UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CLEBER SOUZA UGHINI

Criando Roadmaps a partir de Estados de Configuração Uniformemente Distribuídos

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Luciana Porcher Nedel
Orientadora

Porto Alegre, setembro de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Ughini, Cleber Souza

Criando Roadmaps a partir de Estados de Configuração Uniformemente Distribuídos / Cleber Souza Ughini. – Porto Alegre: PPGC da UFRGS, 2007.

55 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientadora: Luciana Porcher Nedel.

1. Mapas de caminhos. 2. Roadmap. 3. Planejamento de movimento. 4. Navegação 3D. 5. Computação gráfica. I. Nedel, Luciana Porcher. II. Título.

AGRADECIMENTOS

Agradeço à minha orientadora, Luciana Porcher Nedel por toda a supervisão, apoio, aulas e discussões que ajudaram a gerar e guiar esse e outros trabalhos durante o período em que estive cursando o mestrado.

Agradeço também aos outros professores do grupo de computação gráfica: Carla, Manuel e João, pelas aulas, revisões, conversas e exemplos que contribuiram para a minha formação como pesquisador e profissional.

Agradeço aos meus colegas de mestrado: Francisco, Fausto e Andréia pelo companheirismo e apoio sempre presente em trabalhos e discussões que surgiram durante o curso. Agradeço também aos outros membros do grupo de pesquisa: Bruno, Leandro, Fábio, Marcus, Marcos, Renato, Rodrigo, André, Carlos, Rafael, Vitor, Giovane, Dalton e Marta pela ajuda dada, dúvidas respondidas e pela criação de um ambiente favorável ao desenvolvimento do meu trabalho, assim como pelas comemorações, churrascos e jogos de futebol organizados.

Agradeço ao CNPQ pela bolsa e recursos fornecidos e a todos os profissionais do PPGC e do Instituto de Informática que ajudam a manter toda a infraestrutura que me possibilitaram desenvolver esse trabalho.

Agradeço à minha noiva, Sílvia Fischmann Osorio a quem amo muito e que me motivou desde o início para a conclusão desse trabalho. Agradeço à minha mãe, Eli Souza Ughini, pelo apoio e exemplo de vida e perseverança. Agradeço aos meus irmãos Gilbeli e Eduardo, pelo companheirismo e tantos momentos de alegria. Agredeço por fim a todos os meus amigos pelo pouco ou muito que levo comigo de cada um deles, e o que me torna cada dia mais feliz.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CG	Computação Gráfica
DOF	Degrees of Freedom
FSM	Finite State Machine
RRT	Rapid-exploring Random Tree
PRM	Probabilistic Roadmaps
ADRM	Adaptative Deterministic Roadmap
EC	Estado de Configuração

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A geração de bons movimentos em tempo real para corpos com muitos graus de liberdade ainda é um desafio. Uma quantidade elevada de graus de liberdade aumenta de forma exponencial a quantidade de posições diferentes que um determinado corpo pode obter. Fazer uso dessa quantidade de possibilidades para gerar movimentos complexos pode ser extremamente útil para planejamento de movimentos de robôs ou personagens virtuais, porém incrivelmente caro em termos computacionais.

Existem muitos algoritmos que se baseiam no uso de mapas de caminhos (chamados *roadmaps*) para trabalhar com corpos com muitos graus de liberdade. Um *roadmap* funciona como uma coletânea de poses de um corpo interligadas entre si, onde cada ligação representa uma possibilidade de transição livre de colisões. Geralmente as técnicas que utilizam *roadmaps* usam abordagens determinísticas ou aleatórias para atingir o objetivo. Através de métodos determinísticos é possível explorar de forma mais uniforme o espaço de configuração, garantindo uma melhor cobertura e qualidade do roadmap. Já as abordagens aleatórias, geralmente permitem um melhor desempenho e, principalmente, tornam viáveis a aplicação de uma solução para corpos com muitos graus de liberdade.

Neste trabalho é proposto um método determinístico adaptável para a geração de *roadmaps* (ADRM) que provê uma cobertura adequada do espaço de configuração em um tempo perfeitamente aceitável em comparação a outros métodos. Para obter isso, é feita em primeiro lugar uma classificação de todos os DOFs do modelo e, então, essa classificação é usada como parâmetro para decidir quantas amostras serão geradas de cada DOF. A combinação entre as amostras de todos os DOFs gera a quantidade total de amostras.

Para validação do novo método foram executados diversos testes em ambientes distintos. Os testes foram avaliados através da comparação com outras técnicas existentes, em quesitos como tempo de geração e cobertura do espaço de configuração. Os resultados demonstram que o método atinge uma cobertura do espaço de configuração muito boa, em um tempo aceitável.

Palavras-chave: Mapas de caminhos, roadmap, planejamento de movimento, navegação 3D, computação gráfica.

Creating Roadmaps from Uniform Distributed Configuration States

ABSTRACT

The creation of good real time movements for bodies with many degrees of freedom (DOF) still remains a challenge. A great amount of DOFs increase, in an exponential way, the quantity of different positions that a body can assume. Making use of that amount of possibilities to generate complex movements can be useful for planning robots' movements or even to animate virtual characters, however it is extremely expensive in computational terms.

There are many algorithms that are based on the use of roadmaps to work with bodies with many degrees of freedom. A roadmap works as a collection of valid body's positions interconnected, where each connection represents a possibility of a transaction free of collisions. Usually, the techniques which make use of roadmaps follow deterministic or probabilistic approaches to get to the objective. Trough deterministic methods it is possible to explore in a more uniform way the configuration's space, assuring a better covering and quality of the roadmap. Therefore, probabilistic (or random) approaches allow a better performance and, mainly, make possibly the application of a solution for bodies with higher degrees of freedom.

This work proposes a deterministic method applicable to roadmaps generation (ADRM) which provides an adequate covering of the configuration's space in a completely acceptable time range comparing to other rates. To achieve this goal, first of all a classification of all of the DOFs of the model is made and, then, this classification is used as a parameter to decide how many samples will be generated of each DOF. The combining between the samples of all of the DOFs generates the total amount of samples.

To validate the new method, several tests were executed at different environments. The tests were evaluated trough the comparison with other existents techniques, using criteria like the time spent in generating a roadmap and covering of the space of configuration. The results show us that the method achieves a satisfactory covering of the space configuration in an acceptable time range.

Keywords: roadmap, path planning, 3D navigation, computer graphics.

1 INTRODUÇÃO

Encontrar o melhor caminho ou simplesmente um caminho qualquer entre dois pontos tem sido tema de diversos estudos. A dimensão das abordagens cresce consideravelmente quando a ilustração de um ponto pode representar um complexo esqueleto com diversas juntas, cada uma com n graus de liberdade, cada qual com um valor associado. Ou um veículo em um espaço tri-dimensional onde suas momentâneas forças de atuação representam um estado em um mapa de possibilidades (GO; VU; KUFFNER, 2004).

A necessidade de encontrar caminhos que conduzam um corpo de uma posição para outra é frequente em diversas áreas: na animação é o ponto principal, onde um movimento deve atender a requisitos exigentes, procurando passar o máximo de naturalidade a quem assiste; quando usado na área de jogos, o planejamento de movimentos não precisa sempre buscar o máximo realismo possível, entretanto deve ser rápido, permitindo a interação em tempo real com o jogador; já na robótica, os movimentos devem ser precisos, minuciosos, já que erros podem vir a provocar acidentes graves, danos ao robô, ou até mesmo perda da produção de um bem caro.

1.1 Abordagens para Planejamento de Caminhos

Pesquisas recentes na área de animação de personagens têm se concentrado na geração de movimentos realistas usando captura de movimentos, onde um ator real produz uma série de ações determinadas para serem gravadas e armazenadas para depois serem mapeadas em um personagem (KALLMANN et al., 2003; LEE et al., 2002; CHOI; LEE; SHIN, 2003; LAU; KUFFNER, 2004). Cada movimento capturado pode ser encadeado com outros, gerando assim uma ação em um nível maior. Uma máquina de estados então pode ser criada para encadear uma ação à outra, permitindo que um personagem execute-as livremente, respeitando apenas as regras de transições, como pode ser visto na Figura 1.1.

Contudo, dados de captura de movimentos são difíceis de serem reutilizados, e sempre deve ser feita uma captura para cada ação que for prevista, com o agravante de que todas ações devem ser previstas. Após a captura dos dados ainda é necessário fazer um longo e cuidadoso processamento nestes para ajustar suas transições e aplicá-los às devidas situações, conforme se encaixarem.

Entre outras alternativas na área de planejamento de caminhos, muitas soluções trazem bons resultados, como campos potenciais numéricos (BARRAQUAND; LATOMBE, 1989). A idéia do uso de campos potencias consiste em definir uma força de atração para um ponto objetivo que fará com que o personagem ou um robô seja atraído em direção a ele. Em paralelo também existem forças de repulsão emitidas pelos objetos e paredes para evitar que o personagem tenha uma colisão contra esses. A Figura 1.2 mostra um

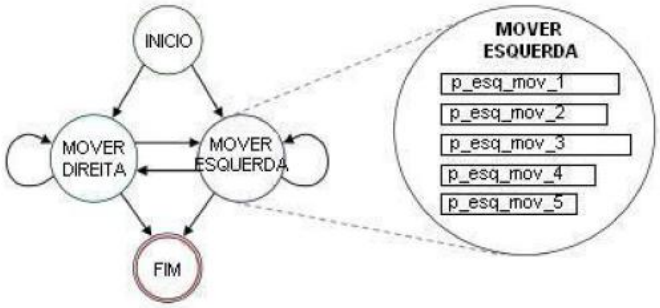

Figura 1.1: Máquina de estados finitos com duas ações simples representando opções de uma caminhada.

exemplo da aplicação dessas forças.

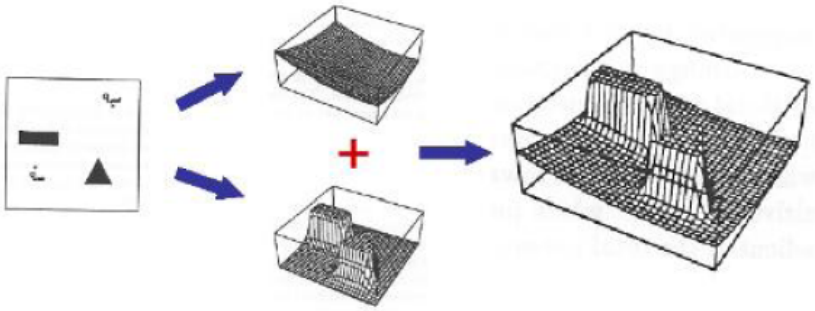

Figura 1.2: Campo Potencial definido por forças de atração e repulsão (LATOMBE, 1991)

Observa-se que os obstáculos representados no cenário se tornam regiões com alto potencial, assim repelindo o personagem, já o ponto objetivo define um declive, induzindo o personagem a ele.

A solução de campos potenciais ainda apresenta um problema de mínimos locais, que são regiões no mapa de forças onde a força de atração e a de repulsão se anulam, gerando uma posição onde o personagem ficaria parado. Existem diversas abordagens que visam tratar isso. Apesar dos ótimos resultados para planejamento de movimentos em mapas, a aplicação de campos potencias em corpos articulados não é trivial e ainda não existem bons resultados de animações nessa categoria.

Em um corpo, articulado ou não, a cada possibilidade de movimento, seja de rotação ou translação, é dado o nome de grau de liberdade, mais comumente referenciado como DOF (Degree Of Freedom). Para cada valor diferente que for dado a cada DOF de um corpo, esse corpo poderá assumir uma posição e/ou localização diferente no seu espaço de configuração. O espaço de configuração de um corpo é o conjunto de todas as posições possíveis que um corpo pode assumir. Cada uma dessas posições dentro do espaço de configuração é chamada de estado de configuração, referenciado nesse texto por EC. Quando um ou mais DOFs do corpo em questão não possuir valores discretos, a quantidade de ECs dentro de um espaço de configuração é infinita.

Um mapa de caminhos (mais conhecido por *roadmap*) é um conjunto de estados de configuração de um corpo onde alguns deles (ou todos) estão interconectados entre si. Cada conexão entre dois estados representa uma forma válida, livre de colisão, de mudar de um EC ao outro (Figura 1.3).

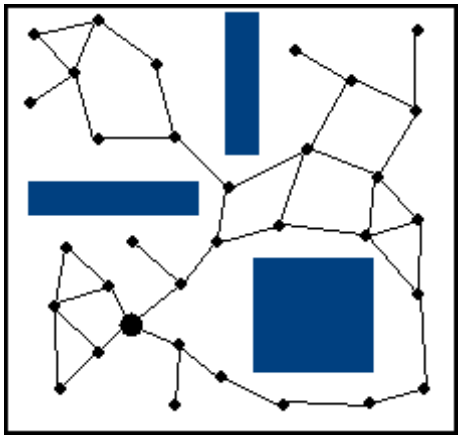

Figura 1.3: Mapa de caminhos plano para resolução de problemas de planejamento de movimentos translacionais 2D.

A geração de um *roadmap* pode explorar quase todas as possibilidades de configuração de um corpo no espaço, levando em consideração todos seus graus de liberdade. Para uma aplicação interativa, onde o próprio ambiente pode assumir configurações diferentes, não seria possível prever todas suas configurações dado uma grande dimensão de possibilidades (LAU; KUFFNER, 2004).

Entre as técnicas para gerar *roadmaps* hoje em dia trabalhadas, poucas focam uma distribuição uniforme dos estados de configuração, gerando os novos estados aleatoriamente até o algoritmo atingir um estado de satisfação (um número de ECs específico). Em geral esses algoritmos precisam de muitos ciclos de iterações para atingir esse estado de satisfação, e com poucos ciclos, acabam não conseguindo um *roadmap* abrangente o bastante para explorar o ambiente adequadamente.

Roadmaps podem ser gerados para resolver um ou múltiplos casos de planejamento de movimento. Quando ele é feito para apenas um caso, trata-se de um *roadmap* de consulta simples (*single-query roadmap*) e toma como parâmetro um estado de configuração de início e um estado de configuração objetivo. Após o processamento, retorna um caminho como resultado e é descartado em seguida. Quando o *roadmap* serve para tratar mais de um caso, ele toma apenas um estado de configuração inicial como parâmetro e tenta expandir ao máximo as possibilidades de configuração interligando-as. O *roadmap* é armazenado e sempre que uma consulta precisar ser feita, basta passar um EC inicial e um objetivo que o caminho é encontrado através dele. Esse tipo de *roadmap* é geralmente referido como *roadmap* para múltiplas consultas (*multiple-querys roadmap*).

O custo computacional envolvido na geração dos *roadmaps* para múltiplas consultas geralmente é alto, tornando-os inviáveis de serem gerados em tempo real. Os *roadmaps* para consulta simples possuem algoritmos de geração mais rápidos e eventualmente podem ser gerados em tempo real. A complexidade do corpo de acordo com seu número de graus de liberdade também influi diretamente no tempo de construção do *roadmap* (KALLMANN; BARGMANN; MATARIC, 2004). Para ambientes relativamente estáticos, os *roadmaps* podem ser gerados em uma etapa de pré-computação e então utilizados posteriormente com um bom desempenho.

O uso de *roadmaps* é amplo em aplicações para ambientes 3D (KALLMANN et al., 2003; CHOI; LEE; SHIN, 2003; KALLMANN; BARGMANN; MATARIC, 2004), incluindo animação de humanóides em aplicações gráficas, de forma a obter movimentos realistas (KALLMANN et al., 2003). Seu uso também é muito freqüente para controle de

robôs no mundo real que executam as mais diversas tarefas.

1.2 Objetivos

O objetivo principal desse trabalho é apresentar uma nova abordagem desenvolvida para gerar *roadmaps*. A nova abordagem deve ser aplicável a corpos com qualquer tipo e número de DOFs em tempo aceitável. Além disso, esse trabalho também tem por objetivo fazer um comparativo entre essa e outras abordagens, como forma de gerar resultados que validem seu uso.

A principal estratégia adotada para gerar o *roadmap* é trabalhar de forma a gerar os ECs deterministicamente, tentando tirar o máximo proveito do desempenho. O método de geração de novos estados de configuração considera todo o espaço de configuração do corpo, planejando a construção de todos os estados de configuração de tal forma que garantidamente eles não sejam redundantes e mantenham uma distribuição uniforme no espaço de configuração. A abordagem baseada em mapas de caminhos foi escolhida devido à sua aplicabilidade a corpos articulados e capacidade de gerar boas animações em tempo real após sua computação.

1.3 Organização

Este trabalho está organizado em 5 capítulos incluindo esta introdução. No Capítulo 2 é feito um levantamento de trabalhos que usam mapas de caminhos, incluindo os trabalhos que serviram de inspiração para o desenvolvimento deste e que têm maior impacto na sua área. Entre eles são apresentados os *Probabilistic Roadmaps*, técnica introdutora ao conceito de mapas de caminhos, e outras que surgiram a partir de sua evolução.

No Capítulo 3 o novo método desenvolvido é detalhado como solução para planejamento de movimentos e caminhos. A técnica é descrita para cada passo desde a geração do *roadmap* (o que inclui etapas de amostragem, validação e conexão) até a solução de casos de planejamento relacionados.

O Capítulo 4 apresenta alguns cenários de teste criados. Para executar os testes foram implementadas a técnica desenvolvida e outras duas. Os resultados obtidos mostram comparações entre o desempenho das técnicas e também resultados visuais dos roadmaps gerados para cada uma. O Capítulo 5 conclui este trabalho com comentários e sugestões sobre trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Recentemente, diversos algoritmos não-determinísticos fazendo uso de um *roadmap* pré-computado foram propostos. O pioneiro foi o algoritmo de mapa de caminhos com amostragem aleatória (ou Probabilistic Roadmaps - PRM) (KAVRAKI; LATOMBE, 1994) onde é introduzida a nova abordagem para planejamento de movimento de robôs com vários graus de liberdade. A partir daí outros algoritmos foram propostos mantendo o uso de um *roadmap* para encontrar caminhos. Em um trabalho de LaValle (LAVALLE, 1998) é feita uma proposta que usa critérios aleatórios para explorar os possíveis estados de configuração, com um desempenho consideravelmente bom para ser aplicado a corpos com muitos graus de liberdade. As duas técnicas abriram precedentes para diversos outros trabalhos modificando ou estendendo o uso dessas duas.

Como efeito colateral o enfoque determinístico perdeu força nas pesquisas e tem sido deixado de lado por geralmente não conseguir atender bem a corpos com muitos graus de liberdade (LAVALLE, 2006). Em um trabalho mais recente (BRANICKY et al., 2001) é feita uma análise de desempenho para métodos determinísticos contra o tradicional PRM, como forma de avaliar o rumo que está sendo tomado.

Neste capítulo serão detalhadas as técnicas mais tradicionais e algumas de suas variações. Na Seção 2.1 é apresentado o algoritmo de mapa de caminhos com amostragem aleatória, mais conhecido por sua sigla PRM. Muitas variações foram propostas em cima dessa técnica e algumas delas são apresentadas em suas subseções. Na Seção 2.2 é introduzida a técnica de exploração rápida em árvores de crescimento aleatório, popularmente conhecida por RRT (*Rapidly-Exploring Random Trees*), e algumas modificações propostas sobre elas. A Seção 2.3 apresenta algumas soluções de natureza determinística recentes, contrabalançando as argumentações probabilísticas.

2.1 Mapa de Caminhos com Amostragem Aleatória

Essa técnica discutida por Kavraki e Latombe (KAVRAKI; LATOMBE, 1994) introduziu a possibilidade de usar um mapa de estados que é previamente calculado para, a partir deste, encontrar um caminho válido entre duas configurações. O pré-processamento é feito apenas uma vez para um dado ambiente estático, gerando um conjunto de pontos que representam configurações possíveis do corpo a ser trabalhado nesse ambiente. A geração das configurações é feita aleatoriamente partindo de uma configuração inicial e sendo cuidadosamente selecionada para não permitir configurações não-válidas para o ambiente. Após o mapa de possibilidades ter sido gerado, o custo para gerar um caminho válido entre duas configurações é baixo, o que torna o algoritmo atraente para ser usado em robôs com muitos graus de liberdade.

Para gerar o *roadmap* são seguidas quatro etapas descritas a seguir.

1. Geração dos nodos: configurações aleatórias são geradas para o corpo considerando o espaço livre no ambiente a ser trabalhado. Após ser gerada uma configuração para cada grau de liberdade do corpo, são testadas colisões contra o ambiente para verificar se a configuração é válida. Deve ser tomado algum cuidado para que as configurações não se repitam ou fiquem muito próximas uma da outra. Cada configuração válida é chamada de nodo.

2. Interconexão dos nodos gerados: usando um planejador simples, que apenas evita colisões, os nodos são verificados um a um e ligados aos k nodos mais próximos (k é dado como parâmetro). Isso gera uma rede de conexões que deve abranger todos os nodos.

3. Refinamento de regiões: no fim da segunda fase, tem-se um grafo que pode ter regiões mais concentradas e outras mais esparsas. As regiões mais concentradas podem conter caminhos com maior custo em relação ao número de nodos percorridos para chegar de uma configuração à outra. O objetivo dessa fase é reduzir esses custos criando mais conexões de longo alcance dentro dessas regiões, assim também podendo unir regiões antes desconexas.

4. Redução do número de componentes: Após a execução da etapa 3, o número de conexões e nodos pode ser excessivo e em alguns caminhos desnecessários. Na tentativa de redução de caminhos, são verificados pares de nodos próximos e se estão conectados. Caso sim, eles são fundidos e sua conexão é armazenada. O processo é repetido algumas vezes para cada par de componentes. No final desse processo tem-se o *roadmap*.

A Figura 2.1 mostra um exemplo de resultado que pode-se esperar de um estado de configuração e quando se mostra ao mesmo tempo todos os ECs de um *roadmp* para um modelo de 7 DOFs.

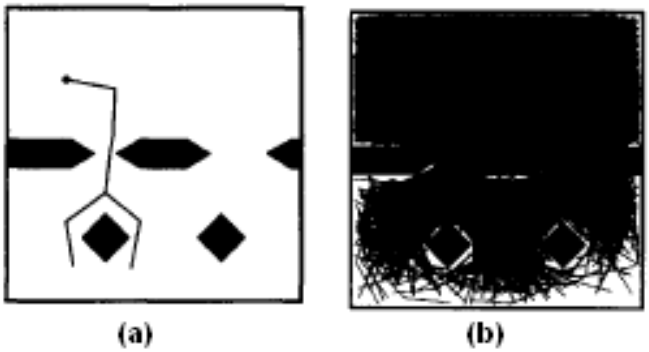

Figura 2.1: PRM resultante: (a) Um estado de configuração pertencence ao *roadmap*. (b) Todos os estados de configuração do roadmap sobrepostos. (KAVRAKI; LATOMBE, 1994).

Com o *roadmap* gerado a tarefa de encontrar o caminho fica bem rápida. Dadas duas configurações *O* e *G*, sendo a primeira a configuração inicial e a segunda a configuração objetivo, primeiro deve-se unir a configuração *O* à configuração mais próxima no *roadmap*. Os nodos são ordenados pela distância de *O*. Assim se o primeiro não for possível

de conectar, parte-se para o próximo da lista. O critério de distância é dado por alguma função definida que considera os limites máximos e mínimos de cada grau de liberdade. Caso nenhuma configuração seja atingível, executa-se um deslocamento aleatório da configuração *O* e então tenta-se unir novamente. A magnitude do deslocamento pode ser dada como parâmetro. Esse processo pode ser executado algumas vezes até encontrar uma ligação com o *roadmap* se necessário. Analogamente, o mesmo procedimento descrito é executado para a configuração *G*.

Considere *A* e *B* sendo nodos pertencentes ao *roadmap* R conectados a *O* e *G*. Uma rápida busca no *roadmap* encontra um caminho entre *A* e *B*, e portanto, entre *O* e *G*. Para fins de realismo, pode-se suavizar os movimentos seguindo o *roadmap* usando alguma técnica padrão de suavização, como interpolação de Bézier ou interpolação ponderada.

Caso o algoritmo não encontre nenhum caminho para unir os pontos de origem e destino, retorna falha, considerando um dos pontos inatingível pelo *roadmap*.

A técnica de Mapa de Caminhos com Amostragem Aleatória (KAVRAKI; LATOMBE, 1994) se tornou muito popular e passou a ser amplamente usada. Com o tempo, diversas variações da idéia original foram surgindo, seja para melhorar seu desempenho ou adaptar seu uso em situações mais específicas. A seguir, são apresentadas duas variações como exemplo.

2.1.1 PRM Baseado em Critérios de Visibilidade

O PRM baseado em critérios de visibilidade (*Visibility-Based Probabilistic Roadmap*) geralmente referenciado por VB-PRM (NISSOUX; SIMéON; LAUMOND, 2000) surge como uma variação ao tradicional *Probabilistic Roadmap* apresentado anteriormente. A idéia básica é explorar o espaço livre configurando-o em domínios de visibilidade, gerando *roadmaps* menores.

No PRM, o algoritmo gera configurações livres de colisão aleatoriamente e depois tenta uní-las usando um método de planejamento de movimentos qualquer. Nesta abordagem, o algoritmo considera aspectos de visibilidade para trazer melhores resultados. Enquanto cada configuração válida gerada pelo PRM é integrada ao *roadmap*, o VB-PRM mantém apenas configurações que não visíveis entre si, ou seja, que não são passíveis de uma transição direta de uma à outra. Após isso outras configurações são então geradas, de forma a conectar as anteriores. Dessa forma, essa abordagem acaba por gerar um *roadmap* com um pequeno número de nodos.

Melhor descrevendo o funcionamento da técnica: considere um espaço de configuração C, o espaço de trabalho livre (sem obstáculos) nesse ambiente é denotado por C_{free}, e seu espaço complementar C_{obs}. Considere uma função $L(x, x')$ que calcula um caminho reto (uma linha reta ou uma interpolação linear) entre as configurações x e x' sem passar por C_{obs}. Define-se o domínio de visibilidade $Vis(x)$ da configuração x como sendo:

$$Vis(x) = \{x' \in C_{free} \ \ tal\ que \ \ L(x, x') \in C_{free}\} \tag{2.1}$$

Cada EC x gerado dessa forma é chamado de *guard*. Em um primeiro passo geram-se *guards* suficientes para cobrir C_{free}, permitindo que os domínios de visibilidade das *guards* possam se interseccionar. O processo de amostragem das *guards* deve ter um número de interações limite, devido ao fato do número de amostras ser limitado pelo critério de visibilidade. Após a geração das *guards*, o processo de amostragem recomeça. Nesta segunda etapa, quando um EC for amostrado dentro da área de intersecção de duas ou mais *guards* ele é armazenado para fazer a ligação entre elas. A esse EC é dado o nome de *connection*. A Figura 2.2 ilustra melhor esse cenário.

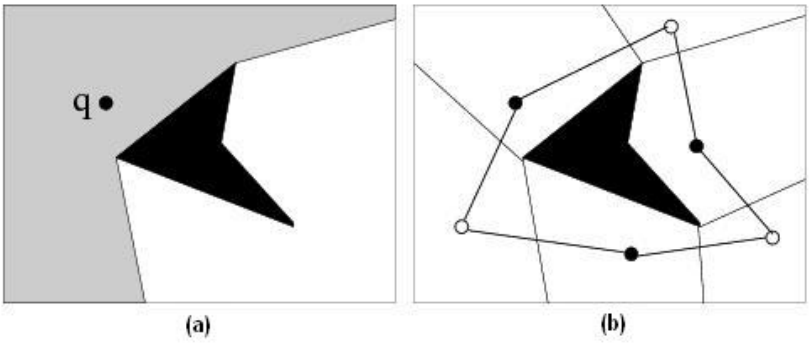

Figura 2.2: Domínios de visibilidade: (a) a área hachurada representa o domínio de visibilidade da configuração q, (b) *guards* (pontos pretos) ligados por *connections* (pontos brancos). O objeto ao centro representa C_{obs}. (NISSOUX; SIMéON; LAUMOND, 2000).

Para cenários densos, onde C_{obs} é grande, pode-se ter dificuldades para definir o número máximo de interações de forma que ele seja suficiente para cobrir C_{free} adequadamente. Para uso em corpos articulados esse algoritmo pode não prover resultados satisfatórios, já que deve-se definir um critério de visibilidade que pode não ser muito trivial.

2.1.2 *Adaptação de PRM para Mapas Imprecisos*

Nos casos tradicionais onde o PRM é aplicado, o ambiente inteiro e correto deve ser conhecido. Assim todo o cálculo do *roadmap* é feito para todas as regiões do ambiente. Nessa variação da técnica desenvolvida por Missiuro (MISSIURO; ROY, 2006), assume-se que o ambiente não é totalmente conhecido, ou que ele é conhecido, mas não se sabe com precisão sobre os obstáculos.

A técnica trabalha adicionando um filtro na etapa que gera os nodos do *roadmap* e um filtro na etapa de planejamento de movimento. A função do filtro na primeira etapa é detectar quais estados de configuração possuem uma grande probabilidade de entrar em colisão. Essa função toma como parâmetro o próprio estado de configuração, o mapa do ambiente conhecido e um parâmetro de densidade sobre o que se conhece do ambiente. Todo estado de configuração gerado é testado contra essa função.

Na etapa de planejamento de movimento também é aplicado um filtro para escolher os caminhos do *roadmap*. Esse filtro visa escolher os estados de configuração que possuem a menor probabilidade de colisão com o ambiente, gerando assim não o caminho mais curto dentro do *roadmap*, mas aquele que pode ser considerado o mais seguramente livre de colisões.

2.2 *Exploração Rápida em Árvores de Crescimento Aleatório*

Em um trabalho de La Valle (LAVALLE, 1998) é introduzida a técnica conhecida como *rapidly-exploring random trees* (RRT). O foco do desenvolvimento das RRTs é para melhor tratar casos não-holonômicos e um grande número de graus de liberdade.Um movimento não-holonômico é onde geralmente há uma continuidade dos movimentos de um corpo (como deslocamento de carros e outros veículos sobre rodas). Uma RRT é iterativamente expandida, aplicando comandos que controlam o corpo para diversas direções de forma contínua, ao contrário do PRM que trabalha conexões ponto a ponto isoladamente.

Para gerar o *roadmap*, considere um ambiente X a ser explorado. Denota-se o espaço livre desse ambiente como X_{free}, cujo complemento é o espaço ocupado por obstáculos X_{obs}. Partindo de um estado de configuração inicial x_{init}, o algoritmo deve designar diferentes valores para os graus de liberdade a cada iteração, respeitando um valor máximo de incremento em relação a x_{init}. O valor desse incremento pode ser dado como parâmetro à aplicação. Cada nova configuração gerada deve ser testada contra X_{obs} para evitar colisões, cabendo uma nova tentativa a cada configuração inválida.

Para garantir um resultado de natureza não-holonômica, uma equação na forma $x' = f(x, u)$ garante a coerência entre o estado anterior e o que se sucede. O parâmetro x pode ser visto como o estado anterior e u como o comando dado ao corpo na configuração atual, para gerar a próxima configuração x'. Para gerar uma RRT de natureza holonômica basta definir x como sendo a configuração atual, sendo u livre das restrições de continuidade. Uma diferença entre as duas formas de tratar a expansão da RRT pode ser observado na Figura 2.3. Em ambos os casos, x_{init} é designado por O.

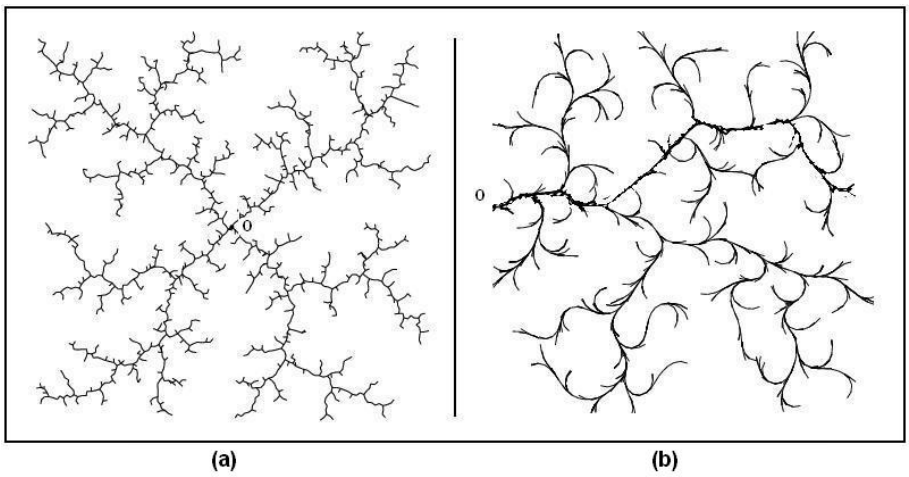

Figura 2.3: RRT: (a) resultado holonômico, (b) resultado não-holonômico. (LAVALLE, 1998).

Observa-se no resultado não-holonômico a inexistência de mudanças bruscas de direção respeitando-se sempre uma curvatura máxima a partir do estado de origem.

As RRTs apresentam uma série de propriedades interessantes que as tornam ideais de serem usadas em diversas situações. Entre elas tem-se:

1. A expansão da RRT é fortemente direcionada para porções mais inexploradas do espaço livre. O que torna sua cobertura do espaço livre mais completa.

2. A distribuição dos vértices da RRT respeita densidades progressivas, levando a um comportamento consistente. Evitando que regiões fiquem com muitos nodos e outras poucos.

3. Uma RRT pode ser considerada bem completa, sob condições gerais. Atendendo a casos holonômicos e não-holonômicos.

4. O algoritmo da RRT é relativamente simples, o que o torna fácil de entender e implementar e também facilita análises de desempenho.

5. A RRT sempre permanece conectada a todos os nodos gerados, mesmo que o número de arestas entre os nodos seja mínimo.

6. A RRT pode ser considerada e implementada como um módulo de planejamento de movimento que pode ser adaptado e incorporado a uma grande variedade de sistemas de planejamento.

7. Todo o algoritmo de planejamento de movimento pode ser implementado sem a necessidade de conhecimento aprofundado da aplicação destino e necessidades de mudança durante seu uso, aumentando a aplicabilidade das RRTs.

Posteriormente foram propostas diversas extensões para o algoritmo base de RRT. Algumas mais populares e recentes são mostradas a seguir.

2.2.1 RRT-Connect

Essa variação introduzida por Kuffner (KUFFNER; LAVALLE, 2000) implementa a idéia base da RRT que deve partir de um EC inicial, adicionando uma segunda RRT que parte do EC destino do corpo. Como parâmetro para guiar o crescimento das duas árvores, é dado às duas a direção crescente da outra, assim elas buscam sempre se encontrar. O processo retorna uma solução quando as duas árvores crescentes se encontram, indicando que existe um caminho entre os dois ECs passados como parâmetro. A Figura 2.4 ilustra o crescimento das duas árvores.

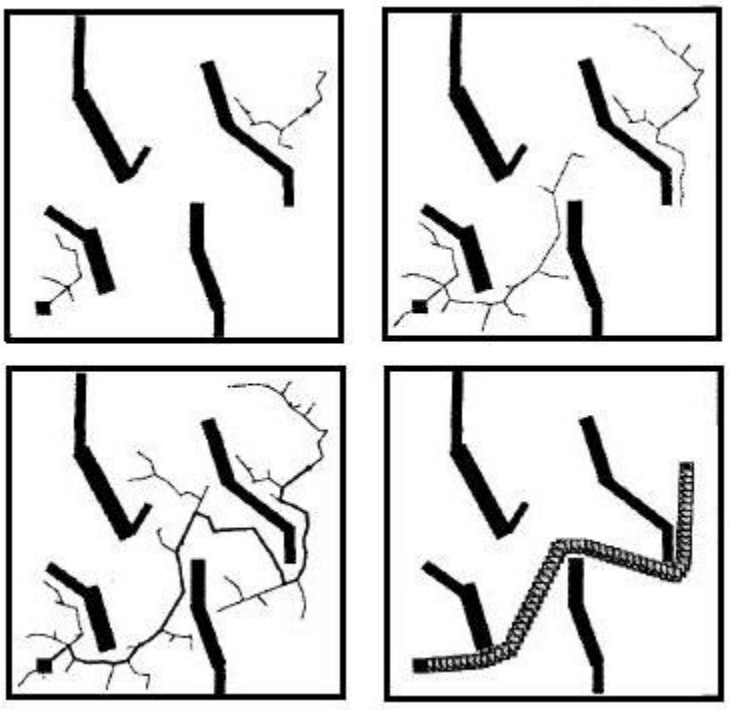

Figura 2.4: RRT-connect encontrando um caminho entre dois pontos (KUFFNER; LAVALLE, 2000).

Resultados obtidos com esse método geram um mapa e encontra o caminho entre dois estados de configuração muito mais rápido que a técnica original, contudo seu uso acaba sendo muito mais restrito que o da RRT tradicional, devendo ser aplicado apenas quando se tem um estado de configuração destino conhecido.

Uma vez gerado o *roadmap*, ele serve apenas para essa situação, o que o torna útil apenas para planejamento em tempo real, restringindo seu uso a corpos menos complexos, uma vez que corpos com muitos graus de liberdade tendem a exigir mais tempo de processamento. Outra restrição dessa variação, é que ela pode gerar problemas com movimentos não-holonômicos no momento de conectar as duas árvores em expansão.

2.2.2 RRTs com Domínios Dinâmicos

No trabalho desenvolvido por Yershova (YERSHOVA et al., 2005), é feita uma análise da expansão da RRT tradicional em relação aos diagramas de Voronoi, feita originalmente no trabalho de Lindemann (LINDEMANN; LAVALLE, 2004). A expansão de uma RRT para casos holonômicos, tende a selecionar nodos de tal forma que eles capturem o maior volume possível em um diagrama de Voronoi no espaço de configuração dado. A Figura 2.5 (a) mostra a RRT original dividida pelo diagrama de Voronoi. A Figura 2.5 (b) mostra apenas o espaço visível dentro do obstáculo.

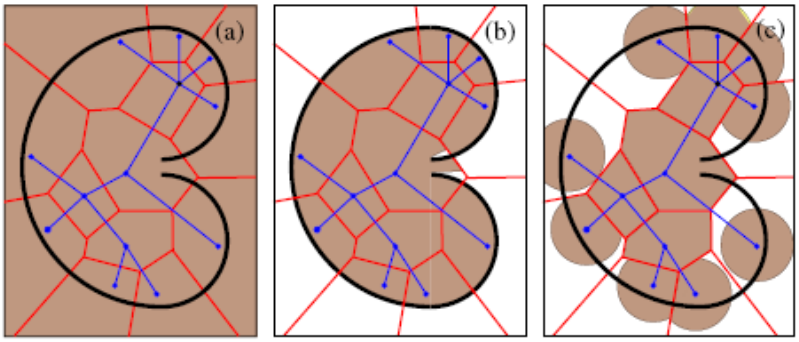

Figura 2.5: (a) RRT original, (b) regiões de Voronoi visíveis, (c) RRT com domínios dinâmicos (YERSHOVA et al., 2005).

A idéia dos domínios dinâmicos é restringir um raio máximo para as regiões que intersectam as bordas, formando um domínio menor que o que originalmente seria tomado, conforme visto na Figura 2.5 (c). Assim os espaços livres para as regiões de expansão permitem que o crescimento da árvore se direcione para as regiões mais propensas a conter o caminho não explorado.

2.2.3 RRT-blossom

Outra variação, introduzida por Kalisiak (KALISIAK; M. PANNE, 2006), associa o uso de campos potenciais ao crescimento da RRT. O mecanismo principal dos campos potencias está em estabelecer regiões de alto gradiente e regiões de baixo gradiente no mapa, fazendo com que o corpo acabe convergindo para o destino. Apesar de haver vários problemas associados à mínimos locais, existem métodos de contorná-los.

Na implementação da RRT-blossom, uma pequena diferença de gradientes é introduzida entre os pontos inicial e objetivo, para guiar o crescimento inicial da árvore. Conforme a árvore se expande e colide com obstáculos, valores altos de gradiente são atribuídos a esses pontos, para que eles não voltem a ser explorados. Dessa forma a árvore é guiada mais diretamente às regiões livres e não-exploradas do espaço de configuração.

A natureza desse algoritmo o torna muito mais útil a ser usado em ambientes estreitos (como túneis ou canos), onde a RRT tradicional perderia muito tempo com colisões. O

uso em situações não-holonômicas pode ser mantido, já que o algoritmo parte de um ponto inicial apenas, e o *roadmap* gerado pode vir a ser reutilizado para achar outros caminhos, para isso bastando não adicionar a diferença de gradientes inicial.

2.3 Abordagens Determinísticas

A construção de *roadmaps* de forma determinística (comumente chamados de *grids*[1] até então) esteve em alta nos anos 80. O brilho das novas técnicas que geravam amostragens aleatórias permitindo aplicação em robôs com números elevados de graus de liberdade acabou deixando as aplicações determinísticas um pouco ofuscadas. Entretanto ainda surgem pesquisas (BRANICKY et al., 2001) visando avaliar o real ganho de desempenho dado pelos métodos não-determinísticos. Em paralelo outros métodos determinísticos continuam a ser explorados como forma de tentar superar dificuldades conhecidas nas atuais formas de gerar *roadmaps*.

2.3.1 Quasi-Randomized Path Planning

Um estudo foi feito por um grupo de autores conceituados (BRANICKY et al., 2001) para propor uma forma de fazer a amostragem de modo determinístico usando a tradicional metodologia do PRM. A idéia se diferencia do PRM apenas na primeira etapa, onde os estados de configuração são gerados. A Q-PRM, como foi batizada, usou alguns métodos matemáticos conhecidos de distribuição uniforme ou pseudo-uniforme para calcular os estados de configuração do *roadmap*. Após ter os estados gerados, as etapas seguintes de união e planejamento de movimento foram igualmente aplicadas.

Quatro métodos de distribuição foram considerados para a primeira etapa: os conjuntos de Halton (HALTON, 1960); os conjuntos de Hammersley (HAMMERSLEY, 1960); as distribuições de malha, que visam gerar um conjunto de pontos onde a discrepância é máxima; as distribuições de Sukharev (SUKHAREV, 1971), que objetivam uma máxima distribuição dos pontos.

Entende-se por discrepância máxima o cuidado em gerar valores únicos para cada valor dos DOFs do corpo, de tal forma que não haja nenhuma relação de proporção linear, geométrica ou exponencial entre eles. Assim sendo, cada valor atribuído a cada grau de liberdade nunca deve se repetir. Já por distribuição máxima, entende-se que a preocupação recai na distância entre os ECs gerados. O objetivo então é distribuir o máximo possível o conjunto de ECs no seu espaço de configuração, de modo que nenhuma distância entre dois ECs seja menor que a distância entre outros (LAVALLE; BRANICKY; LINDEMANN, 2004). Um exemplo das referidas distribuições para $\Re^2$ pode ser visto na Figura 2.6 onde os diagramas de Voronoi são usados para salientar a distribuição.

Os resultados obtidos nos testes feitos indicaram que gerar os estados de configuração de forma determinística acabou sendo consideravelmente mais rápido que de forma aleatória, e a distribuição sobre o espaço de configuração também foi melhor, encontrando soluções boas mesmo para casos onde a abordagem aleatória não conseguia encontrar um caminho. Além disso, viu-se que com os métodos que visam máxima distribuição, o tempo de geração foi mais rápido que os que visam máxima discrepância. Contudo quanto à solução para planejamento de movimento, fica a dúvida se a discrepância máxima não gera melhores resultados.

[1] Os algoritmos determinísticos usados para fazer *roadmaps* datam desde 1935, a partir das seqüências de Van der Corput. Devido à sua distribuição regular em forma de grade para espaços $\Re^2$, eram chamados de *grid*. O uso do termo *roadmap* surgiu em 1994 com o desenvolvimentos do *probabilistic roadmaps*.

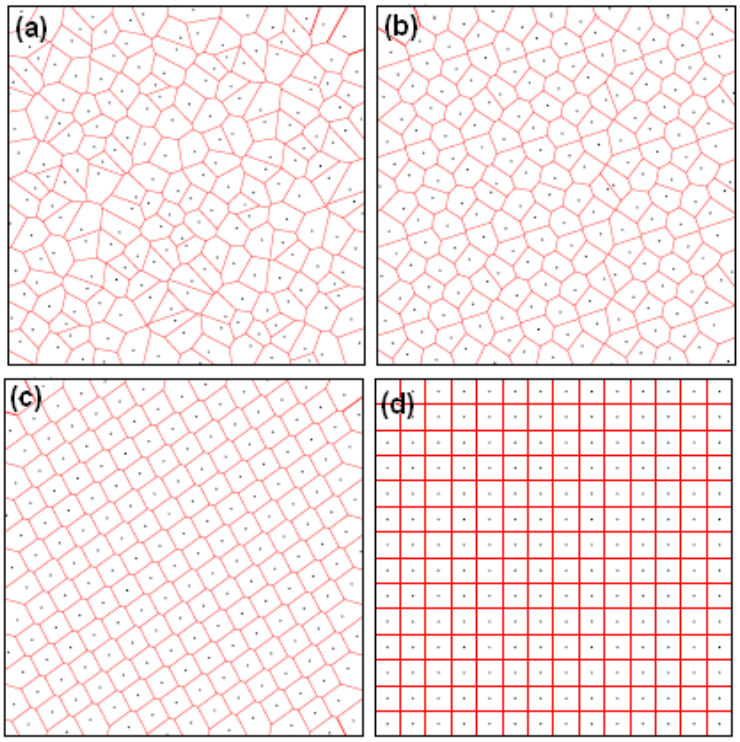

Figura 2.6: (a)conjunto de Halton; (b) conjunto de Hammersley; (c) malha; (d) conjunto de Sukharev (LAVALLE; BRANICKY; LINDEMANN, 2004).

Todos os testes e comparações foram realizados para o espaço $\Re^2$ e, por suposição, tomados como melhores mesmo para espaços de configuração maiores. Contudo, as distribuições uniformes (ou pseudo-uniformes) tem um crescimento exponencial em relação ao número de nodos quando se aumenta o número de graus de liberdade, o que é dado como principal argumento para defender o uso das distribuições aleatórias.

2.3.2 Estados de Configuração Distribuídos pela Superfície de Objetos

No trabalho desenvolvido por Amato (AMATO; WU, 1996), o espaço de configuração ocupado é ponto de partida para escolha dos estados de configuração. Para cada obstáculo contido no ambiente, é feita uma varredura em seu perímetro para escolher estados de configuração muito próximos aos objetos, mas que não estejam em estado de colisão. O método base consiste em calcular o centro de cada objeto do ambiente, e então escolher um ângulo de variação para aplicar aos DOFs. Com o cálculo da variação escolhida, é traçada uma linha do centro em direção à borda do objeto. Onde a linha interseccionar a borda é escolhido o estado de configuração (Figura 2.7 (a)).

De acordo com a distância da borda em relação ao centro, especialmente em objetos longos e finos, pode haver uma grande distância entre estados de configuração ao longo do perímetro. Para corrigir isso, são adicionadas variações intermediárias entre as linhas iniciais de acordo com sua distância da borda ao centro, tentando assim manter a distribuição mais uniforme (Figura 2.7 (b)).

Após a fase de geração dos estados de configuração, ainda é necessário fazer a união entre eles, testando cada EC gerado contra todos os outros. (Figura 2.7 (c)). Na fase de planejamento de movimento o caminho é encontrado como em qualquer *roadmap* tradicional.

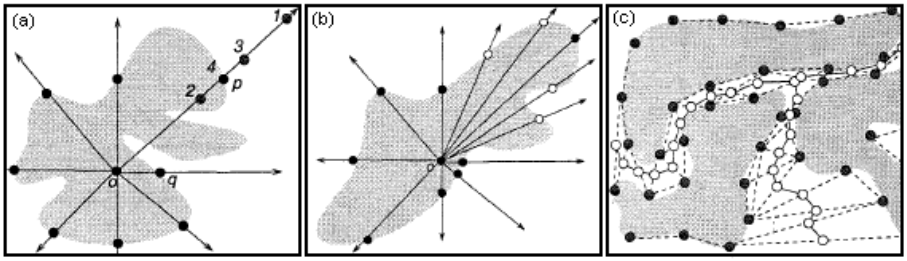

Figura 2.7: (a) escolha dos estados de configuração a partir da borda dos obstáculos. (b) otimização de estados distantes (c) ligação entre os estados (AMATO; WU, 1996)

Os custos computacionais desse método acabam sendo maiores que o convencional na fase de geração, já que cálculos mais avançados são feitos para cada EC a ser gerado, e a função de validade não deve apenas considerar se o EC é válido ou não mas deve levar em conta a proximidade com as bordas de cada objeto. A fase de conexão dos nodos tem um desempenho igual ao do PRM, já que todos nodos são testados contra todos. Otimizações adicionais como visibilidade entre objetos podem aumentar mais a eficiência dessa fase.

A aplicabilidade da técnica, devido à sua natureza, acaba ficando mais restrita a ambientes com corredores estreitos no espaço de configuração, onde seu desempenho e resultados acabam sendo melhores. Para ambientes mais abertos o uso de outra técnica com certeza será mais eficiente.

2.4 Resumo

Todas as abordagens tem seu ponto forte e a escolha de cada uma vai depender de um uso específico. Algumas apresentam soluções mais genéricas e podem ser aplicadas a uma série de casos, já a outras são impostas restrições maiores. A Tabela 2.1 confronta características gerais dos métodos vistos.

Tabela 2.1: comparação entre métodos

Método	Natureza	Geração	Uso em corpos articulados	Cobertura
PRM	probabilística	pré-processado	bom	boa
VB-PRM	probabilística	pré-processado	ruim	regular
RRT	probabilística	tempo real	bom	boa
RRT-Connect	probabilística	tempo real	regular	regular
PRM-Sukharev	determinística	pré-processado	regular	boa

O foco desse trabalho está em buscar uma forma de gerar um *roadmap* que garanta a melhor cobertura possível do espaço de configuração para qualquer tipo de grau de liberdade presente no corpo trabalhado. Por isso foi escolhida uma forma determinística de gerar os nodos do *roadmap*, o que garante sua distribuição. As técnicas vistas nesse capítulo serviram de inspiração para o desenvolvimento de diversas etapas do processo de geração do *roadmap* como um todo.

3 ROADMAPS DETERMINÍSTICOS ADAPTÁVEIS

Os métodos tradicionais de distribuição uniforme ou pseudo-uniforme podem ser aplicados a espaços de configuração maiores sob a penalidade do custo exponencial para cada grau de liberdade a mais. Contudo, para corpos articulados com muitos DOFs, os resultados gerados podem não explorar uniformemente o espaço de configuração como um todo. Isso se deve ao fato de que cada estado de configuração a ser gerado trabalha cada DOF isoladamente transportando-o para um espaço normalizado e depois combinando com os outros DOFs. Isso faz com que todos os DOFs tenham o mesmo peso na distribuição, não importando se no espaço de configuração eles tenham um intervalo de valores grande ou pequeno.

O método aqui proposto visa gerar um número adequado de amostras, evitando que amostras demais sejam geradas desnecessariamente para DOFs pouco significativos, mesmo quando trabalhando com modelos que possuam um grande número de DOFs. Classificar e definir métricas que definam o quanto um DOF é importante para o corpo é uma etapa do método.

3.1 Visão Geral da Técnica

O processo de criação e uso de um *roadmap* é constituído de uma série de etapas até que se obtenha como retorno um caminho a ser traçado. Para criar o *roadmap*, em primeiro lugar, deve-se gerar os estados de configuração que irão compor sua estrutura. Os ECs são gerados de forma determinística, mantendo-se o cuidado para que eles possuam uma distância uniforme entre si. Detalhes desse procedimento serão vistos na Seção 3.2.

Com o conjunto de ECs válidos gerados, a próxima etapa é conectá-los. A conexão entre os ECs define os caminhos que serão gerados internamente no *roadmap* e afeta seu uso para o planejamento de movimentos. Pontos importantes como a distância entre dois ECs candidatos a serem conectados devem ser considerados, evitando que se criem conexões inválidas ou em excesso, o que poderia afetar o desempenho no planejamento de movimentos. A Seção 3.3 explica o processo de conexão dos ECs.

Todo o processo, tanto a geração quanto a conexão é acompanhado de uma função de validade. Essa função indica se um determinado EC é considerado válido ou não, ou seja, se o corpo ao qual os DOFs pertencem, pode ser configurado com ele. A forma como essa função trabalha é melhor descrita na Seção 3.4.

Com o *roadmap* pronto pode-se finalmente usá-lo para planejar quantos movimentos diferentes se desejar. A forma como o *roadmap* resultante é usado para isso é vista na Seção 3.5.

3.2 Gerando os Estados de Configuração

A distribuição adaptativa baseada no espaço de configuração gera os estados de configuração tomando o espaço de configuração de todos os DOFs e calculando quais são mais significativos de serem expandidos em relação aos outros. Assim, uma subdivisão diferenciada ocorre para cada DOF, permitindo que DOFs com espaços de configuração pouco significativos para o corpo como um todo, ou com um espaço de configuração muito pequeno sejam subdivididos em seu espaço de configuração tanto quanto os outros com um espaço de configuração maior.

Entre os parâmetros considerados para esse cálculo está a magnitude de abertura de cada DOF de mesma natureza, a relação hierárquica entre os DOFs (se existir) e um valor que indica qual a diferença de peso que deve ser atribuída a DOFs em diferentes pontos da hierarquia (1 para que todos sejam tratados igualmente). DOFs de naturezas diferentes (i.e. translacional, rotacional ou discretos) recebem o mesmo peso no momento de ser considerados, já que seus valores reais no espaço de configuração representam coisas diferentes. Isso pode ser perfeitamente ajustado com um valor que prioriza uma natureza ou outra.

A Figura 3.1(a) mostra como uma distribuição simples trataria um corpo articulado com dois DOFs. Mesmo havendo uma diferença angular muito grande entre os dois DOFs rotacionais, seu espaço de configuração é dividido igualmente, gerando saltos muito grandes na junta J2. A Figura 3.1(b) mostra como ficariam os espaços subdivididos de forma diferenciada, considerando o espaço de configuração não normalizado. Nesse caso, a hierarquia não está sendo considerada.

Como pode-se observar, a cobertura do espaço de configuração é muito maior no exemplo da Figura 3.1(b). Para facilitar a visualização, a Figura 3.2 mostra um resultado de natureza translacional. Quando não existe nenhuma diferença, ou uma diferença pequena entre o espaços de configuração dos DOFs, os resultados de ambas as distribuições tendem a ser semelhantes.

Considere o espaço de configuração C composto de n graus de liberdade. O espaço de configuração livre é denotado por C_{free}, o espaço ocupado por obstáculos é denotado por C_{obs}. Existe três diferentes formas de definir como as amostras serão geradas. A primeira seria definindo um valor fixo de subdivisões (s) a ser dado a todos os DOFs. Assim o número de amostras finais é dado pelo cálculo de s^n, o que cresce exponencialmente de acordo com n, custoso para corpos com muitos graus de liberdade. A distribuição adotada segue a linha das amostragens de Sukharev, onde a dispersão máxima é buscada. A Figura 3.3 mostra os valores de um espaço unitário $[0, 1]$ quando subdividido em 2 e 3 valores de amostragem.

Dessa forma se garante a dispersão máxima, o que significa que, dado qualquer ponto do espaço de configuração global, é garantida a menor distância máxima desse ponto para um estado de configuração presente no *roadmap* para a quantia de amostras permitida.

Essa primeira forma é a tradicionalmente abordada em outras técnicas, gerando resultados conforme a Figura 3.2(a). Portanto o foco será mantido nas próximas duas a serem apresentadas.

Uma segunda forma de gerar os estados é definir um intervalo (t) preciso a ser distanciado entre duas amostras de um espaço de configuração para cada um dos espaços de configuração. Esse intervalo deve ser definido para cada natureza (rotações ou translações) diferente de grau de liberdade, em graus ou radianos por exemplo, ou no caso translacional, em unidades de medida de distância linear. Dessa forma, o valor de s é calculado para cada DOF como mostrado na Equação 3.1.

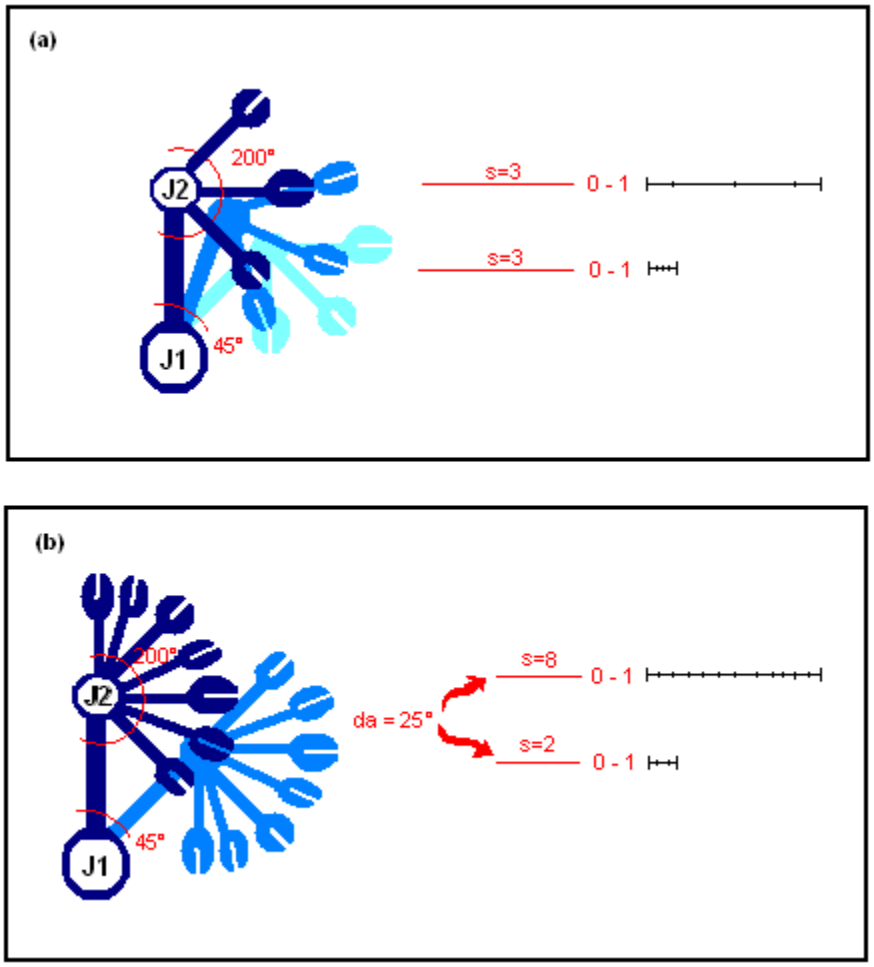

Figura 3.1: (a) subdivisão uniforme entre os DOFs (b) subdivisão diferenciada para cada DOF.

$$s_i = \frac{e_i}{t} \tag{3.1}$$

O valor e_i é o intervalo de valores no espaço de configuração do DOF i correspondente. O número total de estados de configuração gerados pode ser obtido através da multiplicação dos valores conforme a Equação 3.2.

$$r = s_1 \cdot s_2 \cdot s_3 \cdot \ldots \cdot s_n \tag{3.2}$$

Nesse sistema, a hierarquia entre os DOFs do corpo é ignorada, tendo todos a mesma prioridade. O número de estados de configuração gerados ao final também tende a crescer em ordem exponencial de acordo com o número de DOFs.

Após ter o s de cada DOF computado, basta subdividir cada espaço individualmente, conforme ilustrado na Figura 3.3 e combinado com cada valor de todos os outros DOFs. O *roadmap* resultante já garante uma distribuição uniforme como visto na Figura 3.2(b). Do número final de ECs gerados pela combinação de todos os DOFs deve-se ainda subtrair os ECs inválidos, resultando por fim no número de nodos que comporão o *roadmap*. O conceito de um EC inválido é apresentado na Seção 3.4.

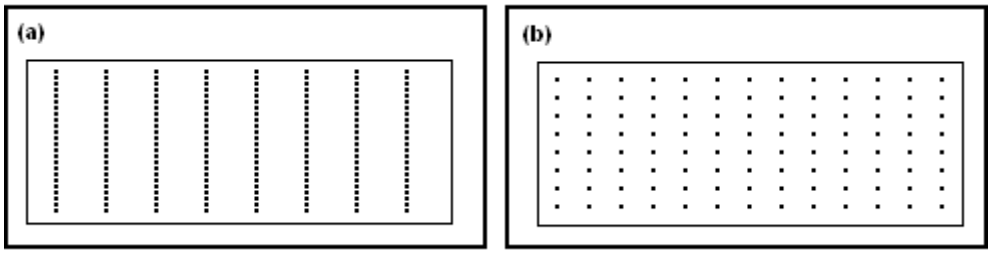

Figura 3.2: (a) subdivisão uniforme entre os DOFs no espaço $\Re^2$, (b) subdivisão diferenciada para cada DOF no espaço $\Re^2$.

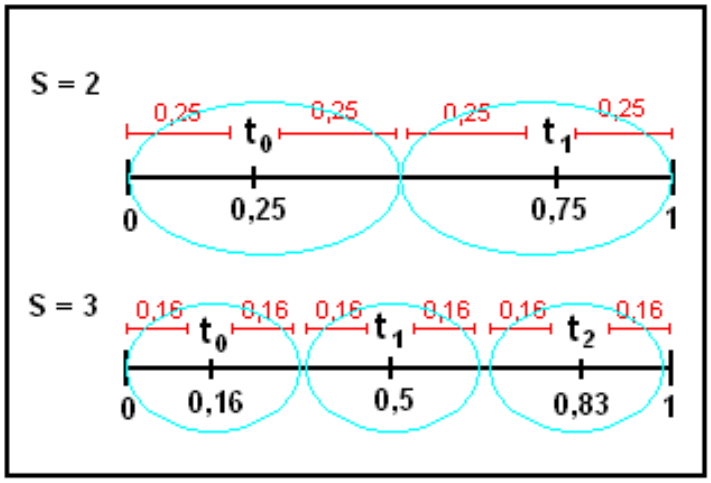

Figura 3.3: subdivisão do espaço normalizado para valores 2 e 3 de subdivisões no espaço $\Re^1$.

A terceira forma de gerar os estados de configuração toma como parâmetro, além do espaço de configuração do corpo, um número r de nodos a ser gerado. Em adicional pode-se passar um parâmetro w de peso sobre os diferentes tipos de grau de liberdade, o que irá priorizar um sobre outro caso exista mais de um para o mesmo corpo. Esse peso trabalha modificando a proporção de valor calculado para cada tipo de grau de liberdade, que é dado na Equação 3.3 para cada tipo f de grau de liberdade no corpo.

$$r' = \sqrt[f]{r} \tag{3.3}$$

Os valores gerados com a modificação do peso, quando multiplicados, acabam aproximando apenas r para valores de f maiores que 1. Com r' calculado é feita uma distribuição entre todos os DOFs para obter o s correspondente para cada um. A distribuição é feita primeiro calculando o somatório dos valores no espaço de configuração, dado da Equação 3.4.

$$E = \sum_{i=0}^{n'} e_i \tag{3.4}$$

Depois, o somatório é dividido pelo menor espaço encontrado entre os valores formados, gerando um f' que serve como parâmetro para computar qual o s final de cada DOF que é dado na Equação 3.5.

$$s_i = \frac{e_i}{min(e)} \cdot \sqrt[f']{r'} \tag{3.5}$$

O valor final é arredondado (se necessário) para ser usado na subdivisão do espaço. As informações de hierarquia entre os DOFs atuam modificando o valor de e_i para o DOF i correspondente. Neste caso existe um valor h que indica o peso de acordo com a hierarquia, e para cada DOF é calculada uma profundidade p que indica a maior profundidade encontrada em sua sub-árvore hierárquica. O novo valor atribuído a e_i é dado pela Equação 3.6.

$$e'_i = e_i \cdot h^p \tag{3.6}$$

Assim, pode-se inclusive decidir por dar mais peso aos DOFs raiz com $h > 1$, mais peso aos DOFs folha com $0 < h < 1$, ou não definir prioridade hierárquica mantendo $h = 1$.

Evidentemente, para cada estado gerado pelo processo é feito um teste com uma função de validade (ver Seção 3.4) que testa se o estado de configuração é válido. O conceito de validade pode variar para cada cenário, podendo ser apenas um teste de colisão ou de equilíbrio do corpo. Caso o estado seja dado como inválido ele é descartado ou armazenado para posterior reavaliação.

3.3 Conexão dos Estados

Após ter o grupo de estados de configuração gerados, é necessário conectá-los criando caminhos no *roadmap*. Quando o processo de amostragem gera os estados aleatoriamente é necessário dar um valor de distância máxima para avaliar se dois estados serão conectados ou não e então cada EC gerado é testado contra cada outro EC e se estiver dentro da distância máxima estabelecida, é feito o teste para conectá-los.

No caso de ter gerado os ECs de forma determinística e uniformemente distribuídos, uma distância padrão entre eles é conhecida e o teste de todos os ECs gerados contra todos os outros é dispensado, reduzindo a verificação de distância a um custo zero. Essa distância padrão é equivalente a um salto n-dimensional, o que significa que dois estados serão testados para conexão apenas se para cada grau de liberdade houver no máximo variação de uma vez a distância padrão resultante da subdivisão do seu espaço de configuração. Pode-se considerar esses grupos de nodos dentro da distância padrão como se estivessem virtualmente conectados, bastando avaliar a validade dessas conexões para efetivá-las.

Com uma função de validade definida, cada uma dessas conexões é testada fazendo uma interpolação linear entre os dois estados e verificando se os estados intermediário são válidos também. Caso os testes de validade nos estados de configuração intermediários retornem sucesso, a conexão é então feita. Se em algum momento um teste retornar como falso, a conexão é ignorada. A Figura 3.4 mostra como são consideradas as conexões inicialmente e como ficariam após os testes de validade.

A função de conexão considera uma margem de distância para decidir quantos estados intermediários serão gerados para validar a conexão. Assim, se a distância entre dois estados for menor que a margem estabelecida, as conexões podem ser feitas diretamente, sem necessidade de teste. Essa margem pode ser um parâmetro que indique a largura máxima de um obstáculo do ambiente. Assim a partir do momento em que a quantidade de nodos gerados para o *roadmap* for muito grande, a distância entre todos os nodos conectáveis estará abaixo da margem estabelecida, bastando fazer a conexão trivialmente,

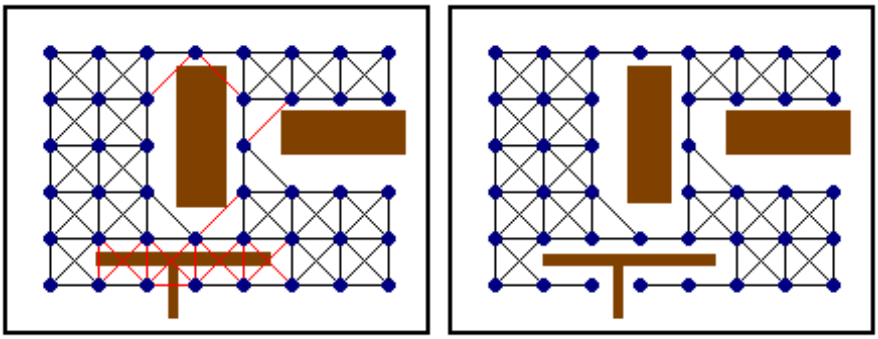

Figura 3.4: Avaliação das conexões pré-estabelecidas e eliminação das conexões inválidas em um ambiente 2D onde a área branca representa o espaço de configuração livre e a área escura os obstáculos.

e reduzindo o tempo de conexão substancialmente. Um exemplo disso pode ser visto na Figura 3.5 onde o cenário da Figura 3.4 recebe mais amostras. Como pode-se ver as duas imagens são muito mais parecidas, o que indica que menos conexões inválidas são inicialmente geradas, tornando mais rápido seu processo de verificação.

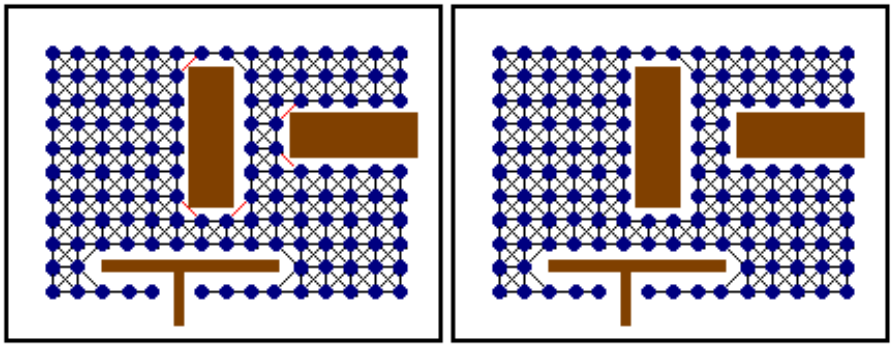

Figura 3.5: Um conjunto de amostras denso que reduz substancialmente a avaliação das conexões.

Ocasionalmente, alguns ECs podem vir a ficar sem conexão com outros, tornando-os inatingíveis. Esses estados são posteriormente testados contra todos os outros para tentar encontrar uma conexão válida. Caso essa última tentativa falhe, o EC é então considerado inatingível e descartado.

3.3.1 Distância entre dois Estados de Configuração

O conceito de distância entre dois pontos é perfeitamente aplicável quando se trabalha com graus de liberdade translacionais, bastando utilizar a distância Euclidiana entre um ponto qualquer do modelo em dois diferentes estados de configuração. Porém, esse conceito deve evoluir quando se trabalha com um grau mais complexo, em corpos com muitos DOFs, sendo vários deles rotacionais. A Figura 3.6 mostra um mesmo corpo em dois estados de configuração diferentes: (a) e (b). Pode-se ver claramente que as duas posições diferem muito em relação ao grau de flexão de suas juntas, conforme ressaltado pelos arcos. Porém quando sobrepostas, Figura (c), vê-se claramente o *end effector* (mão direita) repousando no mesmo local, o que mostra que a distância Euclidiana não é suficiente para medir a diferença entre os estados de configuração.

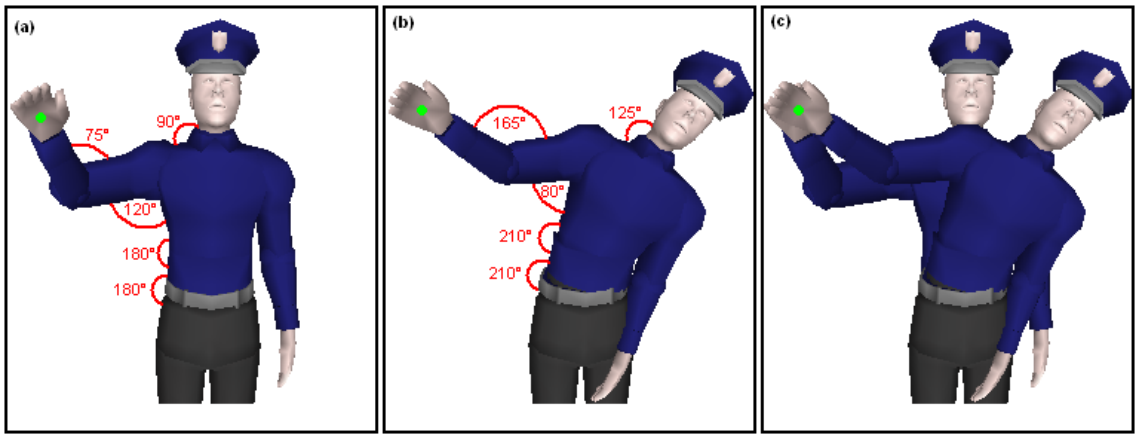

Figura 3.6: Corpo articulado com ângulo de abertura das juntas em graus. Apesar de todas as juntas ressaltadas estarem com valores diferentes, a distância Euclidiana do *end effector* é 0.

Para isso, o cálculo da distância entre dois estados de configuração é feito baseado em esforços, o que em uma junta com grau de liberdade rotacional consiste na soma das diferenças dos valores no espaço de configuração entre cada grau de liberdade. No caso da Figura 3.6 a distância poderia ser dada pela Equação 3.7

$$|75^o - 165^o| + |90^o - 125^o| + |120^o - 80^o| + |75^o - 165^o| + ... \tag{3.7}$$

Diferentes naturezas de grau de liberdade são tratadas de forma diferenciada também, já que trabalham em magnitudes de valores diferentes (distâncias angulares ou lineares).

Da mesma forma, o conceito de largura máxima de um obstáculo no ambiente é modificado, atendendo às medidas de esforço estabelecidas para poder se adaptar à etapa de conexão dos nodos do *roadmap*.

3.4 Função de Validade

A função de validade é um conjunto de testes que verificam se um determinado estado de configuração do modelo trabalhado é aceitável para o ambiente estabelecido. Os testes podem ser testes de colisão simples, usando *bounding boxes*, ou mais complexos chegando no nível de vértices. Algumas funções, para corpos humanóides geralmente, podem testar um estado de equilíbrio do corpo, calculando seu centro de massa e coerência na posição de suas juntas.

O teste de validade desenvolvido para os testes executados nesse trabalho consistiu de testes de colisão usando *bounding boxes* hierárquicas. Onde uma única *bounding box* inicial é feita para cada objeto a ser testado contra colisão, e caso o teste retorne falso, cada *bounding box* é subdividida em 8 *bounding boxes* refinando o teste e aproximando as *bounding boxes* cada vez mais do formato preciso do corpo, podendo chegar a existir uma *bounding box* para cada vértice do objeto, o que seria o grau de refinamento máximo. A Figura 3.7 mostra como são consideradas as *bounding boxes* em diferentes níveis sobre um mesmo objeto.

Para os testes executados foi considerado empiricamente um máximo de 3 níveis de subdivisão das *bounding boxes* para refinamento dos testes, por ter um custo computacional acessível e atender bem às necessidades de verificação de colisões.

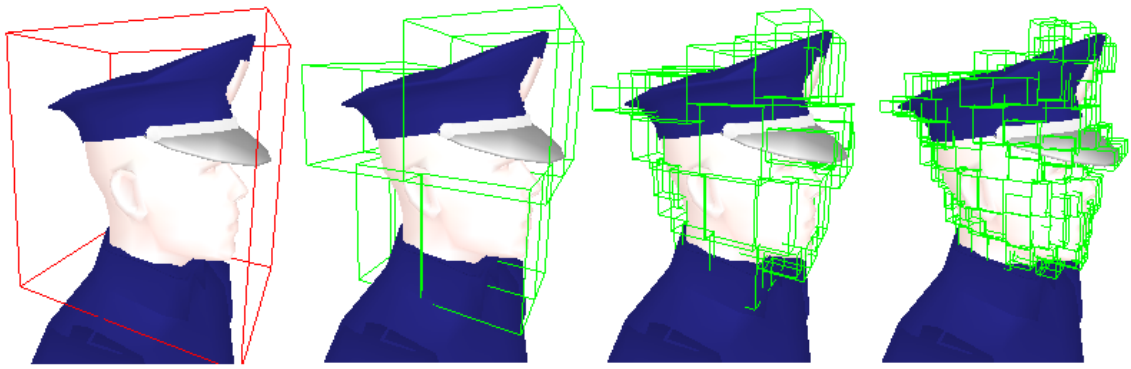

Figura 3.7: 4 diferentes níveis de refinamento de uma *bounding box* sobre um objeto.

3.5 Planejamento de Movimentos

Encontrar um caminho entre dois estados de configuração usando o *roadmap* é simples. A partir do momento em que um *roadmap* consiste em um conjunto de estados de configuração interconectados, onde cada conexão representa uma transição direta válida entre eles, basta usar um algoritmo qualquer de caminhamento em grafos como A* para encontrar um caminho válido entre dois nodos qualquer do *roadmap*.

Assim, para a aplicação de planejamento de movimento basta fornecer um EC de origem e um EC objetivo. Para ambos, um de cada vez, será buscado no *roadmap* o EC mais próximo, e então verificado se uma transição direta entre eles é possível. Se ambos ECs fornecidos forem possíveis de conectar ao *roadmap* dessa forma, um caminho entre eles é então garantido, já que internamente no *roadmap* sempre há um caminho entre dois nodos.

Neste ponto recai uma das maiores vantagens de se usar uma distribuição uniforme dos ECs ao invés de uma aleatória. Com uma distribuição uniforme, a cobertura do espaço de configuração global é maior, aumentando as chances de existir uma conexão válida entre qualquer EC fornecido e um existente no *roadmap*, garantindo uma solução válida.

4 RESULTADOS OBTIDOS

Com o objetivo de avaliar o método proposto foram executados uma série de testes em diferentes ambientes, usando modelos com diferentes tipos e número de graus de liberdade. Os testes são comparativos, analisando o desempenho da técnica proposta e de outras técnicas. Também foram feitos testes isoladamente apenas da técnica proposta, para ponderar o uso da técnica em uma mesma situação com diferentes exigências.

4.1 Técnicas Implementadas

O novo método foi implementado e nomeado de ADRM, acrônimo para *Adaptable Deterministic Roadmap*. Toda sua implementação seguiu a linha descrita no capitulo 3.

Adicionalmente, foram implementados os métodos PRM original, conforme visto na Seção 2.1, para permitir a comparação do método desenvolvido com um de natureza não determinística tradicional; e o PRM com sua fase de amostragem modificada, usando a distribuição de Sukharev, representando uma amostragem determinística. Essa segunda utiliza a mesma metodologia encontrada no PRM para as etapas que sucedem a fase de amostragem, e foi implementada também para testes no trabalho realizado por Branicky *et al* em 2001 (BRANICKY et al., 2001) para fins de comparação também.

Para evitar que fatores externos afetassem os resultados dos testes, todas os métodos utilizam a mesma função de validade, mesmos modelos e mesmo computador.

4.2 Aplicação e Modelos de Testes

Foram desenvolvidos cinco cenários para a execução dos testes, alguns deles são cenários clássicos no quadro de planejamento de movimentos e servem muito bem ao propósito de realizar comparações entre as diferentes técnicas implementadas. Cada um desses cenários considera um conjunto de graus de liberdade diferentes em diferentes modelos. Também foram implementados cenários mais complexos, populados por muitos objetos e com vários graus de liberdade, com o intuito de aplicar a nova técnica a situações mais comuns onde soluções de planejamento de movimentos podem ser mais úteis, como na geração de animações complexas.

O primeiro cenário é um mapa planar clássico de proporções retangulares dividido em duas regiões por uma passagem estreita e sinuosa. O avatar desse cenário é um quadrado que possui dois graus de liberdade (x, y) translacionais apenas, como pode ser visto na Figura 4.1.

O objetivo desse cenário é encontrar um caminho válido entre as duas regiões divididas e foi usado para comparar o desempenho entre as diferentes técnicas em modelos

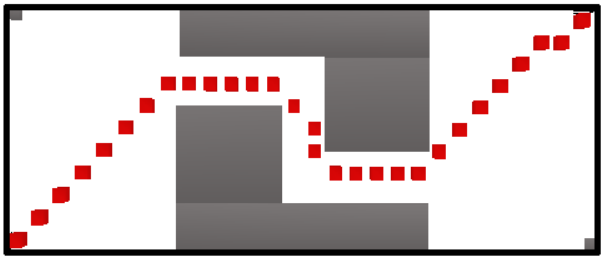

Figura 4.1: cenário 1, um objeto quadrado se deslocando em um mapa 2D.

com graus de liberdade apenas translacionais.

O segundo cenário desenvolvido é também um modelo para duas dimensões, mas dessa vez com graus de liberdade rotacionais. O objeto usado é um braço robótico que possui 3 juntas, cada uma com um grau de liberdade que permite rotações sobre um eixo perpendicular ao plano, e tem sua base presa a um ponto do cenário. O restante do cenário é composto por duas caixas que servem como obstáculo. O cenário 2 pode ser visto na Figura 4.2.

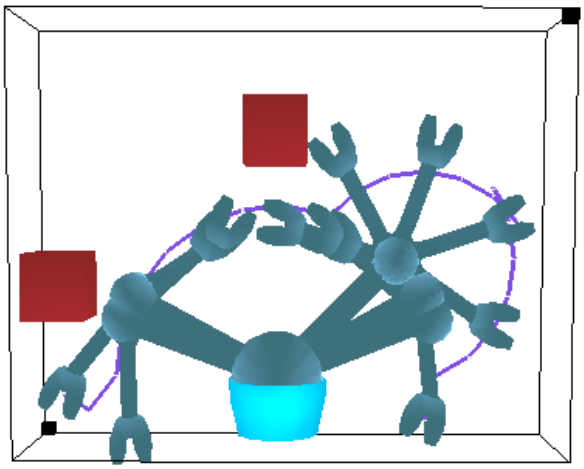

Figura 4.2: cenário 2, um braço robótico limitado a duas dimensões com 3 graus de liberdade.

Os testes nesse cenário serviram para comparar o comportamento de cada método em um modelo com DOFs apenas rotacionais em um ambiente 2D.

O terceiro cenário é composto por um braço robótico articulado, que possui 3 graus de liberdade, 2 na junta base (roll e yaw) e 1 na junta central (yaw), e alguns objetos soltos para servir de obstáculo. Uma imagem do terceiro cenário pode ser vista na Figura 4.3.

Os testes nesse cenário visam comparar o comportamento de cada método em um modelo com DOFs apenas rotacionais em um ambiente 3D. Como objetivos do cenário foram dados alguns ECs onde a garra do braço deveria alcançar pontos extremos do cenário desviando dos obstáculos presentes.

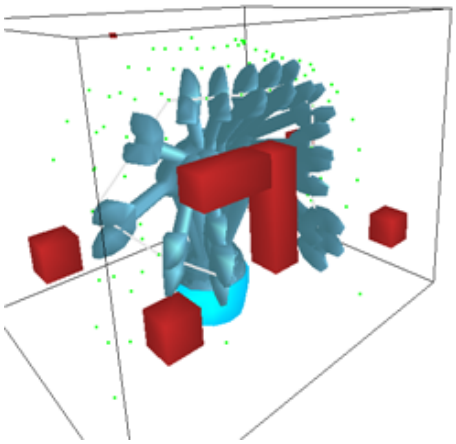

Figura 4.3: cenário 3, um braço robótico em ambiente com alguns obstáculos.

O quarto cenário é composto por um ambiente dividido ao meio por uma barreira que possui um buraco no centro, e uma peça em formato de "L" que deve transpor-se de um lado ao outro do cenário utilizando-se para isso de 6 graus de liberdade, três translacionais (eixos x, y e z) e três rotacionais (rotações sobre os eixos x, y e z). Um resultado nesse cenário pode ser visto na Figura 4.4.

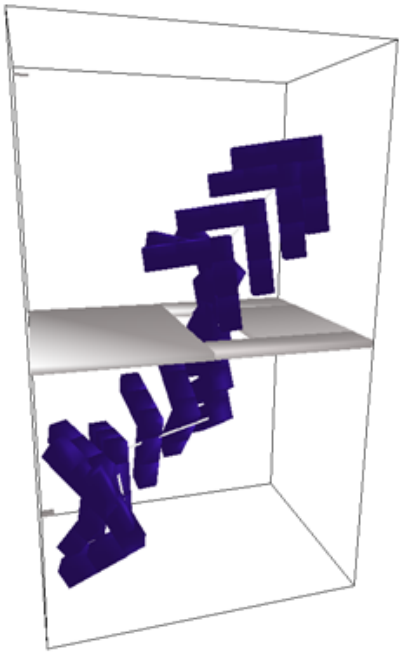

Figura 4.4: cenário 4, peça em formato de "L" passando por fresta em parede.

O principal objetivo desse cenário é testar o comportamento de objetos com diferentes tipos de grau de liberdade em um ambiente 3D para cada método implementado.

Para testar a eficiência e escalabilidade da técnica foi desenvolvido um cenário complexo: um escritório com diversos objetos e um humanóide com 11 DOFs. O cenário pode ser visto em dois ângulos diferentes na Figura 4.5.

Para os testes nesse cenário não foi dado nenhum objetivo de movimento, apenas foram gerados roadmaps de modo a explorar a estrutura complexa do cenário o melhor

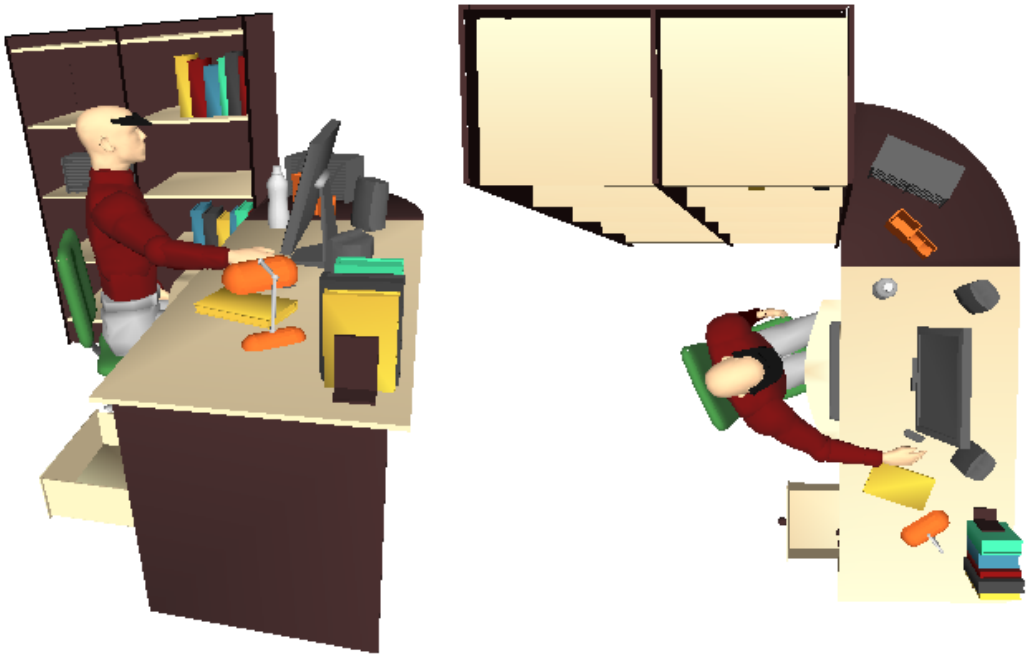

Figura 4.5: cenário 5, corpo complexo com 11 DOFs em ambiente densamente populado.

possível, verificando o alcance do personagem em pontos difíceis.

4.3 Resultados Obtidos

Os três métodos descritos na Seção 4.1 foram implementados e testados nos cenários 1, 2, 3 e 4. Foi tomado o cuidado para que, em cada cenário, os três métodos gerassem o mesmo, ou ao menos aproximadamente o mesmo número de nodos. O número de nodos gerados pelas técnicas determinísticas é determinado por uma multiplicação entre a quantidade de valores gerados para cada um de seus DOFs, menos os casos inválidos (onde há colisão). Desta forma, é muito difícil obter exatamente a mesma quantidade de nodos com todas as técnicas implementadas.

Um dos principais objetivos foi analisar a cobertura do *roadmap* gerado sobre o espaço de configuração, e também o desempenho do método proposto em relação aos outros. Para analisar a cobertura do *roadmap* resultante, após gerados os *roadmaps* foram gerados aleatoriamente uma grande quantidade de estados de configuração e tomada a distância entre cada EC gerado e o EC mais próximo de cada *roadmap*. Dessa forma supõe-se que o *roadmap* que possuir as menores distâncias entre os ECs gerados aleatoriamente e um de seus ECs é o que fornece a melhor cobertura. A média das distâncias encontradas e a distância máxima encontrada para cada *roadmap* está descrita na Tabela 4.1.

Para facilitar a análise da cobertura, foi gerado um mapa de cores onde a distância de um EC gerado aleatoriamente para o EC do *roadmap* mais próximo serviu para colorir um ponto do espaço de configuração. A Figura 4.6 mostra o *roadmap* gerado para cada método, todos eles com um total de 47 nodos cada. A Figura 4.7 mostra o mapa de cores quando foram gerados 10.000 pontos de amostragem.

Os pontos coloridos em verde representam os ECs que possuem uma distância muito pequena em relação a um EC no *roadmap*. Conforme essa distância aumenta, a cor do ponto vai mudando até atingir a cor vermelha, que representa a máxima distância medida.

Tabela 4.1: distâncias média/máxima

Cenário	nodos	ADRM	PRM	Sukharev
1	185	0,21 / 0,47	0,71 / 1,84	0,25 / 0,57
2	340	0,14 / 0,25	0,14 / 0,38	0,15 / 0,37
3	100	1,89 / 3,80	3,24 / 7,68	2,50 / 4,12
4	260	1,10 / 2,32	2,21 / 5,15	1,50 / 3,72

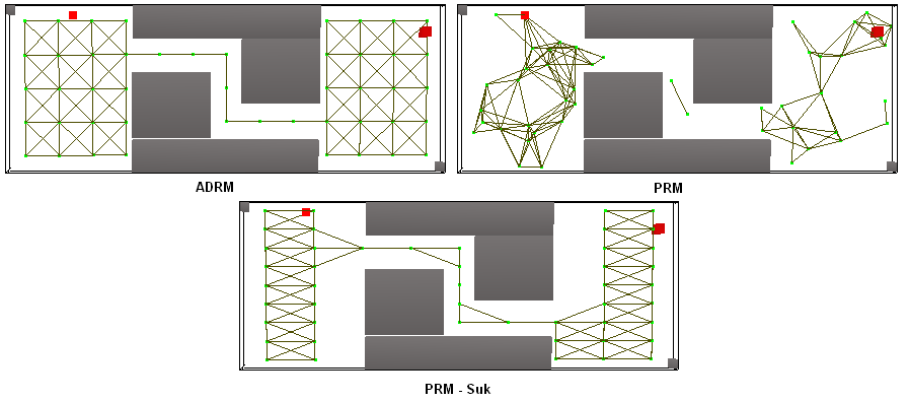

Figura 4.6: *roadmaps* gerados para cada método no cenário 1.

Os *roadmaps* mostrados nas Figuras 4.6 e 4.7 foram gerados com poucos nodos, a fim de evidenciar as diferenças entre os métodos. Resultados gerados com mais nodos foram também gerados para o mesmo cenário e podem ser vistos nas Figuras 4.8 e 4.9.

A Figura 4.10 mostra um gráfico de cores para o cenário 3, onde foi estabelecido um limiar para mostrar apenas os pontos mais próximos da distância máxima. Dessa forma, pode-se analisar quantitativamente os pontos que ficam longe do *roadmap*, justificando a média de distância como uma boa métrica de cobertura.

Dessa forma, quanto menor for a distância entre qualquer ponto do espaço de configuração e um outro pertencente ao *roadmap*, maior será a chance de encontrar uma ligação com ele, aumentando sua cobertura e as chances de encontrar uma solução para o planejamento de qualquer movimento.

Para analisar o desempenho, foi guardado o tempo de geração total do *roadmap* para cada método. O tempo armazenado inclui o tempo de geração dos nodos somado ao tempo de ligação entre eles e é informado na Tabela 4.2.

Apesar de um *roadmap* para múltiplas consultas ser gerado em tempo de pré-processamento, o que não interfere no desempenho da aplicação que o usa, o tempo de construção desse pode ser relevante quando for possível economizar horas ou dias na computação de um *roadmap* mais complexo, com muitos graus de liberdade e conexões.

O desempenho ainda pode ser medido em outros aspectos, como o tempo levado para encontrar um caminho no *roadmap*. Entretanto o uso de um *roadmap* é muito rápido, fornecendo solução em tempo interativo (menor que 0.1 seg) em todos os casos. Entretanto, foi encontrada uma diferença no tamanho linear (soma das distâncias dos ECs do caminho encontrado) do caminho traçado por cada solução, que pode ser visto na Tabela 4.3.

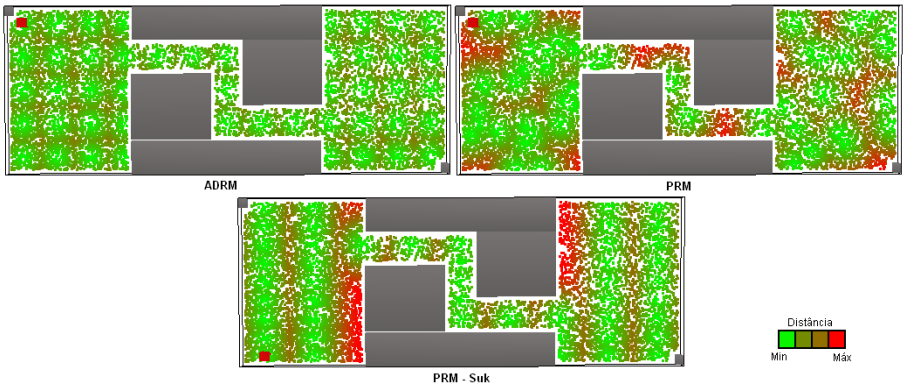

Figura 4.7: mapa de cores do cenário 1.

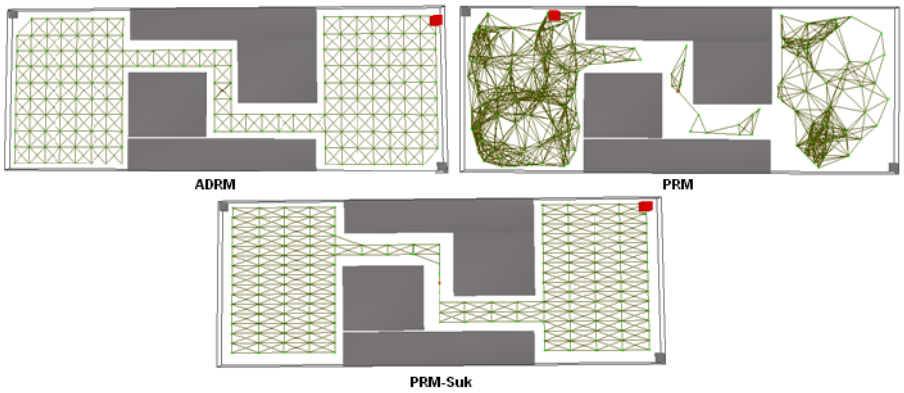

Figura 4.8: *roadmaps* com 180 amostras gerados para cada método no cenário 1.

Um tamanho de caminho menor pode indicar que a solução busca ir mais diretamente ao ponto, otimizando o movimento. Quando se aumenta significativamente o número de nodos de cada *roadmap*, o tamanho do caminho encontrado tende a ser o mesmo.

Os testes foram executados em um Computador AMD Athlon 64 3700+, com 2,21 Ghz e 2 Gb de memória RAM com adaptador gráfico GF 6600GT. Os métodos foram implementados usando Visual Studio .NET 2003 e o V-ART toolkit (V-ART, 2006). Os cenários e modelos usados foram modelados em Blender v2.41 e exportados com o plugin V-ART for Blender.

4.4 Análise dos Resultados

A análise dos resultados obtidos nos diferentes cenários para cada método implementado, nos permite fazer uma série de constatações sob diversos pontos.

4.4.1 Cobertura do Espaço de Configuração

Os *roadmaps* resultantes do método desenvolvido geraram em todos os casos uma boa cobertura do espaço de configuração. Isso pode ser facilmente verificado na Figura 4.6 onde todos os *roadmaps* possuem apenas 47 nodos. Neste mesmo cenário é possível verificar uma maior regularidade na disposição dos estados de configuração sobre o espaço de configuração, devido à sua adaptabilidade a espaços não-regulares. Ainda, se comparado com o PRM, o ADRM foi capaz de encontrar um *roadmap* que atendesse às

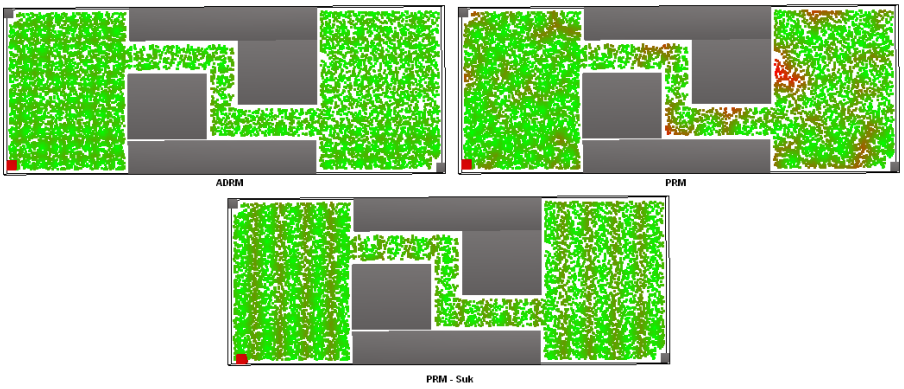

Figura 4.9: mapa de cores do cenário 1 com 180 amostras.

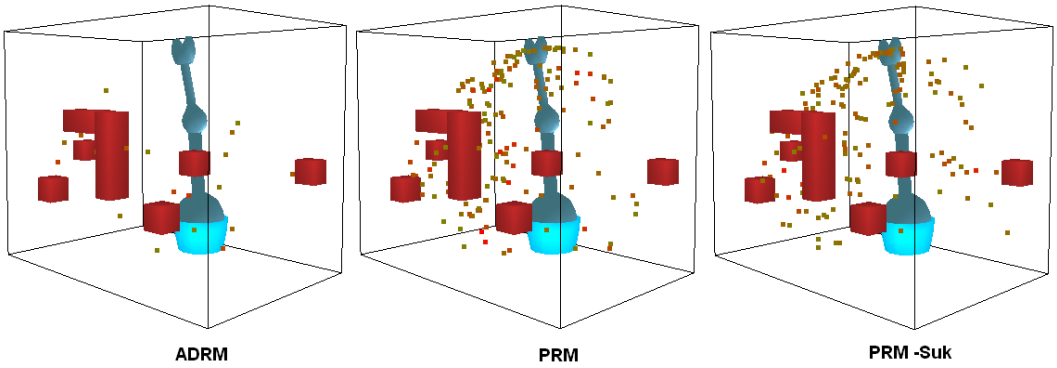

Figura 4.10: gráfico de cores do cenário 3 apenas com pontos distantes.

necessidades de planejamento de caminhos mesmo com muito poucos nodos. Já a solução não-determinística teve que ser recalculada mais de uma vez até que uma solução fosse encontrada. No segundo cenário, com apenas 100 nodos, o ADRM também foi capaz de atingir qualquer ponto alcançável pelo PRM.

Os dados coletados para comparar os métodos no sentindo de cobertura do espaço (tabela 4.1), mostram que o ADRM obteve, em geral, valores menores para as médias de distância e distâncias máximas em relação a um conjunto grande de ECs amostrados, ou seja, muito poucos pontos distantes do *roadmap* como pode ser conferido nas Figuras 4.7 e 4.10. Essas distâncias menores são muito desejáveis, pois garantem uma melhor cobertura do espaço de configuração, aumentando as chances de atingir qualquer ponto deste sem o risco de colisões.

4.4.2 Generalização

O ADRM pode ser aplicado a robôs com diferente número e tipos (translacionais ou rotacionais) de grau de liberdade e a vários ambientes distintos, como pode ser verificado pelos cenários de teste apresentados. O método de amostragem de nodos se ajusta às necessidades do usuário, priorizando DOFs translacionais ou rotacionais, se desejado. Para isso, há apenas a necessidade de informar ao método os limites dos espaços de configuração envolvidos, para que o algoritmo possa ponderar suas subdivisões. Para ambientes que possuem objetos dinâmicos passíveis de colisão, o *roadmap* pode ser computado previamente e, em tempo de navegação, detectar colisões e recalcular caminhos em tempo real, já que a etapa de planejamento de movimento é feita interativamente.

Tabela 4.2: tempo de criação do *Roadmap* (em segundos)

Cenário	nodos	ADRM	PRM	Sukharev
1	185	9,40	17,10	11,10
2	340	103,35	114,71	149,08
3	100	97,20	87,10	161,40
4	260	40,08	69,87	43,21

Tabela 4.3: tamanho médio do caminho encontrado (em unidades de medida)

Cenário	nodos	ADRM	PRM	Sukharev
1	185	24,60	27,32	25,79
2	340	18,90	18,27	18,72
3	100	15,23	20,21	16,65
4	260	17,82	18,94	19,38

4.4.3 Ponderação Hierárquica

Como descrito na Seção 3.2, existe uma forma de priorizar um grau de liberdade sobre o outro conforme sua posição na hierarquia dos DOFs, permitindo que um DOF mais próximo à raiz seja melhor desenvolvido que outros devido ao peso das alterações que ele provoca no corpo. A Figura 4.11 mostra como cada grau de liberdade de um corpo afeta este separadamente.

Apesar de todos os DOFs terem o mesmo ângulo de abertura, eles poderiam ser tratados diferentemente, priorizando aqueles que provocam mais alterações no corpo, melhorando a cobertura do *roadmap*. A Figura 4.12 mostra como fica a distribuição do corpo quando a ponderação hierárquica é ou não considerada para uma mesma situação.

A Figura 4.12 (a) mostra o resultado de um roadmap com 27 nodos, já a Figura 4.12 (b) mostra um com apenas 24. Porém, mesmo assim, o segundo apresenta uma melhor distribuição devido aos diferentes pesos atribuídos.

4.4.4 Eficiência e Escalabilidade

O ADRM foi desenvolvido para tratar qualquer quantidade de dados, deixando problemas de desempenho e limitações para o computador que o executa. Os tempos coletados durante os testes e mostrados na Tabela 4.2 mostram que o tempo gasto para gerar o *roadmap* é compatível com os de outras técnicas, apresentando até valores sensivelmente menores.

A Tabela 4.4 mostra um teste de esforço do algoritmo no cenário 5, onde foi passado um valor diferente de intervalo para todos os DOFs em cada caso. Quanto menor o valor desse intervalo, mais subdivisões cada espaço de configuração de cada DOF sofrerá, gerando mais nodos.

O tempo de geração do *roadmap* final depende exponecialmente do número de no-

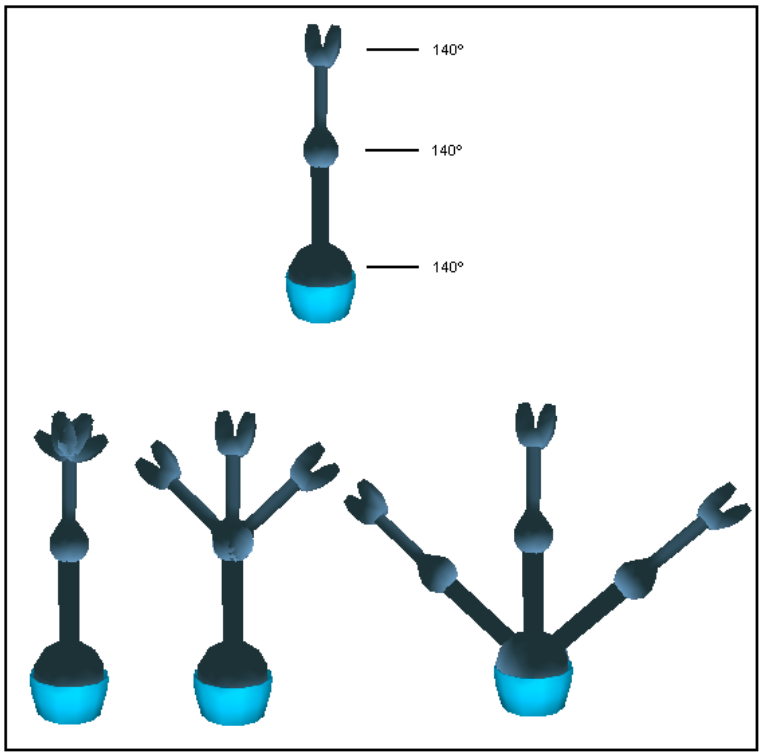

Figura 4.11: efeito da alteração de cada grau de liberdade separadamente sobre o corpo. Cada DOF possui o mesmo ângulo de abertura, porêm o que está mais próximo à raiz tem mais efeito sobre o *end effector*.

dos a serem gerados. O processo de conexão foi executado duas vezes, uma utilizando normalmente a função de validade, e outra ignorando-a, apenas fazendo a conexão trivialmente de acordo com a distância. A Tabela 4.5 mostra os tempos coletados para os mesmos casos da tabela anterior.

Analisando os dados, pode-se notar que o tempo não cresce em tempo exponencial em relação aos número de nodos quando se usa o teste de validade, isso acontece porque o número de testes entre os pares de ECs a serem conectados é reduzido devido à maior proximidade entre eles. O gráfico da Figura 4.13 mostra que conforme o intervalo angular entre os ECs for sendo reduzido, a tendência é a igualdalde dos dois tempos medidos, pois quando todas as distâncias entre os ECs vizinhos forem muito pequenas, não será necessário fazer nenhum teste entre eles, bastando conectá-los trivialmente.

4.4.5 Consumo de Memória

A quantidade de memória necessária é diretamente proporcional ao número de nodos que comporão o *roadmap*. No *roadmap* são armazenados apenas os valores dos DOFs, assim como em cada outra técnica. As informações de limites do espaço de configuração são usadas apenas durante a computação do *roadmap* e descartadas para o armazenamento. Durante esse processamento, os picos de memória não avançam uma quantidade significativa em relação ao necessário para armazenar os dados, e portanto o uso de memória não representa nenhum gargalo para o método.

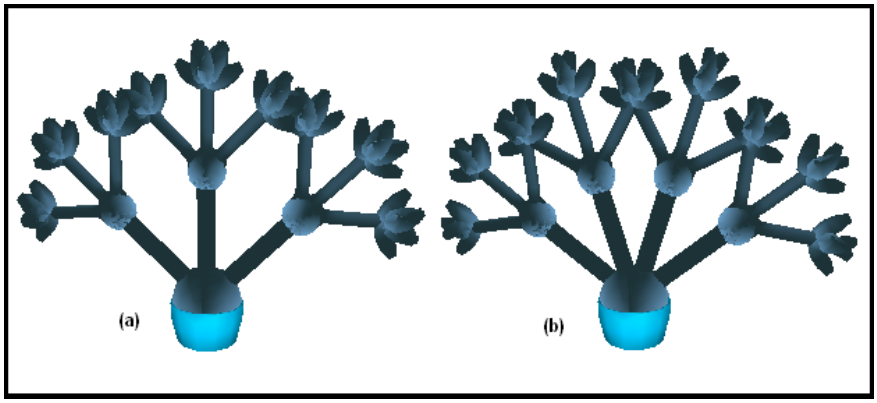

Figura 4.12: (a) resultado da distribuição considerando apenas o espaço de configuração; (b) resultado da distribuição considerando também a hierarquia.

Tabela 4.4: teste de escalabilidade

Intervalo (em rad)	nodos gerados	Tempo de Geração (em segundos)
1,0	445	167,27
0,9	583	201,06
0,8	769	263,61
0,7	1.324	504,96
0,6	9.970	5.844,31
0,5	54.507	30.830,00

4.4.6 Espaços de Configuração Superiores

O método pode ser usado para tratar corpos com muitos graus de liberdade, contornando os problemas geralmente encontrados em métodos determinísticos. Isso se deve ao fato do próprio algoritmo poder identificar DOFs prioritários a serem tratados em detrimento de outros pouco significativos para o corpo em geral. Quando se deseja obter um número pequeno de estados de configuração mesmo para um corpo com muitos DOFs, o algoritmo acaba ignorando alguns DOFs, tratando o corpo como se ele tivesse menos DOFs.

4.5 Considerações Finais

Esse trabalho gerou um artigo que foi submetido ao *International Conference on Robotics and Automation* (ICRA 2007). O artigo e animações geradas ao longo desse trabalho podem ser encontrados em http://www.inf.ufrgs.br/∼ughini/adrm. O artigo também pode ser visto no apêndice A.

Tabela 4.5: tempos de conexão

Intervalo (em rad)	Tempo de Conexão (sem teste de validade) em segundos	Tempo de Conexão (com teste de validade) em segundos
1,0	0,341	3.083,2
0,9	0,426	3.797,7
0,8	0,665	4.911,5
0,7	1,690	11.023,0
0,6	85,023	42.329,0
0,5	4.042,37	133.832,0

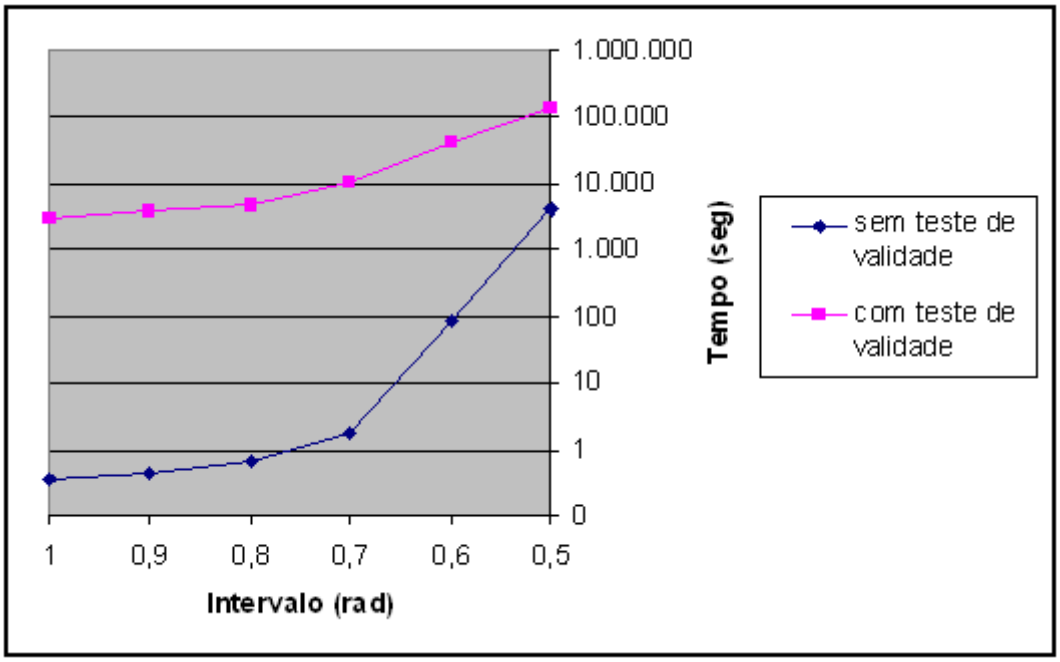

Figura 4.13: curva de crescimento do tempo conforme diminuição do intervalo das subdivisões dos DOFs.

5 CONCLUSÕES

Neste trabalho foi apresentado um novo método determinístico para gerar *roadmaps* como uma solução para o problema de planejamento de movimentos. O novo método foi batizado de ADRM, acrônimo para Adaptative Deterministic *Roadmap*, e visa gerar uma quantidade adequada de amostras para qualquer modelo trabalhado independente do número de graus de liberdade que este possuir. Em um primeiro momento, o método coleta dados sobre cada grau de liberdade do modelo e sua hierarquia e faz uma classificação de prioridades sobre cada DOF. Então, essa classificação é utilizada como uma métrica para decidir quais DOFs são mais significativos de serem tratados no modelo, permitindo a priorização destes em detrimento de outros. Assim, mesmo para corpos com grande número de graus de liberdade, é possível gerar um *roadmap* com um número mais reduzido de nodos, porém com nodos melhor selecionados em relação a outros métodos existentes.

Os resultados obtidos na aplicação com os diferentes modelos e cenários são perfeitamente aceitáveis, como pode ser comprovado com as métricas coletadas durante os testes e gráficos comparativos mostrados. Estes ilustram uma boa cobertura do espaço de configuração, procurando atingir todos os pontos cabíveis entre os valores limites dos DOFs, em um tempo computacional bastante razoável, conforme os tempos medidos. Em adicional, analisando qualitativamente os resultados obtidos de muitas animações geradas usando o ADRM, pode-se concluir que os caminhos retornados pela técnica em geral são naturais, se aproximando do que seria executado por uma pessoa real.

O gargalo verificado nos nossos testes recaiu principalmente sobre a função de validade, que testava colisões usando um algoritmo baseado em *multi-level hierarchical bounding boxes*, o que pode ser oneroso considerando ambientes muito populados com corpos mais complexos. Os tempos medidos poderiam ser substancialmente reduzidos caso uma função de validade mais simples fosse aplicada. Quando trabalhando com cenários mais densos com essa função de validade, o desempenho deve ser pior já que a probabilidade de colisões deve aumentar fazendo com que a função busque fazer testes mais refinados e, conseqüentemente, mais pesados. Entretanto como todos os métodos implementados usaram a mesma função de validade, esse peso acabou sendo igual para todos, não interferindo nas comparações medidas.

Em trabalhos futuros devem ser executados testes em ambientes dinâmicos, investigando a hipótese de que o replanejamento das conexões do *roadmap* não deve consumir um tempo substancial, a partir do momento e que existem poucos nodos no *roadmap*. Também estão sendo investigados novos critérios para atribuir prioridades aos DOFs. Ainda, está sendo avaliada a possibilidade de usar o ADRM como uma matriz base para robôs explorarem ambientes desconhecidos, de tal forma que a fase de amostragem indique objetivos e a fase de conexão explore os possíveis movimentos a partir de um estado inicial.

REFERÊNCIAS

AMATO, N.; WU, Y. A randomized roadmap method for path and manipulation planning. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.113–120.

BARRAQUAND, J.; LATOMBE, J. C. **Numerical Potential Field Techniques for Robot Path Planning**. Stanford, CA, USA: Department of Computer Science, Stanford University, 1989. (Report No. STAN-CS-89-1285).

BRANICKY, M. S.; LAVALLE, S. M.; OLSON, K.; YANG, L. Quasi-Randomized Path Planning. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.1481–1487.

CHOI, M. G.; LEE, J.; SHIN, S. Y. Planning biped locomotion using motion capture data and probabilistic roadmaps. **ACM Trans. Graph.**, New York, NY, USA, v.22, n.2, p.182–203, 2003.

GO, J.; VU, T.; KUFFNER, J. Autonomous Behaviors for Interactive Vehicle Animations. In: ACM SIGGRAPH SYMPOSIUM ON COMPUTER ANIMATION, 2004. **Proceedings...** New York: ACM, 2004.

HALTON, J. H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. **Numerische Mathematik**, [S.l.], v.2, n.1, p.84–90, Jan. 1960.

HAMMERSLEY, J. M. Monte Carlo methods for solving multivariable problems. **Annals of the New York Academy of Science**, [S.l.], p.844–874, 1960.

KALISIAK, M.; M. PANNE van de. RRT-blossom: rrt with a local flood-fill behavior. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2006. **Proceedings...** Piscataway: N.I.: IEEE, 2006. p.1237–1242.

KALLMANN, M.; AUBEL, A.; ABACI, T.; THALMANN, D. Planning Collision-Free Reaching Motions for Interactive Object Manipulation and Grasping. In: EUROGRAPHICS, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.313–322.

KALLMANN, M.; BARGMANN, R.; MATARIC, M. Planning the Sequencing of Movement Primitives. In: INTERNATIONAL CONFERENCE ON SIMULATION OF ADAPTIVE BEHAVIOR, 2004. **Proceedings...** [S.l.: s.n.], 2004.

KAVRAKI, L.; LATOMBE, J. C. Randomized preprocessing of configuration for fast path planning. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1994. **Proceedings...** [S.l.: s.n.], 1994. p.2138–2145.

KUFFNER, J.; LAVALLE, S. M. RRT-connect: an efficient approach to single-query path planning. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.995–1001.

LATOMBE, J. C. **Robot Motion Planning**. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

LAU, M.; KUFFNER, J. J. Behavior Planning for Character Animation. In: ACM SIGGRAPH SYMPOSIUM ON COMPUTER ANIMATION, 2004. **Proceedings...** New York: ACM, 2004.

LAVALLE, S. M. **Rapidly-Exploring Random Trees**: a new tool for path planning. [S.l.]: Iowa State University, Department of Computer Science, 1998. (Report No. 98-11).

LAVALLE, S. M. **Planning Algorithms**. Cambridge, U.K.: Cambridge University Press, 2006. Disponível em: <http://planning.cs.uiuc.edu/>. Acesso em: ago 2006.

LAVALLE, S. M.; BRANICKY, M. S.; LINDEMANN, S. R. On the relationship between classical grid search and probabilistic roadmaps. **International Journal of Robotics Research**, [S.l.], v.23, n.7/8, p.673–692, July/Aug. 2004.

LEE, J. et al. Interactive control of avatars animated with human motion data. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 29., 2002. **Proceedings...** New York: ACM, 2002. p.491–500.

LINDEMANN, S. R.; LAVALLE, S. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.3251 – 3257.

MISSIURO, P.; ROY, N. Adapting probabilistic roadmaps to handle uncertain maps. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.1261–1267.

NISSOUX, C.; SIMéON, T.; LAUMOND, J. P. Visibility-Based Probabilistic Roadmaps for Motion Planning. **Advanced Robotics Journal**, [S.l.], v.14, p.477 – 493, 2000.

SUKHAREV, A. G. Optimal strategies of the search for an extremum. **USSR Computational Mathematics and Mathematical Physics**, [S.l.], p.910–924, 1971.

V-ART. **The V-ART project**. Disponível em: <http://www.inf.ufrgs.br/cg/v-art>. Acesso em: dez 2006.

YERSHOVA, A. et al. Dynamic-domain rrts: efficient exploration by controlling the sampling domain. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.3856–3861.

ANEXO ARTIGO SUBMETIDO

O artigo a seguir foi submetido ao International Conference on Robotics and Automation (ICRA 2007).