UM MÉTODO DE PLANEJAMENTO DE TRAJETÓRIA PARA ROBÔS MÓVEIS ATRAVÉS DE PASSEIOS ALEATÓRIOS ADAPTATIVOS E MAPA DE ROTAS

Bruno Vilhena Adôrno*, Geovany Araújo Borges*

*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 - Brasília - CEP 70910-900 - Brasil

Emails: bvabr@yahoo.com.br, gaborges@unb.br

Abstract— This paper proposes a new stochastic algorithm for mobile robot path planning. This algorithm is a hybrid version of Probabilistic Roadmap (PRM) and Adaptive Random Walking (ARW) methods. The main improvement captures the best features of PRM and ARW, related to high path quality and low computing time. An implementation of this algorithm is presented and the results are compared with the original algorithms.

Keywords— Path planning, mobile robotics, random walking, stochastic processes

Resumo— Este artigo propõe um novo algoritmo estocástico para o problema de planejamento de trajetória de robôs móveis. O algoritmo é um híbrido do Mapa de Rotas Probabilístico (MRP) e Passeio Aleatório Adaptativo (PAA). No entanto, ele captura as melhores características dos algoritmos originais, que são a alta qualidade da trajetória gerada, como também o baixo tempo computacional requerido. Uma implementação para o algoritmo proposto é apresentada e resultados obtidos com esse algoritmo são comparados com aqueles obtidos com os algoritmos em que ele foi baseado.

Palavras-chave— Planejamento de trajetória, robótica móvel, passeio aleatório, processos estocásticos.

1 Introdução

O problema do planejamento de trajetória no contexto da robótica móvel vem sendo estudado por muitos pesquisadores desde o final da década de 60. Ele consiste basicamente em, dados pontos de origem e destino, criar uma seqüência de ações que fará com que o robô saia do ponto de origem e chegue ao ponto de destino.

Este problema tem mostrado ser bastante difícil, dado que sua solução requer um tempo exponencial em relação ao número de graus de liberdade do robô, além de ser feita sem a intervenção humana (Barraquand et al., 1997), (Kavraki et al., 1996). Entretanto, devido à criação de algoritmos mais eficientes e o crescente aumento do poder computacional dos processadores, o problema do planejamento de trajetória extrapolou o universo da robótica móvel e passou a ter utilidade em outras aplicações, tais como animação de atores artificiais para filmes ou jogos de computador, robôs cirurgiões (Barraquand et al., 1997), problemas de montagem/desmontagem (Lingelbach, 2004) e biologia molecular (Kavraki et al., 1998).

Várias abordagens foram propostas desde o início das pesquisas sobre o problema do planejamento de trajetória. Contudo, os algoritmos completos se mostraram impraticáveis devido ao tempo de processamento que requerem (Barraquand et al., 1997), (Berg and Overmars, 2004). Apesar deste fato, recentemente houve uma ênfase nas pesquisas de algoritmos não somente probabilisticamente completos (Barraquand et al., 1997) como também de solução completa e até mesmo determinísticos, apesar de o grande enfoque ainda ser nas abordagens estocásticas (Carpin and Pillonetto, 2005).

Uma grande vantagem dos algoritmos estocásticos quando comparados aos determinísticos é a eficiência no tempo de execução (Barraquand et al., 1997). Contudo, uma grande desvantagem é a ausência de repetibilidade, ou seja, duas execuções não irão ocorrer de forma idêntica. Isso dificulta a depuração dos resultados e às vezes algumas falhas podem passar desapercebidas devido à aleatoriedade do processo (Lindemann and LaValle, 2003). Além disso, a trajetória resultante gerada por muitos planejadores estocásticos tende a ter uma baixa qualidade, havendo a necessidade de fazer um pós-processamento para suavizá-la (Geraerts and Overmars, 2004), (Adorno et al., 2005).

Este trabalho apresenta um algoritmo de planejamento de trajetória cujo enfoque é a grande eficiência em tempo de execução. Ele foi baseado na fusão dos métodos PAA (Carpin and Pillonetto, 2005) e MRP (Kavraki et al., 1996), de forma a gerar trajetórias sem a necessidade de um pré-processamento do ambiente e armazenando-as em um mapa de rotas, podendo assim utilizá-las posteriormente. Portanto, o algoritmo proposto será referenciado por Mapa de Rotas criado por Passeio Aleatório Adaptativo (MRPAA). O resultado de sua implementação será comparado com os resultados obtidos com o PAA e ainda com a implementação do algoritmo MRP realizada em (Adorno et al., 2005).

O restante do artigo está organizado da seguinte forma: a Seção 2 faz a revisão bibliográfica

de dois métodos distintos de planejadores de trajetória estocásticos; a Seção 3 apresenta o Mapa de Rotas criado por Passeio Aleatório Adaptativo e a Seção 4 mostra uma avaliação dos resultados das implementações do PAA, MRP e MRPAA. Por fim, a Seção 5 apresenta as conclusões e propostas de trabalhos futuros.

2 Revisão Bibliográfica

2.1 Mapa de Rotas Probabilístico

O Mapa de Rotas Probabilístico (ou *Probabilistic Roadmap*, do inglês) foi inicialmente concebido em (Kavraki et al., 1996) e é dividido em duas fases: a fase de aprendizado e de questionamento. Ele é definido através de um grafo $R(N, E)$ que se localiza no espaço livre C_{free} do espaço de configurações C_{space} de dimensão m, em que m é o número de graus de liberdade do robô, N e E são os nós e as bordas de R, respectivamente.

A etapa de aprendizado consiste em expandir o grafo na região C_{free} de forma a tentar cobrir toda a parte livre do mapa, para assim poder gerar trajetórias rapidamente na etapa de questionamento. O objetivo é que um tempo maior seja usado numa etapa de pré-processamento do mapa através da fase de aprendizado, de forma que as trajetórias requisitadas na etapa de questionamento sejam quase instantâneas.

Na fase de aprendizado, gera-se sucessivamente várias configurações aleatórias no espaço livre C_{free}. Cada nova configuração c gerada é armazenada em um nó N do grafo R e então um planejador local verifica os vizinhos n_c aos quais c pode se conectar. Um exemplo de planejador local extremamente simples é aquele que discretiza o segmento de reta que liga c a cada um de seus vizinhos n_c, colocando então configurações intermediárias do robô, verificando colisões com obstáculos (Kavraki et al., 1996). Caso não haja colisão, se c e n_c não estiverem em um mesmo componente, uma borda E ligando esses dois nós é gerada no grafo R.

Um componente é dado por um conjunto de nós N ligados entre si através de bordas E. O objetivo de não ligar o novo nó gerado a um novo vizinho, caso ambos estejam no mesmo componente, é evitar criar ciclos no grafo, facilitando então nas buscas de trajetórias na etapa de questionamento.

Na fase de questionamento, dado um ponto de início Ps e um ponto de destino Pg, estes se ligam aos pontos Ps' e Pg' mais próximos no grafo R, respectivamente. Uma primeira verificação é feita para garantir que Ps' e Pg' estejam no mesmo componente.

Caso estejam em componentes diferentes, significa que não há uma trajetória ligando o ponto de origem ao de destino. Contudo, se estiverem em componentes iguais é gerada uma trajetória formada pela concatenação de Ps a Ps', todos os nós ligando Ps' a Pg' e Pg' a Pg.

2.2 Passeio Aleatório Adaptativo (PAA)

O Passeio Aleatório Adaptativo (ou *Adaptive Random Walk*, do inglês) apresentado em (Carpin and Pillonetto, 2005) se encontra na categoria dos algoritmos de planejamento de trajetória de questionamento único, ou seja, dado um ponto de início e um ponto de destino, uma trajetória é gerada em função desta escolha, sem a etapa de pré-processamento que ocorre em outros tipos de algoritmos, como, por exemplo, o MRP.

O PAA é dado por um processo estocástico de tempo discreto caracterizado por um passeio aleatório que cresce de um ponto de origem $\mathbf{x}_{start}$ com o objetivo de alcançar o ponto de destino $\mathbf{x}_{goal}$. A cada nova iteração do algoritmo, uma configuração $\mathbf{v}_k$ na vizinhança da última configuração é gerada de acordo com uma distribuição normal de média nula e uma matriz de covariâncias adaptativa $\mathbf{\Sigma}_k^2$, definida mais à frente por (5). Caso o planejador local descrito na Seção 2.1 e formalizado mais adiante consiga ligar as duas configurações, $\mathbf{v}_k$ é adicionado à cadeia de elementos gerada pelo processo estocástico e a matriz de covariância adaptativa é atualizada. Caso o planejador local não consiga ligar as duas configurações, $\mathbf{v}_k$ é descartado.

O objetivo principal da atualização da matriz de covariâncias é fazer com que as amostras geradas em torno da última configuração do passeio aleatório sejam polarizadas pela história H do processo, de forma a fazer com que essa geração de amostras seja adequada ao formato da região do mapa onde esses pontos se encontram. É essa atualização da matriz de covariâncias que determina a adaptabilidade do passeio aleatório (Carpin and Pillonetto, 2005).

A história H do processo simplesmente define o número de elementos da cadeia gerada pelo processo que serão utilizados para a atualização da matriz de covariâncias. O ajuste deste parâmetro é feito de forma empírica e depende da topologia do mapa no qual esse planejador vai atuar, sendo que grandes mudanças em seu valor não promovem uma melhoria no desempenho do algoritmo (Carpin and Pillonetto, 2005). Contudo, a existência de adaptabilidade atualiza a matriz de covariâncias e direciona o algoritmo para uma melhor amostragem.

Uma maneira de tornar este algoritmo mais eficiente no que se refere ao tempo de execução é fazer uma implementação bidirecional, ou seja, dois passeios aleatórios são gerados, de forma que um inicia no ponto de origem e o outro inicia no ponto de destino (Carpin and Pillonetto, 2005).

Caso as duas caminhadas aleatórias se encontrem, a trajetória final será a união das duas.

Caso o passeio originado no ponto x_{start} encontre o ponto final, ou caso a caminhada originada no ponto $\mathbf{x}_{goal}$ encontre o ponto inicial, então uma solução terá sido alcançada.

Outra estratégia utilizada para otimizar o PAA é o oportunismo (Carpin and Pillonetto, 2005). Essa estratégia faz com que a cada número fixo de iterações o algoritmo tente conectar o último elemento gerado pelo passeio aleatório ao seu ponto de destino.

Um problema do PAA é a qualidade muito baixa da trajetória resultante. Em geral ela possui muitos ciclos e desvios desnecessários. Uma maneira de amenizar este problema é fazer um pós-processamento da trajetória com o objetivo de suavizá-la.

A ênfase deste algoritmo é na eficiência de tempo de execução e na sua garantia de convergência. A formalização da convergência do algoritmo é dada em (Carpin and Pillonetto, 2005) e não será explicitamente abordada neste trabalho.

O planejador de trajetórias é definido no espaço de configurações C_{space} e no seu subconjunto C_{free}, de forma que são dadas uma configuração inicial $\mathbf{x}_{start}$ e uma final $\mathbf{x}_{goal}$, ambas vetores coluna de dimensão $N \times 1$, em que N representa o número de graus de liberdade do robô.

Tem-se ainda que, sendo $\mathbf{x}_{start} \subset C_{free}$ e $\mathbf{x}_{goal} \subset C_{free}$, o objetivo é achar um caminho que liga $\mathbf{x}_{start}$ a $\mathbf{x}_{goal}$ através de uma função contínua $\mathbf{f} : [0,1] \rightarrow C_{free}$ tal que $\mathbf{f}(0) = \mathbf{x}_{start}$ e $\mathbf{f}(1) = \mathbf{x}_{goal}$.

Além disso, é definido um planejador local que seja capaz de avaliar se um par de configurações $\{\mathbf{x}_1, \mathbf{x}_2\}$ é válido, o que se verifica se e somente se

$$t\mathbf{x}_1 + (1-t)\mathbf{x}_2 \in C_{free} \tag{1}$$

para todo $t \in [0,1]$. Assim sendo, uma função $\mathbf{g}$ é definida de forma que $\mathbf{g} : C_{space} \times C_{space} \rightarrow C_{space}$, tal que

$$\mathbf{g}(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \mathbf{x}_1 + \mathbf{x}_2 & \text{, se } \{\mathbf{x}_1, \mathbf{x}_1 + \mathbf{x}_2\} \\ & \text{é um par válido} \\ \mathbf{x}_1 & \text{, caso contrário} \end{cases} \tag{2}$$

Define-se ainda um processo estocástico em tempo discreto que gera amostras $\mathbf{x}_k$ em C_{space}, utilizando as seguintes equações recursivas

$$\begin{aligned} &\mathbf{x}_0 = \mathbf{x}_{start} \\ &\mathbf{x}_k = \mathbf{g}\left(\mathbf{x}_{k-1}, \mathbf{v}_k\right), \text{para } k = 1, 2, 3 \ldots \end{aligned} \tag{3}$$

Tem-se ainda que

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_k^2) \tag{4}$$

em que $\mathbf{\Sigma}_k^2$ é a matriz de covariâncias adaptativa do processo, dada por

$$\mathbf{\Sigma}_k^2 = \max\left(diag\left(\mathbf{P}_k\right), \mathbf{\Sigma}_{min}^2\right) \tag{5}$$

com

$$\mathbf{P}_k = \left(\frac{1}{H}\sum_{i=k-H}^{k-1} \mathbf{x}_k \cdot \mathbf{x}_k^T\right) - \bar{\mathbf{x}}_k \cdot \bar{\mathbf{x}}_k^T \tag{6}$$

e

$$\bar{\mathbf{x}}_k = \frac{1}{H}\sum_{i=k-H}^{k-1} \mathbf{x}_i \tag{7}$$

em que H representa o número de amostras que serão levadas em consideração no processo para a realização da média.

Cada elemento da cadeia gerada pelo processo (3) representa uma configuração no espaço de configuração C_{free}. Ainda levando em consideração (3), nota-se que o novo elemento $\mathbf{x}_k$ gerado terá média igual à configuração gerada por $\mathbf{x}_{k-1}$ e matriz de covariâncias dada por (5). É importante ressaltar que (Carpin and Pillonetto, 2005) usa apenas os elementos da diagonal da matriz de covariâncias, o que implica dizer que, para um espaço de configuração bidimensional dado pelas coordenadas (x, y) do robô no mapa, a elipse 3-sigma terá seus focos alinhados sempre na mesma direção dos eixos cartesianos.

3 Mapa de Rotas Criado por Passeio Aleatório Adaptativo (MRPAA)

O algoritmo MRPAA foi baseado na fusão dos métodos MRP e PAA. Ele herda as boas características de exploração de ambientes do PAA para fazer o planejamento da trajetória e depois armazená-las em um mapa de rotas representado por um grafo. Esse mapa de rotas é utilizado posteriormente de forma bem semelhante à da etapa de questionamento do MRP, de forma que questionamentos subseqüentes do MRPAA tenham a possibilidade de fazer a busca da trajetória no grafo, tornando-o bastante eficiente.

Basicamente, dados um ponto de início Ps e um ponto de destino Pg, o algoritmo segue os seguintes passos:

1. Ps e Pg tentam se ligar aos pontos Ps' e Pg' no mapa de rotas através do planejador local.
2. Caso nenhum dos dois consiga se ligar ao mapa de rotas, o PAA bidirecional é utilizado para tentar achar um caminho que ligue esses dois pontos. Caso exista uma trajetória ligando Ps a Pg, esta é suavizada utilizando o método proposto em (Carpin and Pillonetto, 2005) e armazenada explicitamente no mapa de rotas.
3. Caso um dos pontos se ligue ao mapa de rotas e o outro não, um PAA bidirecional tradicional é feito entre esses dois pontos. Caso exista uma trajetória que ligue Ps a Pg, essa trajetória é suavizada e adicionada ao grafo, expandindo o mapa de rotas.

Tabela 1: Tempos de execução do MRPAA

	T_{min}(ms)	T_{max}(s)	$T_{med}(s)$
Labirinto	0,897	1,060	0,351
Mapa real	0,339	0,262	0,060
Corredor	0,955	12,769	1,597

Tabela 2: Tempos de execução do PAA

	T_{min}(ms)	T_{max}(s)	$T_{med}(s)$
Labirinto	237,03	4,561	1,539
Mapa real	33,10	0,365	0,154
Corredor	1008,56	15,284	4,399

4. Caso existam dois ramos diferentes do mapa de rotas que não estão ligados entre si e caso um ponto Ps se ligue ao ponto Ps' em um ramo e o ponto Pg se ligue ao ponto Pg' em outro ramo, um PAA bidirecional é feito entre esses dois pontos. Caso exista uma trajetória ligando Ps a Pg, esta é suavizada e adicionada ao grafo. Sendo assim, esses ramos outrora desconexos passam a ser conectados.

5. Caso Ps e Pg se liguem a Ps' e Pg' e estes estejam em ramos interconectados, é gerada uma trajetória formada pela concatenação de Ps a Ps', todos os nós ligando Ps' a Pg' e Pg' a Pg.

Como a busca em um grafo geralmente é muitas vezes mais rápida que um passeio aleatório adaptativo, a tendência é que ao longo do tempo as diferentes trajetórias sejam geradas mais rapidamente à medida que o mapa de rotas vai se expandindo pelo ambiente. No pior dos casos, um passeio aleatório adaptativo vai ser feito na tentativa de ligar o ponto de origem ao de destino.

O algoritmo de suavização proposto em (Carpin and Pillonetto, 2005) e implementado neste trabalho basicamente faz uma busca binária no conjunto de pontos de passagem que compõe a trajetória, tentando eliminar sempre pontos intermediários que compõem desvios desnecessários na trajetória. Este algoritmo é executado até que não se consiga mais suavizar a trajetória.

4 Resultados

A avaliação dos resultados foi feita usando os mesmos mapas usados em (Adorno et al., 2005). A execução do ambiente de avaliação ocorreu em um computador compatível IBM-PC, com processador Pentium Celeron 2,2 GHz e 512MB de memória RAM.

A avaliação dos algoritmos se deu em duas etapas: uma etapa quantitativa e outra etapa qualitativa. Na etapa quantitativa foi avaliado o desempenho computacional do algoritmo em relação ao tempo de execução. Os resultados numéricos encontram-se nas Tabelas 1, 2, 3 e 4.

Tabela 3: Tempos de execução do MRP - Aprendizado

	T_{min}(s)	T_{max}(s)	$T_{med}(s)$
Labirinto	2,706	3,264	3,093
Mapa real	0,127	0,146	0,137
Corredor	63,985	70,990	68,638
Cor. Gauss	1,350	1,648	1,457

Tabela 4: Tempos de execução do MRP - Questionamento

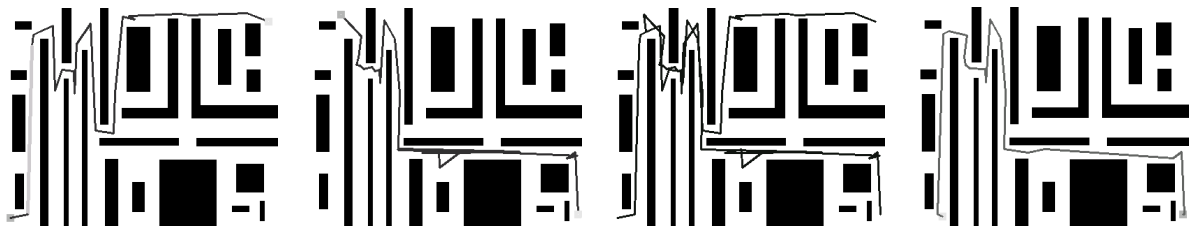
	T_{min}(ms)	T_{max}(ms)	$T_{med}(ms)$
Labirinto	0,572	1,590	0,917
Mapa real	0,417	1,078	0,638
Corredor	0,666	0,841	0,722
Cor. Gauss	0,505	0,815	0,673

Para avaliação dos algoritmos PAA e MRPAA, os tempos de execução considerados foram referentes ao planejamento da trajetória e à sua suavização. Além disso, para cada ambiente, oito trajetórias foram geradas, de forma que os pontos de início de cada trajetória foram gerados sucessivamente no sentido anti-horário e os pontos de destino foram posicionados na extremidade oposta do mapa. Sendo assim, a primeira trajetória começou pelo vértice superior direito e terminou no vértice inferior esquerdo. A segunda trajetória começou no meio da borda superior do mapa e terminou no meio da borda inferior deste ambiente e assim sucessivamente, até que a última trajetória começasse no meio da borda direita e terminasse no meio da borda esquerda do mapa.

Para avaliação do algoritmo MRP foram consideradas três etapas de aprendizado, sendo que para cada uma dessas etapas quatro trajetórias foram geradas, seguindo a mesma metodologia anterior. Sendo assim, a última trajetória gerada começou no meio da borda esquerda e terminou no meio da borda direita. Além disso, para o ambiente do corredor usado em (Adorno et al., 2005) também foi utilizado o método de amostragem gaussiana proposto em (Boor et al., 1999) e implementado em (Adorno et al., 2005).

A avaliação qualitativa foi feita apenas para o algoritmo proposto MRPAA, sendo que foram avaliados a qualidade das trajetórias resultantes, assim como o procedimento de geração do mapa de rotas. Para tanto, foram utilizados um ambiente real e outro ambiente artificial.

A Figura 1(a) mostra um mapa real do tipo grade de ocupação obtido a partir de experimentos com o robô Omni (Borges, 2002). Para o primeiro questionamento, como não existe um mapa de rotas o MRPAA atua como um PAA bidirecional, sendo que o tempo necessário para gerar a rota foi



(a) Primeira trajetória (b) Segunda trajetória (c) Grafo resultante (d) Trajetória utilizando o mapa de rotas

Figura 2: Resultados com ambiente artificial para o MRPAA.

(a) Primeira trajetória (b) Segunda trajetória

Figura 1: Resultados com ambiente Real para o MRPAA.

de 127ms. Para o segundo questionamento da Figura 1(b) o MRPAA utilizou o mapa de rotas que já havia sido criado, de forma que foi necessário apenas 0,57ms para gerar esta trajetória. Nota-se que o aumento do desempenho ao utilizar o mapa de rotas foi considerável. Além disso, a trajetória resultante ficou suave e sem desvios desnecessários.

A Figura 2(a) mostra um mapa artificial complexo, com corredores e passagens estreitas, sendo feito o primeiro questionamento ligando duas extremidades do ambiente. Como ainda não existe mapa de rotas, o algoritmo faz um passeio aleatório adaptativo bidirecional e leva 845ms para conseguir achar uma trajetória que ligue o ponto de origem ao destino.

Na Figura 2(b) o ponto de início encontra o mapa de rotas no canto superior esquerdo, mas não o ponto de destino no vértice direito inferior. Sendo assim, novamente um passeio aleatório adaptativo bidirecional liga os dois pontos, gerando a trajetória em 608ms. Nota-se no meio do mapa um ciclo desnecessário na trajetória. Isso se deu porque atualmente o algoritmo de suavização atua nos dois passeios aleatórios isoladamente, sendo que somente depois de suavizar e armazenar cada passeio aleatório é que eles são ligados. Uma maneira de evitar esse tipo de problema é concatenar os dois passeios aleatórios primeiramente e fazer a suavização da trajetória resultante.

A Figura 2(c) mostra o mapa de rotas resultante dos dois questionamentos anteriores. Pode ser observada uma grande redundância neste grafo, o que pode ser minimizado através de um processo de *prunning* do mesmo. Contudo, como o algoritmo de suavização é extremamente rápido, a solução adotada foi suavizar também as trajetórias que utilizam o mapa de rotas.

A Figura 2(d) mostra um questionamento que utilizou o mapa de rotas. Mesmo com toda a redundância e ainda com o ciclo desnecessário no meio do mapa, a trajetória resultante ficou suave e foi gerada em 2,9 ms.

Dessa forma, nota-se que o algoritmo proposto provê boas soluções em um tempo praticável. Além disso, ele consegue extrair as boas características de cada um dos algoritmos no qual ele foi baseado, isto é, o passeio aleatório adaptativo consegue fazer uma boa exploração de ambientes, enquanto que o mapa de rotas é utilizado para armazenar as trajetórias já realizadas, evitando que elas sejam calculadas novamente.

Quando comparado com o algoritmo PAA proposto em (Carpin and Pillonetto, 2005), o MRPAA provê trajetórias semelhantes, uma vez que o mapa de rotas é expandido através de passeios aleatórios adaptativos e o método de suavização é o mesmo. No entanto, o tempo médio de execução é bem inferior, uma vez que o MRPAA começa a utilizar o mapa de rotas com mais frequência a medida que aumenta o número de questionamentos realizados e o ambiente vai sendo explorado. Caso o ambiente tenha sido completamente explorado, o MRPAA torna-se muitas vezes superior, uma vez que ele passa a utilizar somente o mapa de rotas.

Já quando comparado com o algoritmo MRP proposto em (Kavraki et al., 1996), nota-se que para uma primeira trajetória o MRPAA consegue ser mais rápido, uma vez que o MRP usa uma grande parte do tempo de processamento para aprendizado do ambiente e conseqüente criação do mapa de rotas. Contudo, para questionamentos subseqüentes o MRP consegue fazer o cálculo da trajetória em um tempo inferior ao do MRPAA. Porém, à medida que o mapa vai sendo explorado pelo MRPAA, seu desempenho passa a ser semelhante ao do MRP.

Ainda comparando com o MRP, nota-se que o MRPAA tem uma característica importante de não ser tão sensível ao ambiente em que ele tem que explorar quanto o MRP. Um exemplo foi no

teste quantitativo da Tabela 1, em que para o ambiente do corredor o tempo médio de questionamento foi de 1,597s. Uma coisa importante a ser ressaltada é que esta média foi aumentada pela primeira exploração do MRPAA, que durou 12,779s. Contudo, todos os questionamentos posteriores utilizaram o mapa de rotas e foram bem mais rápidos, aproximando-se do tempo mínimo de planejamento. Já o MRP precisou, em média, de 68,638s usando o método de amostragem uniforme para conseguir uma boa conectividade do mapa de rotas que ligasse as duas extremidades do ambiente. Já usando a estratégia de amostragem gaussiana, esse tempo caiu para 1,457s.

Sendo assim, para um bom desempenho em alguns tipos de ambiente, ao se utilizar o MRP deve ser escolhido também o método de amostragem de configurações aleatórias, ou seja, há mais um ajuste de parâmetros a ser feito no algoritmo. Já o MRPAA necessita de menos ajustes para obter um bom desempenho em diferentes tipos de mapas.

5 Conclusões e Trabalhos Futuros

Neste artigo foi proposto um novo método de planejamento de trajetória, o MRPAA, baseado em dois algoritmos estocásticos: MRP e PAA. O método proposto utiliza o PAA para fazer o planejamento de trajetória, sendo que posteriormente elas são suavizadas e armazenadas em um mapa de rotas. Este pode, então, ser utilizado para planejamentos subseqüentes.

Resultados experimentais mostraram o bom desempenho computacional do algoritmo MRPAA quando comparados aos algoritmos nos quais ele se baseou. Além disso, as trajetórias resultantes se mostraram suaves.

Sugere-se como propostas de trabalhos futuros a investigação de métodos eficientes de caracterização dos mapas no sentido de extrair de forma consistente informações sobre regiões mais difíceis. Esta informação *a priori* sobre a topologia do mapa pode ajudar a polarizar a amostragem aleatória do passeio aleatório adaptativo, ajudando a minimizar o tempo de execução do algoritmo e garantido a alcançabilidade das regiões mais difíceis.

Além disso, também sugere-se a implementação de um método para simplificação do grafo criado pelo MRPAA, evitando assim redundâncias no mapa de rotas.

Referências

Adorno, B. V., Aguiar, C. S. R. and Borges, G. A. (2005). Planejamento de trajetória para o robô omni usando o algoritmo mapa de rotas probabilístico, *Simpósio Brasileiro de Automação Inteligente* pp. 1–8.

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). Roadmap-based motion planning in dynamic environments, *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 1598–1605.

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.*

Carpin, S. and Pillonetto, G. (2005). Motion planning using adaptive random walks, *IEEE Transactions on Robotics* **21**(1).

Geraerts, R. and Overmars, M. H. (2004). Clearance based path optimization for motion planning, *Proceedings of the 2004 IEEE Proceedings of the 2004 IEEE International Conference on Robotics & Automation* pp. 2386–2392.

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).

Lindemann, S. R. and LaValle, S. M. (2003). Current issues sampling-based motion planning, *Proceedings International Symposium on Robotics Research* .

Lingelbach, F. (2004). Path planning using probabilistic cell decomposition, *Proceedings of the 2004 IEEE International Conference on Robotics & Automation* pp. 467–472.