Mobile robots

Tuesday, September 15, 1998
AFTERNOON – Track C – **14h00–16h20**
Room Tancredo Neves A

Robôs móveis

Terça-feira, 15 de Setembro de 1998
TARDE I – Série C – **14h00–16h20**
Sala Tancredo Neves A

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 575-580 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

SISTEMA INTELIGENTE DE NAVEGAÇÃO PARA ROBÔS MÓVEIS AUTÔNOMOS

JOSÉ A. D. DE CARVALHO
JOÃO S. S. SOUZA

Escola Técnica Estadual de São Paulo, Centro Estadual de Educação Tecnológica Paula Souza
IEE, ITA, Instituto Tecnológico de Aeronáutica
CDT, Centro de Desenvolvimento de Tecnologia e Recursos Humanos
C.P. 111, 12242-800 , São José dos Campos, SP, BRAZIL
E-mail: jadias@iconet.com.br

Resumo— Este artigo explana e analisa os diversos métodos empregados no controle tradicional e inteligente de robôs móveis autônomos. Planejadores/navegadores usando conceitos de lógica fuzzy, redes neurais, redes neuro-fuzzy e computação evolucionária serão apresentados e comparados quanto ao seus respectivos desempenhos. Ao final é proposto um sistema inteligente de navegação que combina as melhores características de cada método.

Abstract— This paper explains and analyses the several methods used in the traditional and intelligent control of autonomous mobile robots. Planners/navigators that make use of concepts of fuzzy logic, neural networks, fuzzy-neural artificial networks and evolutionary computation are presented and compared in terms of their respective performances. Also, it is proposed an intelligent system for navigation that combines the best characteristics of each method.

Keywords—Mobile robots; navigation systems; fuzzy control; neural network.

1 Introdução

O problema de planejamento de trajetória de robôs móveis é definido em Xiao, J., Z. M., L. Zhang e T. Krzysztof (1997) da seguinte maneira: "dado um robô e uma descrição do ambiente, planejar uma rota entre duas localizações especificadas, a qual seja livre de obstáculos e satisfaça certos critérios de otimização".

O estudo de planejamento de trajetórias pode ser dividida em duas categorias: global e local. No primeiro caso, o modelo do ambiente é definido com precisão, sendo que a navegação é baseada na informação completa conhecida a priori. Diversas propostas clássicas já foram apresentadas nos últimos anos para definir modelos analíticos de ambiente, das quais destacam-se: espaço de configuração, campo potencial e diagrama de Voronoi.

Já a navegação local é baseada nas informações advindas de sensores (ultra-sônicos, infra-vermelhos, laser, câmeras etc) instalados no robô, os quais fornecem dados sobre o ambiente desconhecido. Assim, o modelo é impreciso devido aos erros inerentes dos sensores utilizados.

Se um robô móvel tem como objetivo mover-se em um ambiente desconhecido para alcançar uma meta sem colidir com obstáculos, as informações dos sensores devem ser usadas para construir um modelo do mundo em tempo real para pré-planejar uma trajetória livre de colisão.

Os métodos globais de planejamento de trajetória são eficientes em ambientes completamente conhecidos e estáticos, tendo resultados bastante ineficientes em ambientes complexos e dinâmicos onde a posição dos obstáculos é desconhecida. Neste caso, a navegação local é a mais recomendada.

Para que o robô móvel tenha uma certa autonomia, o principal desafio é reagir às modificações do meio onde se encontra, tratar em tempo real as informações recebidas (manobrando eficientemente no ambiente), e mostrar um comportamento adaptativo e inteligente. Com o intuito de solucionar essas dificuldades, alguns conhecidos paradigmas da inteligência artificial passaram recentemente a ser empregados no planejamento de trajetórias de robôs móveis e construção de navegadores, tais como Lógica Fuzzy, Redes Neurais Artificiais e técnicas de Computação Evolucionária.

Cada uma dessas técnicas possui aspectos tanto positivos como negativos quando utilizadas individualmente. Porém, quando se combinam as vantagens de cada uma, pode-se chegar a resultados bastante interessantes, como o emprego de Redes Neuro-Fuzzy e Redes Neurais com aprendizado por reforço através de algoritmos biológicos .

O objetivo deste artigo é discutir a evolução das diversas técnicas usadas em navegação de robôs móveis, desde os métodos tradicionais de planejamento global até a recente utilização de algoritmos evolucionários na construção de planejadores e navegadores.

O restante do trabalho é organizado da seguinte forma: na seção 2 são apresentadas as técnicas de planejamento e as arquiteturas de controle convencionais desenvolvidas para navegação de AMRs (Robôs Móveis Autônomos). A seção 3 discute os métodos disponíveis atualmente para a implementação de controle inteligente. Finalmente na seção 4 é apresentado um sistema que combina as diversas características de cada método.

2 Planejamento de Trajetórias

Durante as décadas de 70 e 80, inúmeros pesquisadores dedicaram-se a desenvolver propostas computacionais para solucionar o problema de planejamento do movimento de robôs móveis.

2.1 Estratégias tradicionais de planejamento

Em Latombe, J.-C. (1991) os métodos são resumidos em quatro grandes grupos: Roadmap, decomposição em células exatas e aproximadas, e o uso de campo potencial.

Será analisada uma das mais utilizadas que é o campo potencial originalmente proposta por Khatib.

Basicamente, o robô é tratado como uma partícula no espaço de configuração, sobre a influência de um campo potencial criado artificialmente. O valor do campo potencial em uma dada configuração depende da posição do robô e da distância aos obstáculos. A função potencial é dada por:

$$U_{total}(q) = U_{att}(q) + U_{rep}(q) \quad (1)$$

onde q é a configuração atual do robô, U_{att} é o potencial de atração e U_{rep} o potencial de repulsão.

Computando-se os potenciais, pode-se definir os vetores de forças do campos de atração e repulsão respectivamente como o gradiente dos respectivos potenciais:

$$\begin{aligned} \vec{F}_{att}(q) &= -\vec{\nabla} U_{att}(q) \\ \vec{F}_{rep}(q) &= -\vec{\nabla} U_{rep}(q) \end{aligned} \quad (2)$$

Considerando que as funções de potencial são definidas na forma padrão pelas seguintes funções:

$$U_{att}(q) = \frac{1}{2}\zeta\rho^2_{goal}(q) \quad (3)$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \text{se } \rho(q) \le \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases} \quad (4)$$

onde:

$\rho_{goal}(q)$: a distância euclidiana do robô à meta.

ζ : fator de escala que especifica o peso do potencial de atração.

$\rho(q)$: distância mínima do robô ao obstáculo

η : fator de escala que especifica o peso do campo de repulsão.

ρ_0 : constante representando uma distância limite, na qual o campo potencial de repulsão não afeta o robô.

Aplicando (3) e (4) em (2) temos que as forças de atração e repulsão podem ser escritas como:

$$\vec{F}_{att}(q) = -\zeta(q - q_{goal}) \quad (5)$$

$$\vec{F}_{rep}(q) = \begin{cases} \eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\vec{\nabla}\rho(q) & \text{se } \rho(q) \le \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases} \quad (6)$$

onde $\vec{\nabla}\rho(q)$ é um vetor unitário na direção do gradiente de $U_{rep.}$

Assim, o robô deve se mover no espaço de configuração com um movimento de acordo com as forças artificiais mostradas em (5) e (6). A cada nova configuração q, a força F(q) resultante determina a aceleração da partícula. Supondo conhecida a dinâmica do robô, pode-se computar as forças/torques que devem ser aplicadas aos atuadores em cada instante.

O grande problema dos planejadores que usam este método é a existência de mínimo local no campo potencial, resultante da interação entre as forças de repulsão (devido aos obstáculos) e a força de atração (devido ao alvo). Várias aproximações foram propostas para tratar com este problema, das quais ressaltam-se o uso de outras funções potenciais (grade, numéricas) ao invés da padrão, potenciais elípticos, movimentos randônicos e aprendizado global.

Outra técnica bastante interessante é a construção do diagrama de Voronoi, mais aplicado a cenários estáticos, onde o robô conhece a priori sua localização e orientação, bem como a dos obstáculos, e a idéia geral é capturar a conectividade do espaço livre do robô na forma de uma rede de curvas unidimensionais. Uma vez construído, este mapa é usado como conjunto de rotas padrões para a geração da trajetória.

2.2 Arquiteturas de Controle

Existem diversas propostas em relação às arquiteturas para controle de robôs móveis autônomos.

Nas arquiteturas tradicionais, o controle é feito de forma centralizada, não se mostrando eficiente nos ambientes dinâmicos devido às incertezas provenientes das medidas realizadas pelos sensores.

Uma das grandes evoluções nesta área foi o emprego de modelos baseados em agentes comportamentais. No contexto deste artigo, agente pode ser definido como uma entidade que possui estado interno e que funciona continuamente e autonomamente em um ambiente em que passam outros processos e coexistem outros agentes.

Segundo Colombettti, M., M. Dorigo e G. Borghi (1996) o controle baseado em comportamentos baseia-se na decomposição do sistema de controle do movimento do robô móvel em um conjunto de módulos com objetivos específicos.

Figura 1. Comportamentos distintos de um robô móvel para atingir o alvo.

Um dos principais desafios do controle baseado em comportamentos é como coordenar os conflitos e competições entre múltiplos comportamentos reativos eficientes. Na Fig. 1 podemos ver um exemplo que ilustra este princípio. Um robô deve partir de um ponto inicial e atingir uma posição alvo no interior de uma sala em formato de U. Desta forma, durante a trajetória, o sistema de controle deve "pesar" os diversos comportamentos usados como: desviar de obstáculos, seguir um contorno, mover para o alvo etc, de acordo com a informação adquirida pelos sensores.

Os dois comportamentos mais utilizados são: "desviar de obstáculos" e "seguir para a meta". Um dos grandes desafios é compatibilizar estes comportamentos. Se um robô móvel navega em ambiente incerto em direção a uma posição alvo, estes dois comportamentos sempre irão se conflitar. O comportamento de desvio é usado para evitar os obstáculos independente da posição da meta, enquanto o comportamento de busca da meta é usado para achar a posição do alvo independente da localização dos obstáculos.

No início, as primeiras propostas baseavam-se em decomposição serial de módulos funcionais.

Neste caso, cada nível executa um estágio, resultando em processamento encadeado. Se houver uma falha em algum módulo, o sistema entra em colapso. Esta estrutura hierárquica foi muito usada nos robôs de segunda geração e adaptada aos robôs de terceira geração (que são capazes de sensoriar o ambiente). Sua principal desvantagem é que impede a adição de novos níveis ou funções ao sistema.

A outra forma é a estrutura paralela baseada em comportamentos mostrada na Fig. 2.

Um controlador que funcione baseado nesta estrutura é dividido em camadas, onde cada uma é responsável por um tipo de comportamento ou tarefa e possui um caminho completo dos sensores aos atuadores, permitindo a produção de comportamento útil, antes de todo o conjunto estar completamente projetado.

Camargo, R. F. (1991) afirma que os sistemas construídos baseados nesta arquitetura são puramente reativos, e possuem alguns inconvenientes como a impossibilidade de geração de plano de ação já que são as interações entre os comportamentos que definem a conduta global do robô. Além disso, a resolução de conflitos é definida meramente por mecanismo de prioridades o que dificulta a implementação de ações mais complexas.

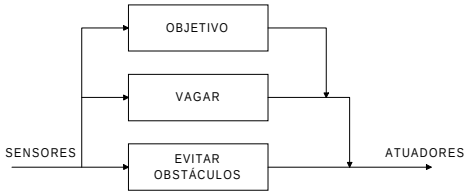


Figura 2. Arquitetura paralela baseada em comportamentos para AMRs

Uma das vantagens em relação à estrutura anterior é que a falha de alguma das camadas, não implica necessariamente na perda total do sistema, sendo que cada camada percebe somente as informações do sensor específico para determinada tarefa que lhe cabe.

Brooks, R.A. (1986) utilizou esta idéia na denominada Arquitetura de Assunção (ou Subordinação, como alguns preferem), onde basicamente um robô móvel teria três comportamentos, definidos da seguinte forma:

- Camada 0: evitar obstáculos
- Camada 1: vagar (gerar movimentos aleatórios)
- Camada 2 : atingir objetivo

Cada camada é composta por máquinas de estado finito que são combinadas através de mecanismos chamados inibidor e supressor, mostrado no exemplo da Fig. 3.

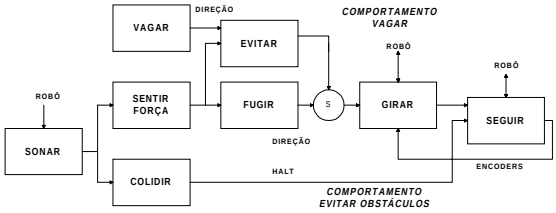


Figura 3. Exemplo de Arquitetura de Assunção para dois comportamentos.

Independente da estrutura de controle a ser usada no projeto do robô móvel, os seguintes critérios devem ser avaliados na escolha:

- Programabilidade: o robô não deve ser simplesmente uma máquina dirigida por estímulos externos, mas uma máquina programável, capaz de executar missões pré-definidas, especificadas por tarefas descritas a um certo nível de abstração.
- Robustez de funcionamento: dispor de estratégias de correção a ser aplicadas quando as condições de operação mudam.
- Evolutividade e reconfigurável: que novas funções possam ser desenvolvidas e adicionadas sem modificar as que já existem, e que diferentes tipos de controle possam ser pesquisados e aplicados.

• Autonomia: capaz de realizar uma missão pelo gerenciamento de seus próprios sistemas.

• Adaptabilidade: capaz de modificar seu comportamento prévio em relação as circunstâncias encontradas. O robô deve possuir capacidade de raciocinar para reconhecer as situações e elaborar as decisões de maneira a produzir/selecionar o comportamento mais adequado para executar a missão.

• Reatividade: capaz de detectar e reagir à eventos assíncronos.

Outras arquiteturas de controle como as propostas por Spence, R. e S. Hutchinson (1995) e Camargo, R. F. (1991), combinam as melhores características aqui apresentadas, sempre visando otimizar os critérios acima comentados.

3 Controle Inteligente

Uma interação adequada dos comportamentos simples resultam em um comportamento mais inteligente e desejável para tratar com situações mais complexas.

Um grande número de metodologias diversas têm sido empregadas para garantir este comportamento inteligente dos AMRs. A seguir apresentamos de maneira resumida as principais técnicas.

3.1 Lógica Fuzzy

Recentemente, diversos trabalhos têm proposto a utilização de controle fuzzy em planejamento de trajetórias para robô móveis. Em comum, temos o fato de que lógica fuzzy é usada para representar o mapeamento entre o espaço de entrada dos sensores e o espaço de ação do robô móvel.

A Fig. 4 mostra a arquitetura de controle fuzzy geral para um robô móvel. Em termos genéricos, um conjunto (ou vários) de sensores (ultra-sônicos, infravermelhos, telêmetro laser ou visão) são usados para modelar o ambiente. De acordo com o tipo de informação a ser manipulada diferentes tipos e quantidade de sensores diferentes devem ser usados.

Como variáveis de saída, normalmente controla-se a velocidade das rodas (acionadas por motor DC), e o ângulo de pilotagem das mesmas (diferença angular entre a direção do movimento do robô e direção do alvo dada pela rota planejada). No controle da direção das rodas pode-se empregar motores de passo. A mudança na direção do robô móvel também pode ser feita pela diferença de velocidade linear aplicada nas rodas.

A proposta de se usar lógica fuzzy tem a vantagem de tratar com várias situações sem o emprego de modelo analítico dos ambientes.

A lógica fuzzy permite que a natureza intuitiva da navegação livre de colisão possa ser facilmente modelada usando terminologia lingüística.

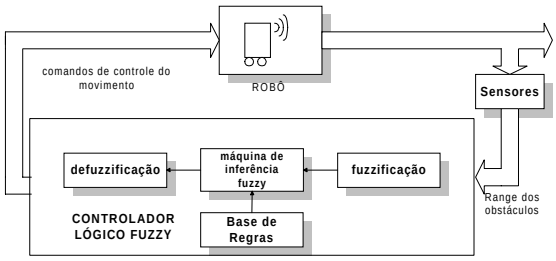


Figura 4. Arquitetura de controle fuzzy geral para robô móvel.

Em Martinez, A., E. Tunstel e M. Jamshidi (1993), a proposta feita pelos autores é que os comportamentos reativos sejam formulados por conjuntos de regras fuzzy, e essas regras integradas em uma única base. A coordenação dos diferentes comportamentos pode ser facilmente realizada por raciocínio fuzzy.

O grande problema dos controladores fuzzy é realmente a quantidade de regras que aumenta na proporção do quadrado do número de entradas de controles. Além disso a sintonia das funções de pertinência e o ajuste/ modificação das regras inviabilizam a sua utilização para certos casos.

Alguns trabalhos como Aoki, T. e M. Matsuno (1994) propõem estratégias diferenciadas para diminuir o número de regras como a utilização hierárquica de regras (combinando a Arquitetura de Assunção).

3.2 Redes Neurais

Como discutido anteriormente, a proposta usando lógica fuzzy tem a vantagem de tratar as incertezas e imprecisões, mas o conhecimento deve ser fornecido na forma de regras *if-then*. Entretanto, mesmo depois de definida e refinada, é difícil tratar todos os casos possíveis com regras específicas.

A utilização de redes neurais artificiais tenta eliminar estas dificuldades, já que não há necessidade de programação do conhecimento. Por exemplo, usando uma rede back-propagation e um conjunto de padrões de treinamento, é possível treinar um robô para navegar em certos ambientes.

Algumas experiências mostram aplicações de redes neurais onde a camada de entrada consiste das informações dos sensores (tipos que dependem da aplicação), camada escondida (que pode ser do tipo recorrente), e camada de saída para controle de velocidade das rodas. A Fig. 5 ilustra um exemplo proposto em Floreano, D. e F. Mondada (1996).

No trabalho de Uebel, L. F. (1994), foi constatado que se uma rede neural é treinada com as mesmas situações usadas para originar o conjunto de regras fuzzy , observam-se as seguintes diferenças entre os dois métodos:

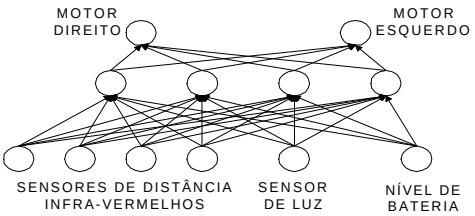


Figura 5. Exemplo de rede neural para controle de robô móvel

- O controle por sistema fuzzy apresenta maior tempo de processamento em relação ao uso de redes neurais, devido a maior quantidade de consulta as regras, inferências e cálculos.
- Os movimentos do robô apresentam-se mais suaves com sistemas fuzzy, já que utilizam velocidades de deslocamento contínuas.
- Quando se diminui o número de regras, o sistema com redes neurais mantém seu desempenho, enquanto o sistema fuzzy degrada o comportamento. Isto devido a capacidade de generalização das redes neurais, o que é interessante para situações mais complexas.

3.3 Redes Neuro-Fuzzy

Redes neurais e sistemas fuzzy possuem suas próprias falhas. Quando se projeta uma rede neural, ela é uma caixa preta que precisa ser definida, através de processo intensivo de computação. Sistemas fuzzy, requerem um profundo entendimento das variáveis fuzzy, das funções de pertinência e bom senso para selecionar as regras que contribuem para a melhor solução.

Os pontos fracos de ambas as técnicas podem ser superadas se incorporarmos operações lógicas fuzzy nas redes neurais, ou aprendizado e classificação das redes neurais em sistemas fuzzy. O resultado é chamado de Rede Neural Fuzzy Artificial (ou FANN - *Fuzzy Artificial Neural Network*). Um ótimo exemplo pode ser encontrado no trabalho de Beom, H. R. e H. S. Cho (1995).

O que distingue uma FANN da rede neural clássica são as seguintes características ideais:

- Todos os números reais que caracterizam uma rede neural (entradas, saídas e pesos) passam a ser números fuzzy.
- A saída de cada neurônio deve ser calculada por aritmética fuzzy e não por somatória.
- Todos os parâmetros dos algoritmos de aprendizado devem ser calculados segundo a lógica fuzzy, inclusive a própria "fuzzificação" do algoritmo de aprendizado.

3.4 Computação Evolucionária

Baseado nos conceitos de Computação Evolucionária (EC) sistemas de planejamento e navegação de robôs móveis têm sido desenvolvidos mais recentemente.

Os sistemas de navegação off-line tradicionais assumem um ambiente perfeitamente conhecido e pesquisam o caminho ótimo baseado em algum critério como o custo por exemplo. Por outro lado, os sistemas on-line são puramente reativos e não tentam otimizar a trajetória, visto o caso da Arquitetura de Assunção mostrado na seção 2.2 .

O avanço das pesquisas na área de computação evolucionária inspirou diversos trabalhos de planejadores de trajetória utilizando algoritmos evolucionários padrão.

Em Xiao, J., Z. M., L. Zhang e T. Krzysztof (1997) encontra-se um aplicação de algoritmo evolucionário para um sistema de planejamento/navegação de robôs móveis.

Basicamente a idéia do algoritmo pode ser resumida como:

- Um cromossomo de uma população *P(t)* de geração *t* representa um caminho do robô até a meta.
- Cada cromossomo é avaliado e o algoritmo entra num loop evolucionário. A avaliação é baseada numa função custo da trajetória que o cromossomo representa. Três critérios são levados em conta: distância mínima percorrida, suavidade da trajetória e requisitos de visibilidade (não aproximar-se muito de obstáculos).
- Operadores (tipo mutação e crossover) são selecionados de uma base e aplicados aos indivíduos para gerar descendentes. Os descendentes produzidos substituem os piores indivíduos da população.
- Se a trajetória gerada tem boa avaliação, o robô é movido enquanto monitora o ambiente para atualizar o mapa.

Este algoritmo roda dois processos em paralelo, ou seja, navega o robô ao longo da melhor trajetória enquanto monitora o ambiente para detectar objetos desconhecidos, e simultaneamente continua o processo de evolução na pesquisa do melhor novo caminho, sempre levando em conta a nova localização do robô e prováveis obstáculos.

4 Sistema Inteligente de Navegação

Pretende-se desenvolver um navegador para operar em ambientes desconhecidos e dinâmicos com boa resposta em tempo real, e atendendo aos requisitos discutidos na seção 2.2.

O sistema de navegação proposto ilustrado na Fig. 6 tem as seguintes características:

- Controlar um robô móvel autônomo com comportamentos de desviar de obstáculos e atingir uma meta (fonte de luz ou de som).
- Utilização de lógica fuzzy para tratamento das informações do sensores, tomada de decisões e geração dos sinais aos atuadores.
- Utilização do método de aprendizado por reforço afim de sintonizar a base de regras do controlador fuzzy.
- Utilização de campo potencial para seleção do comportamento adequado em cada situação.

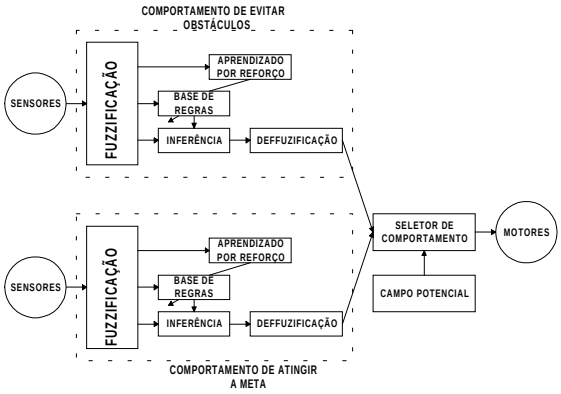


Figura 6. O esquema de controle do navegador proposto

A escolha de aprendizado por reforço ao invés de aprendizado supervisionado, deve-se ao fato que para o comportamento de desviar de obstáculos, é quase impossível determinar as saídas desejadas para cada situação, inviabilizando a criação de padrões para treinamento.

A lógica fuzzy é a melhor maneira de tratar com as incertezas e imprecisões dos sensores, realizando o mapeamento entre os sensores e o espaço de trabalho do robô. A base de regras é construída pelas entradas e saídas dos conjuntos fuzzy. O aprendizado por reforço deve garantir o mapeamento correto.

Para que o robô atinja a meta sem colidir com obstáculos, o comportamento apropriado deve ser selecionado para cada situação. Desta forma, optou-se em usar o método de campo potencial melhorado para realizar esta função, gerando as velocidades corretas aos motores.

Essa estrutura ainda está em fase de detalhamento, devendo posteriormente ser realizada uma simulação para avaliar os resultados. Será utilizada como plataforma experimental o robô móvel miniatura Khepera.

Dependendo dos resultados obtidos algumas modificações poderão ser introduzidas como introdução de um sistema para reflexos básicos, baseado em Arquitetura de Assunção. Isto porque a rede neural usada no aprendizado pode não generalizar para certas situações de mudanças de ambientes.

Nestes casos, os comportamentos reativos seriam gerados como opção para inicializar o aprendizado do controlador.

Futuramente, deve ser implementado um sistema utilizando-se de algoritmos evolucionários como mostrado na seção 3.4 para comparação de desempenho com o sistema proposto.

5 Conclusão

Com este artigo, pretendemos dar ao leitor, uma visão geral do estado da arte em robótica móvel inteligente.

As diversas técnicas foram apresentadas de maneira resumida, e as situações onde as mesmas são empregadas foram discutidas.

Ao final foi mostrado um exemplo de sistema que reúne algumas das metodologias desenvolvidas.

Espera-se obter resultados satisfatórios de planejamento e navegação com a implementação da arquitetura proposta.

Referências Bibliográficas

Aoki, T. e M. Matsuno (1994). Motion Planning for Multiple Obstacles Avoidance of Autonomous Mobile Robot Using Hierarchical Fuzzy Rules. *Proc. IEEE Int. Conf. on Multisensor Fusion and Integ. for Intel. Systems*, Las Vegas,pp. 265-271.

Beom, H. R. e H. S. Cho (1995). A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning. *IEEE Trans. Syst. Man Cyber*., vol. 25, nº 3 , pp. 464-477.

Brooks, R.A. (1986). A robust layered control system for a mobile robot. *IEEE Trans. Robotics Automat*., vol. RA-2 ,nº 1, pp.14-23.

Camargo, R. F. (1991). Architecture Materielle et Logicielle pour le Controle D'Execution d'un Robot Mobile Autonome. These du doctorat, Universite Paul Sabatier, Toulouse,France.

Colombettti, M., M. Dorigo e G. Borghi (1996). Behavior Analysis and Training - A Methodology for Behavior Engineering. *IEEE Trans. Syst. Man Cyber*., vol. 26, nº 3 , pp. 365-380.

Floreano, D. e F. Mondada (1996). Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Trans. Syst. Man Cyber*., vol. 26, nº 3 , pp. 396-407.

Latombe, J.-C. (1991). *Robot Motion Planning* . Kluwer A. P., Norwell.

Martinez, A. , E. Tunstel e M. Jamshidi (1993). Fuzzy Logic Based Collision Avoidance for a Mobile Robot. *IFIS '93 The 3. International Conference on Industrial Fuzzy Control and Intelligent Systems* ,Houston, USA, pp. 66-69.

Spence, R. e S. Hutchinson (1995). An Integrated Architecture for Robot Motion Planning and Control in the Presence of Obstacles with UnknownTrajectories. *IEEE Trans. Syst. Man Cyber*., vol. 25, nº 1 , pp. 100-110.

Uebel, L.F. et all (1994). Controle Inteligente de Robôs Móveis Autônomos: RAM x Fuzzy. *Anais do II Simpósio Brasileiro de Redes Neurais*, pp. 43-48.

Xiao, J., Z. M., L. Zhang e T. Krzysztof (1997). Adaptative Evolutionary Planner/Navigator for Mobile Robots. *IEEE Trans. on Evolutionary Comput.*, vol. 1, nº 1 , pp. 18-28 .

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 581-586 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

UM SISTEMA DE SENSORIAMENTO EXTERNO PARA ROBÔS MÓVEIS COM CONTROLE BASEADO EM AGENTES

EDUARDO O. FREIRE, TEODIANO F. BASTOS FILHO, ROGER A. C. FREITAS, HANSJÖRG A. SCHNEEBELI, MÁRIO SARCINELLI FILHO

Departamento de Engenharia Elétrica, Universidade Federal do Espírito Santo
CP 01-9011, Vitória/ES, 29060-970 – Brasil
Fax: +55-27-335-2650
E-mail: tfbastos@ele.ufes.br, hans@ele.ufes.br, sarcinel@ele.ufes.br

Resumo— A disponibilidade de dados sensoriais é muito importante para os sistemas de navegação de robôs móveis autônomos. Para robôs móveis autônomos com controle baseado em agentes, um sistema de sensoriamento ultra-sônico combinado com visão artificial é aqui abordado. Esse sistema de sensoriamento disponibiliza informações sobre o tipo de obstáculo detectado bem como sua distância ao robô. Tal sistema de sensoriamento externo é capaz de evitar a geração de informação redundante ou desnecessária, assim como não demanda excessivo esforço computacional para processar as imagens a bordo do robô.

Abstract— The availability of sensorial data is an important issue for the navigation system of autonomous mobile robots. For agent-based controlled autonomous mobile robots, a sensing system that combines ultrasonic sensors and artificial vision is here addressed. It makes available information about the type of obstacle detected in the trajectory of the robot, as well as the distance from the detected object to the robot. This external sensing system is also able to avoid the generation of redundant and unnecessary information, besides not demanding too much computational effort to process the acquired images onboard the robot.

Keywords— Mobile robots; robot navigation; robot vision; sensor systems; agents.

1 Introdução

Um robô móvel a rodas na configuração tração diferencial está em construção no Departamento de Engenharia Elétrica da Universidade Federal do Espírito Santo. Este robô é uma plataforma circular com quatro rodas: duas delas são montadas em um eixo comum e são independentemente acionadas por motores DC, enquanto as outras duas são rodas livres. A Figura 1 mostra tal robô, chamado Brutus. Circundando a plataforma, pode-se observar um anel de transdutores ultra-sônicos. Uma câmara monocromática também foi instalada no topo do robô, no centro da plataforma, dotando-o de capacidade visual monocular.

Esse robô móvel utiliza um microcontrolador MC-68332 para controlar os diversos processos envolvidos na movimentação do robô, isto é, sensoriamento interno e planejamento de trajetória. Adicionalmente, o robô também possuirá um microcontrolador MC-68HC11 para controlar o anel de sensores de ultra-som, e uma outra placa à base do microcontrolador MC-68332, para o processamento das imagens adquiridas pela câmara.

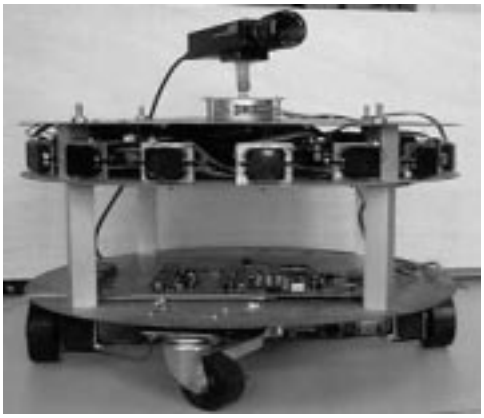


Figura 1: O robô móvel Brutus.

O robô foi projetado para evitar qualquer colisão ao se mover para a execução de uma tarefa em um ambiente semi-estruturado (por ambiente semi-estruturado entenda-se um ambiente cujos limites não são perfeitamente conhecidos pelo robô, mas que tem uma configuração definida e solo regular). Isto é possível graças ao sistema de sensoriamento desenvolvido. O ambiente de operação do robô também deve ter um número reduzido de objetos conhecidos (paredes, cantos, cadeiras e mesas), embora objetos desconhecidos também possam aparecer na trajetória do robô. Quando da detecção de um obstáculo, o sistema de sensoriamento detecta o objeto e passa a informação ao sistema de controle do robô, que adota uma ação exploratória (reconhecimento do objeto) com vistas a redefinir sua trajetória.

2 Estrutura para Construção de Sistemas de Controle Baseados em Agentes

Schneebeli (1992) sugeriu uma abstração para controladores baseados em agentes, que seriam representados por módulos concorrentes e intercomunicantes distribuídos em três categorias: agentes sensores (agentes sensores primitivos e agentes sensores virtuais), agentes comportamentos e agentes atuadores (agentes atuadores primitivos e agentes atuadores virtuais). Eles são mostrados na Figura 2. Nessa estrutura, assim como em Brooks (1986), um grupo de módulos (ou agentes) é responsável por uma atividade (comportamento) a ser executada pelo robô. Cada atividade representa um objetivo do sistema. A inter-

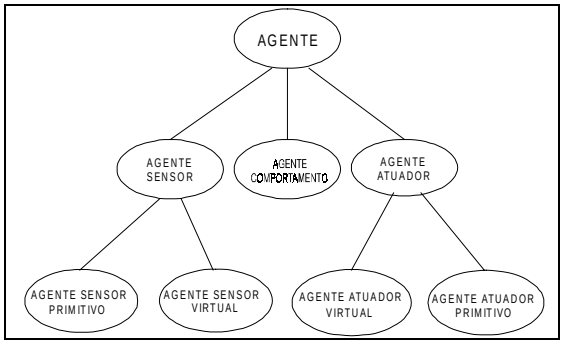


Figura 2: Tipos de agentes (Xavier, 1996).

ação entre os vários módulos define o objetivo principal.

Essa divisão em módulos permite uma abstração que faz com que a finalidade de cada módulo seja próxima à sua representação, dando a cada categoria de agentes algumas características particulares, relacionadas ao seu estado e comunicação, e associadas à sua função específica. Por exemplo: os agentes atuadores têm a característica de serem comandados por apenas um agente comportamento de cada vez. Os sensores físicos são representados por agentes sensores primitivos, e elementos tais como *sensor de câmara*, que têm uma certa característica sensorial mas não se conectam diretamente com nenhum sensor físico, são representados por agentes sensores virtuais. Os módulos que definem ações ou comportamentos são representados pelos agentes comportamento, e tarefas tais como *ir para frente* são representadas por agentes atuadores virtuais. Finalmente, os atuadores físicos são representados por agentes atuadores primitivos.

A estrutura para construção de sistemas de controle baseados em agentes desenvolvida por Xavier e Schneebeli (1995) implementa a abstração de Schneebeli (1992) utilizando a linguagem de programação C++ e bibliotecas concorrentes. Logo, uma estrutura concorrente orientada a objetos é construída. Essa estrutura é muito adequada para a modelagem de sistemas baseados em comportamentos, pois ela reúne características como modularidade, encapsulamento, herança, concorrência, autonomia e a eficiência da linguagem C++.

Na estrutura acima mencionada, os módulos de controle (agentes) são construídos como instâncias de classes definidas pelo usuário e derivadas das classes sensor, comportamento e atuador. Então, a definição de apenas uma classe pode ser usada para criar múltiplos agentes através da instanciação, mesmo em tempo de execução. Nesse caso, a diferença entre os agentes criados deverá ser dada pelos argumentos passados ao construtor da classe, e pelas pontes de comunicação estabelecidas (Xavier, 1996).

A instanciação e desinstanciação dinâmicas dos agentes, aliadas aos mecanismos de conexão intermodular definidos na estrutura (Yonezawa, 1988), provêem uma capacidade dinâmica de reconfiguração do sistema que pode ser usada para auxiliar o robô a gerenciar diversas situações utilizando diferentes estratégias de controle.

Os mecanismos de herança e a modularidade disponibilizados pela estrutura também garantem uma fácil expansão do sistema de controle. Logo, a criação de novos tipos de agentes pode ser feita derivando classes das já existentes classes sensor, comportamento e atuador, minimizando a necessidade de rescrever código. Por sua vez, o encapsulamento de dados e funções em classes e objetos garante a modularidade do sistema.

3 Medição da Distância ao Objeto Detectado

A técnica utilizada para realizar a medição de distância baseia-se em sensoriamento ultra-sônico e é largamente conhecida como Método de Pulso-Eco (Bastos, 1994). Utilizando esta técnica, a distância entre o sensor e o obstáculo detectado é dada por

$$x = \frac{1}{2}ct \qquad (1)$$

onde c é a velocidade do som no ar e t é o tempo de vôo associado ao sinal do eco ultra-sônico.

A Figura 3 mostra a performance do sistema ultra-sônico implementado na medida de distâncias. A experimentação foi feita usando uma parede (plano infinito) a uma distância de 50 cm do sensor ultra-sônico. A medição foi efetuada mil vezes, e o valor médio de distância obtido foi de 50,09 cm e o desvio padrão foi de 0,14 cm. A precisão obtida, então, é suficiente para aplicações relativas à navegação de robôs móveis, principalmente se considerada a simplicidade da estrutura usada para obtê-los (Freire, 1997).

É preciso destacar, neste ponto, que o sistema ultra-sônico utilizado, constituído de 16 transdutores espaçados entre si por 22,5°, permite detectar a presença de obstáculos em qualquer posição ao redor do robô, dada a largura do lóbulo de cada transdutor (Freire, 1997). Mais ainda, vários sensores podem detectar o mesmo objeto, e cada um pode fornecer uma medida diferente para a distância do obstáculo

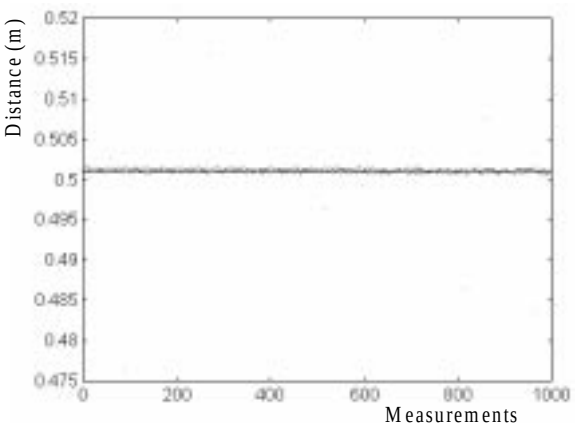


Figura 3: Medição de distância (Freire, 1997).

ao robô. Então, é arbitrado que o sensor que detectou o objeto é aquele que recebeu o maior valor de pico do eco.

Em outras palavras, pode-se identificar o sensor que recebe o maior valor de pico do eco refletido pelo obstáculo como estando bem alinhado com o obstáculo, e assim determinar em que direção está tal obstáculo, assim como sua distância até tal sensor. Estas são as informações fornecidas ao sistema de controle pelo anel de 16 sensores ultra-sônicos.

4 Estratégia para Ativação dos Sensores Ultra-Sônicos

Para a ativação dos sensores ultra-sônicos, utiliza-se uma ativação não periódica dos 16 agentes sensores, associados aos transdutores (Freire, 1997). Para fazer isso, a Estrutura Concorrente Orientada a Objetos desenvolvida por Xavier e Schneebeli (1995), para a construção de sistemas baseados em agentes, foi utilizada. Os transdutores são disparados de acordo com a prioridade associada a cada um. Por exemplo, dá-se maior prioridade ao sensor frontal do robô e menor ao seu sensor de ré, sempre que o robô se movimenta à frente. Quando a prioridade aumenta, a freqüência de disparo do transdutor também aumenta.

A prioridade associada ao *k*-ésimo transdutor ($p_k(x)$) varia de acordo com (Freire, 1997)

$$p_k(x) = \frac{a}{x} - b\frac{dx}{dt} + c$$
$$c = \{0.5[cos(\alpha + n\pi) + cos(4\alpha + n\pi)] + 1\}^2 \quad (2)$$

onde x é a distância do transdutor ao obstáculo detectado; c é a prioridade mínima associada a cada agente sensor (c>1); α é o deslocamento angular do *k*-ésimo transdutor em relação ao transdutor de referência; n=0 significa que a parte frontal do robô está na mesma direção do transdutor de referência (T1); n=1 significa que a parte frontal do robô está na direção oposta (o valor de n é determinado por um agente atuador virtual chamado *Activate_motor*), e a e b são constantes ajustáveis.

5 O Sistema de Visão Artificial

Para que o robô móvel seja capaz de reconhecer um objeto detectado em sua trajetória, pode-se utilizar um sistema de visão artificial (Brooks, 1985; Horn, 1989). Ele aumenta a quantidade de informação a respeito do ambiente, tornando mais segura a navegação do robô e permitindo que ele execute tarefas mais complexas. Porém, ele tem um custo computacional elevado (Vicente, 1997).

Para permitir o reconhecimento de alguns obstáculos, então, incorporou-se uma câmara de vídeo monocromática ao robô Brutus, dotando-o de visão monocular (ver Figura 1), com a câmara colocada na plataforma superior do robô. O que se busca, aliando o sistema de sensores ultra-sônicos à visão monocular, é permitir o reconhecimento de um conjunto limitado de obstáculos, sem porém sobrecarregar o sistema computacional de bordo, como ocorreria no caso de se utilizar um sistema de visão estéreo (Vicente, 1997).

A tarefa de reconhecimento de objetos aqui implementada ocorre da seguinte maneira: quando um dos sensores ultra-sônicos detecta um obstáculo próximo ao robô (no caso, a uma distância menor que 1m de um dos 16 sensores), o sistema de controle envia uma ordem para girar a câmara em direção ao sensor ultra-sônico que detectou o obstáculo, para que seja efetuado o seu reconhecimento. Ao mesmo tempo, o sistema de acionamento dos motores das rodas é paralisado, ou seja, o robô fica parado enquanto reconhece o obstáculo detectado. Feito o reconhecimento, ele reinicia seu movimento, com uma nova trajetória definida em função do obstáculo,.

Observe-se que a distância limite de 1m para que o robô reconheça um obstáculo é definida na construção dos agentes, e pode ser utilizada de forma a reduzir a freqüência com que as imagens são adquiridas, reduzindo-se assim o volume de processamento e o número de pausas do robô, principalmente se o número de obstáculos não é grande (ou seja, quando o ambiente de trabalho do robô não é densamente povoado por obstáculos).

Por outro lado, se um obstáculo próximo é detectado, a imagem que dele se captura é sempre tomada de uma distância mais ou menos fixa (no caso, de cerca de 1m). Para tratar as imagens dos objetos detectados na trajetória do robô, é usado um conjunto bastante simples de técnicas: primeiro, a imagem é filtrada usando-se uma máscara de detecção de linhas verticais; em seguida, é realizada uma operação de limiar (*threshold*) na imagem assim obtida (Gonzales e Woods, 1993), e finalmente a imagem resultante é processada usando-se uma rotina que isola apenas as suas linhas mais significativas, assim eliminando o ruído remanescente das operações anteriores. Com base na imagem resultante, composta apenas de linhas básicas que definem o contorno vertical do objeto, é que se faz o seu reconhecimento. Para ilustrar tal processamento, a Figura 4 mostra a imagem correspondente a um pé de cadeira que é adquirida pela câmara (Figura 4a) e a imagem resultante do processamento acima (Figura 4b). Da Figura 4b, vê-se claramente que tal processamento, embora bastante simples, permite reconhecer os objetos desejados a partir das linhas básicas isoladas. Experimentos realizados com tal sistema permitiram reconhecer, com cem por cento de acerto, além do pé de cadeira acima ilustrado, pés de mesa, paredes, portas, cantos e bordas (tendo como parâmetros de reconhecimento a distância entre as linhas verticais e o número dessas linhas).

6 Um Exemplo de Sistema de Controle Baseado em Agentes

Para mostrar como o sistema de sensoriamento externo desenvolvido pode ser integrado à unidade de

Figura 4a: Imagem inicial de um pé de cadeira.

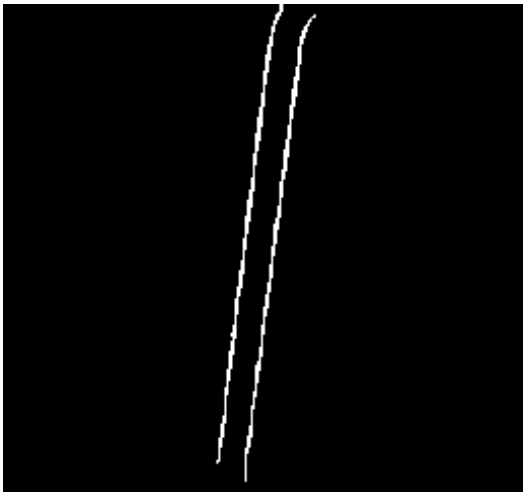


Figura 4b: Imagem do pé de cadeira da Figura 4a após o processamento.

controle central, um exemplo de sistema de controle baseado em agentes, que é capaz de usar a informação provida pelo sistema de sensoriamento, é mostrado na Figura 5.

Tal sistema de controle é composto por 16 agentes sensores primitivos chamados S_i, i=1,...,16, dois agentes sensores primitivos chamados *Encoder1* e *Encoder2*, dois agentes sensores virtuais, um chamado *Ultrasonic_Sensor* e o outro *Camera_Sensor*, três agentes comportamento, chamados *Colision_avoidance*, *Recognizer* e *Trajectory_planner*, três agentes atuadores virtuais, chamados *Go_ahead*, *Rotate* e *Activate_motor*, e, finalmente, dois agentes atuadores primitivos, chamados *Cam_Motor* e *PID*.

Esses agentes estão conectados para poderem implementar quatro comportamentos ou níveis de controle, denominados *Motor_Control*, *Movement*, *Obstacle_Recognition* e *Obstacle_Deviation*.

Cada agente S_i, i=1,...,16, é responsável por disparar um dos transdutores ultra-sônicos e processar o sinal de eco (medir a distância, digitalizar a envoltória, e medir o seu valor de pico). Essas informações, adicionadas à prioridade absoluta associada a cada agente S_k (calculada através da equação (2)), são passadas como uma mensagem ao agente *Ultrasonic_sensor*, que mantém duas listas, uma com a distância medida e outra com a prioridade absoluta associada a cada agente S_k. Quando o agente *Ultrasonic_sensor* recebe uma mensagem que contém uma distância menor que 1m (indicando que um obstáculo a esta distância foi detectado), ele envia uma mensagem ativando os agentes *Colision_avoidance* e *Camera_Sensor*.

Quando um agente S_i envia uma mensagem ao *Ultrasonic_sensor* ele recebe como resposta outra mensagem informando a soma das prioridades absolutas associadas aos agentes *S*. Desta forma, cada agente S_k é capaz de calcular sua prioridade relativa, pela divisão da sua prioridade absoluta pelo valor da soma das prioridades totais informadas pelo agente *Ultrasonic_sensor*. O inverso desta prioridade relativa é utilizado para programar seu relógio interno (objeto *Clock*), mudando o intervalo de tempo em que o agente é ativado (Freire, 1997). Uma vez que o hardware é compartilhado por todos os sensores ultra-sônicos, quando um agente S_k é ativado ele passa a uma fila e fica esperando a sua vez. Quando a prioridade associada ao agente S_k aumenta, o intervalo de tempo entre cada ativação se reduz e o agente S_k vai à fila mais freqüentemente.

Se o agente *Ultrasonic_sensor* recebe uma mensagem que contém uma distância menor que 40cm, o agente *Camera_Sensor* se desativa e o agente *Colision_avoidance* começa o procedimento para evitar obstáculos. Inicialmente, ele envia uma mensagem de pausa ao agente *Go_ahead*, que pára o robô enquanto ele gira sobre seu eixo central, devido a uma mensagem também enviada pelo agente *Colision_avoidance*, agora ao agente *Rotate*. A direção de rotação é determinada de forma a afastar o robô do obstáculo. Após algum tempo, medido pelo objeto *Clock* associado ao agente *Colision_avoidance*, esse agente envia uma mensagem desativando o agente *Rotate* e, então, a mensagem de pausa enviada ao agente *Go_ahead* é suspensa. A partir deste instante o robô volta a explorar o ambiente.

Se a rotação efetuada não for suficiente para evitar a colisão com o obstáculo, certamente outro agente *S* o detectará a uma distância menor que 40cm e o processo novamente se repetirá. O agente *Colision_avoidance* tem uma entrada chamada *canal* que pode receber mensagens de todos os agentes *S* e realizar algumas operações sobre elas, tais como calcular o valor médio, o maior valor e o menor valor. Nesse caso, o *canal* informa ao agente *Colision_avoidance* a menor distância recebida (direção mais provável para a colisão).

Porém, se o agente *Ultrasonic_sensor* recebe uma mensagem que contém uma medida de distância na faixa de 40cm a 1m, o agente *Colision_avoidance* se desativa, e o agente *Camera_Sensor* começa o procedimento de reconhecimento do obstáculo. A

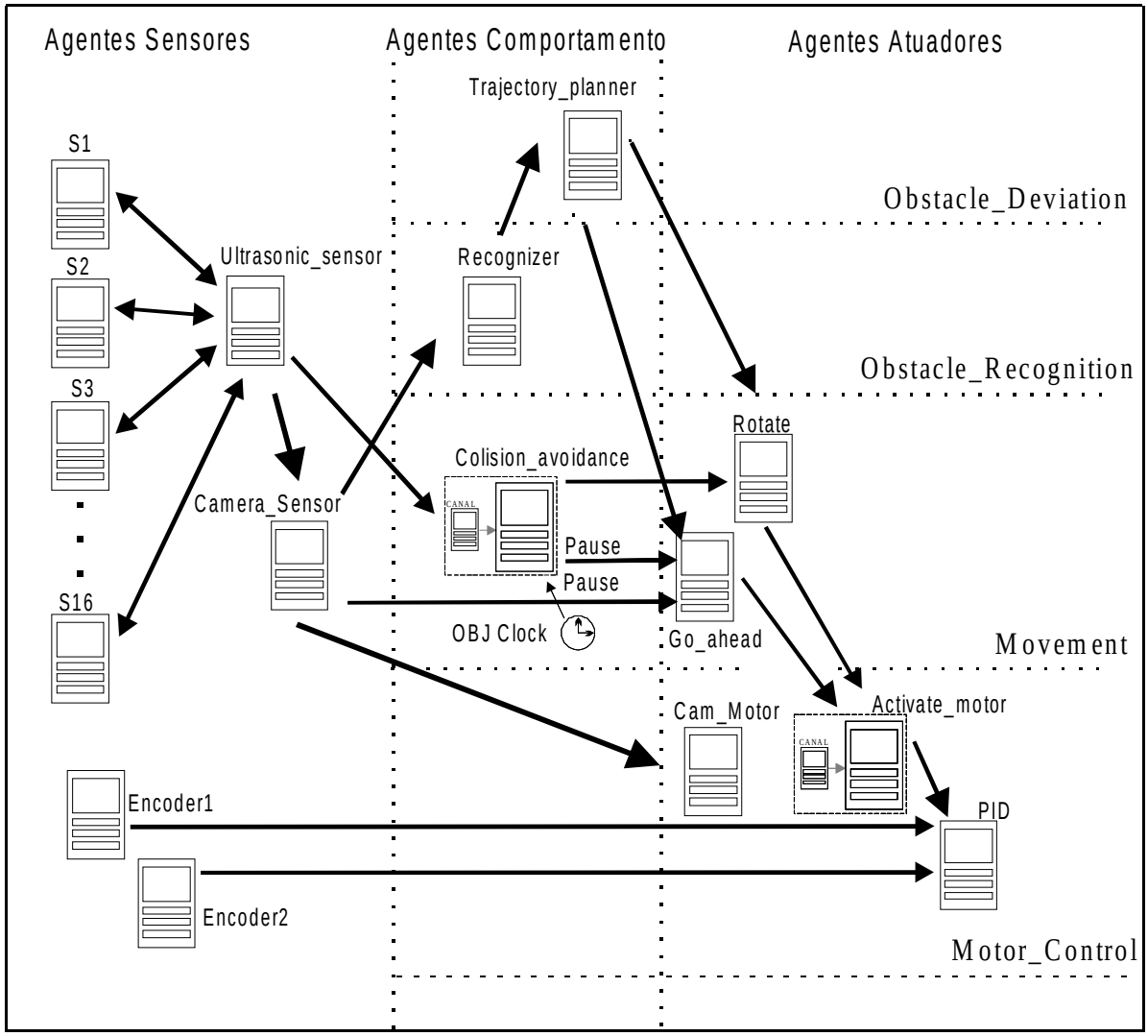


Figura 5: Um exemplo de controle baseado em agentes.

primeira etapa é o envio de uma mensagem de pausa ao agente *Go_ahead*, que então pára os motores. Em seguida, o agente *Camera_Sensor* ativa o agente *Cam_Motor*, que gira a câmara em direção ao agente S_k que detectou o obstáculo. Então, o agente *Camera_Sensor* captura uma imagem do obstáculo detectado enviando o resultado ao agente *Recognizer*. Este agente executa o reconhecimento do obstáculo. Uma vez reconhecido o obstáculo, uma nova trajetória deve ser definida para o robô, a qual contemplará o desvio do obstáculo ou outra atitude, dependendo de qual é o obstáculo (por exemplo, seguir através da porta, se ela estiver aberta). Essa nova trajetória é definida por um agente denominado *Trajectory_planner*, o qual é ativado pelo agente *Recognizer*, quando este identifica o obstáculo (mesmo que o obstáculo não seja identificado, o agente *Recognizer* ativa o agente *Trajectory_planner*, que define uma atitude padrão de desvio do obstáculo). Finalmente, uma mensagem enviada pelo agente *Trajectory_planner* ativa os agentes *Go_ahead* e *Rotate*, permitindo que o robô volte a se mover.

O agente *Activate_motor* também possui uma entrada *canal* que informa ao agente a soma das componentes de velocidade enviados pelos agentes *Go_ahead* e *Rotate*. O agente *Activate_motor* calcula a velocidade que deve ser aplicada a cada motor DC e envia a informação ao agente *PID*, que é responsável pelo controle de ambos os motores.

7 Conclusão

O sistema de sensoriamento ultra-sônico combinado com visão monocular aqui proposto demonstrou ser capaz de prover as informações necessárias ao sistema de navegação do robô móvel Brutus em um ambiente semi-estruturado. Ele mostrou também ser eficiente para reconhecer um pequeno conjunto de objetos que podem estar presentes no ambiente de trabalho do robô. Combinada com o sistema ultra-sônico, a visão monocular assegura reduzido volume de processamento a bordo do robô, permitindo seu deslocamento mais rápido, ao mesmo tempo que assegura flexibilidade suficiente ao seu sistema de navegação. Como exemplo, é mostrado um sistema de controle

baseado em agentes que já integra o sistema sensorial aqui proposto.

Agradecimento

Os autores agradecem ao Dr. Benjamin Kuchen, ao Dr. Ricardo Carelli, ao Ing. Alfredo Castro e ao Ing. Adrián Orellana, do Instituto de Automática da Universidad Nacional de San Juan, Argentina, pela cessão do software DECVISION de processamento de imagens, desenvolvido naquele instituto.

Referências Bibliográficas

Brooks, R. A. (1985). Visual Map for a Mobile Robot. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 824-829, IEEE CS Press, California, EUA.

Bastos, T. F. (1994). Seguimiento y Análisis de Entornos de Soldadura por Arco Automatizado Mediante Ultrasonidos. Tese de Doutorado, Universidade Complutense de Madrid, Madrid, Espanha.

Brooks, R. A. (1986). Achieving Artificial Intelligence Through Building Robots. A. I. Memo 899, MIT, Massachusetts, EUA.

Freire, E. O. (1997). Desenvolvimento de um Sistema de Sensoriamento Ultra-Sônico Para um Robô Móvel com Controle Baseado em Agentes. Dissertação de Mestrado, Universidade Federal do Espírito Santo, Vitória/ES, Brasil.

Gonzales, R. C. e R. E. Woods (1993). Digital Image Processing. Addison-Wesley Publishing Company.

Horn, B.K.P. (1989). Robot Vision. McGraw-Hill, New York, EUA.

Schneebeli, H. A. (1992). Die Steuerung von Mehrfinger-Greifersystemen. Tese de Doutorado, Universidade de Karlsruhe, Karlsruhe, Alemanha.

Vicente, L. A. (1997). Estrategias de Correspondencia Jerárquica y Métodos Directos de Autocalibración para un Sistema Estereoscópico Binocular. Tese de Doutorado, Universidade Complutense de Madrid, Madrid, Espanha.

Xavier, J. E. M. (1996). Uma Estrutura para a Construção de Sistemas de Controle Baseados em Agentes para Robôs Móveis. Dissertação de Mestrado, Universidade Federal do Espírito Santo, Vitória/ES, Brasil.

Xavier, J. E. M. e H. J. Schneebeli (1995). Estrutura para a Construção de Sistemas de Controle Baseados em Agentes para Robôs Móveis. Anais do II Simpósio Brasileiro de Automação Inteligente (II SBAI), Curitiba, PR, pp. 97-102.

Yonezawa, A., E. Shibayaama, T. Takada, Y. Honda (1988). Modeling and Programming in an Object-Oriented Concurrent Language. In: Object Oriented Concurrent Programming (Yonezawa, A., M. Tokoro), pp. 55-90. MIT Press, Massachusetts, EUA.

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 587-592 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

NAVEGAÇÃO EM AMBIENTE FECHADO E INICIALMENTE DESCONHECIDO UTILIZANDO APRENDIZAGEM POR REFORÇO

ARTHUR P. DE S. BRAGA , ALUIZIO F. R. ARAÚJO

Universidade de São Paulo, Departamento de Engenharia Elétrica

Av. Dr. Carlos Botelho, 1465, 13560-250, São Carlos, SP, BRASIL

E-mails: **arthurp@sel.eesc.usp.br, aluizioa@sel.eesc.sc.usp.br**

Resumo — Este artigo propõe e implementa um modelo de agente de Aprendizagem por Reforço que pode tratar múltiplos objetivos e navegar em ambientes dinâmicos e inicialmente desconhecidos. O agente aprende seus comportamentos através de interações de tentativa-e-erro com um ambiente dinâmico. O conhecimento adquirido do ambiente é codificado em duas superfícies: uma de premiação e uma de penalização. A primeira trata principalmente com o planejamento que leva ao estado meta, a última gera reações que permitem o desvio dos obstáculos. Os resultados obtidos em simulação sugerem que o agente é capaz de alcançar o alvo a partir de qualquer ponto do ambiente, evitando pontos de mínimos locais. Além disso, o agente pode evitar obstáculos dinâmicos não apresentados durante a fase de treinamento. Entretanto, o agente ainda pode ficar preso na presença de um obstáculo dinâmico e freqüentemente não seleciona o melhor caminho para navegar.

Abstract — This article proposes and implements a Reinforcement Learning Model for an agent that can manage multiple objectives and navigate in dynamic and initially unknown environments. The agent learns behavior through trial-and-error interactions with a dynamic environment. The environmental knowledge is encoded in two surfaces: the reward and the penalty surfaces. The former deals primarily with planning to reach the goal whilst the latter deals mainly with reaction to avoid obstacles. The simulation results suggest that the agent is able to reach a target from any point within the environment, avoiding local optimum points. Furthermore, the agent can avoid dynamic obstacles that were not present in the learning phase. However, the agent still may get stuck close to a dynamic obstacle and it often does not chose the best path to navigate.

Key Words — Reinforcement Learning; Temporal Difference; Learning algorithms; navigation.

1 Introdução

O objetivo desse trabalho é implementar um agente que aprenda, sem a interferência de um tutor, uma tarefa em um ambiente dinâmico e inicialmente desconhecido. A tarefa em questão é a navegação de um robô móvel que deve ser aprendida por métodos que busquem cumprir o objetivo final (planejamento), considerando objetivos locais (reação). Portanto o caminho para chegar ao alvo deve ser aprendido e implementado em um mecanismo único.

Aprendizagem por Reforço (AR) caracteriza-se pela absorção de conhecimento por um agente através de interação deste com o meio-ambiente em que está imerso (Kaelbling et al., 1996). Os agentes AR mantêm uma função denominada política que mapeia situações em ações. A tomada de decisões utiliza o cálculo do valor instantâneo da política. O imediatismo na tomada de decisões aproxima sistemas AR de sistemas reativos (Brooks, 1986). Entretanto, os agentes AR se distinguem da maioria dos sistemas reativos por aprenderem. Isto é, tais agentes assimilam suas estratégias de controle por tentativa e erro, interagindo com o ambiente, do qual recebe sinais de retorno.

Neste trabalho, o agente aprende a navegar em ambientes fechados que apresentem tanto obstáculos fixos quanto dinâmicos. Para tal, utiliza um método AR considerando duas avaliações: uma de premiação e uma de penalização. A superfície de premiação é gerada a partir da identificação da posição meta no ambiente, sem mapear obstáculos explicitamente. A superfície de penalização é montada a partir de informação local fornecida por estados vizinhos ao atual. A composição das superfícies orienta a navegação do robô móvel. O método AR para solução deste problema é uma adaptação de Aprendizagem por Diferença Temporal (TD) (Barto et al. (1983), Tesauro (1995) e Millán (1996)) porque ele não demanda o modelo do meio ambiente e permite atualização da avaliação a cada mudança de estado.

Os resultados das simulações sugerem que o agente aprende rapidamente a atingir uma meta treinada partindo de qualquer ponto do ambiente. Além disto, o agente consegue evitar obstáculos não presentes na fase de treinamento. Como limitações relevantes, os resultados sugerem que normalmente o agente não segue o melhor caminho e ele pode ficar preso junto a um obstáculo dinâmico.

Na próxima seção apresenta-se a tarefa deste trabalho. Esta origina um conjunto de requisitos

para o agente, discutidos na Seção 3. A solução é proposta na Seção 4 e os resultados são relatados na Seção 5. Finalmente a Seção 6 apresenta as conclusões.

2 Navegação em Ambiente Dinâmico

O foco deste projeto está na proposição de um agente, um robô, capaz de aprender autonomamente, por tentativa-e-erro, o comportamento para gerar caminhos em um ambiente dinâmico que é inicialmente desconhecido e que possui um número finito de estados possíveis.

O ambiente escolhido[1] reproduz, de maneira simplificada, o Laboratório de Sistemas Inteligentes (LASI) do Departamento de Engenharia Elética USP/SC: Espaço dividido em dois compartimentos havendo uma única passagem entre os mesmos. A Figura 1 esboça este ambiente com um exemplo de pontos aleatoriamente escolhidos para aprendizagem. Nela, o ponto assinalado por um **X** denota uma dada meta enquanto que os pontos assinalados por círculos denotam pontos iniciais para treinamento. O robô, deverá alcançar a meta a partir de qualquer ponto do ambiente, depois de uma etapa de treinamento. Um ponto inicial é tomado como referência e um ambiente, de início totalmente desconhecido pelo agente, é mapeado em um *grid* bidimensional.

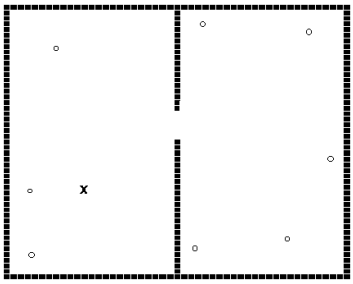


Figura 1 - Esboço do ambiente em questão.

A interação robô-ambiente fornecerá os dados necessários ao agente para cumprir dois objetivos básicos: buscar um caminho que ligue um ponto de partida a um ponto alvo desejado (planejamento) e garantir que no trajeto o robô seja capaz de evitar obstáculos (comportamento reativo).

Koenig e Simmons (1996) definem este como um **problema de exploração direcionado à meta** (PEDM). Nele, o agente deve encontrar um caminho para o estado meta em um espaço de estados inicialmente desconhecido, ou apenas parcialmente conhecido. Este caminho encontrado não precisa ser o caminho ótimo.

[1] Simulações realizadas em outros ambientes apresentaram resultados semelhantes.

3 Requisitos do Agente

A adoção de um controle puramente reativo é uma opção de solução. Porém, Millán (1996) lembra que sistemas puramente reativos (Brooks, 1986; Agre e Chapman, 1987) apresentam duas limitações principais para navegação autônoma em ambientes inicialmente desconhecidos. A primeira, e mais importante, é que estes sistemas podem gerar trajetórias ineficientes por selecionarem a próxima ação simplesmente como uma função das leituras do estado atual. A segunda limitação é a dificuldade em programar tais sistemas.

Whitehead e Ballard (1991) argumentam que o imediatismo na tomada de decisões é uma característica comum aos sistemas AR e aos sistemas reativos. Contudo, a maioria dos sistemas reativos apenas se adaptam, enquanto que os sistemas AR têm **capacidade de aprendizagem**. Isto é, a tomada de decisões em sistemas reativos é codificada de forma explícita por seus projetistas (Brooks, 1986) ou baseada na determinação modelos de mundo expresso por em um conjunto de regras reativas (Fikes et al., 1972; Laird et al., 1986). Em contrapartida, os sistemas AR podem não utilizar conhecimento prévio na tomada de decisões o que impede o emprego de algoritmos de IA clássica como A* (Winston, 1992). Eles aprendem estratégias de controle por tentativa-e-erro interagindo com o mundo que lhes provê realimentações na forma de reforços.

Koenig e Simmons (1996) argumentam que a representação dada ao sinal de reforço e a determinação das estimativas de retorno iniciais de cada estado são cruciais para habilitar o sistema a **lidar com ambientes inicialmente desconhecidos**. Duas são as representações na literatura para o reforço dado ao agente que aprende os caminhos no espaço de estados.

Representação por premiação da meta: O agente é recompensado quando atinge o estado meta, mas não é premiado ou penalizado de outra forma (Sutton, 1990; Whitehead, 1991). Formalmente:

$$r(s,a,s') = \begin{cases} 1, \text{caso } s \text{ for o estado meta.} \\ 0, \text{para qualquer outro estado.} \end{cases} \quad (1)$$

onde para cada estado *s*, $s \in S$, há um conjunto de ações associadas A(*s*), no qual uma ação escolhida, *a*, leva o sistema a um novo estado *s'*.

Nesta representação o cálculo da premiação acumulada utiliza um fator de desconto, γ, para aprender os menores caminhos no espaço de estados. O crescimento de γ é diretamente proporcional ao número de ações de um caminho que leve à meta. Logo, o agente deve alcançar o estado meta com a

menor quantidade possível de ações de forma a obter a máxima premiação acumulada.

Representação por penalização das ações: O agente é penalizado por cada ação que executa. Portanto, em geral, esta estrutura de representação é mais densa que a anterior, pois existem poucos estados meta em um ambiente (Koenig, 1991; Barto et al., 1995). Formalmente:

$$r(s,a,s') = -1, \quad \forall\, s, s' \in \text{S e } a \in \text{A} \qquad (2)$$

Para o aprendizado de caminhos mais curtos, o uso do desconto é facultativo. Logo, independente da utilização do fator de desconto no cálculo da premiação acumulada, o agente busca o estado meta através do menor número possível de ações, minimizando a penalização do agente.

Em conclusão, para Koenig e Simmons (1996) algoritmos AR resolvem um PEDM desde que o agente selecione suas ações segundo uma distribuição de probabilidades uniforme e a representação do sinal de reforço seja densa de forma a reduzir a busca no espaço de estados.

Grande parte dos sistemas AR aprendem suas funções de avaliação por meio de método TD. Os métodos de diferença temporal são uma classe de procedimentos de aprendizagem incremental para problemas de predição (Sutton, 1988). Os métodos TD são guiados pelo erro, ou diferença, entre predições temporárias sucessivas, logo, aprende-se sempre que houver mudança na predição.

Para **gerenciar objetivos distintos** e conflitantes encontrados na navegação de robôs móveis foram propostos métodos de campo potencial (MCP) (Kim e Khosla, 1992; Rimon e Koditscheck, 1992). Estes métodos propõem que a navegação de um robô móvel se baseie em forças imaginárias agindo sobre ele. Obstáculos exercem forças repulsivas, enquanto que uma meta provê uma força atrativa. A resultante ***R*** do somatório destas forças determina a próxima posição do robô. Em geral, estes métodos demandam um modelo de mundo e geram mínimos locais.

Conflitos entre objetivos em navegação autônoma dinâmica ocorrem também durante a fase de operação. O surgimento de um obstáculo dinâmico deve levar o agente a estabelecer um compromisso entre a política já aprendida e um comportamento reativo que o faria livrar-se do obstáculo inesperado.

A **capacidade de aprendizagem incremental** dos métodos TD é desejável para superar o conflito entre planejamento e comportamento reativo. A política aprendida pode ser modificada toda vez que o sinal de reforço indicar a presença de um novo obstáculo, incorporando os comportamentos reativos necessários ao desvio do obstáculo, e criando com isso uma nova política.

4 Modelo Proposto

Esta seção define os componentes de um agente AR levando em conta os requisitos da Seção 3.

O problema de navegação pode ser visualizado como uma árvore de decisão: cada nó corresponde a um possível estado *s*. A heurística que guiará a busca do agente na árvore de decisão será baseada em avaliações das possíveis ações a serem tomadas em cada estado *s*. Como o espaço de trabalho é inicialmente desconhecido, o agente deve aprender a avaliar as ações. Para que este aprendizado ocorra de maneira autônoma, utiliza-se AR.

Em AR, o agente ao interagir com seu ambiente de trabalho recebe deste último um sinal de retorno *r* que quantifica a contribuição instantânea de uma ação *a* em um estado *s* para alcançar um objetivo pré-definido. A avaliação de um estado *s* sob uma política π, denotado por $V^{\pi}(s)$, estima o retorno total esperado ao iniciar o processo no estado *s* e seguir a política π (Sutton e Barto, 1998)

Neste processo de navegação são considerados três tipos de estado: **estados livres** definidos como regiões do ambiente onde não há a presença de obstáculos (**estado obstáculo**) ou alvo (**estado meta**). A representação proposta para a informação fornecida por pelo meio, sinal *r*, é definida da seguinte forma:

$$r = \begin{cases} +1, \text{ se for o estado meta;} \\ -1, \text{ se for um estado com obstaculo;} \\ 0, \text{ se for um estado livre.} \end{cases} \qquad (3)$$

O diagrama da Figura 2 apresenta como se processa a interação agente-ambiente.

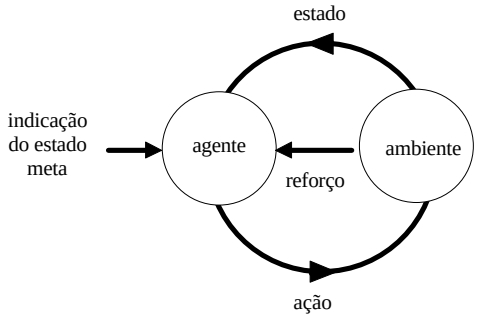


Figura 2 - Funcionamento do agente.

A estratégia escolhida para a solução deste problema é aprendizagem por diferença temporal (TD). Ela tem maior aplicação prática principalmente por sua característica incremental e por ser independente de um modelo do mundo. A

regra de atualização mais simples em métodos TD (Sutton e Barto, 1998) é:

$$\Delta V_t(s_t) = \alpha \left[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \right] \qquad (4)$$

sendo:

$V(s)$	- avaliação do estado s.
$\Delta V(s)$	- erro na avaliação do estado s.
s_t	- estado no instante t.
r_{t+1}	- retorno do ambiente no instante t+1.
α	- tamanho do passo de aprendizagem.
γ	- fator de desconto.

Neste trabalho, o agente aprende a realizar a tarefa de navegação por AR seguindo avaliações positivas e negativas. Deste modo, foi adotado que o valor de avaliação para cada estado leva em conta duas componentes: uma de premiação e uma de penalização. O conjunto das premiações e o conjunto das penalizações formam duas superfícies. A superfície de premiação é criada sem mapear explicitamente os obstáculos (tanto fixos quanto dinâmicos), sendo construída a partir da identificação da posição meta no ambiente. Esta superfície representa uma avaliação a longo prazo. A superfície de penalização é aprendida pelo agente baseando-se em informação local (avaliação de curto prazo) fornecida pelos estados vizinhos ao atual. Essa informação local indicará a proximidade de obstáculos e do alvo. A composição dessas duas superfícies gera informação que servirá para o controle da navegação do robô móvel. O algoritmo para o aprendizado dessas duas superfícies é apresentado a seguir.

```
Inicialize com zeros as avaliações da premiação e da penalização;

Repita

  posicione o robô em um estado inicial;

  Repita

    - selecione uma ação a;
    - execute a ação a e mude para um novo estado;
    - verifique se o novo estado é um obstáculo
      caso positivo, atualize a avaliação da
      penalização do estado anterior;
    - armazene o estado atual na trajetória;

  até encontrar um obstáculo ou o alvo

  - verifique se o novo estado é um alvo
          caso positivo, atualize a avaliação da premiação
          ao longo da  trajetória obtida na tentativa;

até encontrar o alvo um número determinado de vezes
```

Figura 3 - Algoritmo implementado em Português procedimental.

A premiação referente ao alcance da meta final e a penalização referente ao desvio de obstáculos são representadas em estruturas distintas e são aprendidas por estratégias diferentes.

Atualização da avaliação da premiação

Como esta avaliação só pode ser feita após a localização do alvo, o agente armazena os estados por onde passou (trajetória) até ter confirmação sensorial de que atingiu o alvo, a partir disso a premiação é distribuída, usando TD, voltando nos estados armazenados conforme a expressão (4). Cada estado só tem sua avaliação atualizada na primeira vez em que ele é visitado.

Atualização da avaliação da penalização

Na aprendizagem da avaliação da penalização, o agente não armazena estados para depois atualizá-los todos de uma vez. Tal avaliação é local, a atualização é feita a cada mudança de estado, levando-se em conta apenas o estado a ser atualizado e o seu sucessor. A expressão para a atualização também é (4), sendo que para tratar com obstáculos dinâmicos as penalizações devem levar em consideração a confirmação sensorial da presença dos obstáculos. Além disto, a superfície de penalização só é atualizada em uma vizinhança dos obstáculos definida pelo alcance dos sensores.

A escolha da ação entre as possíveis ações a serem tomadas a partir de cada estado (Figura 4.a) é baseada em uma composição da premiação (Figura 4.b) e da penalização (Figura 4.c). Essa composição é a mesma durante as fases de aprendizagem e funcionamento do sistema. A estratégia consiste em escolher a ação a ser tomada a cada estado seguindo a seguinte distribuição heurística abaixo:

- 85% das vezes é selecionada a ação com maior valor combinado (premiação + penalização);
- 10% das vezes é selecionada a ação que leva ao sentido contrário da maior penalização;
- 5% das vezes é selecionada a ação que possui maior premiação;
- Em caso de empate a escolha é aleatória.

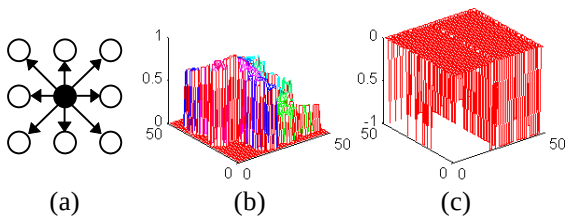


Figura 4 - (a) Máximo número de ações possíveis partindo de um estado (em preto) (b) Exemplo de uma superfície de premiação (c) Exemplo de uma superfície de penalização.

Este sistema está implementado a nível de simulação proporcionando resultados promissores. A próxima seção discute alguns destes resultados.

5 Simulações

A fase de treinamento foi realizada considerando obstáculos fixos, estados iniciais múltiplos e um alvo único. Neste treinamento, o agente aprende a mapear seu ambiente de trabalho estático. Na fase de teste, diversos pontos de partida (treinados e não-treinados) são tomados, igualmente distribuídos entre os dois compartimentos, e obstáculos dinâmicos não treinados, com formatos diversos, são acrescentados ao ambientes. O obstáculo dinâmico aparece apenas na fase de testes, não sendo conhecido pelo agente durante o treinamento.

Foram realizados dois tipos de testes com o agente. O primeiro consistiu na observação do comportamento do agente após a fase de treinamento diante dos pontos iniciais treinados e de pontos não treinados. O segundo tipo buscou verificar como o aparecimento de obstáculos não presentes na fase de treinamento afeta o rendimento do robô.

Todas as etapas do agente foram implementadas e simuladas no aplicativo MATLAB, tendo sido criada uma interface GUI para acompanhar os comportados gerados. A Figura 5 apresenta exemplos de trajetórias obtidas a partir das superfícies de avaliação da Figura 4. Note que o agente atua satisfatoriamente tanto com pontos treinados como com pontos não treinados. Entenda-se por atuar satisfatoriamente o fato de que as trajetórias que o agente gera não são ótimas no sentido de serem o menor caminho entre dois pontos estabelecidos, porém são capazes de estabelecer uma ligação entre pontos iniciais apresentados e a meta.

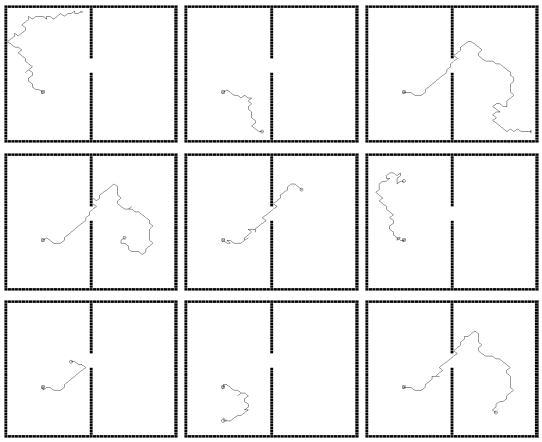


Figura 5 - Exemplos de trajetórias obtidas pelo agente na quais a meta está próxima ao centro da região à esquerda do ambiente.

Nos resultados da Figura 5, percebe-se que certos trechos de trajetórias se repetem para diferentes pontos de partida. O fato da premiação obtida pelos pontos treinados não ter sido uniformemente distribuída no espaço de estados faz com que as trajetórias aprendidas sirvam de atratores para os demais pontos. Assim, ao partir de um ponto inicial não treinado, o agente caminha pelo espaço de estados até atingir algum ponto que tenha sido visitado durante o treinamento. A partir desse ponto, o robô passa a seguir a trajetória a qual aquele ponto pertencia no treinamento até alcançar a meta.

Até o momento os testes do agente foram realizados em um ambiente "estacionário". Com o surgimento de "novos obstáculos não-treinados" no ambiente, como tratar o problema? A solução proposta consiste em durante a fase de teste/funcionamento, antes de cada transição de estados, monitorar os sensores para verificar se existe algum obstáculo no caminho. Caso os sensores acusem a presença de algum obstáculo, a penalização do estado com obstáculo vai para -1. Assim, quando for realizada a escolha da ação, o estado com obstáculo será evitado e um entre os demais vizinhos será escolhido. A partir deste ponto o agente observará as avaliações dos estados vizinhos ao novo estado para o qual ele foi "desviado" e seguirá com a política aprendida ou evitará o obstáculo.

Para testar o comportamento do agente diante de um obstáculo dinâmico realizou-se o seguinte procedimento: visualiza-se a trajetória seguida normalmente pelo robô e adiciona-se um obstáculo que interrompa essa trajetória. A Figura 6 apresenta alguns dos comportamentos obtidos.

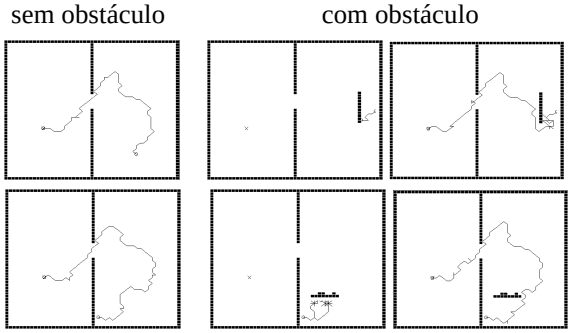


Figura 6 - Dois resultados obtidos em testes do agente no desvio de obstáculos dinâmicos para a mesma meta dos testes anteriores.

Pelas trajetórias acima apresentadas, o tratamento dado pelo agente aos obstáculos dinâmicos ainda é dependente da aleatoriedade da tomada de decisão. Conforme a atuação da componente aleatória da política seja sentida pelo agente é possível ou não desviar o robô da trajetória aprendida de forma a livrá-lo do obstáculo.

6 Conclusões

Este artigo propôs um agente para navegar autonomamente em ambientes dinâmicos e, de início, totalmente desconhecidos. Para o cumprimento adequado da tarefa, algumas

características principais devem estar presentes no agente: capacidade de aprendizagem, capacidade de trabalhar com ambientes inicialmente desconhecidos, capacidade de anexação incremental do conhecimento e capacidade de gerenciamento de múltiplos objetivos. Tais características estão presente no agente deste trabalho que se utiliza de uma representação diferenciada do conhecimento para selecionar suas ações: duas superfícies, uma de premiação e outra de penalização, codificam avaliações de longo (planejamento) e curto (comportamento reativo) prazos dos estados seguintes. Uma política simples de tomada de decisões, que considera essas duas avaliações, é apresentada, sendo as avaliações aprendidas por aprendizagem por reforço, empregando métodos de diferença temporal (TD).

Os resultados obtidos apontam que o agente é capaz de navegar partindo de pontos treinados, e não treinados, alcançando o ponto meta sem a necessidade de exploração exaustiva do ambiente de trabalho. As trajetórias geradas não são, em geral, ótimas e na presença de obstáculos dinâmicos pode ocorrer o surgimento de mínimos locais que prendam o agente em um estado diferente do meta.

Uma etapa de otimização permitiria a melhora do desempenho do agente ao diminuir as trajetórias geradas, e esse é um dos pontos em que os autores estão trabalhando.

Referências Bibliográficas

Agre, P. E. e Chapman, D. (1987) Pengi: An implementation of a theory of activity. *Anais do Sixth National Conference on Artificial Intelligence*. Morgan Kaufmann. Los Altos, CA. Pp: 268-272.

Barto, A. G., Sutton, R. S. e Anderson, C. W. (1983). Neuronlike adaptative elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man, and Cybernetics*, V. 3, N. 5, pp: 834-846.

Barto, A. G., Bradtke, S. J. e Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, V. 73, N. 1, pp: 81-138.

Brooks, R. A. (1986). A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation*, V. RA-2, N.1, pp: 14-23.

Fikes, R. E., Hart, P. E. e Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, V. 3, pp:251-288.

Kaelbling, L. P., Littman, M. L. e Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, V. 4, pp: 237-285.

Kin, J. e Khosla, P. K. (1992). Real-time obstacle avoidance using harmonic potencial function. *IEEE Transactions on Robotics and Automation*, V.8, N.3, pp: 338-349.

Koenig, S. (1991). Optimal probabilistic and decision-theoretic planning using Markovian decision therory. *Tese de mestrado*. Computer Science Department, University of California at Berkeley.

Koenig, S. e Simmons, R. G. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement learning algorithms. *Machine Learning*, V. 22, pp: 227-250.

Laird, J. E., Rosenbloom, P. S. e Newell, A. (1986). Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning*, V. 1, pp: 11-46.

Milán, J. del R. (1996). Rapid, safe, and incremental learning of navigation strategies. *IEEE Transactions os Systems, Man, and Cybernetics*, V. 26, pp: 408-420.

Rimon, E. e Koditschek, D. E. (1992). Exact robot navigation using artificial potencial functions. *IEEE Transactions on Robotics and Automation*, V.8, N. 5, pp: 501-518.

Sutton, R. S. (1988). Learning to predict by methods of temporal differences. *Machine Learning*, V. 8, pp: 9-44.

Sutton, R. S. (1990). Integrated architectures for learning, planning and reacting based on approximating dynamic programming. *Anais do International Conference on Machine Learning*. pp: 216-224.

Sutton, R.S. e Barto, A. (1998). *Introduction to Reinforcement Learning*. MIT Press / Bradford Books, Cambridge, MA.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, V. 38, No. 3., pp: 58-68.

Whitehead, S. D. (1991). A complexity analysis of cooperative mechanisms in reinforcement learning. *Anais do AAAI*. pp: 607-613.

Whitehead, S. D. e Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, V. 7, pp: 45-83.

Winston, P. H. (1992). *Artificial Intelligence*. Reading, UK: Addison Wesley.

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 593-598 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

CONTROLE DE TRAJETÓRIA E ESTABILIZAÇÃO DE ROBÔ MÓVEL NAO-HOLONÔMICO COM REALIMENTAÇÃO VARIANTE NO TEMPO

ALESSANDRO C. VICTORINO, PAULO R. G. KURKA, EURÍPEDES G. O. NOBREGA

DPM - FEM - Universidade Estadual de Campinas
Caixa Postal: 6122, CEP: 13083-970, Campinas, SP
E-mails: victor@fem.unicamp.br, kurka@fem.unicamp.br, egon@fem.unicamp.br

Resumo— Robô Móvel Comandado (RMC) é um sistema mecânico não-holonômico caracterizado por restrições cinemáticas não integráveis. É apresentado o controle de trajetória de um RMC não-holonômico com utilização de Linearização por Realimentação (Feedback Linearization), com resultados de simulação baseados em um robô miniatura fabricado por K-Team no laboratório LAMI, Suécia. A condição do Rank de Brockett postula que um sistema não-holonômico não pode ser estabilizado para uma configuração de repouso através de leis suaves de realimentação. O modelo cinemático de estado e algumas de suas propriedades estruturais para o robô Khepera é inicialmente considerado. Linearização por realimentação dinâmica de estados e controle com realimentação suave dependente do tempo são sucessivamente aplicados em uma estratégia híbrida, afim de se obter a convergência para a trajetória de referência e estabilizar o robô para uma configuração de repouso. A contribuição deste trabalho está na justificativa do momento de comutação entre as duas metodologias de controle na estratégia híbrida, com base na análise do número de condição da matriz de controlabilidade.

Abstract— Nonholonomic Wheeled Mobile Robot (NWMR) is a mechanical system which is characterized by non integrable kinematic constraints. The trajectory control of nonholonomic NWMR using feedback linearization is presented, with simulated results based on a miniature robot made by K-Team, Sweden. Brockett's rank condition shows that a nonholonomic system cannot be stabilized to a rest configuration by means of smooth state feedback laws. Its structural properties are analyzed and the kinematic state space models necessary for the understanding of the behavior of the NWMR are presented. Trajectory tracking problems for mobile robots by means of feedback linearization is then considered. Dynamic feedback linearization and time-varying feedback are successively applied to handle the tracking of a moving reference trajectory and stabilization of the robot to a rest configuration in a hybrid strategy. The contribution lies on the justification of the commutation moment between the two control strategies of the hybrid methodology, based on the condition number analysis of the controllability matrix.

Keywords— Mobile robots, Nonholonomic systems; Dynamic feedback linearization; Time-varying continuous feedback; Hybrid Strategy

1 Introdução

Tem aumentado nos últimos anos a literatura publicada sobre o controle de robôs móveis não-holonômicos com a utilização do método de Linearização com realimentação. O problema de controle é projetar uma lei de realimentação que possa estabilizar o robô em torno de um ponto de equilíbrio. Entretanto, um robô móvel não-holonômico não pode ser estabilizado para uma configuração de repouso através de leis suaves de realimentação, mesmo que seja completamente controlável Brockett et al.(1983),Pomet (1992), Samson (1995). Alguns pesquisadores têm proposto leis de realimentação de estados dependentes do tempo, leis de controle dependentes não somente do vetor de estados do sistema mas também explicitamente da variável temporal. Tal método não é uma extensão natural das soluções tradicionais, mas tem a vantagem de preservar a suavidade na lei de controle. Estas são as chamadas leis de controle não estacionárias, Pomet (1992), Samson (1995), Pomet et al. (1992), Sϕrdarlen (1995).

Este artigo está direcionado para o problema de estabilização em torno de um ponto de equilíbrio, mas também da varredura, ou seguimento, estável a um movimento de referência. A técnica de linearizar completamente o sistema de equações por meio de realimentação estática não é suficiente para assegurar a solução deste problema, não obstante a linearização completa pode ser obtida contanto que a velocidade longitudinal do robô seja diferente de zero Thuloit et al. (1996). A técnica de realimentação dinâmica tem sido de grande importância entre pesquisadores de sistemas não-holonômicos, mais precisamente de robôs móveis, porque garante o controle total e convergência exponencial para a referência de cada variável do sistema.

Propõe-se nesse artigo derivar, das técnicas de controle: Linearização com realimentação dinâmica e realimentação não estacionária, uma estratégia de controle e estabilização e aplicá-la ao modelo do robô móvel não-holonômico Khepera, Kepera (1995). A tarefa imposta ao robô é que ele se movimente de um estado inicial até um estado final seguindo uma trajetória previamente estabelecida. A varredura à trajetória de referência se dá com a utilização do método Linearização com Realimentação Dinâmica, mostrado na seção 3.1; a estabilização do robô para uma configuração de repouso é obtida usando as leis de realimentação descontínuas e variantes no tempo de uma forma híbrida, como se vê na seções 3.2.1 e 3.2.2. Esta metodologia está abordada em Pomet et al. (1992), Thuloit et al. (1996), Campion et al. (1996), mas não se mostram justificativas para a escolha do momento adequado da comutação entre as leis de controle. Mostra-se neste artigo que é importante a

determinação do momento adequado para a comutação, e se apresenta uma metodologia para a estimação deste ponto.

O artigo é organizado da seguinte forma: na seção 2 é apresentado o modelo cinemático do robô sob consideração e algumas propriedades estruturais inerentes aos robôs móveis são consideradas. Na seção 3 são apresentados o controle de varredura e estabilização através de leis de realimentação nos estados, o problema de controle se divide em duas partes. Primeiramente as leis de realimentação estáticas e dinâmicas são desenvolvidas, comparadas e aplicadas ao robô móvel sob consideração na seção 3.1, também seção 3.1 mostra-se os resultados de simulação das leis de realimentação estacionária. As leis de realimentação variante no tempo são discutidas e simuladas na seção 3.2. A tarefa completa para o robô é ,então, simulada. A seção 3.2.2 apresenta a estratégia híbrida e é feito um estudo para se estimar o melhor momento a comutação.

2 Modelo Cinemático de Configuração

2.1 Postura do Robô

A posição do robô é completamente descrita através do vetor ξ de coordenadas de postura:

$$\xi = (x, y, \theta)^T$$

onde x, y são as coordenadas cartesianas da origem do sistema de referência móvel {X1,Y1}, solidário ao robô, em relação ao sistema de referência inercial {X,Y}, e θ é o ângulo de rotação do referencial móvel em relação ao referencial inercial, Fig. 1.

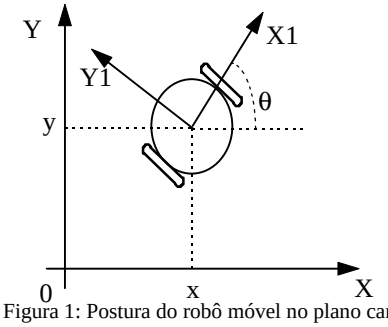


Figura 1: Postura do robô móvel no plano cartesiano.

2.2 Modelo Cinemático

O robô móvel sob consideração tem duas rodas fixas convencionais motorizadas independentemente. É mostrado em D'Andrea-Novel (1995), Isidori (1989) que o movimento desse tipo de robô é descrito pelo seguinte modelo de estado cinemático:

$$\dot{x} = B(x)u$$

onde $x \equiv \xi$; $B(x) = R^T(\theta)\Sigma$; $u = \eta$ e $\Sigma = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$.

O vetor de velocidades das rodas e o vetor de entradas η são relacionadas da seguinte forma:

$$\eta = D\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix};\ D = \begin{bmatrix} r/2 & r/2 \\ r/2R & -r/2R \end{bmatrix},$$

onde $\dot{q} = \{\dot{q}_1, \dot{q}_2\}$ é o vetor de velocidades das rodas, *r* é o raio das rodas e *2R* é a distância entre as rodas motorizadas do robô. O modelo cinemático pode então ser rescrito como,

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} -\frac{r}{2}sin\theta & -\frac{r}{2}sin\theta \\ \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2R} & -\frac{r}{2R} \end{bmatrix} \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{Bmatrix}$$

3 Controle de Trajetória Através de Realimentação de Estados

3.1 Leis de RealimentaçãoInvariantes no Tempo

As leis de realimentação dinâmica de estados são aplicáveis ao problema de seguimento de trajetórias que não contenham pontos de repouso para qualquer robô móvel com 2 ou menos rodas direcionáveis, como mostrado em D'Andrea-Novel (1995). Esta estratégia é aplicada ao modelo cinemático do robô móvel sob consideração.

Primeiramente é aplicado o procedimento de Linearização com realimentação estática D'Andrea-Novel (1995), Isidori (1989), (Slotine e Li, 1991), Marino (1986); seja o modelo cinemático do robô rescrito a seguir:

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} -sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} \eta_1 \\ \eta_2 \end{Bmatrix} \qquad (9)$$

$$\varphi = \begin{Bmatrix} \varphi_1 \\ \varphi_2 \end{Bmatrix} = \begin{Bmatrix} x \\ y \end{Bmatrix} \qquad (10)$$

onde φ é o vetor de funções de saída escolhido.

Diferenciando a função de saída, nós facilmente verificamos a presença da entrada η_1, a velocidade longitudinal do robô, mas a matriz *E* é singular. O vetor de entrada será então "atrasado" gerando uma nova entrada η_3 definida como: $\dot{\eta}_1 = \eta_3$. Isto resulta o seguinte modelo estendido:

$$\begin{aligned}\dot{x} &= -sin\theta\eta_1 \\ \dot{y} &= \cos\theta\eta_1 \\ \dot{\theta} &= \eta_2 \\ \dot{\eta}_1 &= \eta_3\end{aligned} \qquad \chi_e = \begin{bmatrix} x \\ y \\ \theta \\ \eta_1 \end{bmatrix} \qquad u_e = \begin{Bmatrix} \eta_2 \\ \eta_3 \end{Bmatrix} \tag{11}$$

onde χ_e é o vetor estendido de estado, e u_e o novo vetor de entrada.

Prosseguindo com a linearização estática, aplicado sobre o modelo estendido

$$\ddot{\varphi} = \begin{bmatrix} -\cos\theta\eta_1 & -sin\theta \\ -sin\theta\eta_1 & \cos\theta \end{bmatrix} \begin{Bmatrix} \eta_2 \\ \eta_3 \end{Bmatrix}$$
$$\ddot{\varphi} = E(\chi_e)U \tag{12}$$

Nota-se que a matriz E é inversível sempre que $\eta_1 \neq 0$; a velocidade do robô não deve ser zero. Escolhendo a lei de controle U tal que as não linearidades sejam canceladas: $U = E^{-1}(\chi_e)\nu$ onde ν é um vetor de entradas auxiliares que será convenientemente projetado. A aplicação desta lei de controle permite que o modelo estendido, descrito pela equação (11), seja escrito como o seguinte sistema linear,

$$\ddot{\varphi} = \nu \tag{13}$$

Considerando ϕ_{ref} como a trajetória de referência a ser seguida pela função de saída φ, k_1 e k_2 os ganhos de realimentação, então o controle auxiliar ν pode ser projetado como,

$$\begin{aligned}\nu &= \ddot{\varphi}_{ref} - k_2\dot{\tilde{\varphi}} - k_1\tilde{\varphi} \\ \tilde{\varphi} &= \varphi - \varphi_{ref}\end{aligned} \tag{14}$$

onde $\tilde{\varphi} = \varphi - \varphi_{ref}$ é o erro de convergência.

A substituição de (14) em (13) resulta em uma dinâmica exponencialmente convergente para o erro,

$$\ddot{\tilde{\varphi}} + k_2\dot{\tilde{\varphi}} + k_1\tilde{\varphi} = 0$$

A simulação se dá em duas partes, considerando uma trajetória não singular onde pode ser aplicado o método de Linearização por Realimentação Dinâmica; e considerando uma trajetória contendo um ponto de singularidade.

A trajetória de referência não singular a ser seguida pelo robô é:

$$x_{ref} = -\frac{0.0125}{3}t^3 + \frac{0.1}{3}t^2 - 0.8t + 1$$
$$y_{ref} = x_{ref}$$

É não singular porque $\eta_{ref} \neq 0$, $\forall t \geq 0$. Com as seguintes condições iniciais.

$$x(0) = x_{ref}(0) - 0.2 = 0.8m$$
$$y(0) = y_{ref}(0) + 0.2 = 1.2m$$
$$\dot{x}(0) = \dot{y}(0) = \dot{x}_{ref}(0) = \dot{y}_{ref}(0) = -0.8m/s$$

O objetivo do robô é seguir uma linha reta no plano cartesiano *xy*.

Esta trajetória de referência foi projetada com base nas limitações de velocidade dos motores do robô Khepera, sendo a máxima velocidade permitida para cada motor de 127 pulsos/10ms, correspondendo a 1m/s. As figuras 2 e 3 mostram os resultados de simulação do controlador com Linearização por realimentação dinâmica.

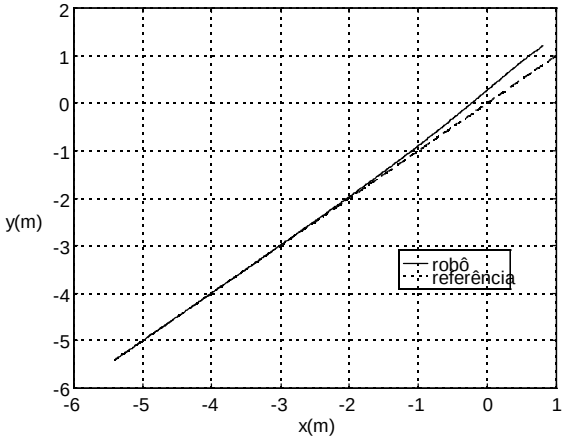


Figura 2: Varredura à trajetória não singular.

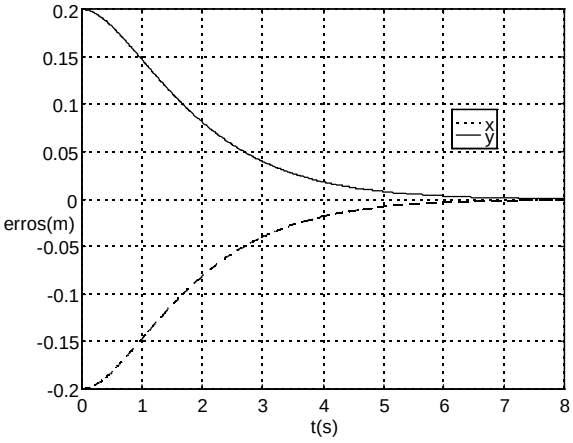


Figure 3- Os erros de convergência.

3.2 Leis de Realimentação Variantes no Tempo e a Estratégia Híbrida

Os métodos de realimentação apresentados acima somente funcionam nos casos onde a condição de Brockett é evitada (trajetória sem configurações de repouso). No presente caso, como visto na seção 3.1.2, a matriz de desacoplamento $E(x,\eta_1)$ é

inversível somente se a entrada original do sistema η_1 for diferente de zero. Como visto na seção 2, a entrada η_1 é a velocidade longitudinal do robô móvel a qual restringe as aplicações do método de linearização por realimentação dinâmica aos casos onde o robô se movimenta sem parar, neste caso o robô móvel não encontra pontos de singularidades. Entretanto, o que se espera do robô móvel é a realização de uma tarefa completa de mover-se de um ponto inicial para um ponto final seguindo uma trajetória previamente estabelecida no plano. Como mencionado na seção 1, esta tarefa tem sido um desafio que requer soluções de controle mais avançadas. Apresenta-se na próxima seção soluções alternativas para a solução deste problema.

3.2.1 Leis de Realimentação Variantes no Tempo

Realimentação variante no tempo é uma estratégia que depende não somente do vetor de estados mas também explicitamente da variável temporal. Um projeto explícito de leis de controle estabilizantes com realimentação variantes no tempo para uma classe de sistemas controláveis é apresentado na referência Pomet (1992). O método de projeto proposto em Pomet (1992) pode ser aplicado somente se o modelo do robô móvel seja completamente controlável, o modelo do robô Khepera é não-holonômico e completamente controlável Thuloit et al. (1996), Campion et al. (1996), (Bloch e McClamroch, 1990). Para estabilizar o robô para uma configuração final, a lei de controle periódica CL3, proposta em Pomet et al. (1992), é aplicada, onde as entradas de controle são as seguintes:

$$u_1 = x\sin\theta - y\cos\theta$$
$$\frac{\dot{\psi}(\theta)}{\lambda}u_2 = x\sin t - (\frac{\psi(\theta)}{\lambda} + x\cos t) \qquad (9)$$

Vê-se em Fig. 4 o resultado de simulação desta lei de controle com $\lambda = 40$ e $\psi(\theta) = \theta$.

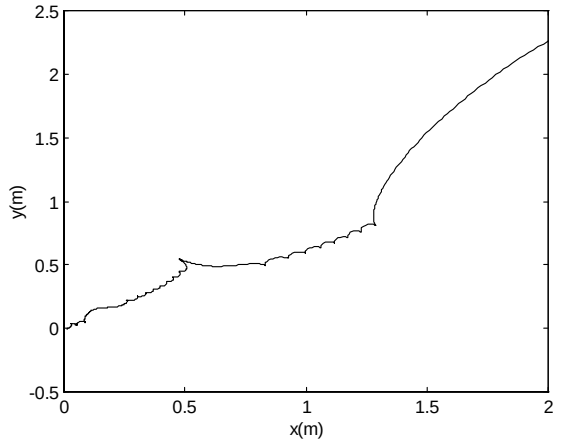


Figura 4: Lei de realimentação variante no tempo e convergência para o ponto de repouso (0,0).

Os estados do robô convergem para a origem, neste caso, fato este considerado impossível com o uso de realimentação de estados invariantes no tempo, como a linearização por realimentação dinâmica.

As leis de controle variantes no tempo não são suficientes para se conseguir a realização de uma tarefa completa para o robô. Com tais leis, o robô pode ser levado de uma estado inicial para um final mas nenhuma trajetória de referência pode ser especificada. Além disso, desde que as trajetórias realizadas com tais leis são muito oscilantes, devido a dependência temporal das funções de Lyapunov, não é possível aplicar tais estratégias se o ponto alvo do robô for muito longe do ponto inicial. Portanto, uma estratégia híbrida deve ser usada. Híbrida porque se faz uso das duas técnicas: Linearização com realimentação dinâmica durante o controle de varredura à trajetória de referência e Realimentação variante no tempo durante o controle de estabilização do robô para o ponto final de repouso.

3.2 A Estratégia Híbrida

Nesta seção se estabelece a tarefa completa para o robô, seguir uma determinada trajetória de referência preestabelecida até um ponto objetivo que é o estado final do movimento. Este ponto objetivo é caracterizado por um estado de repouso onde a velocidade longitudinal do robô, expressa por η_1 em (9), é igual a zero. A lei de realimentação dinâmica, apresentada na seção 3.1, não é apropriada para ser aplicada nessa parte do movimento, pois como se vê no modelo estendido (12), a matriz de controlabilidade, $E(\chi_e, \eta_1)$ dada por:

$$E(\chi_e, \eta_1) = \begin{bmatrix} -\cos\theta\eta_1 & -\sin\theta \\ -\sin\theta\eta_1 & \cos\theta \end{bmatrix}$$

é função de η_1 e não possui inversa se η_1=0, nesse caso a lei de controle dada por $U = E^{-1}(\chi_e, \eta_1)\nu$ não pode ser calculada com precisão.

Considere uma trajetória de referência singular com um ponto final de repouso na origem (0,0) dada por:

$$x_{ref} = y_{ref} = \frac{0.0125}{3}(8-t)^3$$

com as seguintes condições iniciais :

$$x(0) = 2m$$
$$y(0) = 2.26m$$
$$\dot{x}(0) = \dot{y}(0) = -0.8m/s$$

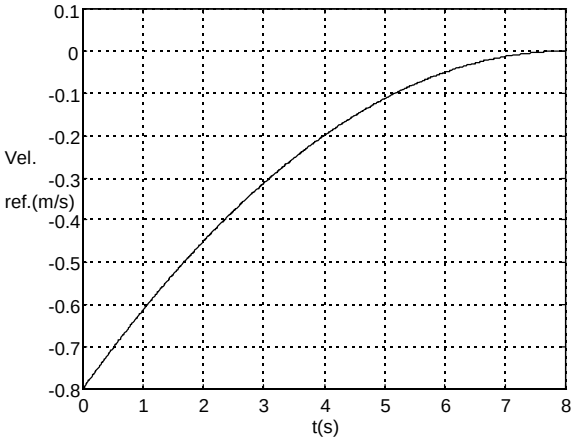


Figura 5: Velocidade de referência em função do tempo.

Em Fig. 5 mostra-se a velocidade de referência em função do tempo, vê-se que em t=8s a velocidade é zero o que constitui um ponto de singularidade na trajetória. Portanto, para que o robô seja conduzido para o ponto final de repouso, se faz necessário o uso de uma estratégia híbrida a qual combina a realimentação dinâmica com a realimentação variante no tempo, a metodologia é a seguinte: O movimento de varredura à trajetória de referência sob o comando de realimentação dinâmica se estende até que o robô esteja em uma circunferência centrada na origem com um certo raio, neste ponto ocorre uma comutação para estratégia de controle variante no tempo para que o estado do robô possa ser estabilizado para a origem, onde a lei de realimentação dinâmica tenderia para o infinito.

O momento da comutação será estimado pela observação da variação do número de condição da matriz de controlabilidade $E(\chi_e, \eta_1)$. A figura 6 mostra o número de condição em função tempo.

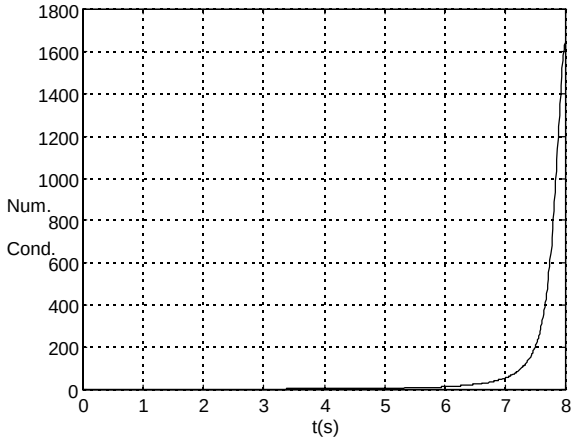


Figura 6: Nº de Condição da matriz de controlabilidade.

Para t>3.5s o número de condição da matriz $E(\chi_e, \eta_1)$ é maior que 2, conclui-se então que a partir desse ponto a matriz de controle se torna extremamente mal condicionada e o calculo de sua inversa é numericamente instável, o que nos indica que este ponto é uma estimativa para se efetuar a comutação entre as estratégias de controle.

A figura 7 mostra a estabilização do robô para o ponto de repouso (0,0) no plano cartesiano. A comutação para a lei de controle variante no tempo ocorre quando o robô atinge a circunferência de raio 0.37m centrada na origem.

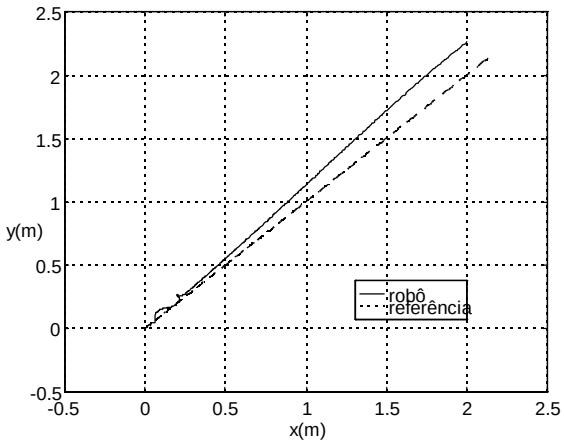


Figura 7: Varredura à trajetória singular.

4 Conclusões

Apresentou-se uma estratégia global combinando lei de realimentação dinâmica para a varredura à trajetória de referência não singular e a lei de realimentação variante no tempo para a estabilização final. Apresentou-se, como uma contribuição original, uma metodologia para se estimar o momento adequado de se efetuar a comutação na estratégia híbrida. Resultados de simulações, usando o MATLAB, são apresentados onde pode ser visto que os erros de varredura são exponencialmente convergentes para zero.

5 Referências

Brockett, R. W., et al. (1983). Asymptotic stability and feedback stabilization, *Differential geometric control theory*, Boston: Birkhauser. pp. 181-191.

Pomet, J. -B. (1992). Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift, *Systems & Control Letters*, 18, pp. 147-158.

Samson, C. (1995). Control of chained systems application to path following and time-varying point-stabilization of mobile robots, *IEEE Trans. Automatic control*, 40(1), pp. 64-77.

Pomet, J. -B. et al. (1992). A hybrid strategy for the feedback stabilization of nonholonomic mobile robots, *IEEE Int. Conf. Robotics and Automation*, Nice, France.

Sϕrdarlen, O. J. (1995). Exponential stabilization of nonholonomic chained systems. *IEEE Trans. on automatic control*, 40(1), pp. 35-49.

Samson, C. et al. (1990). Mobile robot control part 1: feedback control of a nonholonomic wheeled cart in cartesian space, *Report INRIA, ICARE*.

Thuloit, B., et al. (1996). Modeling and feedback control of mobile robots equipped with several steering wheels, *IEEE Trans. robotics and automation*, 12(3), pp. 375-390.

Kepera (1995). *USER MANUAL*. Version 4.06.

Campion, G. et al. (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots, *IEEE Trans. robotics and automation*, 12(1), pp. 47-62.

D'Andrea-Novel, B. (1995). Control of nonholonomic wheeled mobile robots by state feedback linearization, *The int. Journal of robotics research*, 14(6), pp.543-559.

Isidori, A. (1989). *Non Linear Control Systems*. New York: Springer-Verlag, 2nd Ed..

Slotine, J. J. & Li, W.(1991). *Applied nonlinear control*, Englewood Cliffs, NF: Prentice Hall.

Marino, R. (1986). On the largest feedback linearizable subsystem, *Systems & Control Letters*, 6, pp. 345-351.

Bloch, A. M. & McClamroch, N. H. (1990). Contollability and stabilizability properties of a nonholonomic control system, *Proc. 29th Conf. on decision and control*, December.

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 599-603 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

PLANEJAMENTO DE TRAJETÓRIA DE UM AGV UTILIZANDO REDES NEURAIS

ALEXANDRE SCHAMMASS, CARLOS M. O. VALENTE, GLAUCO A. P. CAURIN

Laboratório de Mecatrônica, Depto. de Eng. Mecânica, Universidade São Paulo
P.O. Box 359, 13560-250 São Carlos, SP, BRAZIL
E-mails: schammas@ifqsc.sc.usp.br, cmov@sc.usp.br, caurin@umc.br

Resumo— Este artigo apresenta o desenvolvimento de um sistema de planejamento de trajetória que utiliza redes neurais e métodos clássicos de forma complementar. Objetivamos com este trabalho aliar a flexibilidade e a inteligência dos sistemas biológicos à robustez e confiabilidade das máquinas industriais disponíveis atualmente. O planejador de trajetórias, comum no comando clássico do robô, é realizado por duas redes neurais. A primeira rede gera a seqüência de coordenadas da trajetória e a segunda rede faz o papel de cinemática inversa. Os resultados apresentados aqui correspondem ao estágio atual de implementação de um sistema neural para a coordenação senso-motora de um robô móvel (AGV) que trabalha em ambientes pouco estruturados e variáveis no tempo.

Abstract— This paper presents the development of a trajectory planning system that use neural networks and knowledge from the classic control system theory in a complementary form. The purpose of this approach is to combine the flexibility and the intelligence of biological systems and reliability of current industry machines. The planner, commonly used in classic robot command, is implemented with two neural networks. The first network yields the sequence of trajectory coordinates and the second one performances the inverse kinematics. The results presented in this paper correspond to the state of art in the implementation of a neural system for sensor-motor coordination of an autonomous guided vehicle (AGV) working on time variant semi-structured environment.

Keywords— Autonomous mobile robots, Inverse kinematic problem, Neural networks, Trajectory planning.

1 Introdução

A automação de processos cada vez mais complexos em ambientes não estruturados, como por exemplo hospitais, escritórios ou mesmo residências, exige um nível de inteligência de máquina muito maior do que o encontrado nos robôs e em outros sistemas mecatrônicos atuais.

A capacidade de trabalhar de forma rápida e eficiente com a presença constante de situações excepcionais, incertezas e outras características de ambientes menos estruturados que os cenários tradicionais de chão de fábrica, tem estimulado o uso de metodologias baseadas no funcionamento de sistemas biológicos.

Se a flexibilidade e a inteligência dos sistemas biológicos representam uma necessidade para as máquinas do futuro, não seria coerente no entanto, que estas máquinas apresentassem características inferiores de robustez e confiabilidade quando comparadas às máquinas industriais disponíveis atualmente. Portanto, o que buscamos em nossas pesquisas é utilizar métodos clássicos e métodos baseados em sistemas biológicos de forma complementar.

Seguindo esta linha de pensamento, pesquisamos sistemas mecatrônicos utilizando redes neurais, sem deixar de explorar os conceitos e conhecimentos já disponíveis e obtidos anteriormente por métodos clássicos de modelagem e controle. As formas escolhidas para introduzir o conhecimento prévio no sistema neural são; codificação funcional das redes e escolha da sua arquitetura. As redes desempenham funções semelhantes e trabalham com as mesmas informações que os sistemas convencionais.

Foram realizados testes computacionais (simulações) e testes experimentais com um AGV. Para permitir as simulações computacionais adotou-se um modelo cinemático do AGV, que também foi utilizado para o treinamento das redes. O sistema de redes neurais, que realiza o planejamento e controle de trajetória do robô, explora os aspectos da arquitetura perceptron com 2 e 3 camadas do tipo Feed-forward (Santos, 1994).

Este artigo apresenta também uma evolução em relação ao trabalho apresentado por Caurin (1996). Na seção 4 é mostrado o treinamento da rede neural responsável por parte do planejamento de trajetória (Net 1) levando em consideração as restrições do modelo, obtendo assim um melhor desempenho.

É interessante ressaltar que tanto os programas de simulação como os programas de implementação no robô foram desenvolvidos em Oberon 2. Oberon (Wirth e Reiser, 1992) é a última linguagem descendente da família do Algol 60, e cujos outros membros são o Pascal e o Modula 2. Esta linguagem unifica os conceitos tradicionais da programação procedural com as técnicas necessárias para obter a extensibilidade de dados (orientação a objeto), através dos módulos (Mössenböck, 1993). Além de uma linguagem, Oberon também possui um sistema operacional que proporciona um ambiente de janelas gráficas utilizando hipertexto, o que o facilita muito ao usuário, podendo ser instalado em um computador pessoal ou em uma estação de trabalho.

2 Modelo do Robô

A modelagem do robô é feita de uma forma simplificada, levando em consideração apenas os aspectos cinemáticos, ou seja, não consideramos a dinâmica do sistema (forças de atrito, torque dos motores, momentos de inércia, massa, etc.). O modelo consiste em um chassi retangular com duas rodas trasei-

ras e uma roda dianteira responsável pelo torque e pelo esterçamento (Fig. 1).

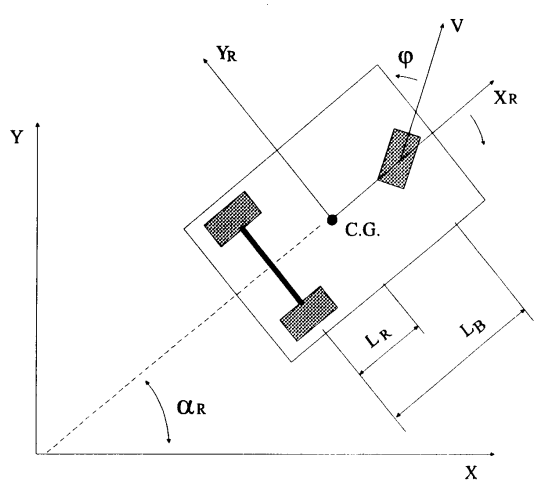


Figura 1. Modelo cinemático do robô.

Através da análise cinemática, obtivemos as equações que relacionam a velocidade (v) e o esterçamento (φ) da roda dianteira com as velocidades absolutas em relação ao sistema de coordenadas inerciais XY. As equações que relacionam a velocidade e esterçamento da roda dianteira (v, φ) e as velocidades relativas são:

$$V_{RX} = v.\cos\varphi \quad (1)$$

$$V_{RY} = v.\operatorname{sen}\varphi.\frac{l_R}{l_b} \quad (2)$$

$$\omega_R = v.\frac{\operatorname{sen}\varphi}{l_b} \quad (3)$$

onde V_{RX} é a velocidade relativa do robô na direção X_R, V_{RY} a velocidade relativa na direção Y_R e ω_R a velocidade angular relativa ao robô, que também será a velocidade angular absoluta em relação ao eixo XY. Os valores l_R e l_b são as dimensões do robô indicadas na Fig. 1.

Assim, obtivemos as equações que determinam as velocidades globais em função das variáveis relativas, como mostra o sistema de matrizes abaixo:

$$\frac{d}{dt}\begin{bmatrix} X \\ Y \\ \alpha_R \end{bmatrix} = \begin{bmatrix} \cos\alpha_R & -l_R \operatorname{sen}\alpha_R \\ \operatorname{sen}\alpha_R & l_R \cos\alpha_R \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} V_{RX} \\ \omega_R \end{bmatrix} \quad (4)$$

onde $\frac{dX}{dt}, \frac{dY}{dt}, \frac{d\alpha_R}{dt}$ são as velocidades absolutas.

Note que a velocidade angular absoluta é igual à relativa. Evidentemente precisamos das posições globais, não das velocidades, porém as equações acima não possuem uma solução literal, apenas numérica (sistema não-holônomo).

3 Sistema Experimental

O SmartRob, ilustrado na Fig. 2, é um pequeno robô móvel com aproximadamente 40 cm de comprimento e três rodas, sendo duas traseiras e uma roda dianteira, responsável pelo movimento e esterçamento do robô, idêntico ao nosso modelo computacional apresentado na Fig.1.

A parte eletrônica é composta por um microcontrolador 68332 BCC com TPU (Time Processing Unit), drivers para sensores, atuadores e também para câmeras tipo CCD que permitem a utilização adicional de 2 sensores de infravermelho e 1 de ultra-som, como também 3 motores de passo, 4 servos-motores e também 2 câmeras de vídeo. Todo o sistema é alimentado por duas baterias de 12 V em série (+24 V) com a capacidade de 3 A/h.

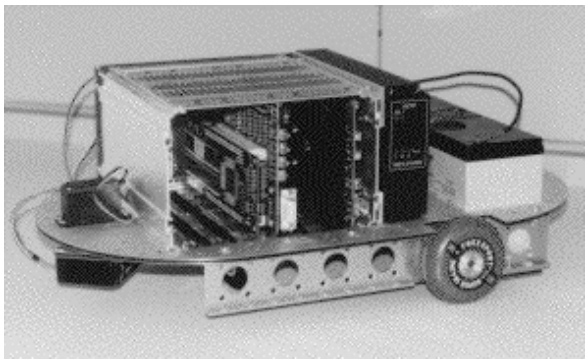


Figura 2. AGV SmartRob utilizando como plataforma de testes para o sistema neural de planejamento e controle

Para programarmos este AGV, utilizamos o ambiente D'NIA, que é um cross-compiler para desenvolvimento de sistemas que exigem tempo real baseado em Oberon 2, dispensando assim o uso de qualquer rotina tipo Assembler. Para carregarmos um programa na memória do robô, apenas conectamos um cabo serial do micro (no caso um Macintosh Performa 6200CD) ao robô. Após o carregamento dos programas objeto o cabo pode ser desconcertado (embora isto não seja necessário) e o robô pode funcionar de forma autônoma.

4 Planejador de Trajetórias

O planejamento da trajetória (sistema que gera os valores desejados para o controlador) é realizado por duas redes separadas como ilustra a Fig. 4. Como já mencionamos na introdução, utilizamos conhecimentos prévios provenientes da teoria de controle clássico para desenvolver o sistema senso-motor do robô.

Olhando de forma macroscópica (Fig. 4) as partes Net1 e Net2 pouco se diferenciam em suas funções de sistemas de controle convencionais de robôs. No entanto, pelo fato das redes neurais serem sistemas inerentemente não-lineares, é de se esperar

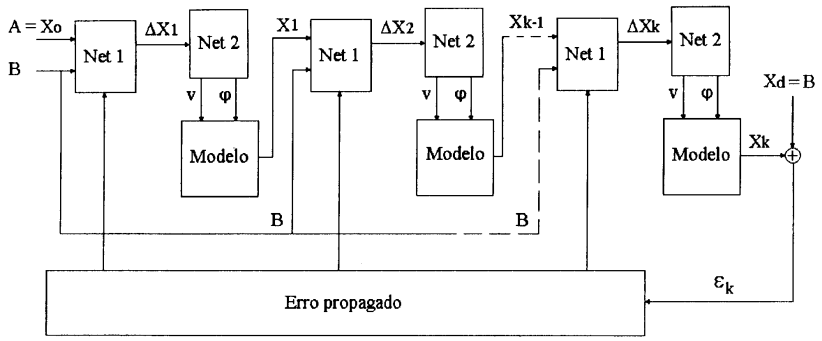


Figura 3. Treinamento da rede Net 1 pelo segundo método.

um desempenho superior em relação aos controladores clássicos.

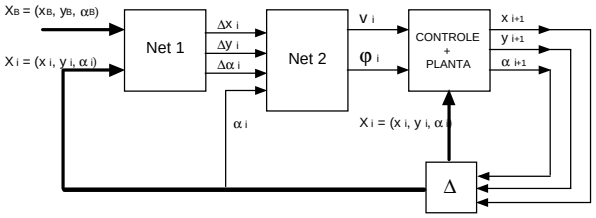


Figura 4. Representação Geral

No caso, o primeiro bloco Net1 recebe do usuário as coordenadas de uma posição para onde o robô deve navegar. A rede Net1 fornece então para Net2 toda a seqüência de coordenadas da trajetória a ser percorrida, assumindo assim o papel de planejador de trajetórias em coordenadas espaciais. O planejador de trajetórias é um componente típico do comando clássico de robôs.

A Net2 é responsável pela transformação dos incrementos em coordenadas cartesianas para incrementos em coordenadas do robô (cinemática inversa).

4.1 Rede Neural 1 (Net 1)

Para treinar a Net 1 não foi utilizado um emulador neural do modelo do robô, como apresentado por Nguyen e Widrow (1990), nem o erro é propagado através de expressões matemáticas muito simples (Caurin, 1996). Neste artigo apresentamos um método de treinamento que leva em conta o modelo do robô e a cinemática inversa (Net 2 já treinada) para o treinamento da rede Net 1. Como ilustra a Fig. 3, as entradas da rede são a posição atual do AGV, dada pelas coordenadas X,Y,α e a posição desejada (X_B, Y_B, α_B). As saídas são as alterações de coordenadas (deslocamentos).

Para ilustrar (Fig. 3) a forma de treinamento adotado de forma simplificada, nos restringimos apenas à coordenada X, lembrando que o mesmo foi feito para as coordenas Y e α. Simulamos a rede para a posição X atual, obtendo um ΔX. Este ΔX é entrada da rede Net 2, que funciona como cinemática inversa, fornecendo ao modelo a velocidade e esterçamento necessários para este executar o deslocamento ΔX. Assim obtemos o X_{i+1} na saída do modelo, que será aplicado novamente à entrada da rede Net 1, repetindo este processo até obtermos X_k. Este valor é então comparado com a posição final desejada X_d, obtendo ε_k. A partir deste erro (variação de deslocamento desejado) podemos aplicar Back Propagation ajustando os pesos da Rede Neural, repetindo este processo até atingirmos um erro aceitável. A vantagem deste método é garantir um erro pequeno no ponto de destino da trajetória pois leva em conta as imperfeições de treinamento da segunda rede e as restrições de movimento do robô.

Em relação à normalização das entradas e saídas, como estamos usando a função tangente hiperbólica, mapeamos a nossa área de movimento em coordenadas (x, y) de -1 a 1, ou seja, um quadrado 2 por 2. A saída foi normalizada para que não ultrapassasse 0,1 em módulo. Como adotamos k_{max} = 20 na maior distância a ser percorrida, é necessário um deslocamento igual a 0,1m (v = 1m/s) em cada uma de suas componentes.

Os resultados neste caso foram satisfatórios, treinando uma rede com 6 entradas, uma camada escondida de 10 neurônios e 3 neurônios de saída (6:10:3). Utilizamos 51 exemplos gerados aleatoriamente e o erro médio quadrático foi de 0,2 em 1000 épocas, o que é suficiente pois pudemos observar o significado deste erro através dos valores gerados pela simulação. É interessante observar que a convergência foi mais rápida em relação à Net 2 por se tratar de um modelo linear.

4.2 Rede Neural 2 (Net 2)

Para o treinamento da rede responsável pela cinemática inversa, utilizamos o modelo do robô recebendo como entrada os valores da velocidade v e do

ângulo de esterçamento φ, gerando assim uma tabela de valores (Δx, Δy, Δα, α_i, α_{i+1}, v, φ).

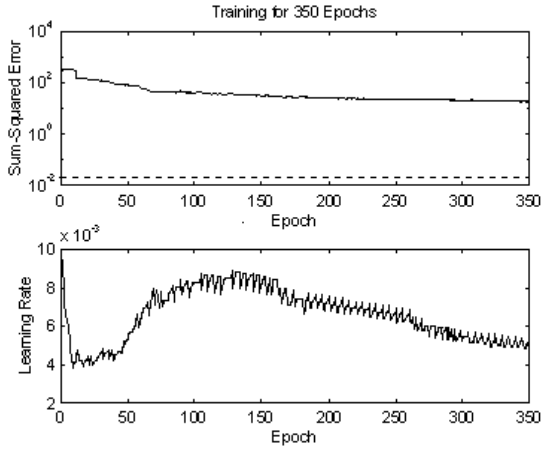


Figura 5. Gráfico do erro médio quadrático e taxa de aprendizado em relação às épocas utilizando back-propagation adaptativo.

No treinamento geramos os exemplos através de trajetórias simples com velocidade v = ±1m/s e esterçamento φ entre ±0,1; ±0,3; ±0,5; ±0,7 radianos, com orientação do robô α variando de -180 até 180 graus. Para isto, fizemos o modelo descrever uma semi circunferência para cada par (v,φ), sendo que em cada curva foram tomados 10 pontos como exemplos do treinamento da rede neural, portanto 160 exemplos.

Neste treinamento utilizamos o Back Propagation com taxa de aprendizado adaptativo (Braspenning et al., 1995), o que apresentou resultados de aprendizado mais rápidos que o Back Propagation comum. Porém, pudemos observar que no final do aprendizado, próximo ao erro mínimo, o treinamento se torna muito instável (Fig. 5).

Portanto, treinamos uma Rede Neural 4:15:15:3 com até 350 épocas através de Back Propagation com taxa de aprendizado adaptativo (Fig. 5) e posteriormente 10000 épocas utilizando Back Propagation com momento a fim de se obter uma estabilidade maior no processo de convergência (Fig. 6). Para ilustrar este treinamento, as Fig. 5 e Fig. 6 representam o comportamento do erro médio quadrático e da taxa de aprendizado em relação às épocas.

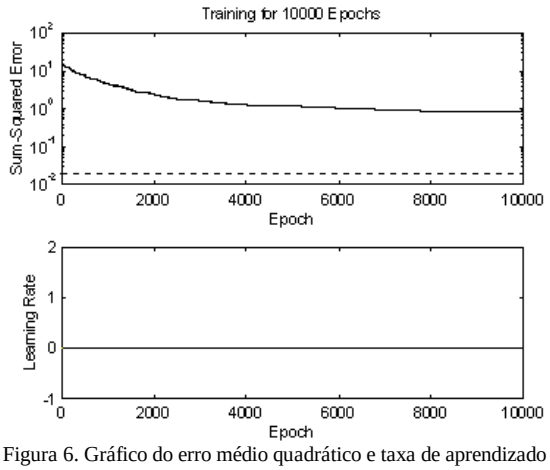


Figura 6. Gráfico do erro médio quadrático e taxa de aprendizado em relação às épocas utilizando back-propagation

A fim de observar o que este erro representa, após treinar a rede para 10000 épocas, tomamos os pesos da Rede Neural treinada e a testamos através do modelo do robô conforme mostra a Fig. 7.

Assim, geramos os deslocamentos (Δx, Δy, Δα) com as mesmas velocidades utilizadas nos exemplos de treinamento. Testamos também a Rede Neural para valores de φ (entre –0,7 a 0,7 radianos) não apresentados no treinamento. Os erros apresentaram-se menores que 5% para a maioria dos pontos e pudemos observar a capacidade de generalização da rede.

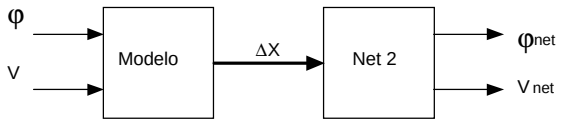


Figura 7: Esquema que mostra a simulação e teste da rede Net2.

Os testes de implementação são idênticos tanto para o modelo simulado do robô como para o AGV experimental. Fornecemos ao programa a posição "A" (x_A, y_A, α_A) onde o robô se encontra inicialmente e a posição final "B" (x_B, y_B, α_B) a qual o robô deve alcançar em k movimentos, onde k é determinado em função da distância a ser percorrida. Os valores da posição atual e final "B" serão as entradas da primeira rede (Net 1) que fornecerá em sua saída um deslocamento de posição ΔS (Δx, Δy, Δα). Este ΔS é então aplicado à entrada da segunda rede (Net 2), obtendo-se na saída a velocidade e esterçamento desejados para realização do movimento ΔS.

A Fig. 8 apresenta o resultado de uma simulação em Oberon. Neste exemplo, temos como posição inicial A = (0; 10; 0) e posição final desejada B = (10; 0; -1,57). Obtivemos a posição final F = (9,3; -1,6; -2,1).

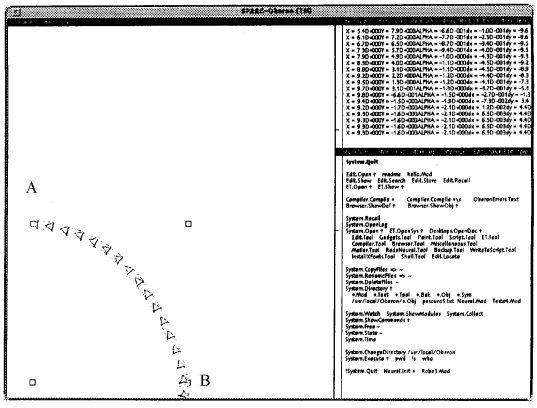


Figura 8: Simulação com o controlador neural implementado

5 Conclusão

Com respeito à primeira rede (Net 1), planejador de trajetória, pudemos concluir que ao dividirmos o controlador neural em duas redes, facilitamos o aprendizado do planejador de trajetória, pois a rede

possui agora um problema linear para ser resolvido. A vantagem do método utilizado para o treinamento é que este considera as imperfeições da rede Net 2 e as limitações de movimento do robô.

Na segunda rede (Net 2), concluímos que os exemplos influenciam bastante no aprendizado. Não podemos ter um grande número de exemplos, pois o aprendizado será mais difícil, porém este número deve ser suficiente para que a rede possa generalizar. Também verificamos que a rede possui uma convergência melhor quando apresentamos os exemplos no treinamento de maneira que cada entrada corresponda a uma única saída, razão que nos levou a mudar a configuração de entrada e saída.

Na implementação concluímos que os resultados são satisfatórios e representam uma melhoria em relação aos resultados apresentados por Caurin (1996) pois utilizamos um novo método para o treinamento da rede neural Net 1. No entanto, verificamos que para trajetórias que exigem grande manobrabilidade do robô os resultados ainda não são satisfatórios. Uma possível solução para o problema seria o aumento do número k de movimentos (vide seção 4) quando se exige maior manobrabilidade.

Com respeito ao AGV experimental concluímos que todo o sistema oferece uma grande facilidade de utilização e flexibilidade para sofisticarmos o sistema conforme nossas necessidades. Isto se deve principalmente pelo fato do sistema utilizar uma linguagem de programação de alto nível.

É interessante ressaltar que este trabalho continua em desenvolvimento, visando como próximo passo a presença de obstáculos. Para isto, serão necessárias algumas modificações na primeira rede (Net 1) utilizando porém a mesma metodologia de treinamento.

Referências Bibliográficas

Braspenning, P. J., Thuijsman, F. e Weijters, A. J. M. M. (1995). *Artificial Neural Networks - An Introduction to ANN Theory and Practice*, Springer.

Caurin, G. P. (1996). Coordenação Senso- Motora de um AGV Utilizando Redes Neurais e Conhecimento Prévio, *Anais do III Simpósio Brasileiro de Redes Neurais*, Recife.

Mössenböck, H. (1993) Object-Oriented Programming in Oberon-2, Springer Verlag

Nguyen, D. H. e Widrow B. (1990). Neural Networks for Self-Learning Control Systems, *IEEE Control Magazine*, April, pp. 18-23.

Santos, V. P. (1994), Aplicação de Redes Neurais no Reconhecimento de Doenças do Coração utilizando o MATLAB, ICMSC-USP.

Wirth, N. e Reiser, M. (1992). *Programming in Oberon. Steps beyond Pascal and Modula-2*. Addison Wesley.

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 605-610 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

CONTROLE EM COORDENADAS POLARES DE ROBÔS MÓVEIS COM RODAS

WALTER F. LAGES

Depto. de Física, Fundação Universidade do Rio Grande
Av. Itália, Km 8, 96201-901 Rio Grande, RS, BRASIL
E-mail: w.fetter@ieee.org

ELDER M. HEMERLY

Depto. de Controle e Conversão de Energia, Instituto Tecnológico de Aeronáutica
CTA - ITA - IEEE, 12228-900 São José dos Campos, SP, BRASIL
E-mail: hemerly@ele.ita.cta.br

Resumo- Sabe-se que transformações de coordenadas adequadas podem ser bastante úteis no desenvolvimento de controladores para sistemas não-holonômicos. A literatura mostra que a transformação para coordenadas polares permite o controle de robôs móveis com acionamento diferencial. No entanto, não existe um método para obtenção de tais transformações. Neste artigo propõe-se transformações de variáveis adicionais que permitem a obtenção de leis de controle para todas as classes de robôs móveis com rodas. As leis de controle obtidas garantem que a trajetória do robô converge assintóticamente para uma determinada posição e orientação.

Abstract- It is well known that a suitable coordinate transform can be very effective to the design of controllers for nonholonomic systems. A literature search shows that differential drive robots can be controlled in polar coordinates. Nonetheless, a method to obtain such transforms is not available. This paper proposes a set of further variable transforms that enable us to obtain control laws for all classes of wheeled mobile robot. The designed control laws ensure asymptotic convergence to specified position and orientation.

Keywords- mobile robots; nonlinear control; nonholonomic systems; automated guided vehicles; robotics

1 Introdução

Os principais métodos para controle de postura (posição e orientação) originam-se no fato de que um sistema não-holonômico sem deriva não pode ser estabilizado em um ponto através de uma realimentação suave invariante no tempo (Brockett, 1983).

Alguns métodos (Walsh et al, 1994 e Thuilot et al., 1996) abandonam a idéia de estabilização em um ponto e procuram obter convergência para uma trajetória. Outros métodos mantêm a exigência de convergência para um ponto às custas de leis de controle mais complexas, como realimentação não suave (Bloch et al., 1992 e Canudas de Wit e Sørdalen, 1992). Utilizando-se uma lei de controle suave, é possível obter-se convergência para um ponto se esta lei de controle for variante no tempo (McCloskey e Murray, 1997). No entanto, leis de controle variantes no tempo geralmente produzem trajetórias demasiadamente oscilantes.

Também são utilizadas leis de controle híbridas (Oelen et al., 1995). Estas leis diferenciam-se das demais por combinarem leis de controle de tempo contínuo e leis de controle de tempo discreto. Na maioria das vezes ocorre chaveamento entre uma lei de controle não suave e uma lei de controle variante no tempo.

Como sugerido em McCloskey e Murray (1997) e McClamroch e Kolmanovsky (1995), o uso de transformações de coordenadas não suaves seguido de uma realimentação suave é promissor. A idéia nesta abordagem é fazer uma transformação de coordenadas de forma que o sistema transformado seja descontínuo, evitando-se assim as limitações impostas por Brockett (1983).

Neste artigo são propostas transformações adicionais que permitem generalizar os resultados de Aicardi et al. (1995) para todas as classes de robôs móveis com rodas. Uma transformação de coordenadas torna o modelo do sistema descontínuo na origem. Através de uma análise *Lyapunov-like*, é obtida uma lei de controle suave e invariante no tempo tal que, em malha fechada, o sistema seja contínuo também na origem.

2 Modelagem

Neste trabalho serão considerados apenas robôs com acionamento através de rodas. Será utilizado o modelo cinemático, obtido através de uma formulação semelhante a Campion et al. (1996).

Como detalhado na Fig. 1, as coordenadas de postura do robô descritas no sistema $\{X_0, Y_0\}$ e a matriz de rotação entre $\{X_c, Y_c\}$ e $\{X_0, Y_0\}$ serão

$$\xi_0 = \begin{bmatrix} x_C & y_C & \theta \end{bmatrix}^T \tag{1}$$

$$^{c}\mathbf{R}_{\mathbf{0}} = \begin{bmatrix} \cos\theta & \text{sen}\theta & 0 \\ -\text{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

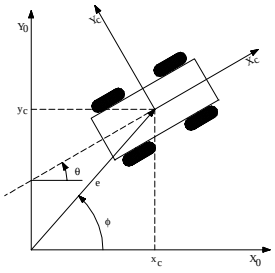


Figura 1 - Definição dos Sistemas de Coordenadas.

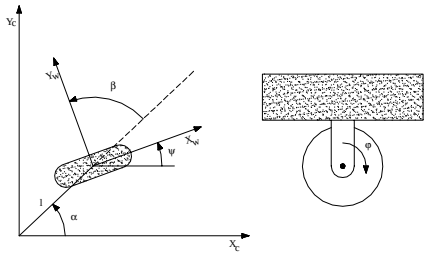


Figura 2 - Definição do Sistema de Coordenadas da Roda Convencional Fixa.

2.1 Descrição das Rodas

2.1.1 Rodas Convencionais Fixas

Da Fig.1 e das expressões (1) e (2), tem-se que a velocidade do robô, em relação ao sistema de coordenadas inercial, descrita no sistema de coordenadas $\{X_c, Y_c\}$ será

$$\dot{\xi}_C = {}^{\mathbf{C}}\mathbf{R}_{\mathbf{0}}\dot{\xi}_0 \qquad (3)$$

Esta mesma velocidade, descrita em um sistema de coordenadas $\{X_w, Y_w\}$ associando ao centro da roda, conforme a Fig. 2, tem a forma

$$\dot{\xi}_w = {}^{\mathbf{w}}\mathbf{R}_{\mathbf{c}}\dot{\xi}_c + \mathbf{l}_{\mathbf{w}} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^{\mathbf{w}}\mathbf{R}_{\mathbf{c}}\dot{\xi}_c \qquad (4)$$

onde $\mathbf{l_w}$ é o braço de giro e ${}^{\mathbf{w}}\mathbf{R_c}$ é a matriz de rotação entre os sistemas de coordenadas $\{X_w, Y_w\}$ e $\{X_c, Y_c\}$.

Utilizando-se (3), (4) pode ser reescrita como

$$\dot{\xi}_W = \begin{bmatrix} \text{sen}(\alpha+\beta) & -\cos(\alpha+\beta) & -l\cos\beta \\ \cos(\alpha+\beta) & \text{sen}(\alpha+\beta) & l\,\text{sen}\beta \\ 0 & 0 & 1 \end{bmatrix} {}^{\mathbf{C}}\mathbf{R}_{\mathbf{0}}\dot{\xi}_0$$

e considerando-se que $\dot{\xi}_W = \begin{bmatrix} r\dot{\varphi} & 0 & \dot{\theta} \end{bmatrix}^T$ chega-se à expressão

$$\begin{bmatrix} -\text{sen}(\alpha+\beta) & \cos(\alpha+\beta) & l\cos\beta \\ \cos(\alpha+\beta) & \text{sen}(\alpha+\beta) & l\,\text{sen}\beta \\ 0 & 0 & 1 \end{bmatrix} {}^{\mathbf{C}}\mathbf{R}_{\mathbf{0}}\dot{\xi}_0 + \qquad (5)$$

$$+\begin{bmatrix} r\dot{\varphi} & 0 & -\dot{\theta} \end{bmatrix}^T = 0$$

que representa as restrições ao movimento do robô devido à roda em questão.

2.1.2 Rodas Convencionais Orientáveis Centradas

Uma roda convencional orientável centrada é uma roda cujo plano pode ser rotacionado em torno de um eixo vertical passando pelo centro da roda. A descrição deste tipo de roda é a mesma utilizada para rodas fixas (caso anterior), porém o ângulo β não é constante, mas variável.

2.1.3 Rodas Convencionais Orientáveis Não-centradas

Neste tipo de roda, a rotação se dá em torno de um eixo vertical que não passa através do centro da roda, conforme a Fig. 3. Neste caso, a expressão (5) torna-se

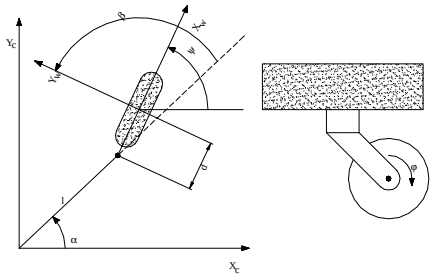


Figura 3 - Roda Convencional Orientável Não-centrada.

$$\begin{bmatrix} -\text{sen}(\alpha+\beta) & \cos(\alpha+\beta) & l\cos\beta \\ \cos(\alpha+\beta) & \text{sen}(\alpha+\beta) & d + l\,\text{sen}\beta \\ 0 & 0 & 1 \end{bmatrix} {}^{\mathbf{C}}\mathbf{R}_{\mathbf{0}}\dot{\xi}_0 + \qquad (6)$$

$$+\begin{bmatrix} r\dot{\varphi} & 0 & -\dot{\theta} \end{bmatrix}^T = 0$$

2.1.4 Rodas Suecas

Para este tipo de roda, é necessário mais um parâmetro para caracterizar a direção, com relação ao plano da roda, do componente nulo da velocidade do ponto de contato. A Fig. 4 mostra a definição dos sistemas de coordenadas utilizados.

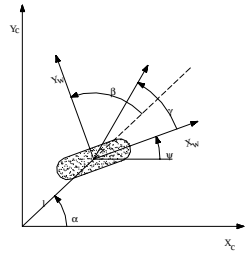


Figura 4 - Definição do Sistema de Coordenadas para Roda Sueca.

Com esta definição de sistemas de coordenadas, a matriz de rotação entre os sistemas de coordenadas $\{X_w, Y_w\}$ e $\{X_c, Y_c\}$ será

$$ {}^{\mathbf{w}}\mathbf{R}_{\mathbf{c}} = \begin{bmatrix} \operatorname{sen}(\alpha+\beta+\gamma) & -\cos(\alpha+\beta+\gamma) & 0 \\ \cos(\alpha+\beta+\gamma) & \operatorname{sen}(\alpha+\beta+\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7) $$

Por outro lado, tem-se que a velocidade do robô descrita no sistema de coordenadas $\{X_w, Y_w\}$ é dada por

$$ \dot{\xi}_w = \begin{bmatrix} r\dot{\varphi}\cos\gamma & \dot{\xi}_{wy} & \dot{\theta} \end{bmatrix}^T $$

Como se trata de uma roda sueca, o componente de velocidade normal é desconhecido, pois a velocidade do ponto de contato nesta direção não é nula. Com isto, a restrição de movimento da roda pode ser escrita como

$$ \begin{bmatrix} -\operatorname{sen}(\alpha+\beta+\gamma) \\ \cos(\alpha+\beta+\gamma) \\ l\cos(\beta+\gamma) \end{bmatrix}^T {}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}\dot{\xi}_0 + r\cos\gamma\dot{\varphi} = 0 \quad (8) $$

Assume-se que para uma roda sueca $\gamma \neq \frac{\pi}{2}$, pois neste caso a roda estaria sujeita a uma restrição idêntica a qual está sujeita uma roda convencional, perdendo o benefício de ser uma roda sueca.

2.2 Restrições à Mobilidade do Robô

Considere-se um robô que possua N rodas, sendo N_f rodas convencionais fixas, N_c rodas centradas, N_{oc} rodas não centradas e N_{sw} rodas suecas. Então, as equações que descrevem as restrições (6-8) podem ser escritas na forma matricial:

$$ \mathbf{J_1}(\beta_C, \beta_{OC}) {}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}\dot{\xi} + \mathbf{J_2}\dot{\varphi} = 0 \quad (9) $$

$$ \mathbf{C_1}(\beta_C, \beta_{OC}) {}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}\dot{\xi} + \mathbf{C_2}\dot{\beta}_{OC} = 0 \quad (10) $$

Considerando-se apenas as primeiras (N_f+N_c) restrições de (10) tem-se

$$ \mathbf{C_1^*}(\beta_C) {}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}\dot{\xi} = 0 \quad (11) $$

sendo $\mathbf{C_1^*}(\beta_C) = \begin{bmatrix} \mathbf{C_{1f}} & \mathbf{C_{1c}}(\beta_c) \end{bmatrix}^T$ e portanto, ${}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}\dot{\xi}$ pertence ao espaço nulo de $\mathbf{C_1^*}(\beta_C)$.

Define-se *grau de mobilidade* como

$$ \delta_m = \dim \mathcal{N}\left(\mathbf{C_1^*}(\beta_C)\right) $$

O *grau de dirigibilidade* é definido como o número de rodas convencionais orientáveis centradas que podem ser orientadas independentemente para dirigir o robô e é dado por

$$ \delta_s = \rho\left(\mathbf{C_{1c}}(\beta_C)\right) $$

Se o robô estiver equipado com mais de δ_s rodas convencionais orientadas centradas, o movimento de N_c-δ_s rodas deverá ser coordenado com as demais, de forma a garantir a existência do centro de rotação instantâneo.

Pode ser provado (Campion et al., 1996) que a configuração das rodas de um robô móvel deve ser tal que as seguintes restrições sejam satisfeitas:

$$ 1 \le \delta_m \le 3,\ 0 \le \delta_s \le 2 \text{ e } 2 \le \delta_m + \delta_s \le 3 \quad (12) $$

Verifica-se que apenas cinco classes de robôs com rodas, caracterizadas pelo par (δ_m, δ_s) satisfazem as restrições (12): as classes (3,0), (2,0), (2,1), (1,1) e (1,2).

2.3 Modelo no Espaço de Estados

De (11) tem-se que ${}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}\dot{\xi}$ pertence ao espaço nulo de $\mathbf{C_1^*}(\beta_C)$. Portanto, lembrando que ${}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}^{-1} = {}^{\mathbf{c}}\mathbf{R}_{\mathbf{0}}^{\mathbf{T}} = {}^{\mathbf{0}}\mathbf{R}_{\mathbf{c}}$, pode-se escrever

$$ \dot{\xi} = {}^{\mathbf{0}}\mathbf{R}_{\mathbf{c}}\Sigma(\beta_C)\eta \quad (13) $$

onde as colunas de $\Sigma(\beta_c)$ formam uma base do espaço nulo de $\mathbf{C_1^*}(\beta_C)$. Pode-se verificar facilmente que as dimensões de $\Sigma(\beta_c)$ e η serão sempre $3x\delta_m$ e $\delta_m x1$. Tem-se então o modelo no espaço de estados do sistema, com as coordenadas de postura ξ como variáveis de estado. Como entradas do sistema tem-se η e β_c. Se o robô não possui rodas orientáveis centradas (N_c=0), a matriz Σ não dependerá de β_c e as entradas (lineares, neste caso) do sistema serão apenas η.

Pode ser provado (Campion et al., 1996) que este modelo é irredutível, ou seja, não existe uma transformação de coordenadas tal que uma das coordenadas é identicamente nula.

3 Transformação de Coordenadas

Nesta seção o modelo descrito pela expressão (13) será convertido para coordenadas polares. Conseqüentemente, a posição do robô será representada através de coordenadas polares (e, ϕ). Observando-se a Fig. 1 é facilmente verificado que

$$ e = \sqrt{x_C^2 + y_C^2} \quad (14) $$

$$ \phi = \operatorname{atan2}(y_C, x_C) \quad (15) $$

$$ x_C = e\cos\phi \quad (16) $$

$$ y_C = e\operatorname{sen}\phi \quad (17) $$

derivando-se (14) e (15) e substituindo-se (16) e (17) obtém-se

$$ \begin{bmatrix} \dot{e} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\phi & \operatorname{sen}\phi \\ -\frac{\operatorname{sen}\phi}{e} & \frac{\cos\phi}{e} \end{bmatrix} \begin{bmatrix} \dot{x}_C \\ \dot{y}_C \end{bmatrix} \quad (18) $$

Definindo-se $\alpha=\theta-\phi$ e $z = \begin{bmatrix} e & \phi & \alpha \end{bmatrix}^T$ pode-se de (13) e (18) escrever o modelo em coordenadas polares

$$ \dot{z} = D(z, \beta_C)\eta \quad (19) $$

$$\text{com } D(z,\beta_c)=\begin{bmatrix} \cos\phi & \text{sen}\phi & 0 \\ -\dfrac{\text{sen}\phi}{e} & \dfrac{\cos\phi}{e} & 0 \\ \dfrac{\text{sen}\phi}{e} & -\dfrac{\cos\phi}{e} & 1 \end{bmatrix}\mathbf{0}\,\mathbf{R_c}\Sigma(\beta_c).$$

4 Leis de Controle

As leis de controle propostas neste artigo baseiam-se em uma análise do tipo de Lyapunov e Barbalat. Sem perda de generalidade, é assumido que deseja-se a convergência de e, ϕ e α para zero.

4.1 Classe (3,0)

Considere-se a seguinte candidata à função de Lyapunov:

$$V=\frac{1}{2}\lambda e^2+\frac{1}{2}(\alpha^2+h\phi^2) \qquad (20)$$

onde λ e h são constantes positivas.

Derivando-se (20) em relação ao tempo e substituindo-se $\dot{e}$, $\dot{\phi}$, e $\dot{\alpha}$ de (19) tem-se

$$\dot{V}=\dot{V}_1+\dot{V}_2$$

sendo

$$\dot{V}_1=\lambda e\cos\alpha\eta_1+h\phi\frac{\cos\alpha}{e}\eta_2$$

$$\dot{V}_2=\alpha\left(\eta_3-\left(\lambda e\frac{\text{sen}\alpha}{\alpha}+\frac{\cos\alpha}{e}\right)\eta_2-\frac{\text{sen}\alpha}{e\alpha}(\alpha-h\phi)\eta_1\right)$$

Este particionamento é feito de forma a deixar em $\dot{V}_1$ os termos que não são identicamente nulos quando α é nulo, ou seja, os termos que não são função explícita de α ou senα e em $\dot{V}_2$ os demais termos.

Pode-se verificar facilmente que $\dot{V}_1$ pode ser tornado não positivo fazendo-se

$$\eta_1=-\gamma_1 e\cos\alpha$$

$$\eta_2=-\gamma_2\phi e\cos\alpha$$

onde γ_1 e γ_2 são constantes positivas. Com isto tem-se que

$$\dot{V}_1=-\lambda\gamma_1 e^2\cos^2\alpha-h\phi^2\cos^2\alpha\le 0$$

$$\dot{V}_2=\alpha\left(\eta_3+\gamma_2\phi\cos\alpha\left(\lambda e^2\frac{\text{sen}\alpha}{\alpha}+\cos\alpha\right)+\right.$$

$$\left.+\gamma_1\cos\alpha\frac{\text{sen}\alpha}{\alpha}(\alpha-h\phi)\right)$$

Logo, fazendo-se

$$\eta_3=-\gamma_3\alpha-\gamma_2\phi\cos\alpha\left(\lambda e^2\frac{\text{sen}\alpha}{\alpha}+\cos\alpha\right)$$

$$-\gamma_1\cos\alpha\frac{\text{sen}\alpha}{\alpha}(\alpha-h\phi)$$

sendo γ_3 uma constante positiva, tem-se

$$\dot{V}=-\lambda\gamma_1 e^2\cos^2\alpha-h\gamma_2\phi^2\cos^2\alpha-\gamma_3\alpha^2\le 0 \quad (21)$$

que juntamente com o fato de que V é contínua e não negativa, garante a estabilidade do sistema em malha fechada. Por outro lado, considerando-se que V é inferiormente limitada e $\dot{V}$ é uniformemente contínua, tem-se pelo lema de Barbalat que $\dot{V}$ tende a zero e portanto, pode-se verificar pela expressão (21) que e, ϕ e α também tendem a zero.

4.2 Classe (2,0)

Agindo-se de forma semelhante ao caso anterior, a candidata a função de Lyapunov é parcionada em

$$\dot{V}_1=\lambda e\cos\alpha\eta_1$$

$$\dot{V}_2=\alpha\left(\eta_2-\frac{\text{sen}\alpha}{e\alpha}(\alpha-h\phi)\eta_1\right)$$

de onde obtém-se

$$\eta_1=-\gamma_1 e\cos\alpha$$

$$\eta_2=-\gamma_2\alpha-\gamma_1\cos\alpha\frac{\text{sen}\alpha}{\alpha}(\alpha-h\phi)$$

resultando

$$\dot{V}=-\lambda\gamma_1 e^2\cos^2\alpha-\gamma_2\alpha^2\le 0$$

Portanto, através dos mesmos argumentos utilizados no caso anterior, pode-se concluir que e e α tendem a zero. A convergência de ϕ para zero pode ser provada observando-se as equações do sistema em malha fechada:

$$\begin{cases} \dot{e}=-\gamma_1 e\cos^2\alpha \\ \dot{\phi}=-\gamma_1\text{sen}\alpha\cos\alpha \\ \dot{\alpha}=-\gamma_2\alpha+\gamma_1 h\phi\cos\alpha\dfrac{\text{sen}\alpha}{\alpha} \end{cases} \qquad (22)$$

Considerando-se a convergência para zero de e e α, conclui-se facilmente de (22) que $\dot{\phi}$ também tende a zero. A convergência de $\dot{\phi}$ para zero, por sua vez, faz com que $\dot{\alpha}$ convirja para algum valor constante dado por $\gamma_1 h\phi^*$. Por outro lado, a continuidade uniforme de $\dot{\alpha}$ juntamente com a convergência para zero de α, garante, pelo lema de Barbalat, que $\dot{\alpha}$ tende a zero. Portanto, ϕ^* tem, necessariamente, que ser zero.

4.3 Classe (2,1)

Neste caso, toma-se como candidata à função de Lyapunov

$$V=\frac{1}{2}\lambda e^2+\frac{1}{2}(\alpha^2+h\phi^2+g\beta_c^2)$$

onde λ, h e g são constantes positivas.

Diferentemente dos casos anteriores, agora $\dot{V}$ é parcionada em três. $\dot{V}_2$ contém os termos que são nulos quando α é nulo e $\dot{V}_3$ contém os termos que são nulos quando β_c é nulo e $\dot{V}_1$ contém os demais termos. Ou seja,

$$\dot{V}_1 = \lambda e \cos(\alpha+\beta_c)\eta_1$$

$$\dot{V}_2 = \alpha\left(\eta_2 - \cos\beta_c \frac{\operatorname{sen}\alpha}{e\alpha}\eta_1(\alpha - h\phi)\right)$$

$$\dot{V}_3 = g\beta_c\left(\zeta - \cos\alpha\frac{\operatorname{sen}\beta_c}{e\beta_c}\eta_1(\alpha - h\phi)\right)$$

e portanto, seguindo o mesmo procedimento utilizado nas classes anteriores tem-se

$$\eta_1 = -\gamma_1 e\cos(\alpha+\beta_c)$$

$$\eta_2 = -\gamma_2\alpha - \gamma_1\cos\beta_c\frac{\operatorname{sen}\alpha}{\alpha}\cos(\alpha+\beta_c)(\alpha - h\phi)$$

$$\eta_3 = -\gamma_3\beta_c - \gamma_1\cos\alpha\frac{\operatorname{sen}\beta_c}{\beta_c}\cos(\alpha+\beta_c)(\alpha - h\phi)$$

conseqüentemente,

$$\dot{V} = -\lambda\gamma_1 e^2\cos^2(\alpha+\beta_c) - \gamma_2\alpha^2 - g\gamma_3\beta_c^2 \le 0$$

de onde pode-se concluir que e, α e β_c convergem para zero. A convergência de ϕ para zero é provada considerando-se as equações dinâmicas do sistema, como na classe (2,0).

4.4 Classe (1,1)

Definindo-se

$$u_1 = \eta\operatorname{sen}\beta_c \quad \text{e} \quad u_2 = \eta\cos\beta_c$$

o modelo (19) assume a forma

$$\begin{bmatrix}\dot{e}\\ \dot{\phi}\\ \dot{\alpha}\end{bmatrix} = \begin{bmatrix} d\cos\alpha & 0\\ \dfrac{d\operatorname{sen}\alpha}{e} & 0\\ -\dfrac{d\operatorname{sen}\alpha}{e} & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

que exceto pela constante d, é a mesma forma do modelo para a classe (2,0). Logo, obtém-se

$$u_1 = -\gamma_1 de\cos\alpha$$

$$u_2 = -\gamma_2\alpha - \gamma_1 d^2\cos\alpha\frac{\operatorname{sen}\alpha}{\alpha}(\alpha - h\phi)$$

Obviamente, as entradas η e β_c podem ser obtidas de u_1 e u_2 através das expressões

$$\eta = \sqrt{u_1^2 + u_2^2} \quad \text{e} \quad \beta_c = \operatorname{atan2}(u_1, u_2)$$

4.5 Classe (1,2)

Com procedimento similar ao adotado para a classe (1,1), para esta classe também serão definidas novas variáveis de entrada:

$$u_1 = \eta\operatorname{sen}(\beta_{c1}+\beta_{c2}) \qquad (23)$$

$$u_2 = 2\eta\operatorname{sen}\beta_{c1}\operatorname{sen}\beta_{c2} \qquad (24)$$

$$u_3 = \eta\operatorname{sen}(\beta_{c1}-\beta_{c2}) \qquad (25)$$

portanto, o modelo do sistema pode ser escrito na forma

$$\begin{bmatrix}\dot{e}\\ \dot{\phi}\\ \dot{\alpha}\end{bmatrix} = \begin{bmatrix} L\cos\alpha & -L\operatorname{sen}\alpha & 0\\ L\dfrac{\operatorname{sen}\alpha}{e} & L\dfrac{\cos\alpha}{e} & 0\\ -L\dfrac{\operatorname{sen}\alpha}{e} & -L\dfrac{\cos\alpha}{e} & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix}$$

que é bastante semelhante ao modelo para a classe (3,0). Logo, através de análise semelhante obtém-se

$$u_1 = -\gamma_1 Le\cos\alpha$$

$$u_2 = -\gamma_2 Le\phi\cos\alpha$$

$$u_3 = -\gamma_3\alpha - \gamma_2 L^2\phi\cos\alpha\left(\lambda e^2\frac{\operatorname{sen}\alpha}{\alpha} + \cos\alpha\right)$$

$$-\gamma_1 L^2\cos\alpha\frac{\operatorname{sen}\alpha}{\alpha}(\alpha - h\phi)$$

Para obter-se η e β_{c1} e β_{c2}, divide-se (24) pela diferença e pela soma de (23) e (25) obtendo-se

$$\beta_{c1} = \operatorname{atan2}(u_2, u_1 - u_3) \qquad (26)$$

$$\beta_{c2} = \operatorname{atan2}(u_2, u_1 + u_3) \qquad (27)$$

Elevando ao quadrado (23)-(25) e somando as três expressões pode-se obter

$$\eta = \pm\sqrt{\frac{u_1^2 + u_2^2 + u_3^2}{2\left(\operatorname{sen}^2\beta_{c1} + \operatorname{sen}^2\beta_{c2}\right)}}$$

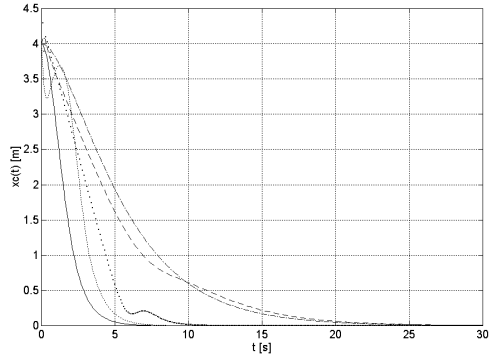
que pode ser calculado após β_{c1} e β_{c2} serem obtidos através de (26)-(27). Convém ressaltar que esta expressão não é válida quando $\beta_{c1}=n_1\pi$ e simultaneamente $\beta_{c2}=n_2\pi$, n_1, n_2=0, 1, 2….No entanto, para estas condições, o próprio modelo (13) não é válido, pois $\delta_m = \dim \aleph\left(\mathbf{C_1^*}(\beta_c)\right)$ será igual a 2 e não mais igual a 1 como é necessário para que o robô pertença a classe (1,2). Ou seja, nesta situação, o robô degenera para um robô pertencente à classe (2,0). O sinal correto de η pode ser determinado de (23) e (25) observando-se que $\operatorname{sen}(\beta_{c1}+\beta_{c2})$ e $\operatorname{sen}(\beta_{c1}-\beta_{c2})$ serão simultaneamente nulos apenas se $\beta_{c1}=\beta_{c2}=0$. Além disso, tem-se que sgn(a/b)=sgn(ab). Portanto, pode-se escrever

$$\operatorname{sgn}(\eta) = \begin{cases} \operatorname{sgn}(u_1\operatorname{sen}(\beta_{c1}+\beta_{c2})), \\ \quad \left|u_1\operatorname{sen}(\beta_{c1}+\beta_{c2})\right| > \left|u_3\operatorname{sen}(\beta_{c1}-\beta_{c2})\right| \\ \operatorname{sgn}(u_3\operatorname{sen}(\beta_{c1}-\beta_{c2})), \quad \text{caso contrário}\end{cases}$$

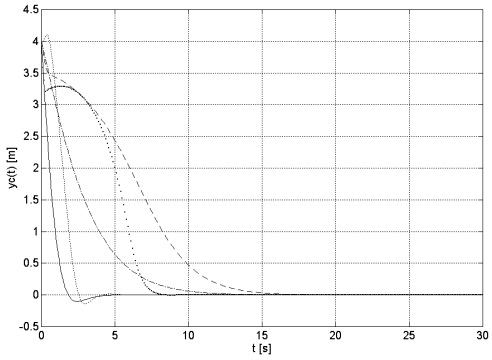
5 Resultados de Simulação

As leis de controle obtidas na seção anterior foram simuladas para um robô típico de cada classe. As condições iniciais foram $x_c(0)$=4m, $y_c(0)$=4m e

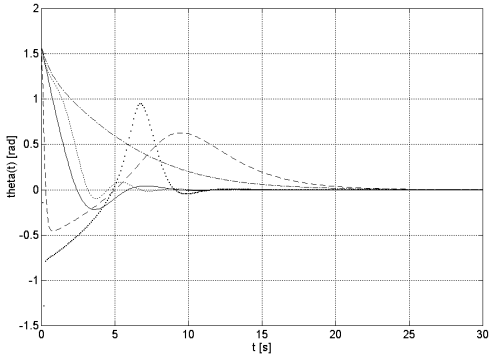
$\theta(0)=\pi/2$ e deseja-se estabilizar o robô na origem. A convergência das variáveis de posição e orientação pode ser verificada na Fig. 5.



(a)



(b)



(c)

Figura 5 - Resultados de Simulação para Robô Classe (3,0) (pontos grandes), Classe (2,0) (contínuo), Classe (2,1) (pontos pequenos), Classe (1,1) (traço ponto) e Classe (1,2) (tracejado). a) Resposta de $x_c(t)$, b) Resposta de $y_c(t)$, c) Resposta de $\theta(t)$.

6 Conclusão

A técnica apresentada neste artigo permite que se obtenha leis de controle que garantam a convergência para um ponto e não apresentem oscilações excessivas. Além disso, as leis de controle obtidas são contínuas e suaves. No entanto, não se pode obter conclusões acerca de qual classe de robô apresenta melhor desempenho, pois a trajetória obtida pode ser bastante diferente, em função dos parâmetros de projeto utilizados no cálculo da lei de controle.

Como considera-se apenas o modelo cinemático dos robôs, as variáveis de entrada são homogêneas a velocidades ou posição. Portanto, a utilização prática das leis de controle apresentadas aqui depende da existência de malhas de controle internas que garantam que as velocidades ou posições comandadas sejam efetivamente aplicadas nas rodas. Isto pode ser feito, por exemplo, utilizando-se controladores PID convencionais.

Referências Bibliográficas

Aicardi, M., Casalino, G., Bicchi, A., Balestrino, A. (1995). Loop Steering of Unicycle-like Vehicles via Lyapunov Techniques, *IEEE Robotics & Automation Magazine*, 2(1):27-35.

Bloch, A. M., Reyhanoglu, M. & McClamroch, N. H. (1992). Control and Stabilization of Nonholonomic Dynamic Systems, *IEEE Trans. on Automatic Control*, 37(11):1746-1756.

Brockett, R. W. (1983). Asymptotic Stability and Feedback Stabilization in *Differential Geometric Control Theory*. Brockett, Millman & Sussman, Eds, Boston, MA: Birkhauser, pp. 181-208.

Campion, G., Bastin, G., D'Andréa-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.

Canudas de Wit, C. & Sørdalen, O. J. (1992). Exponential Stabilization of Mobile Robots with Nonholonomic Constraints, *IEEE Trans. on Automatic Control*, 37(11):1791-1797.

McClamroch, N. H. & Kolmanovsky, I. (1995). Developments in Nonholonomic Control Problems, *IEEE Control Systems Magazine*, 15(6):20-36.

McCloskey, R. T. & Murray, R. M. (1997). Exponential Stabilization of Driftless NonLinear Control Systems Using Homogeneous Feedback, *IEEE Trans. on Automatic Control*, 42(5):614-628.

Oelen, W., Berghuis, H., Nijmeijer, H. & Canudas de Wit, C. (1995). Hybrid Stabilizing Control on a Real Mobile Robot, *IEEE Robotics & Automation Magazine*, 2(2):16-23.

Thuilot, B., Andréa-Novel, B. & Micaelli, A. (1996). Modeling and Feedback Control of Mobile Robots Equipped with Several Steering Wheels, *IEEE Trans. on Robotics and Automation*, 12(3):375-390.

Walsh, G., Tilbury, D., Sastry, S., Murray, R. & Laumond, J. P. (1994). Stabilization of Trajectories for Systems with Nonholonomic Constraints, *IEEE Trans. on automatic Control*, 39(1):216-222.

Proceedings of XII Brazilian Automatic Control Conference - XII CBA
Vol. II, pp. 611-616 - September 14 - 18, 1998 - Uberlândia, MG, Brazil

CONTROL PREDICTIVO PARA SEGUIMIENTO DE CAMINOS EN UN ROBOT DEL TIPO "SYNCHRO-DRIVE"

Julio E. Normey-Rico*, J. Gómez Ortega and I. Alcalá****

**Dpto de Automação e Sistemas. Univ. Fed. de Santa Catarina, Cx. P. 476, CEP 88040-900, Florianopolis-SC, Brasil. E-mail: julio@cartuja.us.es*
E-mail:julio@cartuja.us.es
***Dpto. Ingeniería de Sistemas y Automática, Univ. de Sevilla, Escuela Superior de Ingenieros, Camino de los descubrimientos s/n, 41092 Sevilla, Spain.*
Fax: +34 95 4487340, E-mail: juango@esi.us.es

Resumen– Este artículo muestra la aplicación de una estrategia de control predictivo basado en un predictor de Smith (SPGPC) al problema del seguimiento de caminos de un robot móvil del tipo *synchro drive*. La simplicidad del algoritmo, en comparación con un GPC normal basado en un predictor óptimo, así como sus buenas propiedades de robustez, permiten la utilización de un modelo simple del robot móvil manteniendo el mismo comportamiento nominal que el GPC normal. Por lo tanto, la ley de control final es más fácil y rápida de calcular. Se presentan resultados experimentales realizados sobre un robot móvil Nomad 200 que muestran el comportamiento adecuado y la validez de la estrategia propuesta.

Abstract. This paper presents the application of a Smith predictor based generalised predictive controller (SPGPC) to the path-traking problem of a synchro drive mobile robot. The higher robustness of the SPGPC algorithm, if compared to a generalised predictive controller (GPC) based on an optimal predictor, allows the use of a simple model for the mobile robot. Thus, the final control law is simpler and less time consuming than the GPC, mantaining the same nominal performance and better robustness. Experimental tests carried out on a Nomad 200 mobile robot validate the performance of the proposed strategy.

Keywords. Mobile Robots; Path Tracking; Predictive Control; Robustness.

1. INTRODUCCIÓN

Uno de los aspectos más importantes en el campo de la robótica móvil es el problema del seguimiento de caminos (SC). El objetivo del SC consiste en guiar al robot lo más cerca posible de un camino de referencia, previamente definido, y con el menor esfuerzo de control. Esto último es importante ya que permite disminuir el consumo de energía, lo que aumenta la autonomía del robot. En este trabajo se consideran caminos de referencia explícitos, definidos geométricamente mediante un conjunto de puntos consecutivos o de funciones analíticas (rectas, arcos de circunferencia, etc.) que describen segmentos geométricos unidos entre sí.

El estudio del problema del seguimiento de caminos se justifica tanto en entornos bien conocidos (entornos estructurados), en los que el conocimiento preciso de los obstáculos permite una planificacin previa, como en entornos parcialmente estructurados, en los que además pueden encontrarse obstáculos no previstos en la fase de planificación del camino de referencia. En este caso, se encuentran en la literatura numerosas estrategias de navegación que utilizan un módulo de seguimiento de caminos y que recalculan el camino de referencia en tiempo real cuando se detecta, mediante algún sistema sensorial del entorno, un obstáculo no previsto.

Las estrategias de SC se basan en la estimación del error entre la posición actual del vehículo y el camino de referencia que debe ser seguido. Con el fin de calcular dicho error se debe escoger en cada instante de muestreo un punto de referencia sobre el camino. Hay varias posibilidades para esta elección. La estrategia propuesta por Amidi (Amidi, 1990) consiste en escoger dicho punto a una distancia determinada de la posición ocupada por el robot en cada instante. A dicha distancia se le denomina *look ahead* y es un parámetro

---

* J.E.Normey-Rico es doctorando del Dpto. Ingeniería de Sistemas y Automática, Univ. de Sevilla

* Trabajo financiado en parte por CICYT Contratos TAP95-370 y TAP96-884 y CAPES Contrato BEXO448/95-6

camino de referencia a medida que el robot se va moviendo. Aunque en este trabajo se ha considerado un *look ahead* fijo, también podría considerarse variable en función de ciertos parámetros dinámicos del problema como por ejemplo la velocidad del robot o la curvatura del camino. Por otro lado, algunas técnicas de control óptimo utilizan una estrategia diferente que consiste en utilizar en cada instante de muestreo un conjunto de N puntos consecutivos del camino como referencia en lugar de un sólo punto (Gómez Ortega, 1994; Gómez Ortega and Camacho, 1996).

En cuanto a la estrategia de control, también se pueden encontrar en la literatura diferentes técnicas utilizadas en el problema del SC. En (Nelson and Cox, 1990) se propone un control proporcional a un conjunto de errores diferentes todos ellos relacionados con el seguimiento del camino de referencia. Los diferentes errores se ponderan con diferentes constantes. En (Kanayama *et al.*, 1990) se propone un PID clásico como controlador. También pueden utilizarse aproximaciones geométricas al problema. En este caso, la estrategia de control consiste en el cálculo de una curva analítica (por ejemplo un *spline* (Nelson, 1989) o un polinomio de quinto orden (Shin and Singh, 1990)) que cumpla con ciertas restricciones como por ejemplo curvatura continua, puntos inicial y final determinados, etc. La estructura de la curva es fija y los parámetros deben ser calculados en cada instante de muestreo. En (Amidi, 1990), se propone la estrategia denominada *persecución pura* en la que la curvatura que debe seguir el robot en cada instante es la de una circunferencia que pasa por la posición actual del robot y por un punto situado sobre el camino de referencia a una distancia determinada.

También se han utilizado técnicas de control óptimo. En (Ollero and Amidi, 1991) se propone la técnica del control predictivo generalizado como estrategia de control, utilizándose un modelo lineal del robot. En (Gomez Ortega and Camacho, 1996) también se propone una técnica de control predictivo pero en este caso con un modelo no lineal del robot y utilizando una red neuronal como forma de implementar el controlador en tiempo real.

El problema que se estudia en este artículo es el del seguimiento de un camino de referencia previamente calculado y definido mediante una serie de puntos consecutivos en el plano. Se ha utilizado un robot con un sistema de locomoción con tres ruedas motrices sincronizadas. Como estrategia de control se propone un algoritmo de control predictivo generalizado utilizando un predictor de Smith en lugar del predictor óptimo que se usa en el GPC clásico. Esta estrategia presenta ventajas en cuanto a que , manteniendo un comportamiento nominal igual al del GPC normal, su robustez frente a incertidumbres es mayor.

El artículo se organiza de la siguiente manera. En la sección 2 se describe brevemente la estrategia de control predictivo propuesta. En la sección 3 se analiza el modelo utilizado del robot así como la estructura del controlador para el robot Nomad 200 utilizado en los experimentos. En la sección 4 se realiza un estudio relativo a las mejoras obtenidas con esta estrategia de control en cuanto al tiempo tados experimentales obtenidos, finalizando en la sección 6 con las conclusiones.

2. CONTROL PREDICTIVO GENERALIZADO BASADO EN EL PREDICTOR DE SMITH

El SPGPC fue propuesto en (Normey-Rico *et al.*, 1998*a*) y se ha aplicado al seguimiento de caminos en robots móviles en (Normey-Rico *et al.*, 1998*b*). De la misma forma que en control predictivo generalizado (GPC) (D.W. Clarke and Tuffs, 1987), el objetivo del algoritmo es calcular una secuencia de controles futuros $(U(t), U(t+1|t), ..., U(t+N_u-1|t))$ que minimicen una función de coste J de la forma:

$$J(N_1, N_2, N_u) = \sum_{i=N_1}^{N_2} \mu(i)[\hat{Y}(t+i|t) - Y_d(t+i)]^2 + \sum_{i=1}^{N_u} \lambda(i)[\Delta U(t+i-1|t)]^2$$

donde $\mu(i)$ y $\lambda(i)$ son respectivamente las ponderaciones del error y del control; $\hat{Y}(t+i|t)$ son las predicciones de la salida; $Y_d(t+i)$ son las referencias deseadas; N_u es el horizonte de control y $N = N_2 - N_1$ es el horizonte de predicción.

Una vez obtenida la secuencia de control se usa una metodología de horizonte deslizante, que consiste en aplicar únicamente el control $U(t)$ y repetir el proceso en cada período de muestreo. En la figura 1 se muestra un diagrama de bloques del esquema de control.

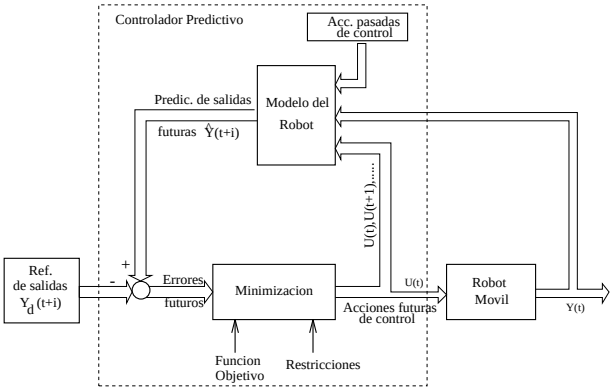


Fig. 1. Control predictivo de robots móviles

Las diferencias entre el control SPGPC propuesto y el GPC tradicional son: (i) en el GPC se utiliza el mismo modelo para generar las predicciones y calcular la ley de control (el modelo usado es un modelo CARIMA), mientras que en el SPGPC se utilizan dos modelos distintos: un modelo incremental para el cálculo de la ley de control y un modelo de bucle abierto para el cálculo de las predicciones; (ii) en el SPGPC las predicciones de bucle abierto en $t+d-i$ son corregidas sumándoles el error entre la salida real y la predecida en $t-i$, donde d es el retardo de la planta:

Cuando el modelo de la planta es lineal y no se consideran restricciones en el control, la ley de control que resulta es lineal. Se puede mostrar (Camacho and Bordons, 1995) que el incremento de la acción de control resultante es proporcional al error entre la respueta libre del sistema f y la referencia w, donde la matriz de ganancias K es función de los parámetros del control y del modelo de la planta:

$$U(t) = U(t-1) + K(f - w)$$

Como el GPC y el SPGPC usan el mismo procedimiento para calcular la ley de control, la matriz K es la misma en ambos controladores y solamente la respuesta libre será calculada de forma distinta. A partir de este análisis es fácil probar (Normey-Rico and Camacho, 1996), (Normey-Rico *et al.*, 1998*b*) que el diagrama de bloques equivalente para ambos algoritmos es el que se muestra en la figura 2.

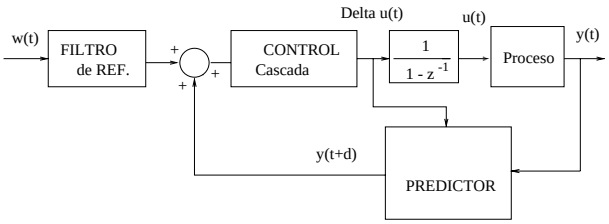


Fig. 2. Esquema de control del GPC y SPGPC

En este diagrama se distinguen claramente las dos etapas del cálculo del control. Primero se calcula la predicción (con un predictor óptimo en el GPC (D.W. Clarke and Tuffs, 1987) y con un predictor de Smith (Smith, 1958) para el SPGPC) y luego se calcula el control con los coeficientes determinados por la optimización de J. Usando este análisis (Normey-Rico *et al.*, 1998*b*) probaron que el SPGPC tiene el mismo comportamiento nominal y mejor robustez que el GPC. En ese trabajo los autores muestran que la máxima incertidumbre admisible para mantener la estabilidad es mayor en el SPGPC, principalmente en altas frecuencias. Este resultado es importante en la practica pues los modelos generalmente usados en control son de bajo orden y por lo tanto siempre existen errores de modelado en alta frecuencia (dinámicas no modeladas y retardos mal estimados). Por otro lado el comportamiento nominal del GPC y del SPGPC es el mismo pues cuando el modelo y el proceso son iguales el predictor no tiene efecto sobre el bucle cerrado. Por estas cualidades el SPGPC permite usar modelos mas simples para el predictor del comportamiento del robot móvil. Esto lleva a leyes de control de fácil implementación y menor tiempo de cómputo, como se analizará en los próximos apartados.

3. MODELO CINEMÁTICO DEL ROBOT MÓVIL NOMAD

El SPGPC propuesto se ha implementado en un robot Nomad 200 fabricado por la empresa Nomadic Technologies (ver figura 3). El sistema de en tres ruedas directrices cuya orientación varía de forma simultánea. Como sistemas sensoriales del entorno el robot incluye tres sistemas de medidas de distancias uno ultrasónico, otro infrarrojo y un tercero basado en un sistema híbrido láser/cámara de vídeo que utiliza una técnica de luz estructurada para el cálculo de las distancias. Estos sistemas no son utilizados en este trabajo.

Fig. 3. The Nomad 200 mobile robot

El control del robot se basa en un sistema multiprocesador siendo un pentium a 130 Mhz el procesador principal que incluye un enlace de radio Ethernet con un ancho de banda de 1.6Mbits/seg. El robot incluye igualmente un sistema para la estimación de la posición basado en la odometría para lo que dispone de codificadores de posición para el cálculo del número de vueltas girado por cada rueda así como de un girocompás para la estimación de la orientación del robot. Esta técnica para la estimación de la posición está sujeta a errores acumulativos por lo que para caminos de referencia largos el sistema debería ser actualizado, con una frecuencia determinada, a partir de la información obtenida de los sensores del entorno. Sin embargo este problema está desacoplado con el problema del seguimiento de trayectorias y no ha sido considerado en este trabajo. Para evitar errores de posición elevados se han utilizado caminos de referencia que no superan los 15 metros de longitud.

El modelo cinemático completo del robot viene dado por las siguientes ecuaciones no lineales, en las que x_g, y_g son las coordenadas del centro del robot referidas a un sistema de referencia global, δ es el ángulo de orientación de las ruedas, V es la velocidad lineal del robot y T es el tiempo de muestreo:

Si $\dot{\delta}(t) \neq 0$, entonces:

$$\delta(t+1) = \delta(t) + \dot{\delta}(t)T$$

$$x_g(t+1) = x_g(t) + \frac{V}{\dot{\delta}(t)}(\sin(\delta(t)+\dot{\delta}(t)T) - \sin(\delta(t)))$$

$$y_g(t+1) = y_g(t) - \frac{V}{\dot{\delta}(t)}(\cos(\delta(t)+\dot{\delta}(t)T) - \cos(\delta(t)))$$

Si $\dot{\delta}(t) = 0$, entonces:

$$\delta(t+1) = \delta(t)$$

$$x_g(t+1) = x_g(t) + VT\ cos\ \delta(t)$$

$$y_g(t+1) = y_g(t) + VT\ sen\ \delta(t)$$

Para $\dot{\delta}(t)$ se ha considerado que sigue al valor de su referencia $\dot{\delta}_r(t)$ con una dinámica lineal determinada por la siguiente expresión:

$$\dot{\delta}(t) = \frac{B}{D}\dot{\delta}_r(t)$$

en la que B y D son polinomios en z^{-1}. El comportamiento real de $\dot{\delta}$ cuando se aplica un escalón en la referencia se muestra en la figura 4

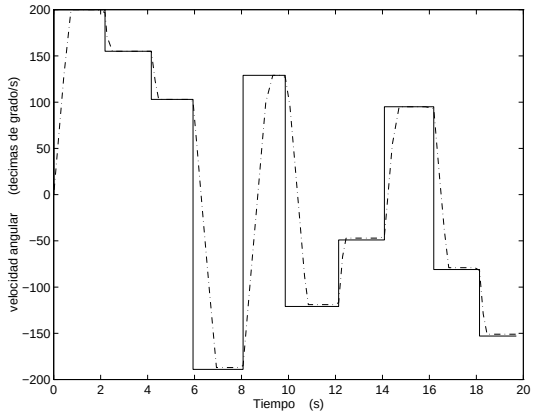


Fig. 4. Comportamiento de $\dot{\delta}$ (línea discontínua) ante escalones en la referencia $\dot{\delta}_r$ (línea contínua)

Sin embargo, el SPGPC requiere un modelo lineal de la cinemática del robot que será utilizado para las predicciones de las posiciones y orientaciones futuras del robot. El modelo lineal utilizado se ha obtenido utilizando la hipótesis de que los incrementos de la orientación de las ruedas entre cada dos instantes de muestreo, $\Delta\delta$, son pequeños. Para ello se ha utilizado un sistema de referencia local ligado al robot (ver figura 5)

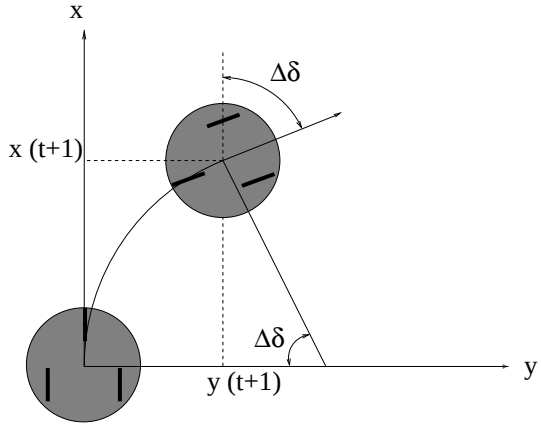


Fig. 5. Sistema de referencia local del robot móvil

El modelo lineal obtenido es el siguiente:

$$\begin{bmatrix} \delta(t) \\ y(t) \end{bmatrix} = \frac{1}{(1-z^{-1})} \begin{bmatrix} T \\ \frac{VT^2}{2} \end{bmatrix} \dot{\delta}_r(t-1-d)$$

en donde $\delta(t)$ es la orientación de las ruedas del robot, $y(t)$ es la posición del robot en el sistema de referencia local (ver figura 5), V es la velocidad lineal del robot, T es el tiempo de muestreo y tardo surge por el hecho de haber simplificado el modelo dinámico para $\dot{\delta}$ en función de $\dot{\delta}_r$ utilizando un escalón más un retardo d en lugar de una dinámica de primer o segundo orden determinada por los polinomios B y D. Este hecho simplifica los cálculos del controlador el cual, por presentar buenas características de robustez permite incluir esta simplificaciones.

Este modelo linealizado sólo es válido para valores pequeños de $\Delta\delta$ lo que implica que cuando el robot no se encuentre en una posición situada sobre el camino de referencia deseado será necesario utilizar caminos de referencia de aproximación. En este trabajo, los caminos de aproximacin se han calculado utilizando la estrategia del *persecución pura*. Es decir, los N puntos siguientes del camino de referencia son calculados en cada instante de muestreo como si el robot estuviese controlado mediante un controlador proporcional calculado según la estrategia citada.

La variable de control utilizada ha sido la velocidad de giro de las ruedas $\dot{\delta}$ que está relacionada con la curvatura γ del robot mediante la siguiente ecuación:

$$\dot{\delta} = \gamma V$$

Sin embargo, aunque el sistema debería ser del tipo *una entrada/una salida* (observar que $y(t) = (VT/2)\delta(t)$), se ha comprobado que el comportamiento del sistema es mucho mejor cuando se consideran dos salidas, $\delta(t)$ e $y(t)$. Esto se debe al hecho de que los caminos de aproximación, y por lo tanto, los caminos de referencia en cada instante, dependen fuertemente de la orientación del robot. Así, un error en $\delta(t)$ daría lugar a caminos de aproximación que no se adaptarían bien al camino de referencia. Por otro lado, si $y(t)$ no fuese penalizado, el robot no sería capaz de aproximarse al camino de referencia deseado.

4. LA ESTRUCTURA DEL SPGPC PARA EL ROBOT NOMAD

En este trabajo los parametros del SPGPC se elijen como: $N_u = N$, $N_1 = d+1$ y $N_2 = d+1+N$. Para calcular los controles futuros $U(t)$, $U(t+1)$,...es necesario calcular las predicciones $\hat{Y}(t+j \mid t)$. Para calcular los coeficientes de la ley de control se usa un modelo incremental para el cálculo de $\hat{Y}(t+d+j \mid t)$:

$$\hat{\delta}(t+j \mid t) = 2\hat{\delta}(t+j-1 \mid t) - \\ -\hat{\delta}(t+j-2 \mid t) + T \triangle \dot{\delta}(t+j-d-1) \quad (1)$$

$$\hat{y}(t+j \mid t) = 2\hat{y}(t+j-1 \mid t) - \hat{y}(t+j-2 \mid t) + \\ + \frac{V(T)^2}{2} \triangle \dot{\delta}(t+j-d-1) \quad (2)$$

o en forma vectorial:

$$\begin{bmatrix} \hat{\delta}(t+d+1 \mid t) \\ \hat{\delta}(t+d+2 \mid t) \\ \vdots \\ \hat{y}(t+d+N \mid t) \end{bmatrix} = G \begin{bmatrix} \triangle\dot{\delta}(t+1) \\ \vdots \\ \triangle\dot{\delta}(t+N-1) \end{bmatrix} + S \begin{bmatrix} \hat{\delta}(t+d \mid t) \\ \hat{\delta}(t+d-1 \mid t) \\ \hat{y}(t+d \mid t) \\ \hat{y}(t+d-1 \mid t) \end{bmatrix} \quad (2)$$

donde G y S son matrices constantes. Usando esta relación en la expresión de J la minimización de la función de coste lleva a:

$$M \begin{bmatrix} \triangle\dot{\delta}(t) \\ \triangle\dot{\delta}(t+1) \\ \vdots \\ \triangle\dot{\delta}(t+N-1) \end{bmatrix} = P_1 \begin{bmatrix} \hat{\delta}(t+d \mid t) \\ \hat{\delta}(t+d-1 \mid t) \\ \hat{y}(t+d \mid t) \\ \hat{y}(t+d-1 \mid t) \end{bmatrix} + P_2 \begin{bmatrix} w(t+d+1) \\ w(t+d+2) \\ \vdots \\ w(t+d+2N) \end{bmatrix} \quad (3)$$

donde $w(t)$ representa la referencia futura en la posición (y_{ref}) y en el ángulo de guiado (δ_{ref}); M, P_1 y P_2 son matrices constantes.

Finalmente, la ley de control resulta::

$$\triangle\dot{\delta}(t) = l_{11}\hat{\delta}(t+d \mid t) + l_{12}\hat{\delta}(t+d-1 \mid t) + l_{21}\hat{y}(t+d \mid t) + l_{22}\hat{y}(t+d-1 \mid t) + \sum_{i=1}^{2N} f_i w(t+d+i) \quad (4)$$

donde los coeficientes l_{ij} y f_i son funciones de V, T, $\mu(i)$ y $\lambda(i)$ y las predicciones se obtienen del modelo de bucle abierto:

$$\hat{\delta}(t+j \mid t) = \hat{\delta}(t+j-1 \mid t) + T\dot{\delta}(t+j-d-1)$$

$$\hat{y}(t+j \mid t) = \hat{y}(t+j-1 \mid t) + V(T)^2/2\dot{\delta}(t+j-d-1)$$

con las correcciones dadas por ($i = 0, 1$):

$$\hat{\delta}(t+d-i \mid t) \leftarrow \hat{\delta}(t+d-i \mid t) + \delta(t-i) - \hat{\delta}(t-i \mid t)$$

$$\hat{y}(t+d-i \mid t) \leftarrow \hat{y}(t+d-i \mid t) + y(t-i) - \hat{y}(t-i \mid t)$$

El diagrama de bloques de ésta estructura se muestra en la figura 4.

Por lo tanto, el SPGPC permite el uso de un modelo muy simple para el robot manteniendo al mismo tiempo buenas prestaciones y robustez. En (Normey-Rico *et al.*, 1998*c*) se muestra como el uso de este modelo simplificado y del predictor de Smith (que es más simple que el óptimo) permiten reducir considerablemente la carga de cálculo del algoritmo, si se compara con un GPC.

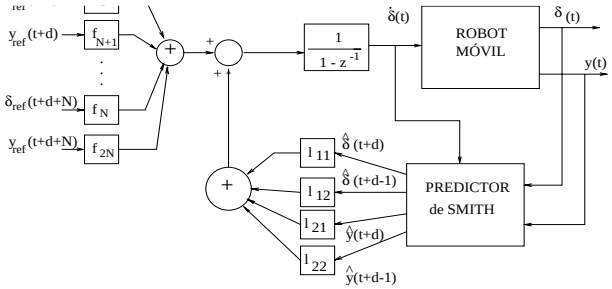


Fig. 6. Estructura del SPGPC para el robot móvil

5. RESULTADOS EXPERIMENTALES

Para los resultados experimentales se utilizaron los siguientes parámetros: $\lambda = 0.5$, $\mu_\delta = 1$, $\mu_y = 1$, $N = 20$, look ahead = 0.6 m, $V = 0.4m/s$ y una aceleración angular $\ddot{\delta}$ de = 30 grados/s^2. El período de muestreo se eligió $T = 0.2$seg. debido a las restricciones de tiempo del sistema operativo Linux utilizado por el robot. Por otro lado como el algoritmo de control puede ejecutarse en un tiempo mucho menor, el período de muestreo también podría elegirse menor si el sistema operativo lo permitiese.

En las figuras 7 y 8 se muestran los buenos resultados experimentales que se obtienen aplicando el control propuesto aún en el caso de seguimiento de curvas con pequeños radios de curvatura.

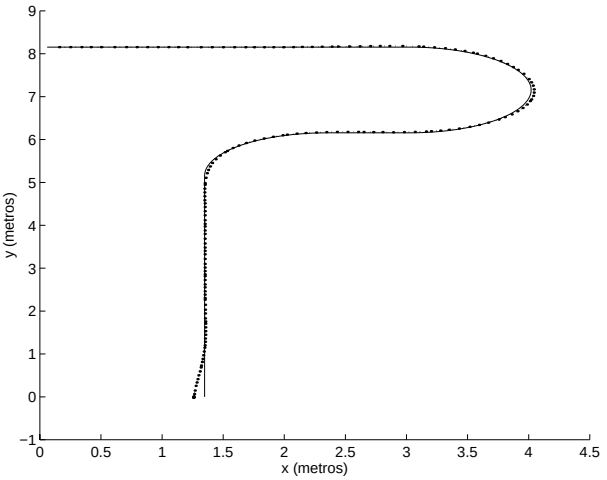


Fig. 7. Seguimiento de caminos del robot Nomad con el SPGPC. Experimento 1 (Referencia en linea continua)

6. CONCLUSIONES

Se analizó en este trabajo la utilización de un control predictivo generalizado basado en el predictor de Smith para el seguimiento de caminos de un robot móvil Nomad. La robustez del algoritmo permite el uso de modelos muy simples para el cálculo de la predicción lo que se traduce en menores tiempos de cálculo del algoritmo implementado en tiempo real. Se analizaron los aspectos prácticos de la implementación del controlador y se mostraron resultados experimentales

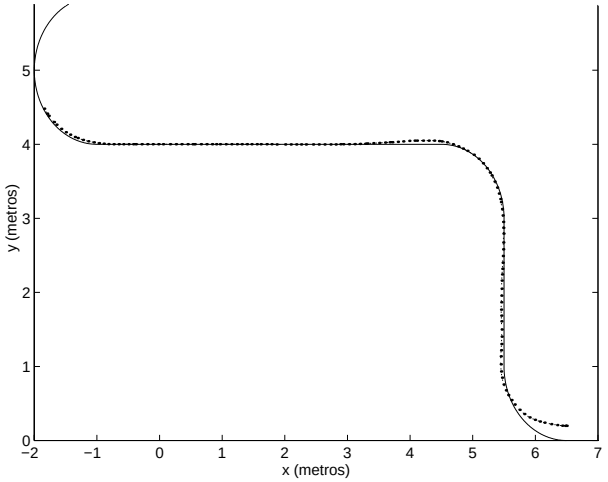


Fig. 8. Seguimiento de caminos del robot Nomad con el SPGPC. Experimento 2 (Referencia en linea continua)

que comprueban la robustez y buenas prestaciones alcanzadas por el control propuesto.

7. REFERENCIAS

Amidi, O. (1990). *Integrated Mobile Robot Control*. Carnegie Mellon University Robotic Institute (Pittsburgh). Technical Report CMU-RI-TR-90-17.

Camacho, E.F. and C. Bordons (1995). *Model Predictive Control in Process Industry*. Springer Verlag.

D.W. Clarke, C. Mothadi and P.S. Tuffs (1987). Generalized Predictive Control. Part I The Basic Algorithm and Part II Extensions and Interpretations. *Automatica* **23**(2), 137–160.

Gómez Ortega, J. (1994). Navegación en robots móviles basada en técnicas de control predicitvo neuronal. PhD thesis. Escuela Sup. de Ingenieros, Univ. de Sevilla.

Gómez Ortega, J. and E. F. Camacho (1996). Mobile Robot Navegation in a Partially Structured Static Enviroment, Using Neural Predictive Control. *Control Engineering Practice* **4**(12), 1669–1679.

Kanayama, Y., Y. Kimura, T. Noguchi and F. Miyakazi (1990). A Stable Tracking Control Method for an Autonomous Mobile Robot. In: *Proc. IEEE Int. Conf. on Robotics and Automation*. pp. 384–389.

Nelson, W. and I. Cox (1990). *Autonomous Robot Vehicles*. Chap. Local Path Control for an Autonomous Vehicle, pp. 38–44. Springer-Verlag.

Nelson, W.L. (1989). Continous Steering-Function Control of Robot Carts. *IEEE Transactions on Industrial Electronics* **36**(3), 330–337.

Normey-Rico, J.E. and E.F. Camacho (1996). A smith predictor based generalised predictive controller. *Technical report GAR 1996/07*.

Normey-Rico, J.E., E.F. Camacho and J. Gómez Ortega (1998*a*). Robustez e predição em controle preditivo generalizado. In: *12th Congreso Brasileiro de Automatica*. Uberlandia (Brasil).

Normey-Rico, J.E., J. Gómez-Ortega and E.F. Camacho (1998*b*). A smith predictor based robot path-tracking. In: *3rd Ifac Symposium on Intelligent Autonomous Vehicles*. Madrid. pp. 471–476.

Normey-Rico, J.E., J. Gómez Ortega and I. Alcalá-Torrego (1998*c*). Low-Time Consuming Implementation of a Predictive Path-Tracking Control for a "Synchro-Drive" Mobile Robot. In: *Proc. 5th International Workshop on Advanced Motion Control*. Coimbra (Portugal). pp. 350–355.

Ollero, A. and O. Amidi (1991). Predictive Path Tracking of Mobile Robots. Aplications to the CMU Navlab. In: *Proc. IEEE Fifth Int. Conf. on Advanced Robotics*. Pisa. pp. 1081–1086.

Shin, D. H. and S. Singh (1990). *Vision and Navigation. The Carnegie Mellon Navlab*. Chap. Vehicle and Path Models for Autonomous Navigation, pp. 283–307. Kluwer Academic Publishers.

Smith, O.J.M (1958). *Feedback Control Systems*. Mc Graw Hill.