PLANEJAMENTO DE TRAJETÓRIA PARA O ROBÔ OMNI UTILIZANDO O ALGORITMO MAPA DE ROTAS PROBABILÍSTICO

Bruno Vilhena Adôrno[*]
bvabr@yahoo.com.br

Carla Silva Rocha Aguiar[*]
carla.rochaguiar@bol.com.br

Geovany Araújo Borges[*]
gaborges@ene.unb.br

[*]Laboratório de Controle e Visão por Computador (LCVC)
Grupo de Robótica, Automação e Visão Computacional (GRAV)
Departamento de Engenharia Elétrica (ENE) - Universidade de Brasília (UnB)
Caixa Postal 04591 - Asa Norte - Brasilia - CEP 70910-900 - Brasil

RESUMO

Este artigo apresenta a implementação de uma técnica probabilística de planejamento de trajetória para um robô móvel omnidirecional. Esta técnica, conhecida por Mapa de Rotas Probabilístico, gera pontos de passagens intermediários que levam o robô de uma posição inicial a uma posição final. A geração de pontos intermediários leva em consideração um mapa do ambiente, que deve ser fornecido *a priori*. A trajetória global resultante é interpolada usando um interpolador de trajetória baseado em curvas de Bezier. Uma estratégia de amostragem gaussiana para as configurações aleatórias visando a redução do custo computacional também é discutida. A análise de resultados de simulação e experimentais usando mapas reais sugerem novas linhas de investigação.

PALAVRAS-CHAVE: Mapa de Rotas Probabilístico, planejamento de trajetória, curvas de Bezier, robótica móvel.

ABSTRACT

This paper presents the implementation of probabilistic roadmap planner for an omnidirectional mobile robot. This technique generates intermediate points in the working space of the mobile robot indicating a colision-free path between the current and the goal position. For this purpose, a priori information is given by an environment map. The intermediate points are interpolated using Bezier curves. In order to reduce computational cost, a Gaussian sampling strategy is applied. The analysis of simulated and real experiments suggest new investigation trends.

KEYWORKS: Probabilistic roadmap, trajectory planning, Bezier curves, mobile robotics.

1 INTRODUÇÃO

O planejamento de trajetória é um problema básico no desenvolvimento de robôs autônomos, uma vez que é uma tarefa onde não pode haver intervenção humana, e tem provado ser um problema difícil em robótica (Barraquand et al., 1997; Kavraki et al., 1996). Sua solução requer um tempo exponencial em relação ao número de graus de liberdade do robô (Barraquand et al., 1997) . Algoritmos completos foram propostos, mas em geral são impraticáveis devido ao processamento que requerem (Barraquand et al., 1997; Berg and Overmars, 2004).

Existem vários outros exemplos de utilização do planejamento de trajetória, com os mais recentes envolvendo tarefas complexas como as de robôs cirurgiões, de exploração espacial, na animação de atores artificiais para filmes ou jogos de computador (Barraquand et al., 1997).

Neste contexto, os problemas de planejamento de trajetória em mapas estáticos e completamente conhecidos são os mais básicos. Atualmente as pesquisas estão se focando mais fortemente em ambientes com incertezas e obstáculos que se movem (Barraquand et al., 1997). As abordagens atuais podem ser divididas basicamente em três grupos (Overmars, 1992): métodos que utilizam mapas de rotas,

métodos de decomposição em células e métodos que utilizam funções de potencial. Entretanto, atualmente outras técnicas já foram desenvolvidas com o propósito de resolver o problema do planejamento de trajetória, como por exemplo aquela baseada em algoritmos genéticos, proposta por (Hu and Yang, 2004).

Um método bastante difundido que utiliza a abordagem de mapas de rotas é o Mapa de Rotas Probabilístico (MRP), conhecido no inglês pela sigla *PRM*. O MRP é normalmente utilizado para resolver os problemas de planejamento de trajetória para robôs com vários graus de liberdade em ambientes estáticos (Kavraki et al., 1996). Inicialmente ele foi concebido para ser aplicado em ambientes estáticos, porém atualmente existem algoritmos baseados no MRP que são utilizados em ambientes dinâmicos. Exemplos de aplicações em ambientes estáticos são: manutenção de tubulações de resfriamento em uma usina nuclear; soldagem ponto-a-ponto na fabricação de carros; e limpeza de fuselagem de aviões (Kavraki et al., 1996). A sua maior ênfase é na eficiência, principalmente de tempo de execução. Para o MRP já existem vários resultados mostrando a relação entre a probabilidade de se achar um caminho e o tempo de execução do algoritmo (Barraquand et al., 1997).

Nos métodos baseados na decomposição em células, realiza-se uma divisão do espaço livre em células simples. A decomposição pode ser tanto exata quanto uma aproximação. Em seguida, células vizinhas são conectadas em um grafo. Entre cada célula uma trajetória pode facilmente ser construída. Para achar uma trajetória entre a origem e o destino o algoritmo determina quem são as células que contém estes pontos e calcula o caminho entre elas.

Os métodos que utilizam funções de potencial trabalham de uma maneira completamente diferente. A sua implementação mais básica consiste em sair da origem e ir ao destino em pequenos passos direcionados sob a ação de forças. O destino tem uma força de atração e os obstáculos possuem forças de repulsão. O problema desta abordagem são os mínimos locais onde as diferentes forças possuem o somatório igual a zero, resultando na parada do robô. Existem diversas maneiras de remediar o problema, porém ou estes métodos são limitados a situações específicas ou então são muito lentos (Overmars, 1992). Em (Overmars, 1992) são empregadas técnicas simples de funções de potencial para construir uma rede aleatória de rotas entre vários pontos, tentando assim conectar a origem e o destino. É um método que mistura mapas de rotas com funções de potencial. É chamado de RPP (*Randomized Path Planner*). Entretanto, um fenômeno adverso deste tipo de planejador é a existência de ciclos limites em ambientes mais complexos.

Neste artigo é apresentada uma implementação do algoritmo MRP para um robô omnidirecional sobre o qual está implantado um interpolador de trajetória baseado em curvas de Bezier (Aragones et al., 2002). É também apresentada uma implementação da estratégia de amostragem gaussiana proposta em (Boor et al., 1999). Uma análise baseada em resultados experimentais é feita entre a estratégia padrão de amostragem aleatória e a estratégia de amostragem gaussiana, considerando-se também a interpolação da trajetória resultante através das curvas de Bezier.

O restante do artigo está organizado da seguinte forma: A Seção 2 descreve o método do Mapa de Rotas Probabilísticas, seguido da Seção 3 que apresenta a estratégia de amostragem gaussiana. A Seção 4 faz uma breve descrição da plataforma experimental, o robô Omni e detalhes de implementação são discutidos na seção 5. Por fim, a Seção 6 apresenta os resultados experimentais que são seguidos das conclusões.

2 MAPA DE ROTAS PROBABILÍSTICAS

O mapa de rotas (ou *roadmap*, do inglês) é definido na configuração livre (C-*free*) do mapa e é armazenado num grafo *R*. As configurações do robô são os nós *N* de *R* e os caminhos entre uma configuração e outra calculados pelo planejador local são as bordas *E* de *R*.

O MRP é constituído de duas fases: *fase de aprendizado* e *fase de questionamento*. Na fase de aprendizado, o robô aprende sobre o mapa e armazena seu conhecimento no grafo *R*. Ele primeiramente constrói o mapa de rotas e em seguida faz um pós-processamento (chamado de expansão) com o objetivo de aumentar a conectividade do mapa de rotas nas regiões mais difíceis do espaço de trabalho C-*space*. Este algoritmo usa uma abordagem probabilística que vai construindo rotas no espaço livre (chamado de C-*free*) de forma incremental. O método heuristicamente identifica as regiões difíceis no espaço livre e gera configurações adicionais do robô de forma a aumentar a conectividade nesses locais. O objetivo final é fazer com que a maior carga de processamento fique na etapa de aprendizado sobre o ambiente e que a fase de questionamento seja muito rápida, praticamente instantânea.

O método é descrito da seguinte forma. Seja o grafo $R = (N, E)$, com *N* representando os nós e *E* as bordas do grafo. A configuração aleatória do robô é armazenada em *N* e a conexão entre dois nós vizinhos é representada por uma borda adicionada em *E*. A conexão é calculada através de um planejador local extremamente rápido, porém não muito poderoso. Um bom exemplo de planejador local que possui essas características é aquele que simplesmente discretiza uma semi-reta que liga dois nós vizinhos e coloca configurações intermediárias do robô com o intuito de fazer checagem de colisões (Kavraki et al., 1996). A trajetória entre dois nós vizinhos não é explicitamente armazenada. Na fase de questionamento o mapa de rotas é usado para resolver problemas de planejamento de trajetória no ambiente em questão.

A etapa de aprendizado consiste em basicamente de duas sub-etapas: a sub-etapa de construção e a sub-etapa de expansão, descritas a seguir. A fase de questionamento, é des-

crita na Seção 2.3.

2.1 Construção

Inicialmente, o grafo está vazio. A partir de então, são geradas repetidamente configurações aleatórias sobre as quais é aplicada uma verificação de colisão. Se não houver colisão, a configuração aleatória é armazenada em *N*. Caso contrário, ela é descartada. Para cada novo nó *c* gerado é selecionado um número limitado de vizinhos N_c para o nó atual e é verificada a conexão de cada um desses nós com o nó atual através do planejador local. O caminho não é explicitamente armazenado, pois o custo computacional do seu cálculo é baixo e é fácil recalculá-lo depois, o que evita armazenamento local de dados.

É importante verificar se um caminho entre a origem e o destino do robô existe ou não. Caso exista, o caminho deve ser determinado o mais rapidamente. Isto requer que o mapa de rotas seja denso o suficiente e que tenha pelo menos um nó no qual seja fácil conectar à origem e ao destino. Além disso, se o planejador local for rápido o suficiente ele poderá ser utilizado na etapa de questionamento.

2.2 Expansão

Caso o número de nós gerados durante a etapa de construção seja grande o suficiente, o mapa de rotas cobrirá grande parte do espaço livre da cena em questão. Entretanto, muitas vezes é necessário a geração de muitos pontos para capturar os locais mais difíceis do espaço de trabalho (C-*space*). Uma maneira de minimizar este problema é aplicar uma heurística para determinar quais são os locais difíceis para então aumentar o número de configurações aleatórias nesta região (Kavraki et al., 1998).

Uma esquema proposto por (Kavraki et al., 1996) consiste em associar um peso positivo $w(c)$ que seja uma medida heurística do grau de dificuldade da região em torno de c. Existem várias maneiras de se medir o valor do peso $w(c)$. Uma das várias possibilidades é contar o número de nós N_c vizinhos a c contidos em uma região pré-definida por uma distância D de c. Se este número de nós for muito pequeno, provavelmente a região é considerada difícil. Daí pode-se chegar à conclusão que $w(c)$ pode ser definida inversalmente proporcional ao número de nós contidos dentro da região definida por $D(c)$. Uma outra possibilidade (Kavraki et al., 1996) seria verificar a distância de c até o primeiro componente que não contém c. Se esta distância for pequena, então c está numa região considerada difícil (ou que pode estar obstruída). Sendo assim, a idéia é que $w(c)$ seja inversamente proporcional a esta distância. Uma terceira alternativa, também proposta em (Kavraki et al., 1998), consiste em verificar o comportamento do planejador local. Por exemplo, se o planejador local falhar bastante em conectar c a outros nós, então isto é uma indicação de que a região é difícil.

2.3 Questionamento

Durante a fase de questionamento, dados os pontos de origem e destino do robô, uma trajetória será gerada a fim de conectar esses dois pontos. Inicialmente o ponto de origem Ps e o ponto de destino Pg são conectados a dois pontos do mapa de rotas $Ps\prime$ e $Pg\prime$. Caso estejam em componentes diferentes, não existirá trajetória praticável e o MRP deverá indicar a falha na geração da trajetória. Caso estes pontos estejam no mesmo componente, o cálculo da trajetória é feito através da concatenação entre Ps e $Ps\prime$, todos os nós que ligam $Ps\prime$ a $Pg\prime$ e $Pg\prime$ a Pg. Os caminhos que foram achados na fase de aprendizado são todos recalculados. Como o planejador local é determinístico, sempre será gerado o mesmo caminho para as mesmas configurações de entrada.

Se os questionamentos de trajetória falharem frequentemente, esta será uma indicação de que o mapa de rotas pode não estar capturando a conectividade do espaço livre de forma adequada. Desta forma, mais tempo poderá ser utilizado para o aprendizado do mapa. Contudo, o mapa de rotas não será construído desde o início, apenas incrementado.

3 ESTRATÉGIA DE AMOSTRAGEM GAUSSIANA

A técnica de amostragem gaussiana foi proposta por (Boor et al., 1999) com o objetivo de melhorar o desempenho dos algoritmos MRP. Ela se baseia no princípio de que os passos computacionalmente mais onerosos do algoritmo MRP correspondem às configurações aleatórias geradas e os testes de conexão entre cada configuração aleatória e seus vizinhos (Boor et al., 1999). De acordo com os autores, o tempo total do algoritmo pode ser aproximado por

$$T = n\,(T_s + T_a) \tag{1}$$

Em que n significa o número de amostras necessárias para resolver o problema, T_s o tempo necessário para criar uma configuração aleatória e T_a o tempo necessário para adicionar a configuração ao grafo (o que inclui checar a conexão com os vizinhos) (Boor et al., 1999).

Na implementação padrão T_s é muito menor que T_a (Boor et al., 1999), de forma que a idéia central deste algoritmo é gerar as configurações aleatórias de uma maneira muito mais criteriosa, diminuindo o número de amostras n necessárias para resolver o problema. Sendo assim, mesmo que T_s aumente, o tempo global é reduzido pela significativa redução de n.

A estratégia de amostragem gaussiana apresenta bom desempenho em situações onde o mapa tem grandes áreas sem obstáculos e algumas poucas passagens estreitas. Sendo assim, o algoritmo amostra basicamente os locais difíceis do mapa, evitando colocar pontos em grandes áreas abertas. Seu funcionamento consiste em criar as configurações aleatórias através dos seguintes passos: inicialmente uma configuração

aleatória c_1 é gerada. Em seguida, uma configuração c_2 é gerada seguindo

$$c_2 \sim N\left(c_1, \delta_d^2 \cdot \mathbf{I}_2\right) \quad (2)$$

ou seja, de distribuição normal com média c_1 e matriz de covariâncias $\delta_d^2 \cdot I^2$, em que $\mathbf{I}_2$ é a matriz identidade de ordem 2. δ_d^2 é a variância da dispersão do ponto c_2 com relação a c_1. Quanto maior δ_d^2 , maior será esta dispersão. Se c_1 pertence ao espaço livre do mapa e c_2 colide com algum obstáculo, então c_1 é adicionado ao grafo. Caso contrário, se c_2 pertence ao espaço livre e c_1 colide com algum obstáculo, então c_2 é adicionado ao grafo. Se nenhuma das condições anteriores forem atendidas, então ambas configurações são descartadas.

4 O ROBÔ OMNIDIRECIONAL OMNI

O robô Omni foi concebido pelo Departamento de Robótica do *Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier* (LIRMM), Montpellier, França. Atualmente Omni está cedido a título de empréstimo para a Universidade de Brasília (UnB). Seu sistema sensorial avançado permitiu a realização de diversos trabalhos em controle de movimento, controle de trajetória, e cartografia e localização simultâneos (Borges, 2002). Na Figura 1 é apresentada uma foto do Omni.

Figura 1: Robô Omni (Borges, 2002).

Esse robô é dotado de avançados sensores proprioceptivos e esteroceptivos. Os sensores proprioceptivos são seis codificadores ópticos incrementais, três codificadores ópticos absolutos e um giroscópio laser de alta precisão. Como sensores exteroceptivos estão um radar a laser 2D, que fornece um mapa de profundidade em 270^o com resolução de $0,6^o$, alcance de $30m$ e precisão de $3cm$ na medição de distâncias. Em sua versão original uma câmera monocromática complementava o conjunto de sensores exteroceptivos. Autalmente esta câmera foi substituída por um par de câmeras estéreo.

Para o planejamento de trajetórias, Omni apresenta uma característica peculiar, dando-lhe vantagem quando da aplicação de algoritmos de planejamento: ele é omnidirecional. Seus 350 kg são movidos por três rodas orientáveis e tracionáveis, possuindo assim seis graus independentes de motorização, superior aos três graus de liberdade de posicionamento. Isto implica em menos restrições para o planejamento de trajetórias quando comparado a robôs a tração diferencial. Um microcomputador Pentium II 300 MHz rodando Windows NT acrescido da extensão tempo real RTX é interfaceado com os sensores e atuadores através de placas de E/S de alta velocidade (12 Mbps).

5 IMPLEMENTAÇÃO DO MRP PARA O ROBÔ OMNI

5.1 Fase de aprendizado

Apenas a sub-etapa de construção foi implementada na fase de aprendizado. O problema de se capturar a conectividade dos locais mais difíceis do espaço livre C-*free* por enquanto foi resolvida adicionando um número maior de configurações aleatórias no mapa.

Nesta implementação, ainda foi determinado de forma empírica o número de configurações aleatórias geradas na etapa de aprendizado, que é um fator determinante para o bom desempenho na etapa de questionamento.

Para cada nova configuração aleatória c é verificada se esta se encontra dentro do mapa e se não colide com algum obstáculo ou com alguma configuração aleatória anterior. Se isto ocorrer, a configuração é descartada. Caso contrário, a configuração é armazenada em N no grafo R. Os valores armazenados são as coordenadas (x, y) do centróide do robô e ainda o componente ao qual a configuração pertence.

Definiu-se na etapa de implementação que os nós vizinhos à c encontram-se dentro de um raio de visibilidade determinado pela distância $D(c)$. Para cada nó vizinho de c é verificado inicialmente se há obstáculo entre esse nó vizinho e c. Esta verificação é feita por um planejador local extremamente simples.

O planejador local simplesmente discretiza uma semi-reta que liga o nó c e o seu vizinho n_c em questão e coloca configurações intermediárias do robô. Caso nenhuma configuração aleatória colida com algum obstáculo, ocorre a verificação para ver se os dois nós estão em componentes separados. Se não pertencerem ao mesmo componente, uma borda ligando c e n_c é adicionada em E e todos os componentes do grafo são atualizados. Caso contrário a borda não é adicionada a fim de evitar ciclos no grafo.

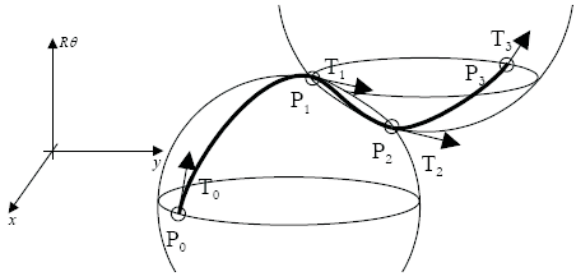

Figura 2: Interpolação da trajetória (Aragones, 2002).

5.2 Fase de questionamento

Na fase de questionamento são fornecidos os pontos de início Ps e destino do robô Pg. Estes pontos são conectados aos componentes mais próximos, nos pontos $Ps\prime$ e $Pg\prime$, respectivamente. Caso $Ps\prime$ e $Pg\prime$ estejam em componentes diferentes, o MRP retorna falha na geração da trajetória. Caso contrário, a geração não suavizada da trajetória é executada através da concatenação entre Ps e $Ps\prime$, $Ps\prime$ a $Pg\prime$ e $Pg\prime$ a Pg usando o planejador local, ou seja, concatenação de retas. A ligação entre $Ps\prime$ e $Pg\prime$ é feita através de uma busca por força bruta no grafo R e a sucessiva concatenação dos pontos intermediários.

Ao final da geração da trajetória global usando o planejador local, é feita a suavização usando um interpolador de trajetórias baseado em curvas de Bezier. Esse gerador é executado em tempo-real no Omni, servindo para gerar referência de configuração operacional para o controlador de trajetória do robô. Sua descrição resumida é feita a seguir.

5.3 Interpolação da trajetória por curva de Bezier

Em geral, o número de pontos intermediários gerado pelo MRP é pequeno, de modo que faz-se necessário dispor de um interpolador para gerar a trajetória a ser seguida pelo robô. Para robôs com restrição de movimento, um gerador apropriado é apresentado em (Fleury et al., 1995). Nessa seção é descrito um gerador baseado em curvas de Bezier, implementado no Omni (Aragones, 2002). O método consiste em construir uma trajetória entre os pontos gerados pelo MRP, ou pontos de passagem, e determinar as velocidades de referência de modo a possibilitar um movimento contínuo com derivadas finitas em todos os pontos do trajeto, resultando assim em um movimento suave.

A trajetória será composta por sucessivas curvas como mostrado na Figura 2. O segmento da trajetória é calculado enquanto o robô se encontra sobre o primeiro ponto de passagem. Dado três pontos consecutivos $(\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2)$ no espaço $\mathbf{R}^3$,$i.e.$, coordenadas (x, y, θ), traça-se um único círculo pelos três pontos. A tangente e a cuvartura do ponto intermediário $\mathbf{P}_1$ são definidos pela sua tangente e pelo raio do centro do círculo a este ponto. Com as informações do ponto anterior $\mathbf{P}_0$ e do ponto intermediário $\mathbf{P}_1$, são derivadas os vetores do pontos de controle $\mathbf{N}_i$ $(i = 1, 2..5)$ no espaço $\mathbf{R}^3$ e a trajetória entre os pontos $\mathbf{P}_0$ e $\mathbf{P}_1$ é interpolada por um polinômio do quinto grau dado por:

$$\begin{aligned}\mathbf{P}(u) = (1-u)^5 \cdot \mathbf{N}_0 + 5 \cdot u \cdot (1-u)^4 \cdot \mathbf{N}_1 \\ + 10 \cdot u^2 \cdot (1-u)^3 \cdot \mathbf{N}_2 + 10 \cdot u^3 \cdot (1-u)^2 \cdot \mathbf{N}_3 \\ + 5 \cdot u^4 \cdot (1-u) \cdot \mathbf{N}_4 + u^5 \cdot \mathbf{N}_5\end{aligned}$$

Nesse polinómio, $\mathbf{P}(u)$ é a expressão vetorial da trajetória parametrizada por u definido no inervalo $[0, 1]$. Tem-se então a postura do robô entre os pontos $\mathbf{P}_0$ e $\mathbf{P}_1$ em função do parâmetro u. Pelas características da trajetória construída, nota-se que os segmentos entre os dois primeiros pontos da trajetória, assim como os dois últimos são aproximados por um arco de círculo.

Uma restrição imposta no algoritmo da interpolação da trajetória é a distância d entre dois pontos de passagem consecutivos. Caso esta distância seja maior que um limiar d_i, os pontos gerados pelo interpolador de trajetória não serão suficientes para garantir a suavidade do movimento. Logo, se esta condição for encontrada, é realizada uma interpolação linear e somente após este processo que o interpolador de Bezier é aplicado para suavizar a curva.

6 RESULTADOS EXPERIMENTAIS

A avaliação dos resultados experimentais foi feita usando dois tipos de mapa. Sua execução foi feita sobre um microcomputador compatível IBM-PC com processador Celeron 2,2GHz com 512kB de memória RAM. A Figura 3 apresenta um mapa real gerado pelo sistema de localização e cartografia implementado no robô Omni e que foi proposto por (Borges, 2002). A trajetória foi gerada pelo planejador local através da concatenação de semi-retas e não está suavizada. Foram necessárias 85 configurações aleatórias para que o espaço livre pudesse ser capturado de forma satisfatória. Além disso, a trajetória tinha 18 pontos intermediários de passagem.

O tempo total para se realizar as etapas de aprendizagem para construir o mapa de rotas e de questionamento foi inferior a 1s.

Na Figura 4 a trajetória foi suavizada através do interpolador de trajetórias que utiliza curvas de Bezier. Apesar de haver um número pequeno de pontos de passagem, a trajetória ficou bem comportada e realmente suavizou a trajetória que tinha sido gerada pelo planejador local. Como o algoritmo implementado ainda não possui nenhuma técnica de otimização tal como a proposta em (Braga and Araújo, 2004), o planejador local fez com que a trajetória fizesse alguns caminhos desnecessários, sendo que o interpolador ainda manteve este problema.

Figura 3: Trajetória não suavizada em um mapa real.

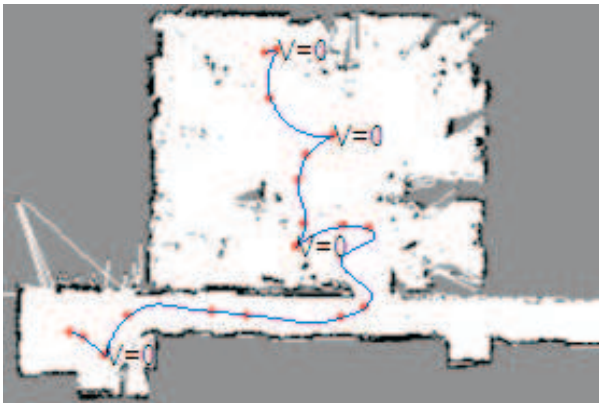

Figura 4: Curva suavizada com o interpolador de trajetórias.

Na Figura 5 foi realizada a geração de pontos de passagem intermediários. Nota-se claramente que a curva ficou mais parecida com a curva gerada pelo planejador local, uma vez que o número de pontos de passagem intermediários aumentou para 55. Ao mesmo tempo que isto mostra uma perda de suavização, mostra também que haverá menos chance de haver colisão na trajetória gerada pelo interpolador de trajetórias, uma vez que a curva não suavizada já havia sido checada para verificar colisões.

A Figura 6 é um mapa artificial com algumas dificuldades específicas. Ele possui passagens estreitas e um labirinto complexo por onde o robô tem que percorrer. Para capturar a conectividade do espaço livre, foram necessárias 288 configurações aleatórias. A trajetória gerada pelo planejador local possui 44 pontos de passagem interconectados por semiretas. Novamente a ausência de uma estratégia para melhorar a solução de navegação pelo planejador local faz com que o trajetória faça desvios desnecessários. A etapa de aprendizagem para a construção do mapa de rotas para a Figura 6 durou menos de 3 segundos e o questionamento foi respondido em menos de 1 segundo.

A Figura 7 mostra a trajetória suavizada usando o interpolador de trajetórias. Como pode ser observado, devido ao número pequeno de pontos de passagem a trajetória saiu um

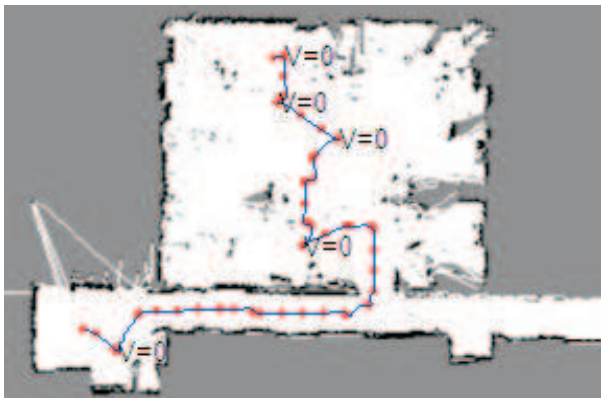

Figura 5: Curva suavizada utilizando geração intermediária de configurações aleatórias.

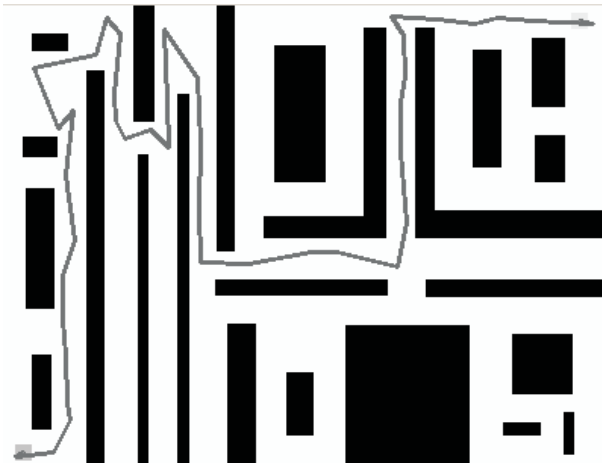

Figura 6: Mapa artificial com trajetória não suavizada.

pouco da rota original em alguns locais. Em algumas situações estes desvios podem fazer com que o robô colida com algum obstáculo, de forma que devem ser evitados.

A Figura 8 mostra o efeito do acréscimo de pontos intermediários na interpolação da trajetória usando curva de Bezier. Novamente, conforme esperado, a trajetória ficou bem comportada, porém menos suavizada, uma vez que o número de pontos de passagem neste caso foi 107.

A Figura 9 mostra uma situação comum nas implementações mais básicas do MRP, onde não há melhoria da trajetória gerada pelo planejador local, tampouco pela trajetória resultante da etapa de questionamento.

O algoritmo muitas vezes escolhe um caminho mais longo ocasionado justamente pela ausência de ciclos. Existem diversas maneiras de se minimizar este problema. Uma delas é fazer a suavização da trajetória ainda na etapa de aprendizado (Kavraki et al., 1996), ou ainda nos caminhos singulares construídos ao final da etapa de questionamento (Kavraki et al., 1996; Braga and Araújo, 2004).

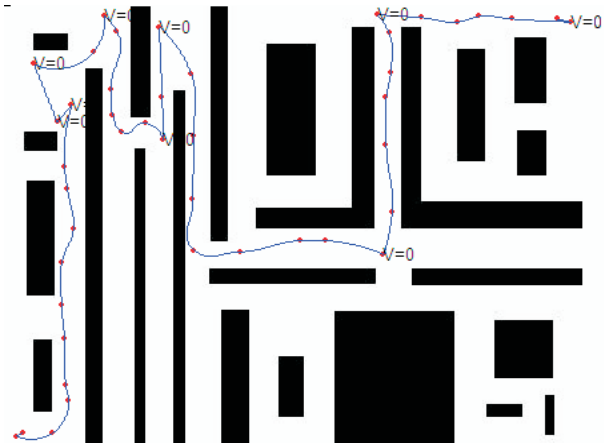

Figura 7: Trajetória suavizada.

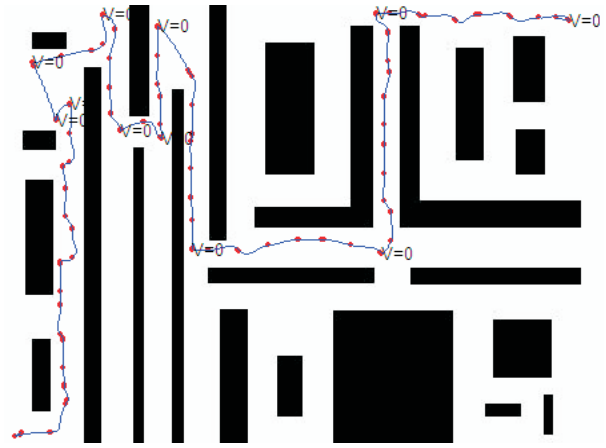

Figura 8: Trajetória suavizada com acréscimo de pontos intermediários.

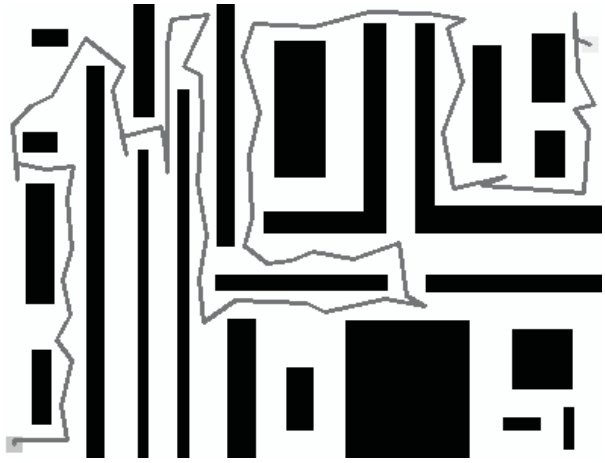

Figura 9: Trajetória com desvios desnecessários.

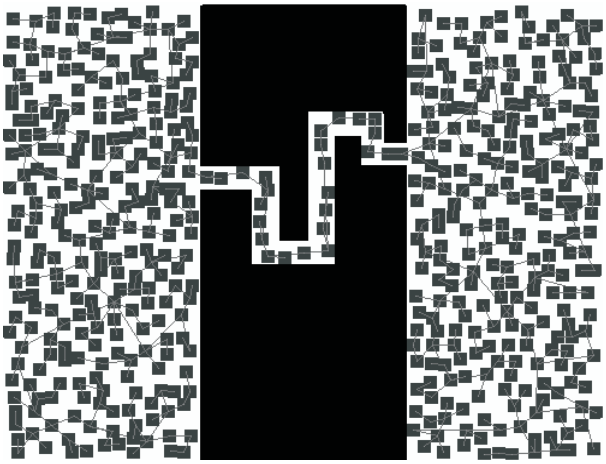

Figura 10: Geração de pontos usando a estratégia padrão de amostragem aleatória.

A Figura 10 mostra o exemplo clássico do mapa onde existem duas grandes áreas sem obstáculos ligadas por um corredor estreito. Foram necessários 61 segundos e 531 configurações aleatórias para amostrar o corredor de forma satisfatória.

A Figura 11 mostra o mesmo mapa, porém a geração de pontos é feita usando a estratégia de amostragem gaussiana apresentada por (Boor et al., 1999). Foram necessários menos de 1s e 104 configurações aleatórias para amostrar o corredor de forma satisfatória. Esses resultados mostram a melhoria de desempenho computacional obtida pela amostragem gaussiana em ambientes com grandes áreas sem obstáculos e algumas poucas passagens estreitas.

A implementação da etapa de questionamento consistiu basicamente em ligar os pontos de início e fim da trajetória (Ps e Pg, respectivamente) aos pontos $Ps\prime$ e $Pg\prime$ do grafo gerado na etapa de aprendizado, concatenar Ps a $Ps\prime$, achar um caminho no grafo entre $Ps\prime$ e $Pg\prime$ e concatenar $Pg\prime$ a Pg.

Na atual implementação o caminho entre $Ps\prime$ e $Pg\prime$ foi encontrado através da força bruta. Entretanto, como os maiores grafos gerados tinham por volta de 550 configurações aleatórias (no caso da Figura 10, por exemplo), essa busca por força bruta ainda ocorreu de forma muito rápida. Todas as situações de questionamento de trajetórias nos mapas mostrados anteriormente foram resolvidas em menos de 1 segundo.

7 CONCLUSÕES

Neste artigo foi apresentada a implementação do algoritmo Mapa de Rotas Probabilístico no planejamento de trajetórias de um robô móvel omnidirecional. Foi discutida a implementação de uma estratégia de suavização de trajetórias baseada em um interpolador por curvas de Bezier. Por último foi apresentada a implementação de uma estratégia de amostragem gaussiana para as configurações aleatórias.

O algoritmo Mapa de Rotas Probabilístico usando a estraté-

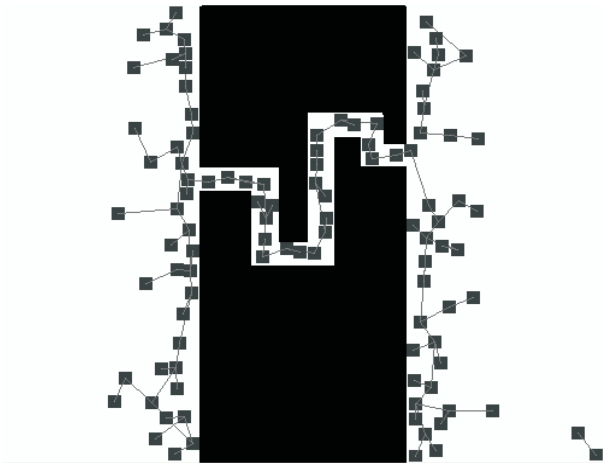

Figura 11: Geração de pontos usando a estratégia de amostragem gaussiana.

gia padrão de amostragem aleatória mostrou ter um bom desempenho para mapas estáticos com vários obstáculos. Contudo, para ambientes com grandes espaços vazios e apenas algumas passagens estreitas a estratégia de amostragem gaussiana obteve um desempenho significativamente maior.

Resultados obtidos em mapas de ambientes reais e ambientes simulados permitiram verificar o desempenho satisfatório deste tipo de planejador. E ainda, foram verificadas limitações que serão alvo de estudo nos próximos trabalhos.

Como trabalhos futuros sugere-se uma melhoria da trajetória global através de métodos eficientes de otimização e suavização de trajetórias e ainda a implementação da etapa de expansão da fase de aprendizado. Além disso, sugere-se uma investigação de outras técnicas de amostragem de pontos aleatórios e ainda um mecanismo mais eficiente para a busca da trajetória no grafo.

AGRADECIMENTOS

O robô Omni foi gentilmente emprestado à Universidade de Brasília (Brasil) pela Université Montpellier II (França), após um acordo entre as duas instituições.

REFERÊNCIAS

Aragones, J. (2002). *Commande et evaluation des performances d'un robot omnidirectionnel a roues*, PhD thesis, Universite Montpellier II Sciences et techniques du Languedoc.

Aragones, J., Borges, G. A. and Fournier, A. (2002). Accuracy improvement for a redundant vehicle, *33rd International Symposium on Robotics*.

Barraquand, J., Kavraki, L., Latombe, J. C. and Li, T. (1997). A random sampling scheme for path planning, *The International Journal of Robotics Research* **16**: 759–774.

Berg, J. P. and Overmars, M. H. (2004). Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners, *IEEE Internation Conference on Robotics and Automation* pp. 453–460.

Boor, V., Overmars, M. H. and Stappen, A. F. (1999). The gaussian sampling strategy for probabilistic roadmap planners, *Proceedings of the 1999 IEEE International Conference on Robotics and Automation* pp. 1018–1023.

Borges, G. A. (2002). *Cartographie de l'environnement et localisation robuste pour la navigation de robots mobiles*, PhD thesis, Université Montpellier II, LIRMM, 161 rue ADA, 34392, Montpellier, Cedex 5, France. *One of the recipients of the 2001/2002 Club EEA prize for the best french thesis in Automatic Control*.

Braga, G. A. and Araújo, A. F. R. (2004). Uma estratégia para a melhoria das soluções de navegação móvel baseadas em mapas topológicos, *Congresso Brasileiro de Automática* .

Fleury, S., Sourères, P., Laumond, J.-P. and Chatila, R. (1995). Primitives for smoothing mobile robot trajectories, *IEEE Transaction on Robotics and Automation* **11**(3): 441–448.

Hu, Y. and Yang, S. X. (2004). A knowledge based genetic algorithm for path planning of a mobile robot, *IEEE International Conference on Robotics and Automation* pp. 4350–4355.

Kavraki, L. E., Kolountzakis, M. N. and Latombe, J. C. (1998). Analysis of probabilistic roadmaps for path planning, *IEEE Transactions on Robotics and Automation* **14**(1).

Kavraki, L. E., Kolountzakis, M. N., Latombe, J. C. and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Transactions on Robotics and Automation* **12**(4).

Overmars, M. H. (1992). A random approach to motion planning, *Technical Report RUU-CS-92-32*, Dept. Comput. Sci., Utrecht University.