PLANEJAMENTO DE TRAJETÓRIAS PARA ROBÔS MÓVEIS EM AMBIENTES DESCONHECIDOS

GUILHERME DE LIMA OTTONI

Engenharia de Computação, Fundação Universidade Federal do Rio Grande
Av. Itália, Km 8, 96201-900 Rio Grande, RS, BRASIL
E-mail: ottoni@ecomp.furg.br

WALTER FETTER LAGES[1]

Depto. de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul
Av. Osvaldo Aranha, 103, 90035-190 Porto Alegre, RS, BRASIL
E-mail: w.fetter@ieee.org

Resumo- Este artigo apresenta um sistema de planejamento de trajetórias para robôs móveis operando em ambientes desconhecidos. O método proposto é baseado na decomposição do ambiente em células aproximadas. As únicas informações necessárias *a priori* para o robô são as suas posições (e orientações) inicial e final. Entretanto, informações sobre obstáculos inicialmente conhecidos também podem ser fornecidas. Os demais dados para o planejamento são obtidos pelo próprio robô através de um sonar de ultra-som. São discutidos detalhes sobre a implementação em tempo real do método, e do robô utilizado para testes. Resultados de simulação e experimentais validam a abordagem utilizada.

Abstract- This paper presents a path planning system for mobile robot working on unknown environments. The proposed method is based on approximated cell decomposition. The initial and final positions (and orientations) are all information that is required a priori. However other information initially known about the environment may also be provided. The remaining data used to develop the path plan are obtained by the robot using an ultrasound sonar. Details about the real time implementation of the method and about the robot are also presented. Simulated and experimental results validate the approach.

Keywords- path planning, decomposition methods, obstacle detection, real-time systems

1 Introdução

Um importante objetivo na área de robótica é a criação de robôs autônomos. Tais robôs devem aceitar descrições de alto nível das tarefas que eles devem fazer, sem a necessidade de maiores intervenções humanas. As descrições de entrada especificam o que o usuário deseja que seja feito, e não como proceder para fazê-lo. Estes robôs são equipados com atuadores e sensores sob controle de um sistema de computação.

O desenvolvimento da tecnologia necessária para robôs autônomos engloba algumas ramificações como raciocínio automatizado, percepção e controle, surgindo vários problemas importantes. Entre eles, encontra-se o planejamento de trajetórias. De uma forma geral, este problema consiste em se descobrir de que forma se pode levar um objeto a partir de uma configuração (posição e direção) inicial até uma configuração final. Um caso particular deste problema é quando o objeto que se deseja movimentar é o próprio robô.

Neste trabalho, serão descritos um método e a implementação de um planejador de trajetórias para robôs móveis em ambientes sobre os quais possui-se pouca informação. Este planejador é executado em tempo real, utilizando os sonares do robô para fazer a detecção dos obstáculos, e ainda interagindo com o controlador para fornecimento das trajetórias produzidas.

Ao final, são apresentados alguns resultados de simulação e experimentais obtidos com o sistema implementado.

2 O Robô Móvel

A figura 1 mostra uma foto do robô sobre o qual foi desenvolvido o presente trabalho. Este robô conta com um série de periféricos, dentre eles dois sonares de ultra-som, utilizados neste projeto para identificação de obstáculos. Ele possui duas rodas não orientáveis acopladas a motores, e outras duas rodas apenas para apoio. O computador é um PC com processador Pentium, o qual utiliza sistema operacional Linux. Para comunicação, ele possui uma placa de rede Ethernet e um modem sem fio. Quanto a sua geometria, ele possui forma cilíndrica com 1,35 m de altura e 0,30 m de raio.

[1] No período de realização deste trabalho, o autor estava vinculado à Fundação Universidade Federal do Rio Grande (FURG).

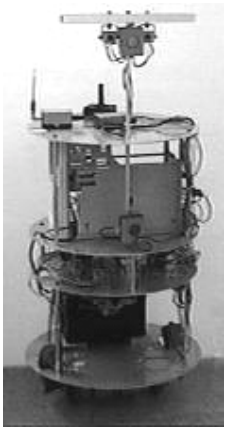


Figura 1. O robô móvel Twil.

Outros dispositivos que este robô possui incluem duas câmeras de vídeo, *joystick* e uma bússola digital. O diagrama de blocos da figura 2 ilustra como todos estes dispositivos interagem entre si.

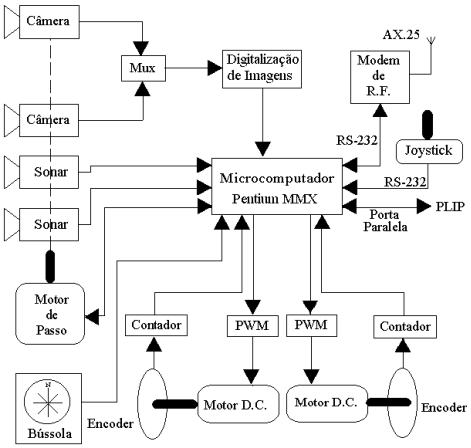


Figura 2. Diagrama de blocos do robô Twil.

3 Método

O método de decomposição em células consiste em dividir o espaço livre do robô em regiões simples (células), de forma que um caminho entre quaisquer duas configurações em uma mesma célula possa ser facilmente obtido. Um grafo não-dirigido representando a relação de adjacência entre as células é construído, e é então efetuada uma busca no mesmo. Este grafo é denominado *grafo de conectividade*. Os vértices que compõem este grafo são as células extraídas do espaço livre do robô. Há uma aresta entre dois vértices se e somente se as células correspondentes a eles são adjacentes. O resultado da busca efetuada é uma seqüência de células denominada *canal.* Um caminho contínuo pode ser computado a partir do canal.

Os métodos baseados em decomposição em células podem ser divididos ainda em *exatos* e *aproximados*:

Métodos exatos de decomposição em células decompõem o espaço livre em um conjunto de células cuja união cobre exatamente o espaço livre.

Métodos aproximados de decomposição em células dividem o espaço livre em um conjunto de células de forma predefinida cuja união está estritamente contida no espaço livre.

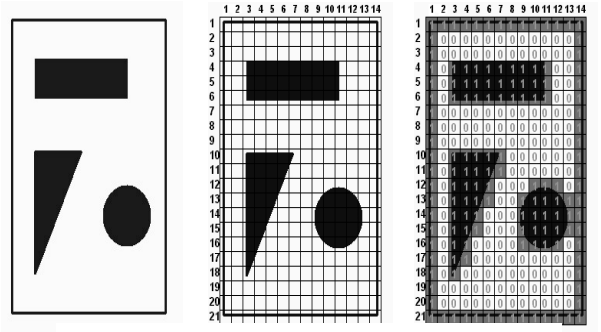
Nota-se que no método aproximado, todas as células possuem a mesma forma, a qual é bastante simples e definida *a priori*. Como conseqüência desta característica, tem-se que em geral não é possível modelar o ambiente na forma exata como ele é, sendo necessário que algumas aproximações sejam feitas. Isto deve ser feito de uma forma conservadora, ou seja, de maneira que garanta que o robô não colida com os obstáculos. Por este motivo, o espaço livre modelado aproximadamente através das células tem que estar estritamente contido no espaço livre real do robô. Pode decorrer disto que não seja encontrado algum caminho entre duas configurações, embora exista. Devido a esta característica, o método aproximado de decomposição em células é dito não completo, ao contrário do método exato, que é completo. Contudo, a principal vantagem do método aproximado sobre o exato é a de fazer a decomposição do ambiente em células através da iteração da mesma computação, que será simples por causa da forma das células. Assim, o método aproximado geralmente dá origem a sistemas mais simples de planejamento de trajetórias do que os que utilizam o método exato. E por isto, o método aproximado tem sido mais utilizado na prática (Latombe, 1991).

Vale acrescentar ainda que com o método aproximado, pode-se ajustar a precisão conforme desejado, mudando, para tanto, apenas o tamanho das células. Por estas razões, o modelo de planejamento de trajetórias implementado neste projeto baseia-se no método de decomposição aproximada em células.

Entretanto, é importante salientar que a possibilidade de ajuste da precisão deste método está intimamente relacionada à quantidade de espaço e de tempo de execução do planejador de trajetórias. Não deve-se definir previamente o tamanho das células, uma vez que este deve ser ajustado com base em diversos parâmetros. Células de tamanho grande reduzem a quantidade de memória necessária e, principalmente, o tempo de execução, o que é muito importante em sistemas que operam em tempo real, como o proposto neste trabalho. Por outro lado, células muito grandes podem causar uma perda de precisão indesejada, fazendo com que não sejam encontradas trajetórias mesmo em situações em que isto é possível.

A forma utilizada para representar o ambiente e seus obstáculos é bastante simples, e consiste em um mapa de *bits*. Este mapa deve conter um número de

dimensões igual ao da modelagem utilizada para o problema. Utiliza-se aqui o problema bidimensional, sendo o mapa de *bits* portanto uma matriz de duas dimensões. O ambiente real, incluindo seus obstáculos, está ilustrado na figura 3.a. Outra forma de representação do ambiente, também amplamente utilizada, é através de uma *quadtree*, como em Zelinsky (1992).



(a) (b) (c)

Figura 3. Decomposição aproximada em células.

Adota-se que cada célula representará uma região quadrada, de lado arbitrário (figura 3.b). Então cada posição do mapa de *bits* conterá 1 se a interseção da célula correspondente com os obstáculos for não nula. A figura 3.c ilustra isto. O planejamento da trajetória deve então escolher uma certa seqüência de células marcadas com 0.

Verifica-se que com este método, não há custo adicional em armazenar o grafo, e o custo de testar se duas células são adjacentes é muito baixo. Isto pode ser testado simplesmente verificando-se as diferenças dos índices correspondentes às células no mapa de *bits*. Se uma destas diferenças for nula, e a outra unitária, então diz-se que as células são adjacentes. Caso contrário, elas não o são.

Devido à geometria bastante simples do robô, pode-se aplicar a técnica de expansão de obstáculos Latombe (1991). Neste caso específico, para cada ponto onde se identifica a presença de um obstáculo é marcada no mapa de *bits* toda uma região quadrada de lado igual ao diâmetro do robô. Assim, pode-se considerar o robô como sendo um único ponto (o seu centro). Se este ponto não colidir com os obstáculos expandidos, certamente o robô real não colidirá com os obstáculos reais do ambiente.

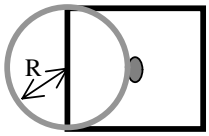


Figura 4. Expansão de obstáculos.

Além disto, este robô pode ser rotacionado sem que haja movimento de translação, o que permite que seja desconsiderada a sua orientação no espaço de configurações, obtendo assim um problema com duas dimensões.

Para efetuar a busca de uma trajetória no espaço de configurações, foi utilizado um algoritmo de busca em largura (Aho et al., 1983). Esta escolha foi feita com base em diversas características, como eficiência, garantia de encontrar uma solução sempre que possível, e facilidade de implementação. Considera-se duas células adjacentes conforme ilustrado na figura 5. As células em branco são parte do espaço livre do robô, enquanto que as escuras indicam a presença de um obstáculo. O robô inicialmente encontra-se na célula central. Então ele poderá se mover para qualquer uma das células adjacentes a esta célula inicial. Duas células são adjacentes caso elas possuam um lado comum e nenhuma delas esteja marcada como obstáculo.

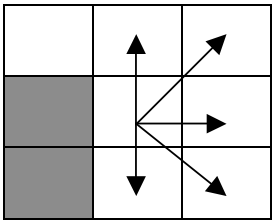


Figura 5. Células vizinhas (adjacentes).

Além disto, foi feita ainda a seguinte consideração. Duas células não marcadas como obstáculo serão consideradas adjacentes se elas possuírem um vértice em comum e as duas células que possuem lado adjacente a ambas as primeiras também não forem marcadas como obstáculo.

A trajetória gerada é fornecida ao controlador ao longo do tempo. Sobre a trajetória, é aplicada um perfil de velocidade constante, cujo valor é um parâmetro fornecido pelo usuário. Se os pontos da trajetória forem relativamente próximos, não será necessário fazer qualquer tipo de interpolação, ficando esta tarefa ao encargo do controlador.

4 Implementação

O sistema foi desenvolvido em linguagem C, utilizando-se *threads* (Leroy, 1996) para modelar cada uma das tarefas envolvidas no processo. A sincronização entre os *threads* foi feita através de semáforos. Os *threads* implementados foram os seguintes:

- *Thread* Principal: inicializa os demais *threads* e os semáforos, além de outras inicializações;
- *Thread* Gerador de Trajetória: faz a geração da trajetória a ser seguida pelo robô;
- *Thread* Fornecedor de Trajetória: passa a trajetória de controle para o controlador, aguardando que ele termine de fornecê-la;
- *Thread* Controlador: faz o controle da trajetória do robô, de acordo com o planejamento anterior;
- *Thread* Detector de Obstáculos: solicita leituras do sonar e marca no mapa de *bits* os obstáculos encontrados; invoca o Gerador de Trajetórias;

• *Thread* Leitor do Sonar: faz uma leitura do ambiente utilizando o sonar e detecta as regiões de profundidade constantes (Renon & Lages, 1999).

Destes, o *thread* controlador (Lages et al., 1998) e o leitor do sonar (Renon & Lages, 1999) foram adaptados de programas já desenvolvidos para o robô Twil para interagir com os demais *threads*.

A estrutura deste sistema com múltiplos *threads* está ilustrada na figura 6. Os *threads* estão representados pelas bolhas. Os retângulos ilustram os dispositivos físicos do robô que interagem com o sistema desenvolvido. As setas indicam o fluxo de comunicação entre eles. O funcionamento destes *threads*, inclusive a comunicação entre eles, são descritos detalhadamente em (Ottoni, 2000). A estrutura dos *threads*, bem como as interações entre eles, estão ilustradas na figura 6.

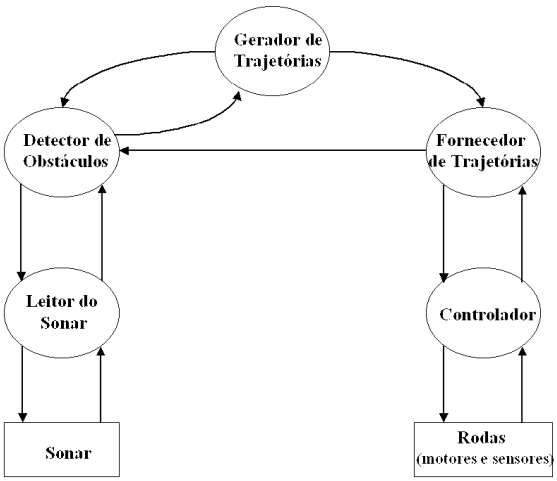


Figura 6. Estrutura de *threads* implementada.

5 Resultados de Simulação

Para testar e validar o método proposto, desenvolveu-se um simulador. Neste, os *threads* de leitura do sonar e controle são simulados por um outro *thread*, o qual tem conhecimento do ambiente real que se está simulando. Assim, este *thread* fornece os obstáculos que são detectáveis a partir de cada posição pelo sonar, e também simula que o robô seguia a trajetória, aproveitando desta situação para já verificar se ocorre alguma colisão do robô com os obstáculos do ambiente.

Na figura 7, tem-se um exemplo testado com o simulador desenvolvido. A figura 7.a é o ambiente real que está sendo simulado. O ponto marcado com *O* é a origem, ou seja, o ponto inicial do robô. O ponto *D* é o de destino. Neste teste, os obstáculos inicialmente fornecidos ao robô foram apenas as paredes que limitam o ambiente. Na figura 7.b, tem-se em cinza claro os obstáculos que foram informados *a priori* (externamente) e os que foram identificados na primeira varredura do ambiente com o sonar. Em preto, está traçada a primeira trajetória gerada pelo planejador de trajetórias, desde a posição inicial até a objetivo. Esta trajetória é seguida até a posição de início da trajetória da figura 7.c. Neste ponto, é feita nova varredura do ambiente através do sonar, e então uma nova trajetória é gerada com base nos obstáculos até agora identificados. Este processo é semelhante ao apresentado por Foux et al. (1993).

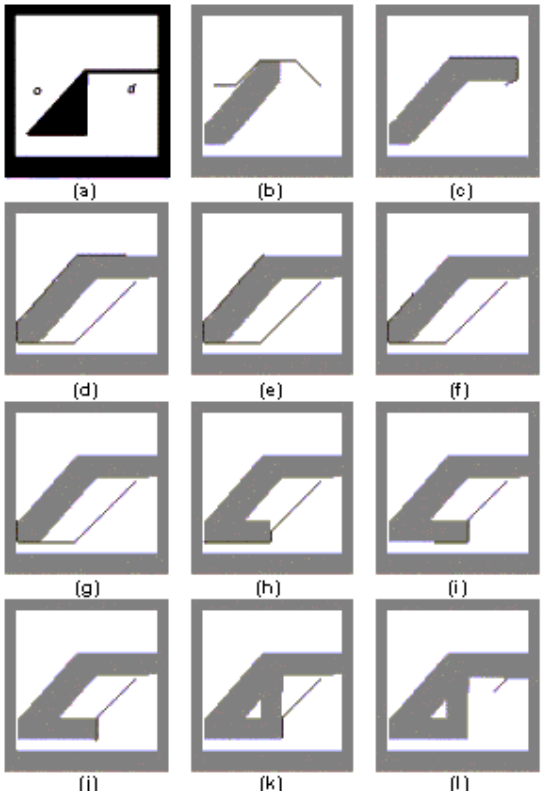


Figura 7. Exemplo de simulação.

A partir destas figuras, pode-se perceber que, ao ser identificado um ponto no ambiente que seja parte de um obstáculo, toda uma região quadrada de lado igual ao diâmetro do robô e centrada no ponto detectado é marcada no mapa como sendo obstáculo. Isto é o que corresponde à expansão dos obstáculos, e que permite que se considere o robô como um único ponto, o seu centro. As figuras 7.d a 7.l mostram as demais trajetórias geradas, para cada posição em que o robô parou para detectar novos obstáculos. A trajetória da figura 7.l é seguida até que o robô alcance a sua posição de destino.

6 Resultados Experimentais

Após os testes em simulação serem efetuados, e os *threads* de interação com os sonares e o controlador serem implementados, o programa desenvolvido foi finalmente testado no robô móvel Twil. Os resultados obtidos foram bastante satisfatórios, sendo que o robô adquiriu assim capacidade de locomover-se autonomamente por entre obstáculos detectados pelo uso de seus sonares. Apenas alguns problemas na detecção dos obstáculos foram identificados, como já era esperado, dada a natureza do método utilizado (uso de sonares). Notou-se também que a performance de tempo do modelo como um todo ficou prejudicada, uma vez que, com o programa utilizado para detecção de obstáculos, é necessário que o robô interrompa seu movimento para efetuar a identificação dos obstáculos através dos sonares.

Segue um exemplo significativo dos testes práticos utilizados, ilustrado na figura 8. Este teste foi efetuado em um dos laboratórios de eletro-eletrônica da FURG. Apesar de relativamente pequeno, este laboratório serviu de base para testes bastante realísticos, pois os obstáculos eram bastante reais, contendo várias irregularidades, o que causa ruído na detecção dos mesmos através do uso de sonares. A figura 8.a mostra apenas o contorno real deste laboratório, mas sem pequenas reentrâncias, as quais não podem ser identificadas através de sonares, mas que não são significativas dadas as dimensões do robô. Cada uma das figuras seguintes ilustra, para cada vez que o robô parou para detectar obstáculos, o mapa de bits do ambiente até então detectado e a trajetória completa gerada (mas que não será seguida até o final, conforme já explicado). Neste caso de teste, nenhum obstáculo do ambiente foi informado *a priori* ao robô. Por este motivo, verificamos a eficácia do sistema desenvolvido neste projeto, dados os resultados satisfatórios ilustrados na figura 8.

Finalmente, na figura 9, tem-se um último exemplo de experimento. Este teste foi feito no laboratório de Eletrotécnica do Departamento de Física da FURG. Este teste serviu principalmente para verificarmos a performance de tempo do sistema desenvolvido, uma vez que se trata de uma sala bastante grande (15 x 15 m) e que possui vários obstáculos. A discretização do ambiente foi feita em células quadradas de lado igual a 5 cm. Este valor mostrou-se adequado, uma vez que permitiu uma boa precisão na discretização do ambiente, não acarretando contudo em uma perda significativa na performance no planejamento das trajetórias.

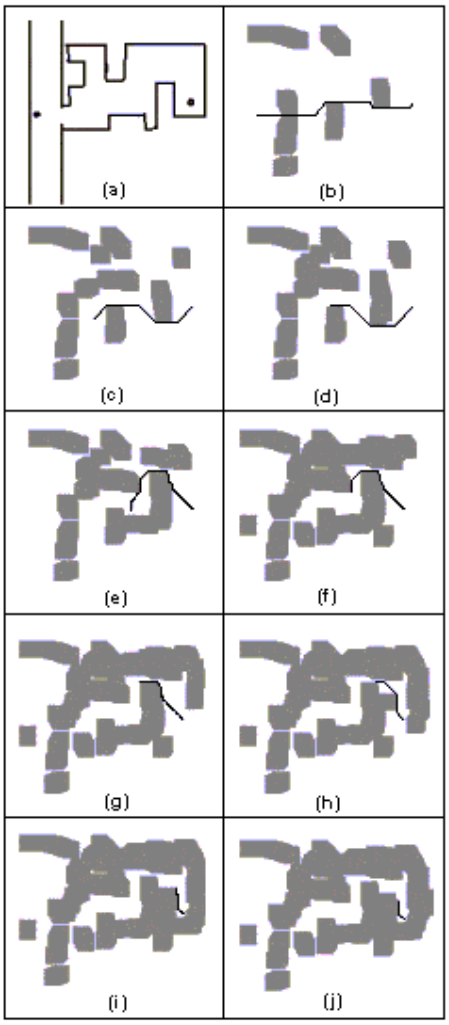


Figura 8. Teste realizado num dos laboratório de eletro-eletrônica.

O resultado obtido foi satisfatório no que tange ao alcance da posição desejada e planejamento da trajetória. Contudo, o tempo que o robô levou para atingir o seu objetivo foi relativamente elevado. Isto deveu-se ao grande número de vezes que ele precisou parar para detectar novos obstáculos. Percebeu-se também em outros testes que, com o método implementado, o robô às vezes não tenta passar em locais pelos quais ele poderia. Isto se deve à imprecisão dos sonares, e também à folga utilizada no raio do robô informado ao programa, necessária como medida de segurança em decorrência do primeiro problema.

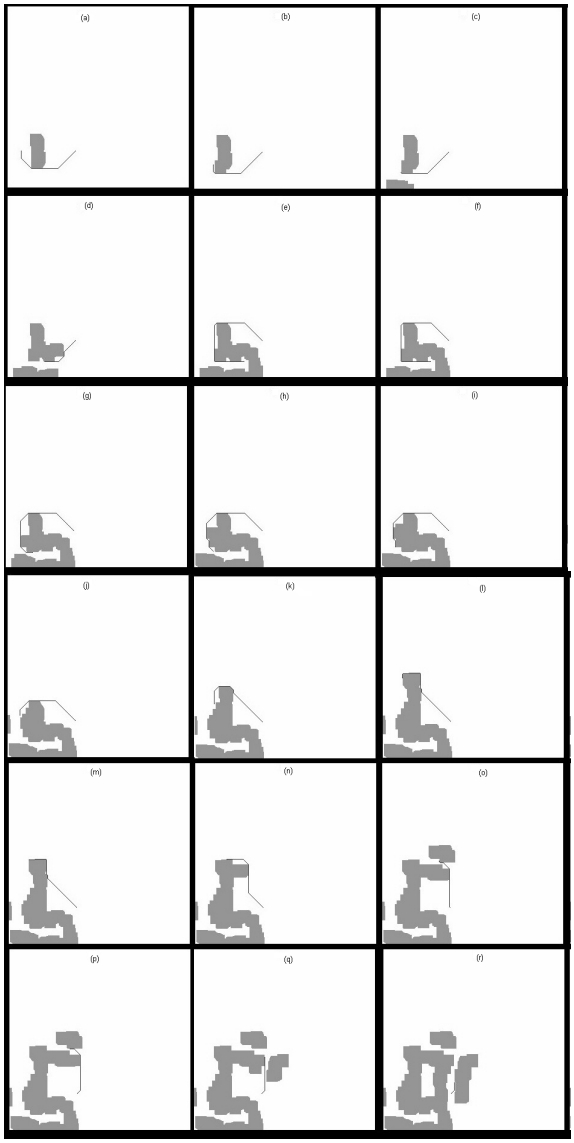


Figura 9. Teste realizado no laboratório de Eletrotécnica.

7 Conclusão

No presente trabalho foi descrito um método relativamente simples de planejamento de trajetórias de robôs móveis em ambientes desconhecidos. Este método foi implementado e aplicado a um robô real. Este planejador de trajetórias funciona em tempo real, interagindo com o sensoriamento do ambiente e controlador. Primeiramente desenvolveu-se um simulador para avaliar o método escolhido frente a situações do tipo proposto. Isto foi bastante válido, pois se pôde fazer alguns testes e ajustes necessários antes de efetivamente aplicar o programa desenvolvido no robô real. Depois disto, foi construído o programa efetivo, fazendo-se então a interface do planejador de trajetórias com os sonares, para detectar os obstáculos, e com o controlador, para fornecimento da trajetória. Conforme os resultados apresentados, tanto de simulação quanto reais, pôde ser verificada a eficiência do sistema desenvolvido. O principal problema encontrado foi referente ao uso dos sonares para detecção dos obstáculos, dada a sua imprecisão inerente. Entretanto, o sistema desenvolvido é suficientemente genérico, de forma que possa ser utilizado juntamente com outro método de detecção de obstáculos, como através de câmeras de vídeo, ou ainda através de métodos híbridos, como o proposto por Song & Tang (1996).

Referências Bibliográficas

Aho, A.V., Ullman, J.D., Hopcroft, J.E. (1983). Data Structures and Algorithms, Addison-Wesley.

Foux, G., Heymann, M., Bruckstein, A. (1993). Two-Dimensional Robot Navigation Among Unknow Stationary Polygonal Obstacles, IEEE Transactions on Robotics and Automation, vol. 9, no. 1.

Lages, W.F., Hemerly, E.M., Pereira, L.F.A. (1998). Controle Linearizante de uma Plataforma Móvel Empregando Realimentação Visual, XI Congresso Brasileiro de Automática, São Paulo.

Latombe, J. C. (1991). Robot Motion Planning, *Kluwer Academic Publishers*.

Leroy, X. (1996). The LinuxThreads Library, http://pauillac.inria.fr/~xleroy/linuxthreads/

Ottoni, G.L. (2000). Planejamento de Trajetórias para Robôs Móveis, Projeto de Graduação em Engenharia de Computação – FURG, Rio Grande.

Renon, F.J., Lages, W.F. (1999). Navegação de robôs móveis utilizando sonares, VIII Congresso de Iniciação Científica FURG/UFPel/UCPel, Rio Grande.

Song, K., Tang, W. (1996). Environment Perception for a Mobile Robot Using Double Ultrasonic Sensors and a CCD Camera, IEEE Transactions on Industrial Electronics, vol. 43, no. 3.

Zelinsky, A. (1992). A Mobile Robot Exploration Algorithm, IEEE Transactions on Robotics and Automation, vol. 8, no. 6.