acompanhado que mais se aproximar de x_{rand} será adicionado na árvore.

Figura 4.1: Abordagem de Planejamento Cinemático-Dinâmico de Movimento baseado em Acompanhamento – Exemplo de expansão.

Figura 4.2: Diagrama de blocos da fase de expansão

Figura 4.3: Diagrama de blocos da fase de simulação de acompanhamento

5 Implementação

O método de planejamento de movimento baseado em RRT, descrito nos algoritmos apresentados, foi implementado utilizando-se a linguagem de programação C++. O planejador de movimentos desenvolvido nesse trabalho foi chamado de `rrt-motion-planner` e para oferecer reusabilidade do código em projetos futuros o software foi implementado segundo o paradigma de programação orientada a objetos. O código fonte desenvolvido ao longo desse trabalho está disponibilizado online e pode ser acessado através do endereço: <http://code.google.com/p/rrt-motion-planner/>.

Foram também desenvolvidos alguns softwares auxiliares que cuidam da obtenção de valores de torques válidos para o robô, execução e acompanhamento da trajetória planejada, visualização das trajetórias planejada e realizada pelo robô P3-AT. As ferramentas e bibliotecas utilizadas na implementação do `rrt-motion-planner` e a discussão mais detalhada da funcionalidade das classes que compõe o planejador de movimentos são descritos nas seções seguintes.

5.1 Ferramentas e bibliotecas

5.1.1 Player/Stage

As trajetórias planejadas pelo software `rrt-motion-planner` são executadas tanto em ambiente computacional, através de simulações, quanto no robô P3-AT. O conjunto de aplicativos Player/Stage foi utilizado nesse trabalho para realizar ambas tarefas.

O software Player permite, através de uma camada de abstração de hardware, que o usuário acesse diferentes tipos de sensores, instalados em plataformas robóticas distintas, sem se preocupar em reescrever um novo código de controle para cada robô diferente. Um arquivo de configuração utilizado na inicialização do Player, informa quais dispositivos estão presentes no robô e fornecem os respectivos drivers para cada dispositivo. Dessa maneira um mesmo código de controle pode ser utilizado para controlar diversos tipos de robôs, bastando para isso

modificar apenas o arquivo de configuração que informa ao player o robô utilizado.

O Stage é um simulador que possibilita o teste do código, antes de aplicá-lo em um robô real. Portanto, a união dessas duas características, quais sejam, reusabilidade de código e ambiente de simulação, fazem da suíte de aplicativos Player/Stage uma poderosa ferramenta na área de robótica (GERKEY; VAUGHAN; HOWARD, 2003).

5.1.2 PQP – Proximity Query Package

Para garantir que o robô mantenha-se a uma distância segura dos obstáculos presentes no ambiente e não entre em colisão, o software `rrt-motion-planner` utiliza a biblioteca Proximity Query Package, PQP (LARSEN et al., 2000). Essa biblioteca utiliza modelos geométricos compostos por um conjunto de triângulos que permitem representar várias formas de obstáculos e robôs, além de realizar três tipos de consultas de proximidade entre os modelos: distância de tolerância, detecção de colisão e distância mínima. Este trabalho utiliza a verificação de distância de tolerância para evitar colisão entre os modelos geométricos do robô e obstáculos.

5.1.3 LEMON

Como visto na Seção 2.4, o algoritmo de planejamento RRT utiliza uma estrutura de dados para armazenar a árvore de expansões e o caminho planejado. A biblioteca LEMON (Egerváry Research Group on Combinatorial Optimization, 2010) implementa várias classes que permitem utilizar diferentes abordagens de grafos. Além disso a biblioteca também possui alguns algoritmos de busca em grafos, de grande utilidade para o software implementado. Essa biblioteca é implementada em C++, que torna fácil a sua utilização pelo software `rrt-motion-planner`, justificando assim o motivo de sua escolha.

5.2 Arquitetura do software

O `rrt-motion-planner` foi desenvolvido com o propósito de lidar com qualquer classe de robô. Uma vez que se conheça o modelo do robô, representado por equações diferenciais, é possível estender o planejamento de movimento para qualquer modelo robótico. Dessa maneira o paradigma de programação orientada a objetos é bastante eficaz, devido a funcionalidades como herança e polimorfismo. O diagrama de classes para o software `rrt-motion-planner` é mostrado na Figura 5.1.

Figura 5.1: Diagrama simplificado de classes – Software rrt-motion-planner

As seguintes classes bases são definidas: `RobotModel` que descreve o modelo matemático do robô (cinemático, dinâmico, baseado em acompanhamento) e `RobotGeometryModel` que descreve as características físicas do robô. A partir dessas classes podem ser criadas classes filhas, permitindo assim descrever diversos modelos de robôs. A classe `EnvModel` cuida da representação do mapa do ambiente e a classe `World` dos testes de verificação de distância de tolerância entre o modelo geométrico do robô e mapa do ambiente. A RRT é a principal classe do software pois trata diretamente do planejamento de movimento baseado no algoritmo RRT.

5.2.1 Classe RRT

Esta classe implementa o método de planejamento de movimento baseado nos algoritmos básicos de RRT apresentados no Capítulo 2. A classe RRT, juntamente com seus atributos e métodos, pode ser vista na Figura 5.2.

Para criar uma instância dessa classe é necessário definir os seguintes parâmetros: estado inicial, estado final desejado e número máximo de tentativas de expansão. Os algoritmos 1 e 2 utilizam uma estrutura de dados para armazenar as expansões em forma de grafo. Cada nó do grafo é um estado que o robô pode assumir e cada aresta do grafo guarda as seguintes informações: estado atual, ação de controle, tempo de aplicação da ação de controle e o estado futuro estimado. A biblioteca LEMON cuida de toda a parte relacionada a grafos, desde a inserção de novos nós até a busca pelo caminho mais curto entre dois nós do grafo.

Como apresentado no Capítulo 3, as coordenadas (x, y) representam o centro de gravidade do robô. A partir dessas coordenadas e do ângulo de orientação, ψ , a classe `VehicleGeome-`

Figura 5.2: Diagrama da classe RRT

`tryModel` calcula os vértices de dois triângulos, que juntos formam um retângulo que modela o corpo do robô. Os testes realizados pela biblioteca PQP, tem o objetivo de evitar a colisão com os obstáculos presentes no ambiente, e dessa forma estabelecer uma distância segura de tolerância, que deve ser respeitada ao longo do planejamento de movimento.

A classe `StateSampler` amostra o espaço de estados e retorna como resultado um estado aleatório x_{rand} . A partir da posição atual do robô e tendo como objetivo local o estado x_{rand} , o método `CheckPathNoCollision` substitui a Linha 5 do Algoritmo 2. Esse método faz a integração numérica das equações diferenciais representadas do modelo do robô e verifica para cada etapa da integração se o estado estimado encontra-se a uma distância segura dos obstáculos, ao invés de realizar isso apenas no final da integração numérica. Como várias entradas de controle são utilizadas nas tentativas de expansão da árvore, utiliza-se uma métrica para escolher o estado futuro estimado que mais aproxima-se de x_{rand} . A métrica utilizada no Algoritmo 2, é implementado pela classe `BiasedMeterP3AT` linha 7.

Para evitar duplicidade de nós no grafo, após o estado x_{chosen} ser escolhido de acordo com a métrica acima apresentada, é realizada uma busca por x_{chosen} no grafo. Essa busca é feita pelo método `CheckDuplicatedNode` que retorna o índice do nó caso ele já esteja adicionado no grafo ou então adiciona o nó. O grafo também armazena uma aresta que contém, além de x_{near} e x_{chosen} , a entrada de controle e o tempo total de integração.

Após esgotar o número máximo de tentativas de expansão ou caso o destino seja alcançado, é realizada uma busca no grafo pelo caminho mais curto entre o estado inicial e o estado final desejado (ou o estado que mais se aproxima do objetivo), realizado pelo método `PathToClosestGoalState`. Esse caminho é então exportado para um arquivo de log, através do método

`ExportPath` para que seja possível a execução da trajetória no ambiente de simulação ou pelo robô P3-AT.

5.2.2 Classe `StateSampler`

No algoritmo 1 linha 4 é necessário amostrar um estado aleatório que guia o crescimento da árvore. Esta classe fornece diferentes tipos de métodos de amostragem do espaço de estados. Desta maneira para problemas distintos pode-se escolher o método de amostragem mais apropriado. Os métodos de amostragem fornecidos são: distribuição gaussiana ou normal, distribuição uniforme e amostragem tendenciosa. Este último método de amostragem retorna o estado tendencioso desejado com uma probabilidade p e com probabilidade $1 - p$ retorna um estado tomado de uma distribuição uniforme.

5.2.3 Classes `DistanceMeter` e `BiasedMeterP3AT`

Após o estado aleatório ser escolhido é necessário encontrar o estado mais próximo já inserido na árvore de expansões. Para realizar estas medidas de distância foi criada a classe `DistanceMeter`, que também é a classe base da classe filha `BiasedMeterP3AT`.

A classe base `DistanceMeter` fornece 3 métodos virtuais: `NearestNodeMetric` que é utilizado para determinar qual nó da árvore encontra-se mais próximo do estado aleatório. Determinando o nó da árvore mais próximo do estado aleatório, são feitas expansões a partir deste nó, aplicando-se as possíveis entradas de controle. Para determinar qual dos nós expandidos encontra-se mais próximo do estado aleatório é utilizado o método `DistanceWeight`. Após encontrar qual nó expandido encontra-se mais próximo do estado aleatório, este nó é inserido na árvore e então o método `GoalStateAchieved` determina se o nó que acabou de ser inserido está próximo o bastante do estado final desejado, encerrando ou não a expansão da árvore.

A classe filha `BiasedMeterP3AT` sobrescreve os três métodos descritos acima, permitindo que o planejador de movimentos se comporte de uma maneira adequada ao robô Pioneer-3AT. O estado que representa o robô é: (x, y, ψ, v, ω) . No método `NearestNodeMetric` o nó mais próximo é escolhido com base na distância euclidiana, uma certa ponderação no ângulo de orientação ψ e na diferença entre as velocidades do estado aleatório e nó da árvore. Caso o estado aleatório esteja dentro de um círculo de raio r cujo centro seja o estado final desejado, as velocidades (v, ω) do estado aleatório são alteradas para o valor das velocidades do estado final desejado, isto é feito para aumentar a probabilidade do estado final ser atingido com as

velocidades pré-determinadas.

O método `DistanceWeight` é utilizado para obter uma medida entre estado a ser atingido e estado expandido através da aplicação de uma entrada de controle. Esta medida leva consideração a distância euclidiana entre os dois estados, a diferença do ângulo de orientação ψ e a diferença da velocidade linear.

5.2.4 Classes `TrackingControl`, `ProportionalControl` e `Pioneer3ATState`

A classe `TrackingControl` é uma classe virtual que possui dois métodos que devem ser reimplementados pelas classes filhas, que são: `run` e `InitializeControllerWeights`. Esta classe tem a função de simular o acompanhamento da trajetória na etapa de planejamento, realizando desta maneira o que chamamos de Planejamento Baseado em Acompanhamento, discutido anteriormente. O método `run` tem o comportamento do algoritmo descrito no Capítulo 4.

A classe `ProportionalControl` é uma classe filha da classe `TrackingControl`. Esta classe implementa um controlador Proporcional e a classe `Pioneer3ATState` é utilizada para cálculos de erros de velocidade e aceleração, que serão utilizados no controlador proporcional.

5.2.5 Classes `RobotGeometryModel`, `EnvModel` e `World`

Essas três classes trabalham em conjunto para representar geometricamente o robô e ambiente, além de realizar testes de distância de tolerância entre os modelos `RobotGeometryModel` e `EnvModel` com o intuito de evitar colisões ao longo do caminho que será planejado. O relacionamento dessas três classes é representando no diagrama de classes da Figura 5.3.

A função da classe `RobotGeometryModel` é representar geometricamente um robô. A classe filha, `VehicleGeometryModel` representa um robô que assemelha-se a um veículo, isto é, um robô cujo corpo pode ser representado por um retângulo. Essa classe recebe informações de largura, comprimento, distância entre os eixos do veículo, a coordenada (x, y) que informa a posição do centro de rotação do veículo e o ângulo de orientação ψ . A partir dessas informações o método `SetVerticesPosition` faz os cálculos, tanto de rotação quanto de translação, para gerar os quatro vértices que representam o corpo do veículo na forma de um retângulo. É importante ressaltar que se houver necessidade de utilizar um robô cujo corpo não se assemelhe a um retângulo, basta criar uma nova classe filha, da classe pai `RobotGeometryModel`.

Figura 5.3: Diagrama de classes – RobotGeometryModel, EnvModel e World

A classe `EnvModel` é responsável por armazenar as informações relacionadas ao mapa do ambiente. Essa classe trabalha em conjunto com a biblioteca PQP. Os obstáculos presentes no ambiente, são armazenados em um arquivo de texto. Cada linha desse arquivo contém três pontos, no espaço cartesiano tridimensional, que formam um triângulo. A partir desse arquivo, a classe `EnvModel` constrói modelos compatíveis com a biblioteca PQP.

O método `GetPosition` da classe `RobotGeometryModel` é utilizado para obter informações a respeito da posição do corpo do robô e construir um modelo do tipo `PQP_Model`. A classe `World` implementa o método `IsVehicleInSafePosition` que faz o teste de distância de tolerância entre o modelo do veículo e o mapa do ambiente representado pelo atributo `obstacles` pertencente a classe `EnvModel`.

5.2.6 Classe RobotModel

A classe base `RobotModel` é responsável por guardar as informações do modelo do robô, tais como: dimensão do espaço de estados, entradas de controle permitidas, equações diferenciais que representam o robô etc. Além disso, implementa apenas métodos virtuais, isto é, todos os métodos por ela implementados devem ser redefinidos em uma classe filha. A Figura 5.4 mostra o diagrama de classes detalhado para a classe `RobotModel`.

Figura 5.4: Diagrama de classes – RobotModel

No software desenvolvido foram implementadas duas classes de robôs, derivadas da classe base **RobotModel**. A classe **CarLikeModel** representa o modelo cinemático de um veículo com rodas esterçantes, utilizando como espaço de estados o vetor (x, y, ψ, v, ω) . A classe **SkidSteerDynamicModel** representa o robô Pioneer 3-AT e leva em consideração a dinâmica do modelo, o espaço de estados é representado pelos parâmetros $(x, y, \psi, v, \omega, \dot{v}, \dot{\omega})$.

A classe **SkidSteerTrackingBased** representa o modelo matemático do robô de rodas deslizantes Pioneer 3AT e agrega também um controlador do tipo **TrackingControl**. Desta forma diferentes tipos de controle podem ser utilizados, bastando criar classes filhas da classe base **TrackingControl**. Esta classe é utilizada na abordagem de planejamento proposta no Capítulo 4 - Planejamento de movimento cinemático-dinâmico baseado em acompanhamento.

O modelo do sistema robótico é representado por um conjunto de equações diferenciais ordinárias. No Algoritmo 2 linha 5, o método `EstimateNewState` estima o estado futuro do robô a partir do estado atual do robô, da entrada de controle e das equações diferenciais (3.9) e (3.5).

O método `EstimateNewState` usa a equação (3.9) para estimar as acelerações linear e angular, a partir do torque aplicado τ . Por integração numérica encontra-se η . De posse do estado atual q_n , do vetor velocidade η e do tempo de integração Δt , estima-se o estado futuro

do robô, q_{n+1} , utilizando-se a equação (3.5). O processo de integração numérica utilizado é o método Runge-Kutta de quarta ordem. A equação (5.1) descreve o método de integração numérica para a equação (3.5).

$$\begin{aligned}\dot{x} &= h(x, \eta, t), \quad x(t_o) = x_n, \\ x_{n+1} &= x_n + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4), \\ t_{n+1} &= t_n + \Delta t.\end{aligned}\tag{5.1}$$

sendo que:

$$\begin{aligned}k_1 &= h(x_n, \eta, t_n), \\ k_2 &= h\left(x_n + \frac{\Delta t}{2}k_1, \eta, t_n + \frac{\Delta t}{2}\right), \\ k_3 &= h\left(x_n + \frac{\Delta t}{2}k_2, \eta, t_n + \frac{\Delta t}{2}\right), \\ k_4 &= h\left(x_n + \Delta tk_3, \eta, t_n + \Delta t\right).\end{aligned}$$

Conhecendo as equações diferenciais que regem o modelo, ao aplicar uma entrada de controle consegue-se estimar o estado futuro do robô. Apenas os estados futuros estimados que não se encontram em colisão, podem ser utilizados na expansão da árvore. Como dito no Capítulo 2 as entradas de controle aplicadas ao modelo devem ser entradas válidas e que não desobedeçam às restrições impostas pelo modelo dinâmico. A próxima Seção apresenta um software auxiliar criado para gerar uma tabela de torques que armazena quais possíveis valores de torques podem ser aplicados ao modelo levando em consideração a velocidade atual do robô.

5.3 Geração de torques

O algoritmo de planejamento espera apenas entradas de controle permitidas, assim há uma ferramenta responsável por gerar uma tabela de torques, a partir de dados como torque mínimo e máximo, valor absoluto da aceleração máxima permitida, valor absoluto da velocidade máxima permitida e valor atual da velocidade do robô.

É válido ressaltar que a discretização dos valores de torques causa um impacto direto no desempenho do algoritmo de planejamento de movimento. O Algoritmo 2 é dependente das entradas de controle do modelo utilizado para representar o robô. Desta maneira ao discretizar os valores de torque a serem aplicados no modelo, limita-se os tipos de movimentos que o robô poderá executar. Tendo isto em mente chega-se a conclusão que quanto maior o número de entradas de controle, maior será a diversidade de possíveis curvaturas a serem realizadas pelo robô. No entanto, a discretização faz-se necessária pois quanto maior o número de entradas de controle mais custoso computacionalmente será o cálculo de todas as curvaturas possíveis.

Três tabelas distintas são geradas. Os valores de torques da primeira tabela são calculados para atuar no movimento linear do robô, isto é, faz com que o robô acelere ou freie, sem interferir no ângulo de orientação. As outras duas tabelas são responsáveis por armazenar valores de torque que auxiliam o robô a realizar curvas para a direita ou para esquerda. Os valores utilizados nestas tabelas foram escolhidos de maneira a maximizar a relação custo-benefício entre número de entradas de controle e custo computacional do algoritmo.

O Algoritmo 3 descreve a geração de torques que fazem o robô acelerar ou freiar (primeira tabela). Para gerar torques que provoquem o movimento de rotação do robô basta substituir a Linha 4 e atribuir valores diferentes para τ_{left} e τ_{right} , além de fornecer velocidade angular máxima, aceleração angular máxima e velocidade angular atual do robô, como parâmetros de entrada para o algoritmo.

Algoritmo 3 Geração de torques válidos

Entrada:
 $v_{lin_abs_max}, acc_{lin_abs_max}, \tau_{min}, \tau_{max}$
Saída:

Tabela de valores de torques válidos.

```

1: for  $v = -v_{lin\_abs\_max}; v \leq v_{lin\_abs\_max}; v = v + v\_step$  do
2:    $x \leftarrow (0, 0, 0, v, 0)$ 
3:   for  $torque = \tau_{min}; torque \leq \tau_{max}; torque = torque + torque\_step$  do
4:      $\tau_{left} \leftarrow \tau_{right} \leftarrow torque$ 
5:      $input\_control \leftarrow (\tau_{left}, \tau_{right})$ 
6:      $x_{new} \leftarrow EstimaNovoEstado(x, input\_control)$ 
7:      $acc_{inst} = f_{abs}(x_{new}.v - x.v)$ 
8:     if  $acc_{inst} \leq acc_{lin\_abs\_max}$  then
9:       Insere o valor do torque na tabela, relacionada a velocidade linear quantizada
       atual  $v$ 
10:    end if
11:  end for
12: end for

```

Como visto no Algoritmo 3 Linha 1 não é possível representar de forma contínua a velocidade do robô, portanto os torques permitidos são gerados apenas para valores quantizados de velocidade. Isto significa que no momento da expansão da árvore, a velocidade atual do robô será arredondada para o valor mais próximo das velocidades quantizadas.

Para uma dada velocidade quantificada v , quinze valores de torques são escolhidos, sendo que são tomados cinco valores por tabela de acordo com os seguintes critérios:

- Tabela 1: dois valores de torques que freiam o robô, dois valores de torques que acelerem o robô e um valor de torque que não altere a velocidade linear.

- Tabela 2 e 3: os cinco valores, para cada tabela, são tomados em ordem decrescente de aceleração angular, isto é, os cinco torques que promovem as maiores acelerações angulares serão escolhidos.

Realizando-se essa escolha para as n velocidades quantificadas possíveis, é criado um arquivo de texto que guarda esses valores de torques. Desta forma, para cada velocidade quantificada há quinze entradas de controle que podem ser aplicadas para estimar os estados futuros.

Os métodos `GenerateInputs` e `GetValidInputs` da classe `RobotModel` tem respectivamente as seguintes funcionalidades:

- Ler o arquivo de texto e armazenar todas as entradas da tabela no vetor `inputs`.
- Selecionar os elementos do vetor `inputs` de acordo com o estado atual do robô, isto é, apenas as entradas de controle permitidas levando em consideração o estado atual do robô.

5.4 Acompanhamento de trajetória

Após o término da execução do algoritmo de planejamento de movimento, é gerado um arquivo de log que contém as ações de controle aplicadas no robô ao longo do percurso. A classe `Tracking` é responsável por enviar esses comandos ao software `Player`. O diagrama de classes simplificado da Figura 5.5 mostra o relacionamento entre as classes utilizadas pelo software de acompanhamento de trajetória.

Figura 5.5: Diagrama de classes - Tracking

O arquivo de configuração escolhido para inicializar o `Player`, permite que o usuário alterne entre o modo de simulação, que utiliza o `Stage`, e a execução da trajetória no robô real. Fo-

ram implementados dois métodos responsáveis por realizar o acompanhamento de trajetória: `PathTracking` e `OpenLoopControl`. O primeiro implementa as equações (3.10) e (3.11) do controlador proporcional apresentadas na Seção 3.3. Já o segundo método aplica as entradas de controle em malha aberta.

5.4.1 Implementação do acompanhamento de trajetória para o Player/Stage

O Player fornece diversas classes que possibilitam acessar os mais diversos tipos de dispositivos comumente encontrados em robôs. No entanto é uma boa prática de programação encapsular classes fornecidas por uma biblioteca, de maneira que se algum método ou atributo da biblioteca for alterado, não seja necessário alterar todo o código, bastando assim alterar apenas a classe encapsuladora. Duas classes foram implementadas para encapsular a comunicação com o Player: `ProxyRobot` e `ProxyPosition`, mostradas no diagrama de classes da Figura 5.6.

Figura 5.6: Diagrama de classes - `ProxyRobot` e `ProxyPosition`

A classe `ProxyPosition` é derivada da classe fornecida pelo Player `Position2dProxy`. O método `GetPose` retorna a posição atual do robô a partir da leitura do odômetro, em uma estrutura do tipo (x, y, ψ) . Os métodos `SetMotorStatus` e `SetOdomPos` são responsáveis por ligar o motor e ajustar a posição atual do odômetro, respectivamente. O método `SetNewSpeed` envia para o robô a nova velocidade (linear e angular). A classe `ProxyRobot` cuida da conexão com o Player, encapsulando a classe `ClientProxy`, fornecida pelo Player, e tem como atributo uma instância da classe `ProxyPosition`. Assim a classe `Tracking` através de uma instância da classe `ProxyRobot` consegue acesso aos comandos de controle do robô.

6 Resultados

A seguir são mostrados os resultados obtidos durante o decorrer deste trabalho. O capítulo está dividido em duas seções. A seção 6.1 trata apenas da fase de planejamento de movimento. Nesta seção serão apresentados o planejamento de movimento cinemático-dinâmico básico e o planejamento de movimento cinemático-dinâmico baseado em acompanhamento.

A seção 6.2 apresenta os resultados experimentais de acompanhamento das trajetórias planejadas na seção anterior. Os resultados experimentais foram levantados a partir da leitura dos sensores embarcados no robô Pioneer 3AT.

Como dito no Capítulo 2, o algoritmo *RRT*, devido sua natureza aleatória, tem como característica uma boa capacidade de exploração do ambiente. Para que o algoritmo alcance o destino mais rapidamente, ao invés de sempre amostrar aleatoriamente um estado para o qual a árvore deverá crescer, o objetivo final desejado é escolhido com uma certa propabilidade p . Esta polarização do crescimento da árvore faz com que o objetivo seja alcançado com menores iterações mas também pode prender a expansão da árvore em um mínimo local. Nos problemas de planejamento mostrados adiante, foi escolhido uma probabilidade de $p = 0.15$. Isto é 85% das tentativas de expansão da árvore serão aleatórias e 15% delas terão como valor o estado final desejado.

É importante ressaltar que o planejador de movimento cinemático-dinâmico básico utiliza uma métrica diferente da utilizada pelo planejador de movimento baseado em acompanhamento. A subseção 5.2.3 descreve em detalhes como são escolhidos os estados expandidos que serão adicionados à árvore *RRT* para ambos os planejadores de movimento.

6.1 Planejamento de movimento

Foram escolhidos dois problemas de planejamento de movimento. No primeiro cenário o robô deve obrigatoriamente atravessar uma passagem estreita para atingir o estado final desejado. No outro cenário o robô deverá realizar um estacionamento paralelo, ou baliza, entre dois

obstáculos. O objetivo é realizar a comparação entre as soluções geradas pelo planejador de movimento cinemático-dinâmico básico e pelo planejador de movimento baseado em acompanhamento.

Para o primeiro problema de planejamento, as passagens tem a largura de 1 *m* e a distância de tolerância adotada foi de 15 *cm*, então as passagens tem uma largura útil de 70 *cm*, sendo que a largura do robô P3-AT é de 50 *cm*.

De posse das seguintes informações: mapa do ambiente, estado inicial do robô, estado final desejado, número máximo de tentativas de expansão da árvore e distância segura dos obstáculos, o mesmo problema de planejamento de movimento foi executado 100 vezes. O número máximo de tentativas de expansão para cada planejamento é $max_trial = 13000$. Um arquivo de log foi gerado para cada planejamento. Como o método RRT explora a característica de aleatoriedade na escolha de x_{rand} para expandir a árvore, diferentes respostas para o mesmo problema de planejamento de movimento podem ser encontradas.

Todo planejamento de movimento é dito bem sucedido se o planejador de movimento consegue encontrar uma trajetória que leva o robô do estado inicial para o estado final sem esgotar o número máximo de tentativas de expansão.

O tempo médio de um planejamento bem sucedido é o tempo gasto para o algoritmo RRT encontrar uma solução que leva o robô do estado inicial ao estado final sem colisões. Caso não haja solução (planejamento mal sucedido) o tempo representará o quanto foi gasto em segundos para esgotar o número máximo de iterações do algoritmo.

Abaixo na tabela 6.1 temos uma comparação dos tempos gastos durante o planejamento das trajetórias. A tabela compara os valores obtidos utilizando o planejador de movimento cinemático-dinâmico básico e a abordagem proposta neste trabalho, o planejador de movimento cinemático-dinâmico baseado em acompanhamento.

Tabela 6.1: Tabela de desempenho - Passagem estreita

	Básico	Baseado em acompanhamento
Planejamentos bem sucedidos	82	73
Tempo médio	2.44 <i>s</i>	9.23 <i>s</i>
Número médio de nós inseridos	1008	4166
Planejamentos mal sucedidos	18	27
Tempo médio	7.5 <i>s</i>	26.25 <i>s</i>
Tempo total	335 <i>s</i>	1383 <i>s</i>

Pode-se observar que o planejador de movimento cinemático-dinâmico básico teve maior desempenho, resolvendo 82 das 100 fases de planejamento de movimento. Isto deve-se ao fato

dos dois planejadores utilizarem métricas diferentes durante a expansão da árvore. O planejador de movimento cinemático-dinâmico baseado em acompanhamento considera que o estado final desejado foi alcançado quando a posição e orientação estão dentro de um certo raio de tolerância e o erro entre a velocidade com que deseja-se alcançar o objetivo e a velocidade final planejada encontra-se dentro de um limite. Por outro lado o planejador de movimento cinemático-dinâmico básico leva em consideração apenas a posição e orientação como parâmetros para decidir se o estado final foi alcançado.

Para permitir que o usuário escolha qual das trajetórias planejadas será a mais apropriada para o problema de planejamento, os arquivos de log gerados para as diferentes trajetórias são ordenados seguindo cinco critérios: distância entre o estado final alcançado pela expansão da RRT e x_{goal} , o tempo gasto de x_{init} até x_{goal} , a distância percorrida, o tempo que o robô se locomove com velocidade negativa e uma medida da mudança de orientação do robô ao longo da trajetória.

As figuras 6.1 e 6.2 mostram o resultado do planejamento de movimento para um mesmo problema de passagem estreita, utilizando as abordagens de planejamento dinâmico e baseado em acompanhamento. O critério de seleção utilizado para a escolha das soluções foi a distância final do estado final desejado.

Figura 6.1: Passagem estreita - Dinâmico

O segundo problema de planejamento que consiste em realizar um estacionamento paralelo ou baliza, a distância livre entre os obstáculos é de $1,20\text{ m}$. Levando em consideração a mesma distância segura mantida dos obstáculos, o robô realiza o estacionamento em um espaço útil de 90 cm . O número máximo de tentativas de expansão para cada planejamento é $max_trial = 20000$. Também foram realizadas 100 tentativas de planejamento de movimento.

Figura 6.2: Passagem estreita - Baseado em acompanhamento

Abaixo na tabela 6.2 temos uma comparação dos tempos gastos durante o planejamento das trajetórias. A tabela compara os valores obtidos utilizando o planejador de movimento cinemático-dinâmico básico e a abordagem proposta neste trabalho baseada em acompanhamento.

Tabela 6.2: Tabela de desempenho - Estacionamento paralelo

	Básico	Baseado em acompanhamento
Planejamentos bem sucedidos	37	51
Tempo médio	3.72 s	12.16 s
Número médio de nós	1384	4618
Tempo total	605 s	1849 s

Dentre os 100 planejamentos realizados para solucionar o problema de estacionamento paralelo, escolheu-se uma solução para cada abordagem. As soluções escolhidas foram as que levaram o menor tempo entre o estado inicial e o objetivo e podem ser vistas nas figuras 6.3 e 6.4.

Com base nas estatísticas de ambos problemas pode-se concluir que o planejamento de movimento baseado em acompanhamento é mais custoso em termos computacionais na ordem de 5 vezes. Isto deve-se ao fato de que o número de nós adicionados em um planejamento bem sucedido baseado em amostragem está próximo a 5 vezes o número de nós adicionados para um planejamento de movimento cinemático-dinâmico básico.

O número de nós presentes na árvore tem impacto no algoritmo 2, onde há a necessidade de descobrir qual nó que está presente na árvore encontra-se mais próximo do estado aleatório amostrado. Desta maneira quanto maior o número de nós que a árvore possuir mais iterações

Figura 6.3: Estacionamento paralelo - Dinâmico

Figura 6.4: Estacionamento paralelo - Baseado em acompanhamento

serão feitas na busca de x_{near} . Além disso ao tentar adicionar um novo nó e uma nova aresta na árvore, é realizada uma verificação a fim de evitar nós e arestas duplicadas, então todos os nós e arestas da árvore são visitados para garantir que não haja nós/arestas repetidos.

6.2 Acompanhamento das trajetórias planejadas

6.2.1 Planejador de movimento cinemático-dinâmico

Foram tomadas duas soluções aleatórias geradas pelo planejador de movimento que leva em consideração a dinâmica do robô. As figuras 6.5 e 6.6 mostram a execução das trajetórias executadas pelo robô Pioneer 3AT.

Para a trajetória representada pela figura 6.5 os erros máximos ao longo da execução são: $|x_e| = 25 \text{ cm}$, $|y_e| = 23 \text{ cm}$ e $|\psi_e| = 17$ graus. Os erros entre o estado final desejado (objetivo) e o estado final alcançado são: $x_e = 2.4 \text{ cm}$, $y_e = 6.2 \text{ cm}$ e $\psi_e = -2$ graus.

Figura 6.5: Passagem estreita - Resultado experimental

Figura 6.6: Estacionamento paralelo - Resultado experimental

Para a trajetória representada pela figura 6.6 os erros máximos ao longo da execução são: $|x_e| = 12 \text{ cm}$, $|y_e| = 18 \text{ cm}$ e $|\psi_e| = 9.3$ graus. O erro entre o estado final desejado (objetivo) e o estado final alcançado são: $x_e = 10.4 \text{ cm}$, $y_e = 10.2 \text{ cm}$ e $\psi_e = -9$ graus.

6.2.2 Planejador de movimento cinemático-dinâmico baseado em acompanhamento

Os testes experimentais também foram realizados para as trajetórias planejadas utilizando a abordagem de planejamento de movimento baseado em acompanhamento. A figura 6.7 mostra a execução da trajetória para o problema de passagem estreita e a figura 6.8 mostra os erros de acompanhamento ao longo da trajetória.

Figura 6.7: Passagem estreita - Resultado experimental

Figura 6.8: Passagem estreita - Resultado experimental

Os erros máximos ao longo do acompanhamento são: $|x_e| = 16 \text{ cm}$, $|y_e| = 17 \text{ cm}$ e $|\psi_e| = 17$ graus. Os erros entre o estado final desejado e o estado final alcançado são: $x_e = 8.2 \text{ cm}$, $y_e = 2.9 \text{ cm}$ e $|\psi_e| = -3$ graus.

Os parâmetros utilizados na fase de planejamento de movimento, que representam o robô Pioneer 3-AT da tabela 6.3, foram extraídos de (KRISHNAMURTHY, 2008).

Tabela 6.3: Parâmetros do Pioneer 3-AT.

Parâmetro	Valor	Unidade
I	0.413	Kgm^2
m	40	Kg
f_r	0.043	
μ	0.506	
x_{CIR}	0.008	m
a	0.138	m
b	0.122	m
c	0.1975	m
r	0.4	m

7 Conclusão

Esse trabalho apresenta a implementação de um planejador cinemático-dinâmico de movimentos, baseado em RRT. Os resultados experimentais apresentados utilizam o robô móvel de rodas deslizantes Pioneer 3-AT. Os experimentos mostraram que o planejador cinemático-dinâmico de movimentos, `rrt-motion-planner`, pode ser usado para solucionar diferentes tipos de problemas de planejamento, tais como estacionamento paralelo e realizar manobras por passagens estreitas.

O software implementado pode ser estendido para qualquer classe de robô, bastando conhecer o conjunto de equações diferenciais ordinárias que regem a transição de estados do modelo do robô e as características físicas do robô.

A abordagem proposta neste trabalho, denominada Planejamento de Movimento Baseado em Acompanhamento tem como característica central utilizar o mesmo controlador tanto no planejamento do movimento quanto no acompanhamento das trajetórias planejadas. Esta abordagem permite levar em consideração o efeito da malha de controle ainda na fase de planejamento.

Os resultados experimentais tem o propósito de fechar o ciclo que envolve as fases de planejamento de movimento e acompanhamento de trajetória. Foram conduzidos duas classes de experimentos: a primeira classe de experimentos utiliza um planejador de movimentos cinemático-dinâmico baseado em RRT, cujas entradas de controle para o modelo dinâmico, que representa o robô, são os torques aplicados nas rodas do robô.

A segunda classe de experimentos utiliza um planejador de movimentos cinemático-dinâmico baseado em acompanhamento, cujas entradas de controle para o modelo dinâmico são as acelerações, linear e angular, a serem aplicadas no robô. Na segunda classe de experimentos há uma certa vantagem em termos de entradas de controle, pois o conceito de aceleração é mais comumente utilizado no dia a dia.

Durante a fase de planejamento o planejador de movimentos baseado em acompanhamento consumiu maiores recursos de memória e tempo de processamento. Embora o planejamento

baseado em acompanhamento seja mais lento, as soluções por ele alcançadas possuem algumas características que auxiliam na tarefa de execução da trajetória planejada. Há a possibilidade de estabelecer uma velocidade final desejada que o robô deve alcançar o objetivo e pode-se definir uma velocidade média desejada ao longo da trajetória. Todas as transições de estados são validadas através do modelo dinâmico do robô.

O modelo usado pelo planejador de movimentos baseado em acompanhamento possui como entradas as acelerações, linear e angular, aplicadas ao robô real, parâmetros que podem ser levantados com maior facilidade do que a geração de tabelas de torque anteriormente utilizadas pelo planejador dinâmico. A principal vantagem do planejamento baseado em acompanhamento é que os efeitos das ações de controle aplicados na etapa de acompanhamento da trajetória são levados em consideração ao se planejar os movimentos.

Para o problema de passagem estreita o planejador de movimento cinemático-dinâmico teve melhor desempenho. Isto deve-se à característica da métrica utilizada pelo planejador de movimento cinemático-dinâmico básico. Como tal planejamento de movimento não leva em consideração a velocidade desejada no estado final, é considerado que o estado final foi atingido quando qualquer estado que esteja dentro de um raio de tolerância levando em conta apenas a posição e orientação do robô é alcançado. Já o planejamento de movimento cinemático-dinâmico tenta alcançar o estado final atingindo a velocidade desejada, neste caso igual a zero. Assim mais uma restrição é imposta para garantir que o estado final desejado foi alcançado.

Para a solução do problema de estacionamento paralelo o planejamento de movimento baseado em acompanhamento mostrou um melhor desempenho se comparado ao planejamento de movimento cinemático-dinâmico básico. Pode-se inferir que isto deve-se ao fato de que a velocidade no estado final desejado é levado em consideração quando é utilizado o planejamento baseado em acompanhamento. Desta maneira ao aproximar-se do destino apenas estados cuja velocidade esteja próxima de zero são utilizados para expandir a árvore. Isto garante que o robô não entrará com uma velocidade demasiadamente grande na vaga de estacionamento, o que poderia causar uma colisão com os obstáculos.

Ambos experimentos foram realizados com o robô P3-AT. Os erros entre estado final planejado e estado final acompanhado mostraram-se com o mesmo desempenho para as trajetórias geradas por ambos planejadores de movimento. No entanto os caminhos obtidos com o planejador de movimentos baseado em acompanhamento mostraram-se mais suaves.

Referências Bibliográficas

- BRUCE, J.; VELOSO, M. Real-time randomized path planning for robot navigation. In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. [S.l.: s.n.], 2002. v. 3, p. 2383 – 2388 vol.3.
- CARACCIOLO, L.; LUCA, A. D.; IANNITTI, S. Trajectory tracking control of a four-wheel differentially driven mobile robot. In: *IEEE International Conference on Robotics & Automation*. Detroit, Michigan: [s.n.], 1999.
- CHOSSET, H.; BURGARD, W.; HUTCHINSON, S.; KANTOR, G.; KAVRAKI, L. E.; LYNCH, K.; THRUN, S. book. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. [S.l.]: MIT Press, 2005.
- DONALD, B.; XAVIER, P.; CANNY, J.; REIF, J. Kinodynamic motion planning. *J. ACM*, ACM, New York, NY, USA, v. 40, n. 5, p. 1048–1066, 1993. ISSN 0004-5411.
- Egerváry Research Group on Combinatorial Optimization. *LEMON Graph Library*. 2010. Disponível em: <<http://lemon.cs.elte.hu/trac/lemon>>.
- FERGUSON, D.; STENTZ, A. Anytime rrts. In: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*. [S.l.: s.n.], 2006. p. 5369 – 5375.
- GERKEY, B. P.; VAUGHAN, R. T.; HOWARD, A. The player/stage project: Tools for multi-robot and distributed sensor systems. In: *In Proceedings of the 11th International Conference on Advanced Robotics*. [S.l.: s.n.], 2003. p. 317–323.
- KOZLOWSKI, K.; PAZDERSKI, D. Modeling and control of a 4-wheel skid-steering mobile robot. *International Journal of Applied Mathematics and Computer Science*, v. 14, n. 4, p. 477–496, 2004.
- KRISHNAMURTHY, D. A. *Modeling and simulation of skid steered robot Pioneer - 3AT*. Dissertação (Mestrado) — Florida State University College of Engineering, 2008.
- LARSEN, E.; GOTTSCHALK, S.; LIN, M. C.; MANOCHA, D. Fast distance queries with rectangular swept sphere volumes. In: *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. [S.l.: s.n.], 2000. v. 4, p. 3719–3726 vol.4.
- LAVALLE, S. M. *Planning algorithms*. [S.l.]: Cambridge University Press, 2006.
- LAVALLE, S. M.; KUFFNER, J. J. Rapidly-exploring random trees: Progress and prospects. In: *Algorithmic and Computational Robotics: New Directions*. [S.l.: s.n.], 2000. p. 293–308.
- LAVALLE, S. M.; KUFFNER, J. J. Randomized kinodynamic planning. *The International Journal of Robotics Research*, v. 20, n. 5, p. 378–400, May 2001.

NOURANI-VATANI, N.; BOSSE, M.; ROBERTS, J.; DUNBABIN, M. Practical path planning and obstacle avoidance for autonomous mowing. In: *In Proc. of the Australasian Conference of Robotics and Automation*. [S.l.: s.n.], 2006.

RODRIGUEZ, S.; TANG, X.; LIEN, J.-M.; AMATO, N. An obstacle-based rapidly-exploring random tree. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. [S.l.: s.n.], 2006. p. 895 –900. ISSN 1050-4729.

VALLEJO, D.; JONES, C.; AMATO, N. M. An adaptive framework for single shot motion planning: A self-tuning system for rigid and articulated robots. In: *In Proceedings of the IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2001. p. 21–26.

VAZ, D. Alves Barbosa de O.; INOUE, R. S.; GRASSI, V. Kinodynamic Motion Planning of a Skid-Steering Mobile Robot Using RRTs. In: *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*. [s.n.], 2010. p. 73–78. Disponível em: <<http://dx.doi.org/10.1109/LARS.2010.27>>.