NAVEGAÇÃO DE ROBÔS MÓVEIS EM AMBIENTES DESCONHECIDOS UTILIZANDO SONARES DE ULTRA-SOM

Guilherme de Lima Ottoni*
ottoni@acm.org

Walter Fetter Lages†
w.fetter@ieee.org

*Department of Computer Science, Princeton University, 35 Olden Street, 08544 Princeton, NJ, USA

†Departamento de Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Av. Osvaldo Aranha, 103
90035-190 Porto Alegre, RS, BRASIL

ABSTRACT

This paper presents a navigation system for mobile robot working on unknown environments. The proposed method is based on approximated cell decomposition. Only the initial and final position and orientation are 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: Autonomous navigation, decomposition methods, obstacle detection, mobile robots.

RESUMO

Este artigo apresenta um sistema de navegação 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 a sua posição e orientação inicial e final. 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.

PALAVRAS-CHAVE: Navegação autônoma, métodos de decomposição, detecção de obstáculos, robôs móveis.

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 desenvolver, 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. Para tanto, 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, são apresentados métodos para a navegação autônoma de robôs móveis em ambientes sobre os quais possui-se pouca ou nenhuma informação. O sistema opera

Artigo submetido em 20/12/2000
1a. Revisão em 7/5/2001; 2a. Revisão 1/4/2002
Aceito sob recomendação do Ed. Assoc. Prof. Paulo E. Miyagi

em tempo real, utilizando os sonares do robô para fazer a detecção dos obstáculos, e interage com o o sistema de controle do robô para rastreamento das trajetórias desejadas.

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 desenvolvido pelos autores (Lages, 1998) e utilizado no presente trabalho. Este robô conta com uma série de periféricos, dentre eles dois sonares de ultra-som, utilizados neste trabalho 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 MMX, utilizando sistema operacional Linux com *kernel* modificado pelos autores para suportar operação em tempo-real. Para comunicação, ele possui uma placa de rede Ethernet e um modem sem fio. Quanto a sua geometria, ele possui forma cilíndrica com 1,35m de altura e 0,30m de raio.

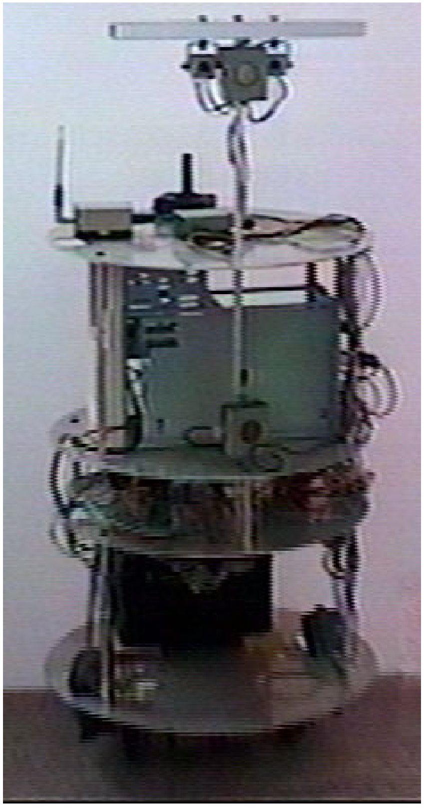

Figura 1: Robô móvel Twil.

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.

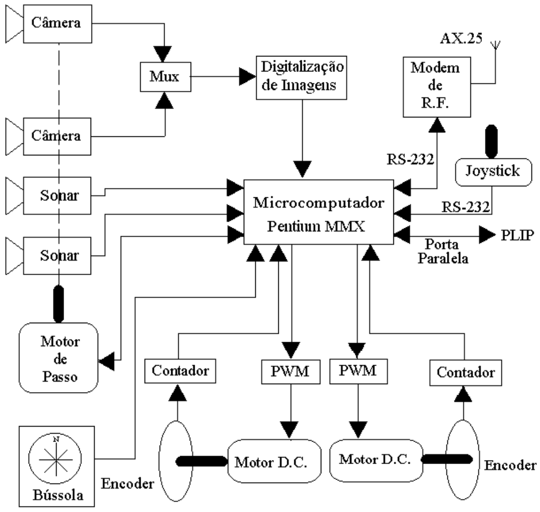

Figura 2: Diagrama de blocos do robô Twil.

Embora o robô disponha de um sistema de aquisição de imagens, o mesmo não foi utilizado neste trabalho, pois os métodos de visão computacional necessários para tal são relativamente complexos, merecendo um espaço exclusivo para que se possa dar o tratamento adequado a este problema. Ressalta-se, no entanto, que grande parte desta complexidade surge devido a problemas relacionados às condições de iluminação do ambiente, à dificuldades para determinação dos parâmetros do sistema de imagem e às ambiguidades apresentadas em grande parte dos procedimentos para obtenção de informações de profundidade a partir de imagens.

3 ARQUITETURA DO SISTEMA

Um diagrama da arquitetura proposta pode ser visto na figura 3. Os módulo implementados em *software* estão representados pelas bolhas. Os retângulos ilustram os dispositivos físicos do robô que interagem com o ambiente.

A arquitetura proposta é formada pelo seguintes blocos:

- Gerador de Trajetória: faz a geração da trajetória a ser rastreada pelo robô. É importante notar que embora a trajetória seja gerada completa, isto é, até a configuração objetivo, ela não será necessariamente rastreada em sua totalidade. Cabe ao fornecedor de trajetória gerar a referência para o controlador apenas referente à parte da trajetória localizada dentro do alcance máximo do sonar. As trajetórias são geradas completas para obter-se a melhor trajetória possível dadas as informações dispo-

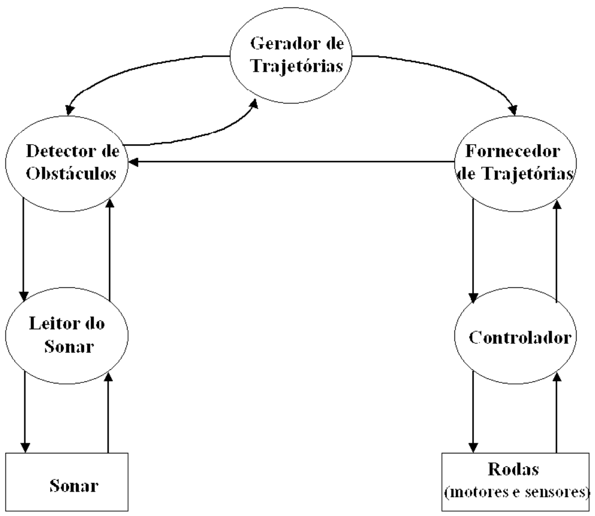

Figura 3: Arquitetura proposta.

níveis, que não incluem as posições ocupadas por todos os obstáculos.

- Fornecedor de Trajetória: gera a referência para o controlador, a partir da trajetória gerada, limitando os movimentos à área já explorada pelo sonar e fornecendo uma referência constante ao atingir-se o ponto final da trajetória gerada.
- Controlador: faz o controle do robô propriamente dito, garantindo o rastreamento da trajetória, assim como a permanencia do robô em um ponto fixo ao final da trajetória;
- Detector de Obstáculos: solicita leituras do sonar e atualiza no mapa de bits os obstáculos encontrados; invoca o Gerador de Trajetórias;
- Leitor do Sonar: faz uma leitura do ambiente utilizando o sonar e detecta as regiões de profundidade constantes (Leonard e Durrant-Whyte, 1992).

Os blocos Gerador de Trajetória, Fornecedor de trajetória e Detector de Obstáculos formam o sub-sistema de planejamento de trajetórias e serão discutidos na seção 5. O processamento dos sinais de sonar e a estratégia de controle serão apresentados nas seções 4 e 6, respectivamente.

4 PROCESSAMENTO DO SONAR

A detecção dos obstáculos através do sonar é realizada utilizando-se o método das regiões de profundidade constante. Este método, baseia-se no *time-of-flight* dos pulsos emitidos pelo sonar. Mais especificamente, define-se como *time-of-flight* o tempo entre a transmissão de um pulso e a recepção do seu eco refletido pelo alvo. Assim, a distância ao alvo é dada por:

$$d = \frac{cT_{of}}{2} \quad c = 331.4\sqrt{\frac{T}{273}}$$

onde d é distância do sensor ao alvo; T_{of} é o *time-of-flight*; c é a velocidade do som; T é a temperatura ambiente em Kelvin.

As distâncias obtidas a partir de leituras dos sonares de ultra som, raramente constituem a reprodução em coordenadas polares do ambiente de prova, como seria, a princípio esperado (Masliah e Albrecht, 1998). Isto se deve à ocorrência de leituras falsas, causadas pelos efeitos de divergência, difração, reflexões múltiplas e absorção (Hollestein, 1992), como ilustrado na figura 4. No entanto, a maioria das superfícies existentes em ambientes construídos pelo homem apresentam reflexão especular (Lim e Leonard, 2000). Este tipo de reflexão produz na varredura do sonar regiões onde a distância ao obstáculo é constante (ou seja, profundidade constante). Assim, determinação dos obstáculos através da identificação de regiões de profundidade constante (RPCs) permite uma maior robustez frente aos erros causados por estes fenômenos.

Uma RPC é definida por um conjunto de leituras do sonar tais que a diferença nas profundidades não seja maior do que um determinado valor σ e tal que a diferença entre o ângulo inicial α_1 e o ângulo final α_n seja maior do que $10°$, ou seja:

$$RPC : \left\{ \alpha_i \middle| \left| p(\alpha_i) - \frac{1}{n}\sum_{j=1}^{n} p(\alpha_j) \right| < \sigma \wedge \alpha_i - \alpha_n \geq 10° \right\}$$

onde $p(\alpha_i)$ é a profundidade na direção α_i.

A figura 5 mostra um corredor onde foi aplicado o método de identificação de obstáculos proposto. Na figura 6 podem ser vistos os dados obtidos do sonar plotados em diagrama polar.

Na figura 7, pode-se identificar mais facilmente as regiões de profundidade constante. Note-se que este gráfico utiliza os mesmos dados da figura 6, porém plotados em forma cartesiana.

Pode-se, agora, identificar a profundidade e orientação do obstáculo (RPC), através da orientação do centro da RPC, ou seja:

$$\alpha = \frac{\alpha_1 + \alpha_n}{2}$$

onde α é o ângulo de orientação da RPC; α_1 é o ângulo inicial da RPC; α_n é o ângulo final da RPC.

As RPCs identificadas a partir da varredura mostrada na figura 6, podem ser vistas na figura 8.

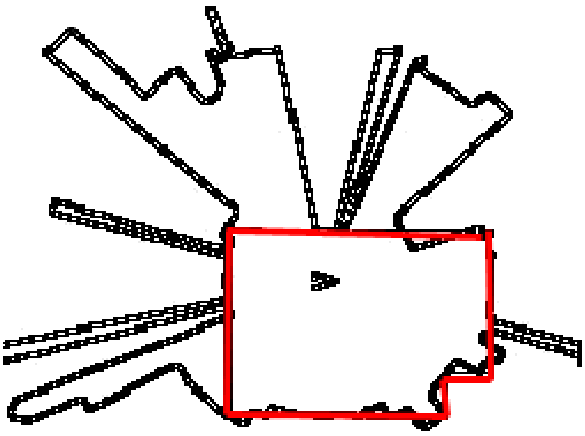

Figura 4: Efeitos causadores de leituras falsas do sonar.

Figura 5: Corredor onde foi feita a varredura do sonar.

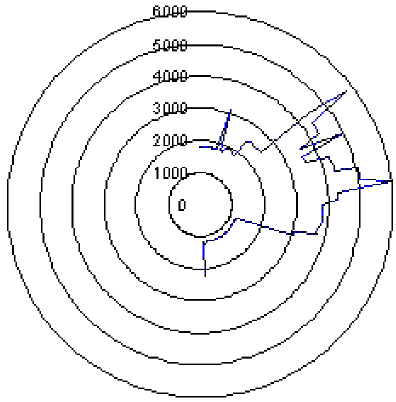

Figura 6: Gráfico polar de uma varredura de 180° do sonar (distância em mm).

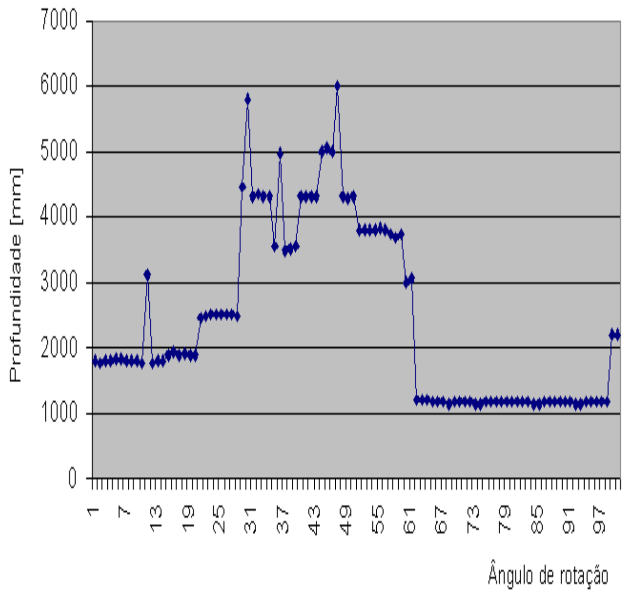

Figura 7: Varredura do sonar plotada na forma cartesiana.

Os obstáculos encontrados são as laterais da parede e o chão, visto que o sensor ultra-sônico, dispersa o som em formato de cone com abertura de 30 graus. Logo, a RPC de profundidade máxima, na figura 8, pode ser determinada, escolhendo-se a altura de montagem do sonar. Isto ocorre porque devido ao ângulo de abertura espacial do sensor, se a altura de montagem do sensor for relativamente baixa (menor do que 2,5m) será obtida uma reflexão no chão antes de atingir-se a profundidade máxima detectável pelo sensor.

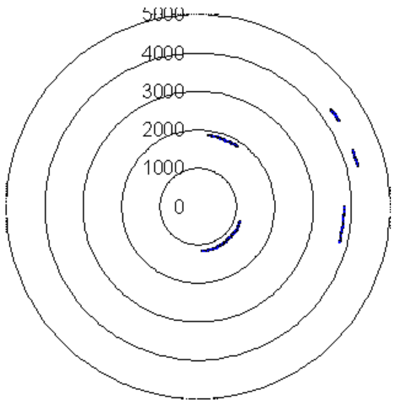

Figura 8: Regiões de profundidade constante identificadas.

5 PLANEJAMENTO DE TRAJETÓRIAS

O planejamento de trajetórias é realizado utilizando-se o método de decomposição em células. Este método 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 9.a. Outra forma de representação do ambiente, também amplamente utilizada, é através de uma quadtree, como em Zelinsky (1992).

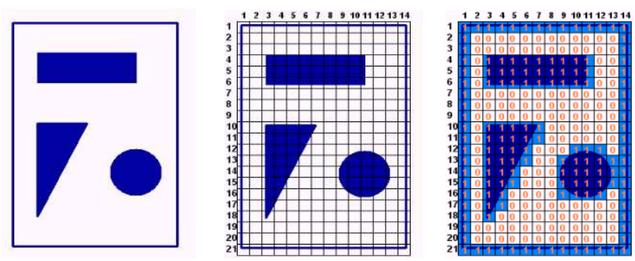

Figura 9: Decomposição aproximada em células.

Adota-se que cada célula representará uma região quadrada, de lado arbitrário (figura 9.b). Então cada posição do mapa de bits conterá 1 se a interseção da célula correspondente com os obstáculos for não nula. A figura 9.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 (figura 10). 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.

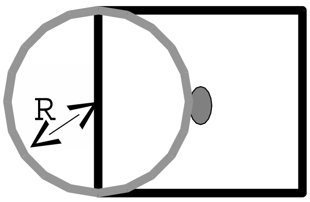

Figura 10: Expansão de obstáculos.

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 11. 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.

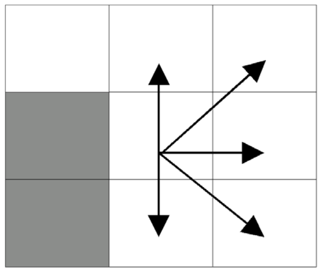

Figura 11: Células adjacentes.

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.

6 CONTROLE

O controlador é semelhante ao apresentado em Lages et al. (1996), empregando linearização por realimentação de estados.

Com base na figura 12, o modelo do robô móvel (Campion et al., 1996) pode ser escrito na forma:

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \operatorname{sen}\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \tag{1}$$

onde x_c e y_c são as coordenadas do centro de massa do robô com relação ao sistema X_0, Y_0; q é a orientação com relação ao eixo X_0;
h_1 é a velocidade linear e h_2 é a velocidade angular.

Considerando-se o sistema definido pela equação de estado (1), pode-se definir a equação de saída do sistema como sendo

$$y(x) = h(x) = \begin{bmatrix} x_r & y_r \end{bmatrix}^T \tag{2}$$

onde (x_r, y_r) são as coordenadas de um ponto de referência (P_r), a princípio arbitrário, sobre a base do robô.

O ponto P_r pode ser descrito no sistema de coordenadas globais X_0, Y_0, através das relações

$$x_r = x_c + x_r^c \cos\theta - y_r^c \operatorname{sen}\theta \tag{3}$$

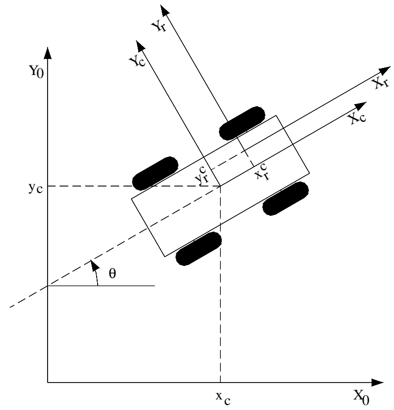

Figura 12: Sistemas de coordenadas.

$$y_r \quad = \quad y_c + x_r^c \operatorname{sen}\theta + y_r^c \cos\theta \tag{4}$$

onde (x_r^c, y_r^c) são as coordenadas de P_r descritas com relação ao sistema de eixos X_c, Y_c, como mostra a figura 12.

Substituindo-se (3) e (4) em (2) e derivando-se em relação ao tempo, obtém-se:

$$\dot{y}(x) = \begin{bmatrix} \dot{x}_c - \dot{\theta}\,(x_r^c \operatorname{sen}\theta + y_r^c \cos\theta) \\ \dot{y}_c + \dot{\theta}\,(x_r^c \cos\theta - y_r^c \operatorname{sen}\theta) \end{bmatrix}$$

e, utilizando-se (1) pode-se escrever:

$$\dot{y}(x) = \begin{bmatrix} \cos\theta & -\,(x_r^c \operatorname{sen}\theta + y_r^c \cos\theta) \\ \operatorname{sen}\theta & (x_r^c \cos\theta - y_r^c \operatorname{sen}\theta) \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}$$

ou ainda, de forma mais compacta, $\dot{y}(x) = E(x)u$, de onde verifica-se que se $det\,(E(x)) \neq 0$, pode-se fazer

$$u(x) = E^{-1}(x)v \tag{5}$$

resultando no sistema $\dot{y}(x) = v$, linear do ponto de vista entrada-saída. Adicionalmente, tem-se que

$$E^{-1}(x) = \frac{1}{x_r^c} \begin{bmatrix} x_r^c \cos\theta - y_r^c \operatorname{sen}\theta & x_r^c \operatorname{sen}\theta + y_r^c \cos\theta \\ -\operatorname{sen}\theta & \cos\theta \end{bmatrix}$$

Portanto, a inversibilidade de $E(x)$ está assegurada desde que escolha-se um ponto de referência tal que $x_r^c \neq 0$.

A entrada equivalente v pode ser gerada por qualquer técnica clássica de controle linear. Neste trabalho, optou-se por utilizar a técnica de modelo-referência, por facilitar o desenvolvimento de versões adaptativas desta lei de controle (Lages e Hemerly, 1998). Assim, considera-se como vetor de entradas equivalentes

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \dot{y}_{m_1} + \alpha_1\,(y_{m_1} - x_r) \\ \dot{y}_{m_2} + \alpha_2\,(y_{m_2} - y_r) \end{bmatrix} \tag{6}$$

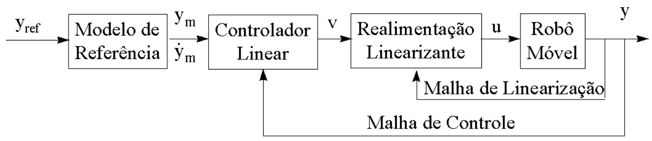

Figura 13: Diagrama de blocos do sistema de controle.

onde y_{m_1} e y_{m_2} são as saídas dos modelos de referência associados a cada componente de (2), descritos pelas seguintes funções de transferência

$$G_{m_1}(s) = \tfrac{\alpha_1}{s+\alpha_1} \quad G_{m_2}(s) = \tfrac{\alpha_2}{s+\alpha_2} \tag{7}$$

Na Figura 13 pode ser visto um diagrama de blocos da estrutura de controle proposta. Estas estrutura de controle é composta por duas malhas de controle. A malha interna é responsável por transformar o sistema em um sistema linear, através da realimentação linearizante definida por (5). O sistema linar resultante, com entrada v e saída y, é controlado pela malha externa que implementa um controlador linear por modelo-referência, descrito pelas expressões (6) e (7).

7 IMPLEMENTAÇÃO EM TEMPO REAL

O sistema foi implementado utilizando-se o sistema operacional Linux com o módulo do *kernel* que gerencia o relógio de tempo real do PC modificado pelos autores. Esta modificação implementa dispositivos virtuais denominados temporizadores de tempo real. Basicamente, estes dispositivos operam do seguinte modo: uma operação de escrita neste dispositivo inicializa um temporizador (com base de tempo dada por *hardware*) com o período desejado. A operação de leitura, bloqueia o processo até que o temporizador expire, garantindo a temporização desejada. Caso a operação de leitura seja executada após o temporizador ter expirado, é retornado um código de erro, indicando que o *deadline* da tarefa foi perdido e portanto o sistema está comprometido, pois o controlador estará operando fora dos parâmetros para os quais foi projetado. A princípio, esta situação nunca deveria ocorrer, mas se for detectada, os atuadores devem ser desligados imediatamente, para evitar danos ao robô ou ao ambiente de trabalho. Obviamente, verificou-se experimentalmente que esta situação não ocorre e que existe uma folga de temporização tal que todos os *threads* são executadas dentro de seu *deadline*.

Adicionalmente, como existe um forte acoplamento entre as diversas tarefas de tempo real, o modelo tradicional de processo do Unix não é apropriado, devido ao grande overhead causado na transferência de dados. Assim, foram utilizados *threads* (Leroy, 1997) para modelar cada uma das tarefas. A sincronização entre os *threads* foi feita através de semáforos.

O funcionamento destes *threads*, incluindo a comunicação entre eles, é descrito detalhadamente em Ottoni (2000).

8 RESULTADOS DE SIMULAÇÃO

Para testar e validar o método proposto, desenvolveu-se um simulador. Neste, além dos *threads* de leitura do sonar e controle, existe um outro *thread*, que simula os obstáculos existentes no ambiente. Asim, este *thread* fornece os obstáculos que são detectáveis a partir de cada posição do robô. Este mesmo *thread* também simula a movimentação do robô e verifica se ocorre alguma colisão do robô com os obstáculos do ambiente.

Na figura 14, tem-se um exemplo testado com o simulador desenvolvido. A figura 14.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 14.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 14.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).

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 14.d a 14.l mostram as demais trajetórias geradas, para cada posição em que o robô parou para detectar novos obstáculos. A trajetória da figura 14.l é seguida até que o robô alcance a sua posição de destino.

9 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 o desempenho 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.

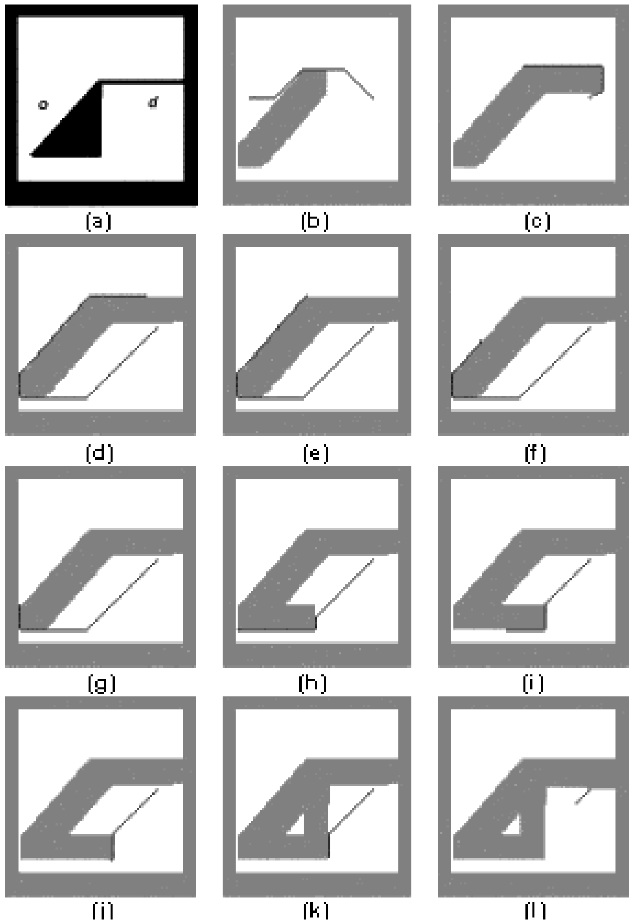

Figura 14: Exemplo de simulação.

Segue um exemplo significativo dos testes práticos utilizados, ilustrado na figura 15. Este teste foi efetuado em um laboratório de eletrônica, contendo bancadas e armários nas paredes, que produzem condições desfavoráveis para o sonar. 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 15.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

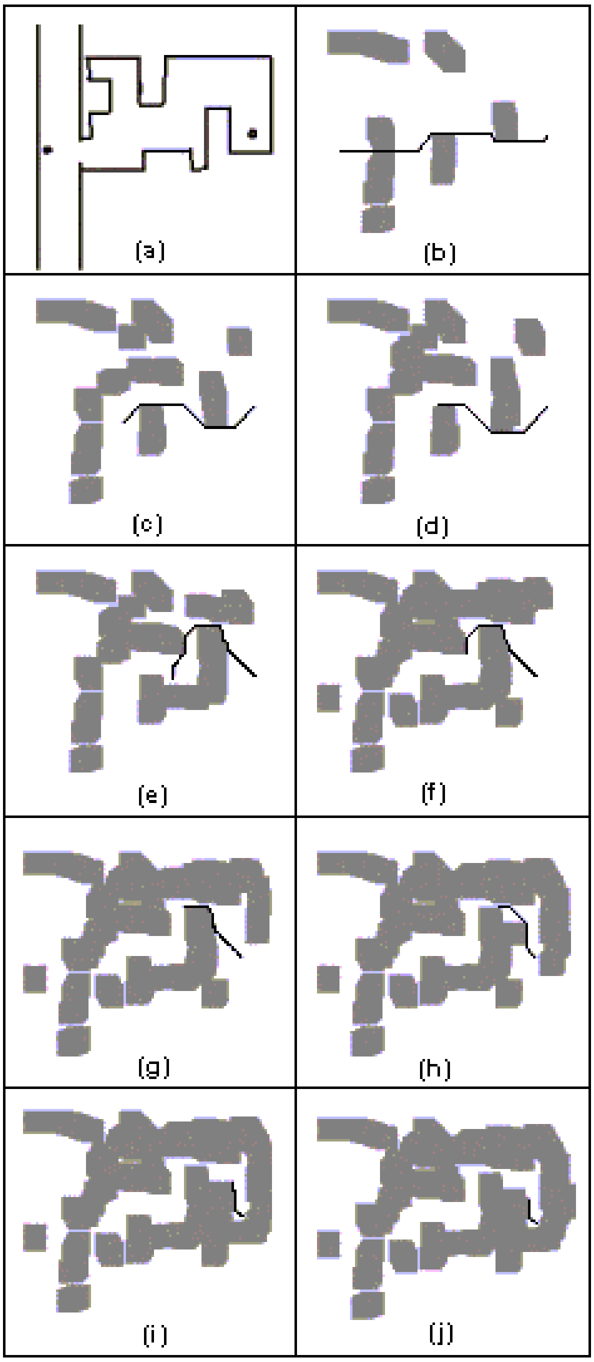

Figura 15: Teste realizado em laboratório.

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.

Finalmente, na figura 16, tem-se um último exemplo de experimento. Este teste foi feito em um laboratório de Eletrotécnica, contendo diversas bancadas de concreto para ensaios de motores, utilizadas como obstáculos. Este teste serviu principalmente para verificarmos o desepenho de tempo do sistema desenvolvido, uma vez que se trata de uma sala bastante grande (15x15m) e que possui vários obstáculos. A discretização do ambiente foi feita em células quadradas de lado igual a 5cm. Este valor mostrou-se adequado, uma vez que permitiu uma boa precisão na discretização do ambiente, não acarretando contudo em uma perda significativa no desempenho no planejamento das trajetórias.

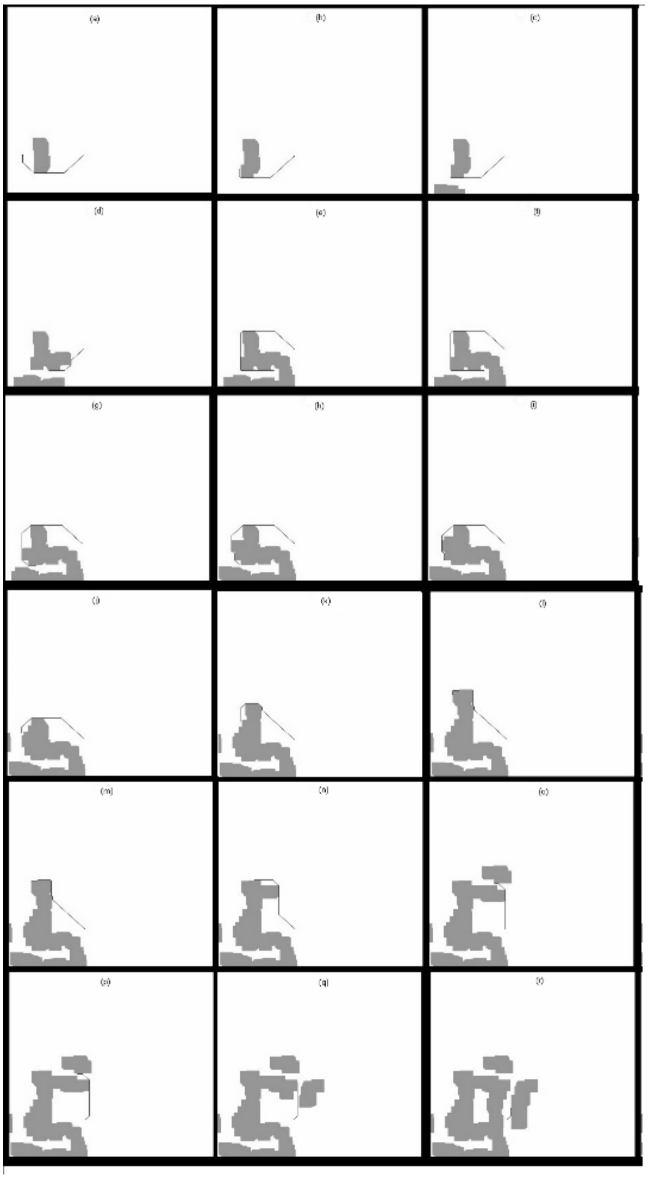

Figura 16: Teste realizado em laboratório.

10 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 e Tang (1996).

REFERÊNCIAS

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

Campion, G., Bastin, G. e D'Anréa-Novel, B. (1996). Structural properties and classification of kinematic and dynamical models of wheeled mobile robots, *IEEE Trans. on Robotics and Automation* **12**(1): 47–62.

Foux, G., Heymann, M. e Bruckstein, A. (1993). Two-dimensional robot navigation among unknown stationary polygonal obstacles, *IEEE Transactions on Robotics and Automation* **9**(1).

Hollestein, A. A. (1992). *Aufdatierung der Position und der Orientierung eines mobilen Roboters*, Doktor abhandlung, Eidgenössischen Technischen Hochschule, Zürich. Diss. ETH Nr. 9803.

Lages, W. F. (1998). *Controle e Estimação de Posição e Orientação de Robôs Móveis*, Tese de doutorado, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP. Disponível em http://www.eletro.ufrgs.br/ fetter/artigos.html.

Lages, W. F. e Hemerly, E. M. (1998). Adaptive linearizing control of mobile robots, *Proceedings of the 5^{th} IFAC Workshop on Intelligent Manufacturing Systems*, International Federation of Automatic Control, Gramado - RS, Brazil.

Lages, W. F., Hemerly, E. M. e Pereira, L. F. A. (1996). Controle linearizante de uma plataforma móvel empregando realimentação visual, *Anais do XI Congresso Brasileiro de Automática*, Sociedade Brasileira de Automática, São Paulo, SP.

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

Leonard, J. J. e Durrant-Whyte, H. F. (1992). *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, Boston.

Leroy, X. (1997). Linuxthreads - posix 1003.1c kernel threads for linux, *available on-line at http://pauillac.inria.fr/x̃leroy/linuxthreads*.

Lim, J. H. e Leonard, J. J. (2000). Mobile robot relocation from echolocation constraints, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(9).

Masliah, M. e Albrecht, R. (1998). The mobile robot surrogate method for developing autonomy, *IEEE Transactions on Robotics and Automation* **14**(2).

Ottoni, G. d. L. (2000). Planejamento de trajetórias para robôs móveis, *Projeto de graduação em engenharia de computação*, Fundação Universidade Federal do Rio Grande, Rio Grande, RS.

Song, K. e Tang, W. (1996). Environment perception for a mobile robot using double ultrasonic sensors and a ccd camera, *IEEE Transactions on Industrial Electronics* **43**(3).

Zelinsky, A. (1992). A mobile robot exploration algorithm, *IEEE Transactions on Robotics and Automation* **8**(6).