SISTEMA PARA NAVEGAÇÃO E GUIAGEM DE ROBÔS MÓVEIS AUTÔNOMOS

Franz A. Sandi L.[1] , Elder M. Hemerly[2] e Walter F. Lages [3]

[1,2]Departamento de Controle e Conversão de Energia
Instituto Tecnológico de Aeronáutica - ITA
CTA - ITA - IEEE 12228-900 São José dos Campos-SP
[3]Departamento de Física, Fundação Universidade do Rio Grande
Avenida Itália s/n, 96201-900 Rio Grande-RS
e-mail: [2]hemerly@ele.ita.cta.br, [3]w.fetter@ieee.org

Resumo: Um sistema para navegação e guiagem de robôs móveis autônomos é proposto e implementado neste trabalho. Para a navegação utiliza-se integração sensorial via filtro de Kalman de dois subsistemas independentes, o *dead-reckoning*, que utiliza a leitura de pulsos de dois *encoders* alocados nas rodas direita e esquerda do veículo, e uma bússola digital. A informação de posição e atitude do veículo é então utilizada por um sistema de guiagem, baseado em controlador tipo *fuzzy*, que é de sintonização simples e prescinde do conhecimento do modelo dinâmico do veículo. O sistema completo é implementado em tempo real e o bom desempenho é exibido em situações de interesse.

Abstract: A navigation and guidance system for autonomous mobile robots is developed in this work. Two independent measuring subsystems are integrated via a Kalman filter, for navigation purposes. The first is a dead-reckoning subsystem based on rotation counts of right and left vehicle's wheels, and the second one is a digital compass. The vehicle position and attitude thereby obtained are fed to a fuzzy guidance controller, aiming at achieving good performance without requiring the complex model of the vehicle's dynamics. Experimental results show that accurate estimates, in orientation and position, are obtained by the system and the guidance controller performs well on realistic paths.

1 INTRODUÇÃO

O crescente interesse por robôs autônomos se deve à grande diversidade de tarefas que podem ser por eles realizadas. Em Asami(94) e Schraft(94), por exemplo, são discutidos os robôs de serviços, como constituindo área emergente de pesquisa e aplicação. Como um exemplo mais específico, podemos citar Mandow *et alii*(96), onde se descreve um robô móvel autônomo para aplicação em agricultura, objetivando substituir o trabalhador humano em atividades inóspitas em estufas, tal como pulverização com inseticidas.

Artigo submetido em 06/11/97
1a. Revisão em 26/02/98; 2a. Revisão em 08/06/98
Aceito sob recomendação do Ed. Cons. Prof.Dr. Ricardo Tanscheit

O problema de controle de robôs móveis exibe dois subproblemas principais: 1) navegação, que designa a determinação de posição e orientação do veículo em um dado instante de tempo, e 2) guiagem, que se refere ao controle da trajetória.

Um método clássico de navegação é o *dead-reckoning*, cuja precisão depende diretamente da qualidade dos sensores utilizados, sendo inevitável a acumulação de erro de posição, o que requer integração com outros tipos de sensores, de modo a se compensar este erro. Por exemplo, em Fuke e Krotkov(1996) integra-se a informação do número de pulsos detectados por *encoders* instalados nas rodas do veículo (odômetro) com a informação de acelerômetros e girômetros solidários ao veículo. Em Murata e Hirose(1993) e Lages *et alii*(1996) são utilizados algoritmos de processamento de imagens que integrados à informação dos *encoders* estimam posição e orientação. Em Borenstein e Feng(1996) é apresentado um método que combina as informações de um girômetro e um odômetro, objetivando melhorar as estimativas da orientação de um veículo móvel. Uma vez que ambos os sensores apresentam erros crescentes no tempo e a interação do veículo com o meio possui características imprevisíveis, faz-se necessária uma cuidadosa modelagem estatística dos erros. Em Barshan e Durrant-Whyte(1995) encontra-se um estudo extensivo da utilização de sensores inerciais, girômetros e acelerômetros, em robótica móvel, com particular ênfase na análise do erro tipo *bias*, pois é o que mais degrada o desempenho de girômetros e acelerômetros.

As informações sobre posição e orientação determinadas no módulo de navegação são fundamentais para o módulo de guiagem, pois permitem a determinação do erro em relação às referências desejadas. Tipicamente essas referências são especificadas via *waypoints*, que são pontos com posição e orientação desejadas para o veículo, havendo uma grande variedade de técnicas de controle para efetuar a guiagem. Por exemplo, em Hemerly e Rodrigues(1994) é utilizado um controlador tipo PI para controle de velocidade das rodas esquerda e direita do veículo. Em Aicardi *et alii*(1995) é empregada uma estratégia de controle obtida via método de Lyapunov, assegurando-se que os erros de orientação e posição

tendam assintoticamente a zero. Uma estratégia de controle exponencialmente estável é proposta em Canudas de Wit e Sordalen(1992), consistindo em se forçar a trajetória do veículo a ser do tipo circunferência passando pelo *waypoint*. Em Vaneck(1997) é utilizada um controlador tipo *fuzzy* para efetuar a guiagem de um barco autônomo, com a grande vantagem de prescindir do conhecimento da dinâmica do veículo e ser de fácil sintonização.

Este trabalho investiga alguns aspectos dos problemas de navegação e guiagem em robótica móvel, e as principais contribuições são:

1. No que se refere à navegação, integra-se a informação de orientação fornecida por um odômetro com a informação de uma bússola digital mediante um filtro de Kalman, que são sensores de baixo custo e portanto de interesse;
2. A implementação em tempo real é efetuada via sistema operacional Linux e o desempenho desta integração é avaliado. A orientação estimada é utilizada para calcular a posição do veículo em cada instante de tempo, de modo a se reduzir a acumulação de erros;
3. Um controlador tipo *fuzzy* para a guiagem do veículo, baseado nas informações de posição e orientação obtidas na fase de navegação, é proposto e implementado, objetivando-se obter bom desempenho sem a necessidade de se usar um modelo matemático do veículo. Com as devidas adaptações para o caso de um veículo móvel, este controlador pode ser visto como uma generalização daquele apresentado em Vaneck(1997), por permitir definir também o perfil de velocidade do veículo.

Na seção 2 é efetuado um resumo sobre o robô móvel utilizado neste trabalho. A seção 3 concerne o módulo de navegação, no qual um filtro de Kalman é utilizado para integração de sensores. Resultados de simulação e experimentais são apresentados. Na seção 4 é descrito o projeto do controlador tipo *fuzzy* para guiagem. A implementação em tempo real é também efetuada e resultados experimentais são reportados. As conclusões são apresentadas na seção 5.

2 DESCRIÇÃO DO ROBÔ MÓVEL

O diagrama de blocos do robô móvel utilizado neste trabalho é mostrado na figura 1. A movimentação é efetuada através de duas rodas dianteiras operando em modo diferencial. Cada uma possui um *encoder* que permite medir sua velocidade angular. Na parte traseira existe uma terceira roda de apoio, que gira livremente. Uma bússola digital é montada no centro do eixo das rodas dianteiras, permitindo medir a orientação do veículo.

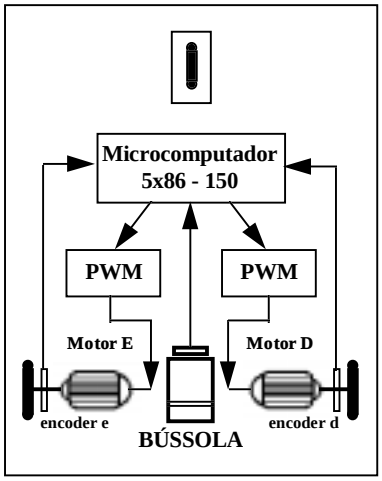


Figura 1- Diagrama de blocos do robô móvel.

Uma fotografia do robô móvel é mostrada figura 2.

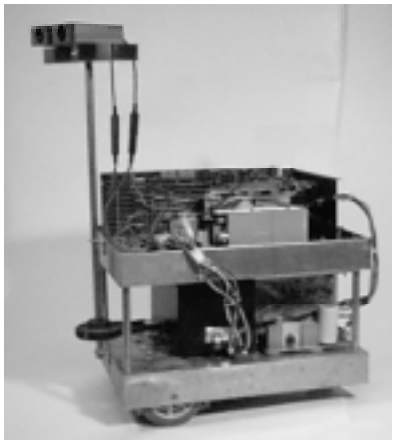


Figura 2- Fotografia do robô móvel utilizado neste trabalho.

O modelo cinemático do veículo exibido na figura 2 é obtido com base na figura 3.

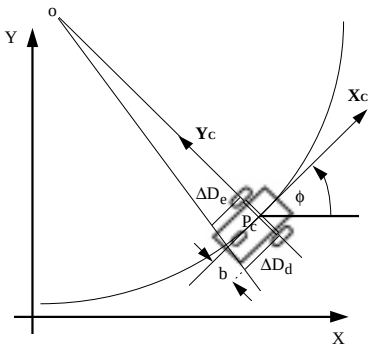


Figura 3- Representação do sistema de coordenadas do veículo (baseado em Murata e Hirose, 1996, Fig. 3, pg. 150).

Na figura 3 tem-se:

P_C : interseção entre o eixo de simetria e das rodas.
ΔDe: deslocamento da roda esquerda no período de amostragem.
ΔDd: deslocamento da roda direita no período de amostragem.
X_C , Y_C: sistema de coordenadas do veículo com origem em Pc.
X, Y: coordenadas do sistema de referência.
ϕ: ângulo medido entre o eixo de simetria do veículo e o eixo X.
b: distância entre as rodas e o eixo de simetria.

Conforme em Murata e Hirose(1993), admitindo-se somente a existência de movimento na direção do eixo de simetria, obtém-se

$$\dot{y}_c \cos\phi - \dot{x}_c \operatorname{sen}\phi = 0 \qquad (1)$$

onde (y_c, x_c) são as coordenadas de P_C em relação às coordenadas de referência (X, Y).

Considerando-se que as rodas do veículo não deslizam, resultam (Yamamoto e Yun, 1994)

$$\dot{\theta}_d r = \dot{y}_c \operatorname{sen}\phi + \dot{x}_c \cos\phi + b\dot{\phi} \qquad (2)$$

$$\dot{\theta}_e r = \dot{y}_c \operatorname{sen}\phi + \dot{x}_c \cos\phi - b\dot{\phi} \qquad (3)$$

onde r é o raio das rodas, θ_d e θ_e são os deslocamentos angulares das rodas direita e esquerda, respectivamente.

Subtraindo-se (3) de (2) obtém-se

$$\dot{\phi} = \frac{\dot{\theta}_d r - \dot{\theta}_e r}{2b} \tag{4}$$

e somando-se (2) e (3) resulta

$$\dot{x}_c \cos\phi + \dot{y}_c \operatorname{sen}\phi = \frac{\dot{\theta}_d r + \dot{\theta}_e r}{2} = \dot{D} \tag{5}$$

onde $\dot{D}$ é a velocidade do veículo na direção do seu eixo de simetria.

Resolvendo-se (1) e (5) para $\dot{x}_c$ e $\dot{y}_c$, obtém-se

$$\dot{x}_c = \dot{D}\cos\phi \tag{6}$$

$$\dot{y}_c = \dot{D}\operatorname{sen}\phi \tag{7}$$

$$\dot{\phi} = \omega \tag{8}$$

onde ω é a velocidade angular do sistema fixo ao veículo com relação ao sistema de coordenadas (X,Y).

Discretizando-se (6), (7) e (8), de forma similar a Yamamoto e Yun(1994), e assumindo-se que a trajetória percorrida por P_C é um arco de circunferência conforme figura 3, obtemos

$$x_c(k+1) = x_c(k) + \Delta s(k)\cos(\phi(k) + \frac{\Delta\phi(k)}{2}) \tag{9}$$

$$y_c(k+1) = y_c(k) + \Delta s(k)\operatorname{sen}(\phi(k) + \frac{\Delta\phi(k)}{2}) \tag{10}$$

$$\phi(k+1) = \phi(k) + \Delta\phi(k) \tag{11}$$

onde

$$\Delta s(k) = \Delta D(k)\frac{\operatorname{sen}\Delta\phi(k)/2}{\Delta\phi(k)/2} \tag{12}$$

$$\Delta D(k) = \frac{\Delta D_d(k) + \Delta D_e(k)}{2} \tag{13}$$

$$\Delta\phi(k) = \frac{\Delta D_d(k) - \Delta D_e(k)}{2b} \tag{14}$$

Convém ressaltar que quando $\Delta\phi(k) = 0$ em (12), resulta $\Delta s(k) = \Delta D(k)$, não havendo portanto indeterminação em (12).

As expressões $\Delta D_d(k)$ e $\Delta D_e(k)$ em (13) e (14) representam os deslocamentos lineares das rodas direita e esquerda no intervalo [k,k+1], respectivamente, dadas por

$$\Delta D_d(k) = 2\pi r\frac{NP_d(k)}{\Pi} \tag{15}$$

$$\Delta D_e(k) = 2\pi r\frac{NP_e(k)}{\Pi} \tag{16}$$

onde Π é o número de pulsos por revolução dos *encoders*, e $NP_d(k)$, $NP_e(k)$ representam o número de pulsos detectados no intervalo [k,k+1].

Convém ressaltar que há erros de modelagem devido ao escorregamento das rodas, às irregularidades do solo, à não exatidão no conhecimento do raio das rodas, à distância entre elas, etc., e que foram modelados como sendo ruído branco de medida na leitura dos *encoders*. Vide Borenstein e Feng(1996) para maiores detalhes.

3 SISTEMA DE NAVEGAÇÃO

Nesta seção é efetuado um resumo de Sandi *et alii*(1997), onde são apresentados detalhes sobre a integração do odômetro e bússola via filtro de Kalman.

Seja $\phi_B(k)$ a leitura no instante k da orientação proporcionada pela bússola digital. O valor verdadeiro $\phi_t(k)$ é corrompido pelo ruído de medida $\nu(k)$, isto é,

$$\phi_B(k) = \phi_t(k) + \nu(k) \tag{17}$$

Neste experimento foi utilizada a bússola digital fabricada pela *Precision Navigation, Inc*., cujas especificações estão em Vector 2X Compass Module(1996).

A integração do odômetro e bússola emprega um filtro de Kalman linear. A orientação calculada com base nos *encoders*, $\phi_E(k)$, pode ser modelada como o valor verdadeiro $\phi_t(k)$ mais um erro de orientação $\delta\phi(k)$, qual seja,

$$\phi_E(k) = \phi_t(k) + \delta\phi(k) \tag{18}$$

O processo de observação y(k) é a diferença de orientação entre ambos os sistemas, (17) e (18), resultando

$$y(k) = \delta\phi(k) + \nu(k) \tag{19}$$

Supondo-se que ruído branco w(k) é adicionado ao incremento de orientação em cada instante de tempo, a orientação $\phi_E(k+1)$ é calculada com base em (11),

$$\phi_E(k+1) = \phi_E(k) + \Delta\phi_t(k) + w(k) \tag{20}$$

onde $\Delta\phi_t(k)$ é o valor verdadeiro do incremento de orientação.

Considerando-se que as magnitudes em (11) são verdadeiras, subtraindo-se (20) de (11) e utilizando (18) obtemos o erro de orientação,

$$\delta\phi(k+1) = \delta\phi(k) + w(k) \tag{21}$$

que constitui a equação de estado para o filtro de Kalman.

Concluimos então que o filtro de Kalman linear é escalar com os ruídos de medida, $\nu(k)$, e de estado, w(k), não-correlacionados, permitindo-nos estimar o erro de orientação, para a seguir subtraí-lo do valor calculado a partir dos *encoders*, como é mostrado na figura 4. A orientação corrigida é retroalimentada para calcular a posição, de modo a se reduzir a acumulação de erros.

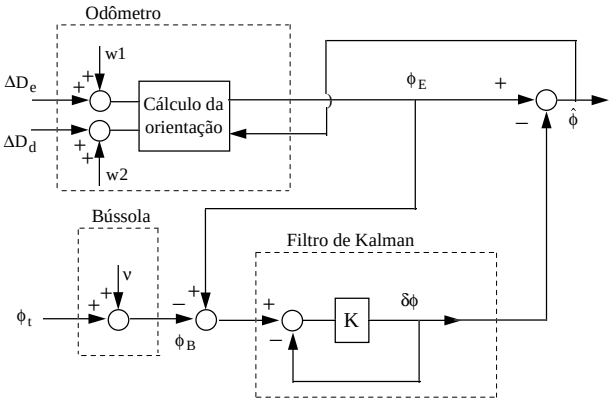


Figura 4- Diagrama de blocos da integração odômetro/bússola via filtro de Kalman.

Resultados relativos à simulação da integração odômetro/bússola podem ser encontrados em Sandi *et alii*(1997). Aqui são apresentados apenas detalhes sobre a implementação em tempo real, também apresentada em Sandi *et alii*(1997), e baseada em microcomputador 5x86-150. Esta implementação é particularmente dificultada pelo fato de que os sensores operam com taxas de amostragem diferentes e não múltiplas. O diagrama de temporização para acionamento da bússola utilizada pode ser visto na figura 5. O símbolo ■ indica processamento pelo microcomputador que consiste apenas da leitura ou escrita de alguns bits nas portas de I/O, portanto é executado em no máximo alguns μs. Os números indicam os *delays* necessários em ms. O símbolo ❒ indica o tempo restante dentro do período de amostragem da bússola de 200ms. Note que embora o processamento para leitura da bússola demore em torno de 135ms, o processamento é necessário apenas durante breves intervalos de alguns μs. A maior parte do tempo necessário para leitura da bússola é constituído por *delays*.

Para implementação em tempo real do sistema de navegação é necessário também efetuar a leitura dos *encoders*, cujo diagrama de temporização pode ser visto na figura 6. Novamente, o símbolo ■ indica processamento pelo microcomputador, da ordem de alguns μs, e o símbolo ❒ indica o tempo restante dentro do período de amostragem dos *encoders*, que é de 50ms.

Conclui-se, então, que a leitura da bússola não pode ser feita pelas técnicas convencionais de *busy-wait*, pois, devido ao seu tempo de processamento em torno de 135ms, isto faria com que os *encoders* não pudessem ser processados a cada 50 ms. Um ambiente multitarefa é portanto necessário para a implementação. Neste caso, a leitura da bússola e a leitura dos *encoders* são feitas por tarefas periódicas diferentes, cada uma com seu tempo de amostragem. Quando uma tarefa necessita efetuar um *delay*, ela é posta para *dormir* durante este tempo, liberando o processador para as demais tarefas.

Este ambiente multitarefa foi implementado neste trabalho utilizando-se o sistema operacional Linux (Yodaiken, 1997). Convém ressaltar que as tarefas não foram implementadas como processos, mas sim como *threads* (Tanenbaum, 1992). Processos operam com espaços de endereçamento diferentes entre si, enquanto *threads* compartilham o mesmo espaço de endereço, além de vários outros recursos. Uma conseqüência direta deste fato é que a troca de dados entre *threads* é feita mais facilmente do que entre processos.

O programa implementado para navegação possui quatro *threads:* 1) leitura da bússola, com período de amostragem de 200ms; 2) leitura dos *encoders*, com período de amostragem de 50ms; 3) fusão de dados, sem período de amostragem fixo, mas sincronizado com os dois *threads* anteriores; 4) interface com o usuário, com período de amostragem de 1s. Este *thread* possui a menor prioridade e portanto executa apenas quando os demais *threads* não necessitam utilizar o processador. Para maiores detalhes sobre a implementação em tempo real do sistema de navegação, vide Sandi *et alii*(1997).

Como resultado experimental, foi executada uma trajetória circular, com raio mantido constante via elo rígido, de um ângulo de 90°. A trajetória nominal percorrida pelo veículo foi então calculada. Quanto aos ruídos de estado e de medida, as covariâncias Q=0.08 rad^2 e R=0.0069 rad^2 foram utilizadas na filtro de Kalman. O valor de Q foi determinado experimentalmente, pois o ruído de estado depende do ruído na leitura dos *encoders*, e esta dependência é dada por (14), onde aparece inclusive a distância b entre as rodas e o eixo de simetria. A figura 7 mostra a trajetória nominal e as posições calculadas com base na integração odômetro/bússola e com o odômetro apenas.

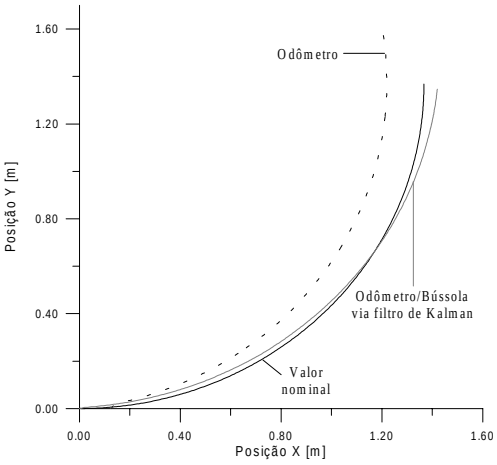


Figura 7- Posições obtidas em tempo real pelo odômetro e pela integração odômetro/bússola, via filtro de Kalman.

A figura 8 mostra as orientações relativas a esta realização.

■	10	■	80 - 100	10	■	5	■	1	■	1	...	1	■	5	■	❒
Disparo			Espera	habilitação			leitura (16 bits)							desabi-litação		livre

Figura 5- Diagrama de temporização da bússola.

■	❒
leitura	livre

Figura 6- Diagrama de temporização dos *encoders*.

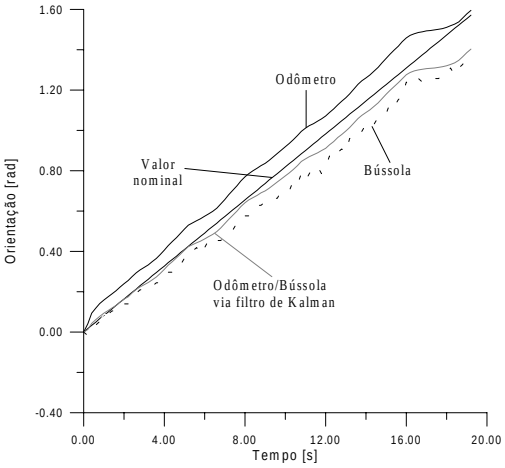


Figura 8- Orientações correspondentes às posições da figura 7.

Note-se que a posição estimada na figura 7 oscila em torno da trajetória nominal, com erros crescendo com o tempo (devido à acumulação do erro de posição), mas sempre está mais próxima do valor nominal do que a posição fornecida pelo odômetro apenas. Obviamente, seria conveniente eliminar a acumulação de erros. Para tanto, faz-se necessário utilizar sensores mais sofisticados, como por exemplo a combinação INS+GPS, vide Scherzinger *et alii*(1997) para detalhes.

4 CONTROLADOR *FUZZY* PARA GUIAGEM

A implementação de um controlador tipo *fuzzy* (Pedrycz, 1989) foi efetuada, pois o mesmo prescinde do modelo dinâmico do veículo. Requer apenas que os comandos de controle, obtidos com base na experiência heurística do operador humano, sejam expressos em variáveis lingüísticas e regras de produção do tipo: Se {condição 1} e {condição 2}, então {ações}. Assim, para converter as variáveis numéricas de entrada do controlador em variáveis lingüisticas é necessário um processamento denominado *fuzzyficação*. De igual modo, para transformar um comando de controle, fornecido como variável lingüistica, em valores numéricos que possam ser interpretados pelo atuador do sistema, é preciso um outro processamento denominado *defuzzyficação*. Todos estes componentes de um controlador fuzzy são mostrados na figura 9.

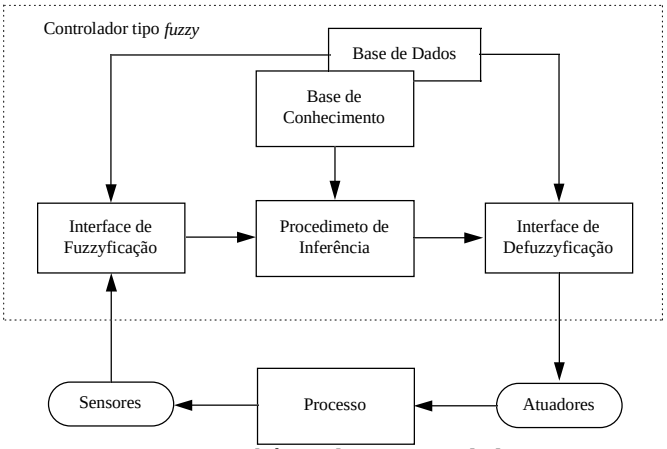
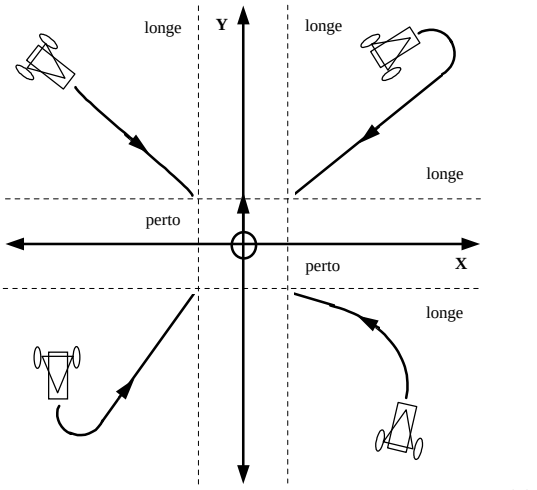


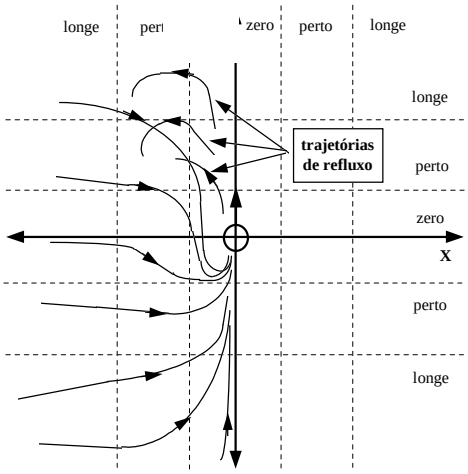
Figura 9- Estrutura básica de um controlador tipo *fuzzy*.

Maiores detalhes sobre lógica *fuzzy* e controladores deste tipo podem ser encontrados na literatura (Mendel, 1995; Gomide e Gudwin, 1994). Nosso objetivo aqui é projetar um controlador para guiagem tipo *fuzzy*, com procedimento similar ao de Vaneck(1997), no qual se obtém as regras do controlador utilizando um "esboço em papel" para visualizar os caminhos que deve seguir um barco autônomo para atingir o objetivo. Supõe-se que o veículo não se movimenta para trás, ou seja, move-se sempre em frente. Será considerado também o controle de velocidade, o que não é efetuado em Vaneck(1997), mas é importante para melhorar o desempenho do sistema de posição e porque o perfil de velocidade pode ser útil em sistemas nos quais se utilizam *landmarks* e processamento de sinal de vídeo para reduzir o erro de posição (Murata e Hirose, 1993), e também em tarefas de inspeção e monitoração.

As regras de inferência do controlador são obtidas levando-se em consideração o raio de giro do veículo e a sua velocidade média. Assim, utiliza-se um esboço em papel das trajetórias a serem seguidas, do ponto de partida qualquer até se atingir o *waypoint*. Aqui se define um *aim point* como sendo uma região em torno do *waypoint*, cuja utilidade será explicitada a seguir. O conjunto de regras aliado à máquina de inferência nebulosa (Nascimento Jr. e Yoneyama, 1997) é concebido aqui visando dois aspectos importantes: 1) minimizar as manobras perto do *waypoint*, e 2) incluir trajetórias que permitam retornar em direção ao *waypoint*, quando forem efetuadas aproximações incorretas, denominadas trajetórias de refluxo. Na figura 10 é mostrado o esboço das trajetórias a serem percorridas pelo veículo tanto para pontos de partida longe do *aim point* (figura 10a), como quando manobras são realizadas próximo a este (figura 10b). Delimitou-se o espaço de posição em três áreas denominadas como *longe*, *perto* e *zero*, segundo a sua proximidade. Na figura 10b são mostradas apenas as trajetórias à esquerda do eixo *y*.



(a)



(b)

Figura 10- Esboço das trajetórias do veículo para vários pontos de partida (baseado em Vaneck, 1997, Fig. 5, pg. 45):
a) Veículo longe do *aim point*; b) Veículo perto do *aim point*.

São previstos tipos específicos de manobras, que dependem da magnitude do erro de posição entre o veículo e o *aim point*:

- Erro de posição grande: manobra-se diretamenta para o *aim point*.
- Erro de posição pequeno: manobra-se de modo a se atingir o *waypoint*.
- Erro de posição perto de zero: realizam-se pequenas correções de orientação.

As entradas do controlador são os erros de posição nos eixos *x* e *y*, e orientação do veículo. Considera-se nesta primeira fase que o veículo tem velocidade constante, e portanto não é necessário utilizar a velocidade como entrada. Os conjuntos nebulosos associados às variáveis de entrada do controlador são definidos por

erro de posição *x* = **{GN, PN, ZE, PP, GP}**
erro de posição *y* = **{GN, PN, ZE, PP, GP}**
erro de orientação = **{GN, MN, PN, ZE, PP, MP, GP}**

e o conjunto associado à variável de saída do controle dado por

orientação = {**GN**, **MN**, **PN**, **ZE**, **PP**, **MP**, **GP**}

onde **GN** é grande negativo, **MN** meio negativo, **PN** pequeno negativo, **ZE** zero, **PP** pequeno positivo, **MP** meio positivo, e **GP** grande positivo. Com isto, o conjunto de regras de inferência utilizado é representado de forma compacta através de tabelas, que são mostradas na figura 11, e são conhecidas como memórias associativas *fuzzy* (Kosko, 1992).

ψ \ X	GN	PN	ZE	PP	GP
GN	PP	PP	GP	MN	MN
MN	PP	PP	MP	MP	MP
PN	PN	PN	PP	PP	PP
ZE	PN	PN	ZE	PP	PP
PP	PN	PN	PN	PP	PP
MP	MN	MN	MN	PN	PN
GP	MP	MP	GN	PN	PN

Y = GN

ψ \ X	GN	PN	ZE	PP	GP
GN	PP	MP	GP	GN	MN
MN	PN	PN	MP	GP	GP
PN	PN	PN	PP	MP	MP
ZE	MN	PN	ZE	PP	MP
PP	MN	MN	PN	PP	PP
MP	GN	GN	MN	PP	PP
GP	MP	GP	GN	MN	PN

Y = PN

ψ \ X	GN	PN	ZE	PP	GP
GN	PN	PN	ZE	MN	MN
MN	PN	MN	MP	MN	MN
PN	MN	MN	PP	GN	GN
ZE	GN	GN	ZE	GP	GP
PP	GP	GP	PN	MP	MP
MP	MP	MP	MN	MP	PP
GP	MP	MP	ZE	PP	PP

Y = ZE

ψ \ X	GN	PN	ZE	PP	GP
GN	ZE	PN	PP	PN	MN
MN	MN	GP	PN	MN	GN
PN	GN	GP	MN	MN	GN
ZE	GN	GP	GP	GN	GP
PP	GP	MP	MP	GN	GP
MP	GP	MP	PP	GN	MP
GP	MP	PP	PN	PP	ZE

Y = PP

ψ \ X	GN	PN	ZE	PP	GP
GN	ZE	MN	ZE	MN	MN
MN	MN	GN	PN	GN	GN
PN	GN	GN	MN	GN	GN
ZE	GN	GP	MP	GN	GN
PP	GP	GP	MP	GP	GP
MP	GP	GP	PP	GP	MP
GP	MP	MP	ZE	MP	ZE

Y = GP

Figura 11- Banco de dados contendo todas as regras de inferência do controlador.

Na figura 11 o erro de posição no eixo *x* é representado por **X**, o erro de posição no eixo *y* por **Y**, o erro de orientação por **ψ** e a ação do controle (comando de orientação) está contida dentro da tabela. Assim, por exemplo, se **X = PN**, **Y = GP** e **ψ = PP**, então o comando de orientação do veículo é **θ = GP**, correspondendo à região sombreada na última tabela da figura 11.

Deste modo, obtemos ações de controle que, quando o veículo se encontra longe do *aim point* (com um grande espaço de manobras), acarretem só pequenas correções na orientação do veículo. Se o veículo estiver perto do *aim point*, adotam-se ações de controle mais agressivas para garantir uma rápida correção da orientação. Finalmente, quando o veículo navega muito perto do objetivo (na região zero), então comandos de orientação moderados são realizados e, se for necessário, navega-se pelas trajetórias de refluxo para manter a característica suave das trajetórias do veículo

Uma vez definidos os conjuntos de variáveis nebulosas, resta-nos atribuir a cada conceito lingüístico um intervalo de valores numéricos, de modo a se efetuar uma conexão entre os valores medidos pelos sensores e a máquina de inferência nebulosa, e entre esta e os atuadores do sistema. Esta conexão é um mapeamento entre valores numéricos e variáveis lingüisticas, que pode ser representado por funções de pertinência, indicando a que elemento da função pertence um valor numérico dado, e com que grau.

Na figura 12 são apresentadas as funções de pertinência correspondentes ao erro de posição (entrada do controlador), usando-se a caracterização triangular proposta por Zadeh (Gomide e Gudwin, 1994), denotadas por $\mu(\cdot)$ e variando entre 0 e 1.

Cada erro de posição na figura 12 apresenta 5 funções de pertinência, que são simétricas com respeito ao valor zero e cuja largura decresce quando o valor absoluto do erro diminui. Isto fornece um melhor desempenho do controlador quando o veículo navega perto do *aim point*.

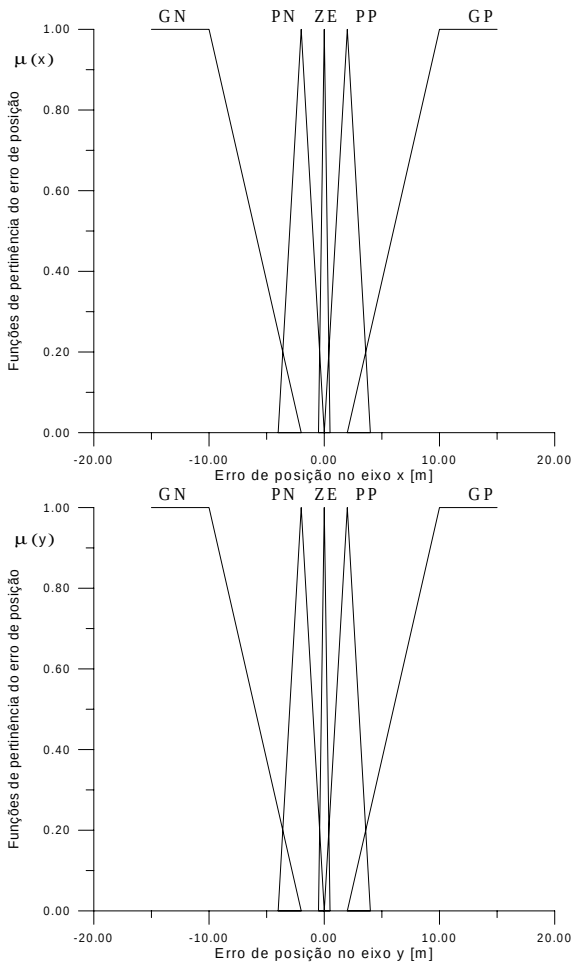


Figura 12- Funções de pertinência relativas ao erro de posição.

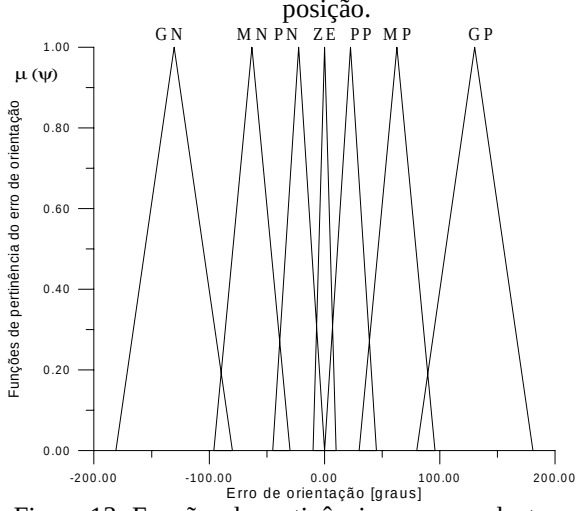


Figura 13- Funções de pertinência correspondentes ao erro de orientação.

Já o erro de orientação apresenta 7 funções de pertinência, conforme mostrado na figura 13, que também são simétricas em relação ao erro nulo.

A saída do controlador é a orientação do veículo com respeito ao seu eixo de simetria. Como este comando de orientação é fornecido a cada instante de tempo, deve ser limitado. Portanto, considerando as dimensões do veículo e seu raio de giro, limitou-se a orientação entre 15° e -15°, como é apresentado na figura 14.

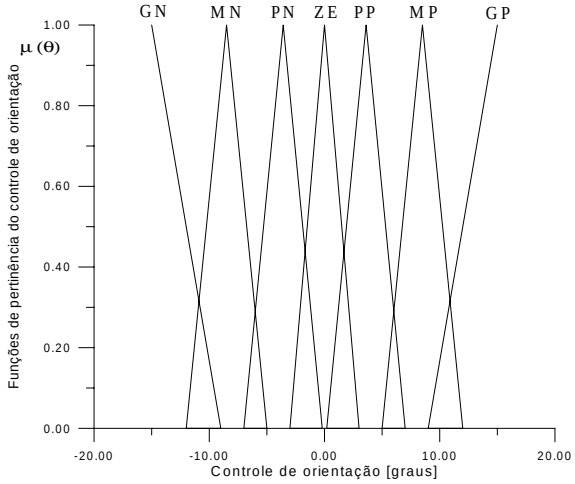


Figura 14- Funções de pertinência relativas ao comando de orientação.

Utilizou-se o método de inferência de correlação mínima para se obter a saída do controlador. Este método, que é resumido na figura 15 para casos particulares de entrada denotados por setas, analisa cada antecedente em uma associação ativa, escolhe o conjunto *fuzzy* com o mínimo grau de pertinência e usa este grau para truncar o conjunto *fuzzy* resultante.

A escolha do processo de d*efuzzyficação* na figura 15 não é crítica em geral, pois, pelas características próprias deste tipo de controlador, as diferenças existentes entre os procedimentos são minimizadas com um ajuste adequado das funções de pertinência. Assim, foi adotado o processo de *defuzzyficação* denominado média dos máximos (Nascimento Jr. e Yoneyama, 1997), que combina as funções de pertinência conseqüentes de modo a calcular um único valor numérico de controle, da seguinte forma,

$$\text{controle} = \frac{h_1 x_1 + h_2 x_2 + \cdots + h_n x_n}{x_1 + x_2 + \cdots + x_n} \qquad (22)$$

onde h_i é o valor do pico máximo da i-ésima função de pertinência conseqüente e x_i o valor numérico correspondente. Note-se que quando a função de pertinência está truncada, h_i é o valor truncado e x_i é o valor médio do intervalo da função.

4.1 Resultados de Simulação com Velocidade Constante

As simulações foram realizadas considerando-se o modelo cinemático da seção 2, e assumindo-se disponíveis os dados de velocidade nas duas rodas do veículo, isto é, sensores sem ruído.Considera-se que as coordenadas instantâneas do veículo constituem o erro de posição x e y, a orientação $\phi(k)$ o erro instantâneo de orientação ψ, e o valor resultante do processo de *defuzzyficação* θ é o comando instantâneo de orientação $\Delta\phi(k)$. Logo, considerando-se constante o incremento de velocidade linear do veículo $\Delta D(k)$, o incremento instantâneo de velocidade em cada roda pode ser calculado com base em (13) e (14), resultando

$$\Delta D_e(k) = \Delta D(k) - b\Delta\phi(k) \qquad (23)$$

$$\Delta D_d(k) = \Delta D(k) + b\Delta\phi(k) \qquad (24)$$

Na figura 16 são mostrados alguns resultados obtidos mantendo velocidade constante de 0,04m/s, com período de amostragem igual a 0,05s. O estado inicial foi mudado de

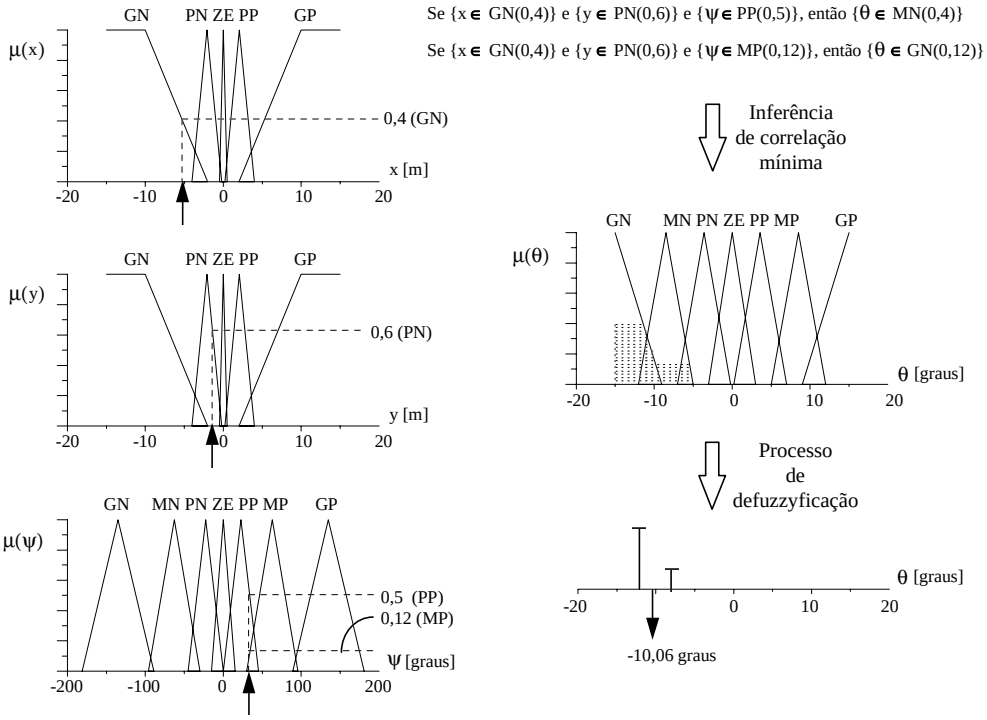


Figura 15- Ilustração do método de inferência de correlação mínima para gerar o sinal de controle de orientação.

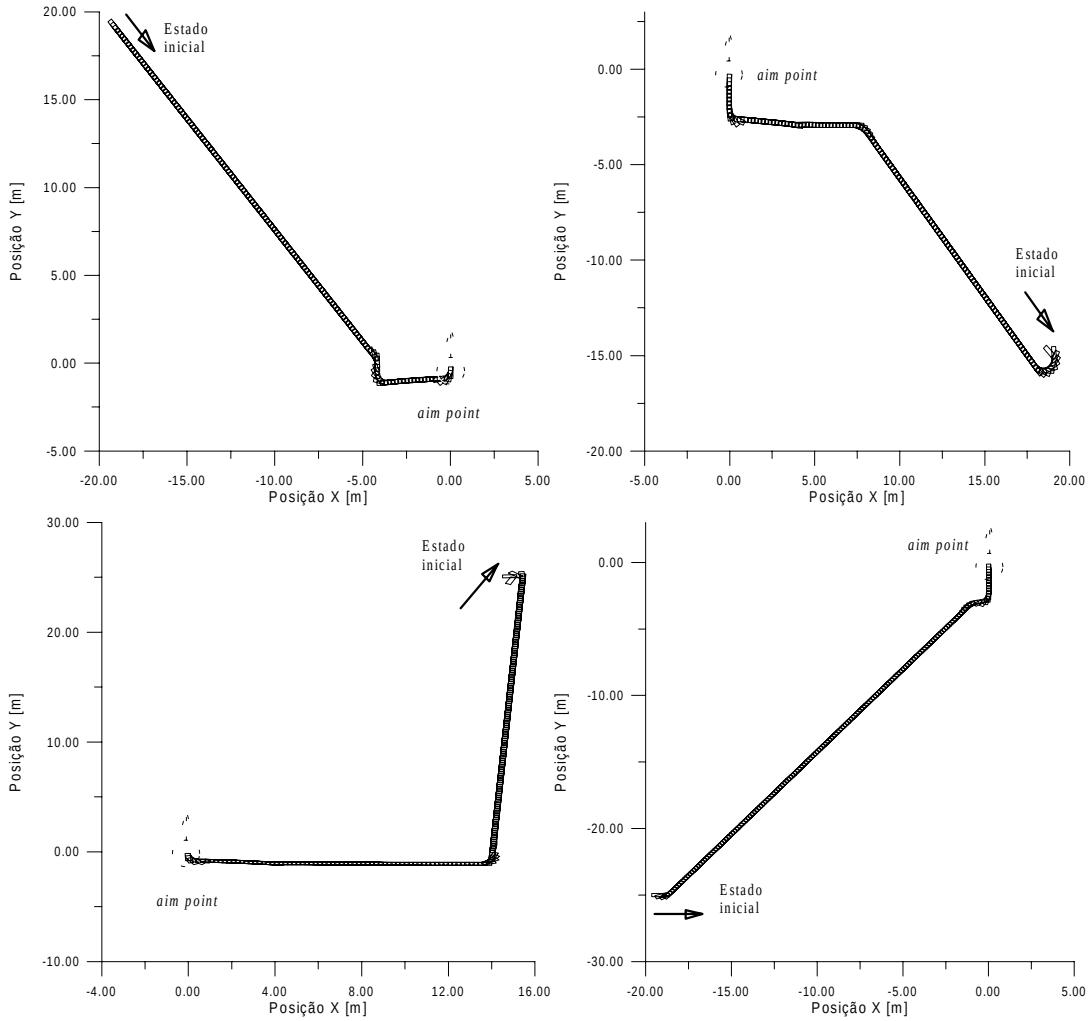


Figura 16- Algumas realizações obtidas via simulação, mantendo-se velocidade constante e mudando-se o estado inicial.

modo a cobrir diversas possibilidades, e os dados foram armazenados a cada 5s, para facilitar a apresentação gráfica.

Se a trajetória for definida por apenas 1 *waypoint* e não houver critério de parada, o veículo atravessará a região delimitando o *aim point*, e retornará logo a seguir por uma trajetória de

refluxo. Assumiu-se como critério de parada a norma ∞ do erro de posição e do erro de orientação, isto é,

$$\| [x , y, \psi] \|_{\infty} \leq 1e\text{-}1 \qquad (25)$$

4.2 Resultados de Simulação com Controle de Velocidade

Pelas características próprias do controle, equações (23) e (24), é possível adicionar conseqüências de velocidade às regras do controlador exposto na seção 4. No entanto, existem várias formas de se implementar um controlador de velocidade tipo *fuzzy* para este caso. Utilizando o conjunto de regras mostrado na figura 11, podem ser geradas conseqüências em termos de velocidade considerando-se como entradas o erro de posição no eixo *x* e o erro de posição no eixo *y*. Podemos ainda utilizar o erro de distância como entrada de um outro controlador tipo *fuzzy* cuja saída serão comandos de velocidade. Desta forma, o número de regras é menor do que no primeiro caso, sendo também de mais simples implementação. Este controlador opera de modo paralelo ao controlador da seção 4.1, e apresenta três regras de inferência, quais sejam,

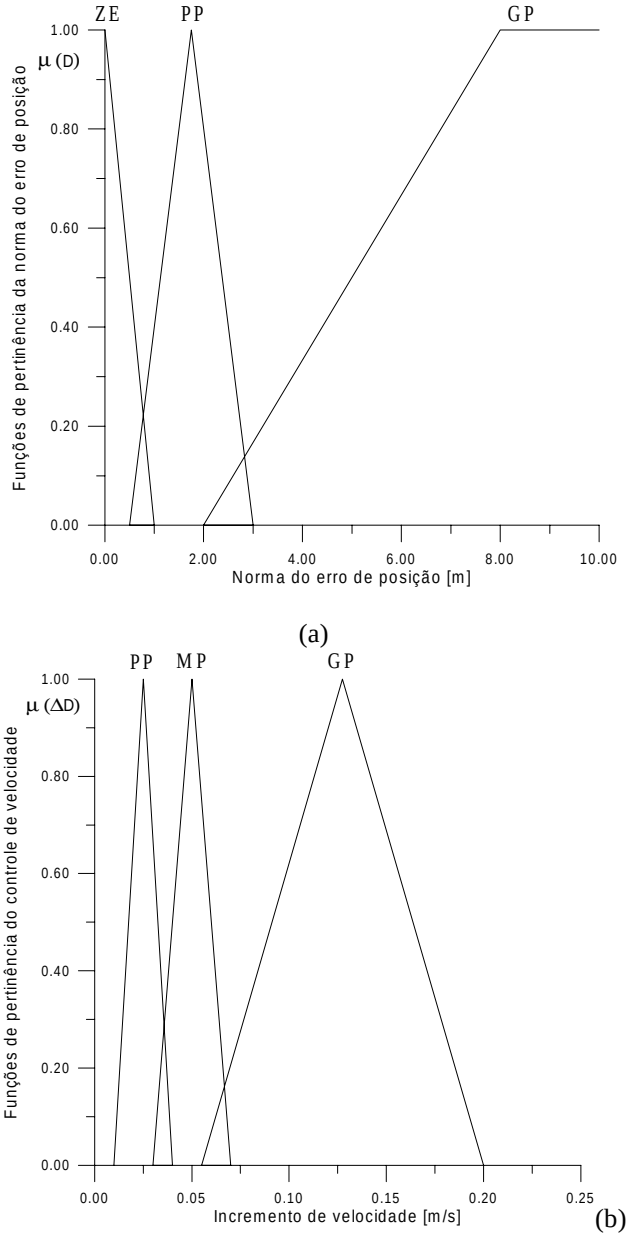


Figura 17- Funções de pertinência para o controlador de velocidade: (a) referentes à entrada, e (b) referentes à saída.

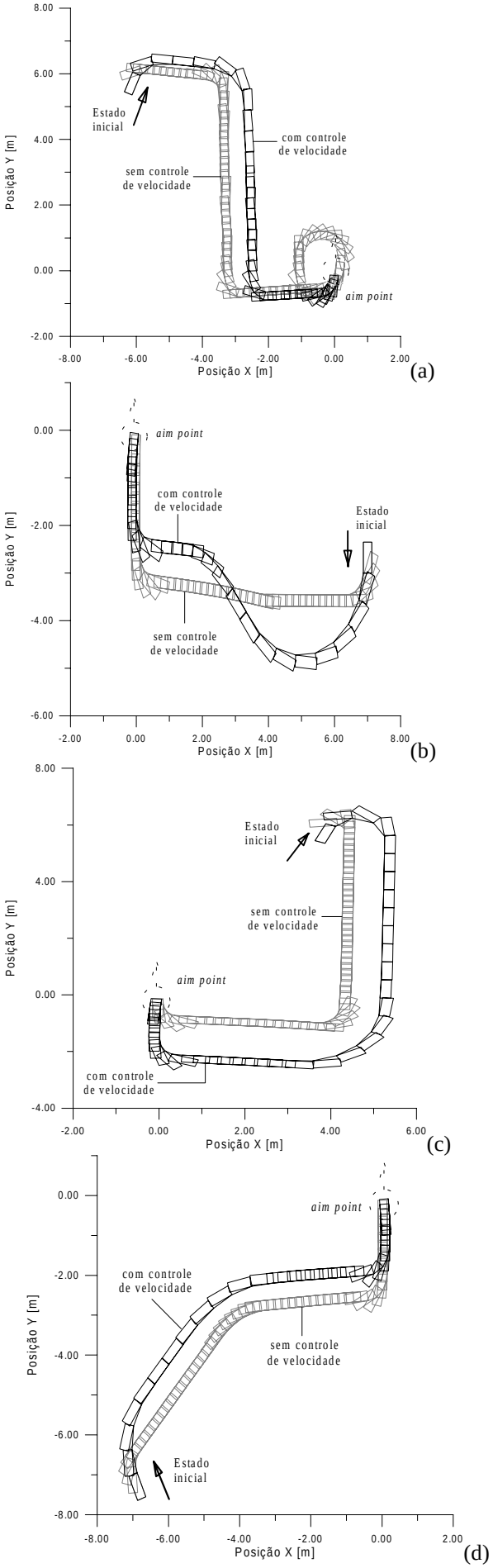


Figura18- Realizações obtidas via simulação comparando-se a guiagem sem e com controle de velocidade.

- **Distância Grande Positiva, então Velocidade Grande Positiva.**
- **Distância Meia Positiva, então Velocidade Meia Positiva.**
- **Distância Pequena Positiva, então Velocidade Pequena Positiva.**

.A figura 17 mostra as funções de pertinência do controlador de velocidade, cuja entrada é a norma infinita do vetor posição, D, e a saída é o incremento de velocidade ΔD.

Convém notar que os comandos de velocidade, na figura 17b, não atingem o valor zero porque podem se apresentar situações nas quais a norma do erro de posição é pequena, mas não o erro de orientação. Assim, quando o erro de posição do veículo é pequeno, reduzimos a velocidade ao mínimo possível para corrigir o erro de orientação, e deixamos o veículo submetido à condição de parada (25). Isto é necessário porque o conjunto de regras na figura 11 foi concebido considerando velocidade constante.

O período de amostragem utilizado nas simulações é de 0,05s. A figura 18 mostra algumas realizações, supondo *waypoint* na origem do sistema de coordenadas, comparando-se os casos onde realizamos guiagem com velocidade constante e quando a velocidade também é controlada. As amostras foram armazenadas com uma taxa de 5s, para facilitar a apresentação gráfica.

Quando é adicionado o controle de velocidade, o raio de giro do veículo aumenta, mostrando que mesmo sendo controles separados, existe um compromisso entre o controle de velocidade e o controle de orientação, que deve ser considerado ao se ajustar os intervalos das funções de pertinência de ambos controladores.

Note-se que em todos os casos na figura 18 o desempenho do sistema de guiagem é melhorado quando incorporamos o controle de velocidade, pois o objetivo é atingido em menor tempo. Na figura 18a pode ser notado que quando se navega com velocidade constante, o veículo não consegue atingir o objetivo com a orientação desejada, tendo que optar pelas trajetórias de refluxo, o que não ocorre quando o controle de velocidade é utilizado.

Em situação mais realista, pretende-se atingir objetivos fora da origem do sistema de coordenadas empregado e com trajetórias definidas por mais de um *aim point*. Nestas situações, basta efetuar uma transformação de coordenadas, que no caso da orientação consiste apenas na adição de um valor constante. Na figura 19 mostra-se uma trajetória consistindo de dois *aim points*.

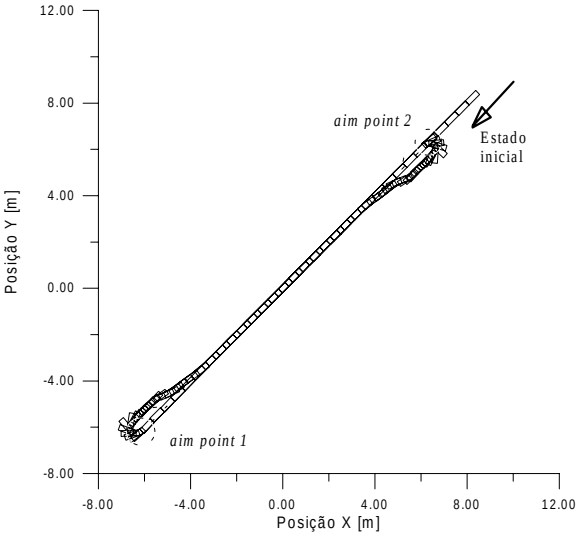


Figura 19- Trajetória consistindo de dois *aim points*.

Desta forma várias trajetórias podem ser percorridas, o que viabiliza o seguimento de uma trajetória qualquer, definida a partir de *aim points*.

5 IMPLEMENTAÇÃO EM TEMPO REAL DO SISTEMA DE GUIAGEM

Com base nas seções 3 e 4, implementou-se em tempo real o controlador de guiagem tipo *fuzzy*, de modo a se avaliar seu desempenho em situações de interesse. Foram especificadas três trajetórias, mostradas nas figuras 20 e 21, onde as amostras para efeito dos gráficos foram obtidas a cada 1s, sendo a taxa de amostragem para efeito do controle igual a 0,05s, conforme seção 4.

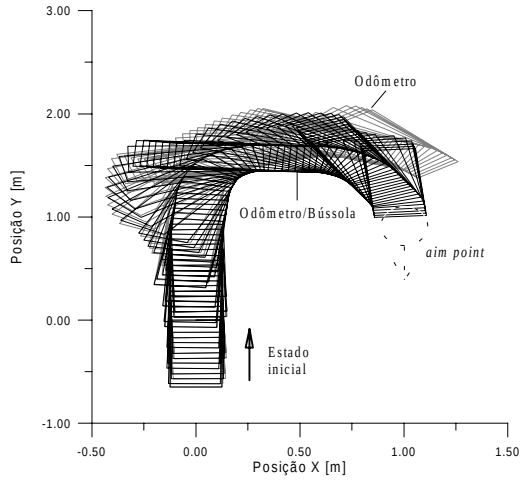
Foram efetuadas seis realizações de cada trajetória indicada nas figuras 20 e 21, calculando-se o erro médio de estado final (posição no eixo $\boldsymbol{X}$, posição no eixo $\boldsymbol{Y}$ e orientação $\boldsymbol{\phi}$) obtido pelo sistema de navegação e o erro médio obtido pelo sistema de guiagem. Considerou-se o sinal do erro sendo positivo quando o valor medido supera o valor nominal. Os resultados são mostrados na tabela 1, onde a posição é dada em metros e a orientação em graus. Para o sistema de navegação, o erro é definido em função das estimativas, de posição e orientação, e os valores efetivamente obtidos ao término da trajetória. Para o cálculo do erro no sistema de guiagem, as estimativas de posição e velocidade são substituídas pelos seus valores desejados.

Considerou-se a seguir uma trajetória #4, composta por dois *aim points*, mostrada na figura 22.

Duas fotografias relativas ao movimento do veículo são mostradas na figura 23, onde também consta o instante de tempo no qual foram adquiridas. Na figura 23a é mostrado o

Tabela 1- Erros médios de estado final considerando-se seis realizações das trajetórias nas figuras 20 e 21.

	Trajetória # 1			Trajetória # 2			Trajetória # 3		
ERRO MÉDIO DE ESTADO FINAL	X [m]	Y [m]	ϕ [°]	X [m]	Y [m]	ϕ [°]	X [m]	Y [m]	ϕ [°]
Sistema de Navegação	-0,02	0,08	3	0,02	-0,09	0,9	0,01	0,09	3,4
Sistema de Guiagem	0,027	0,08	5	0,11	0,09	3	0,11	0,04	3



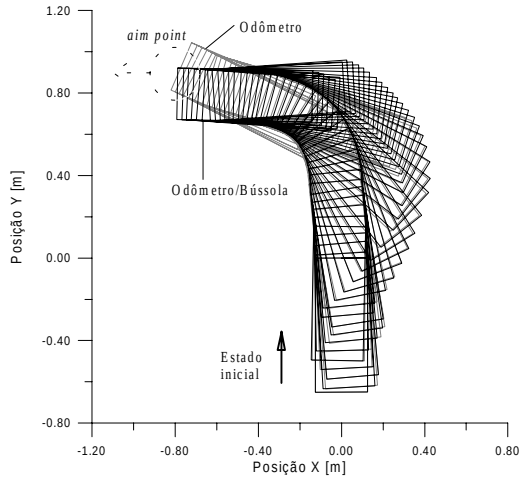
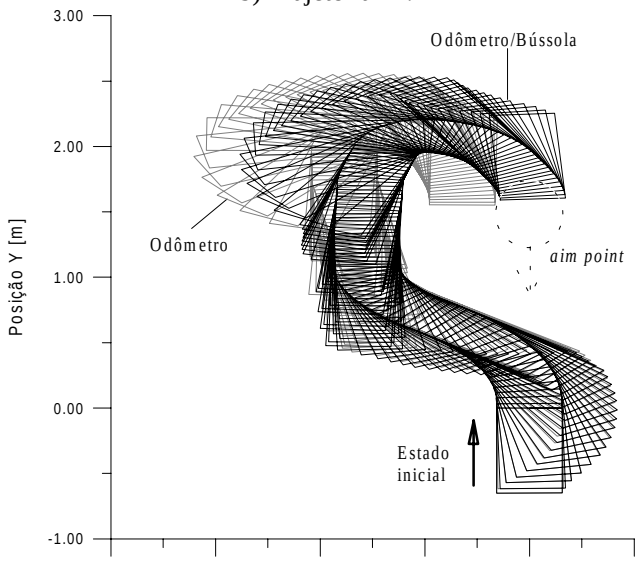


Figura 20- Trajetórias em tempo real: a) Trajetória #1; b) Trajetória #2.



veículo no ponto de partida e os dois *waypoints* definindo a trajetória desejada são marcados com setas pretas. Na figura 23b é mostrada a posição final do veículo para a trajetória # 4.

Para a realização mostrada na figura 23, calcularam-se os erros de estado final e os resultados são apresentados na Tabela 2.

Convém ressaltar que para se avaliar com precisão o erro do sistema de guiagem seria necessário utilizar outro sensor que fornecesse, com maior precisão, a posição e orientação do veículo em todo instante de tempo. Isto porque nossas conclusões nas tabelas 1 e 2 são obtidas só com base na posição final, após o critério de parada ser satisfeito. De

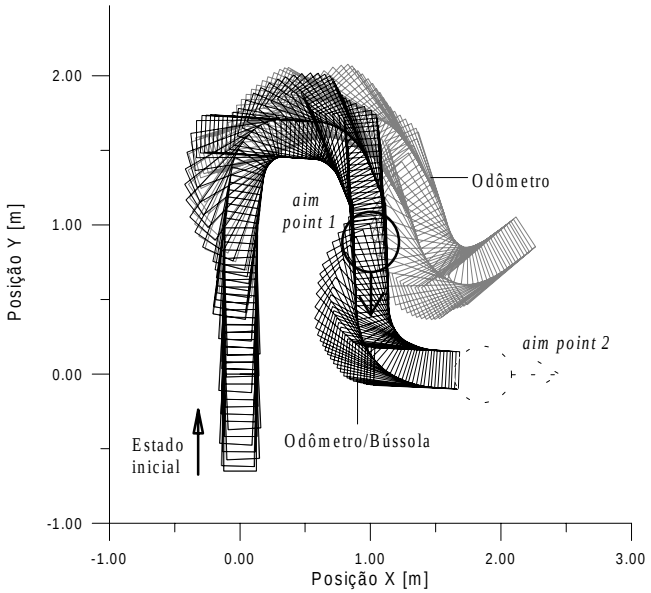
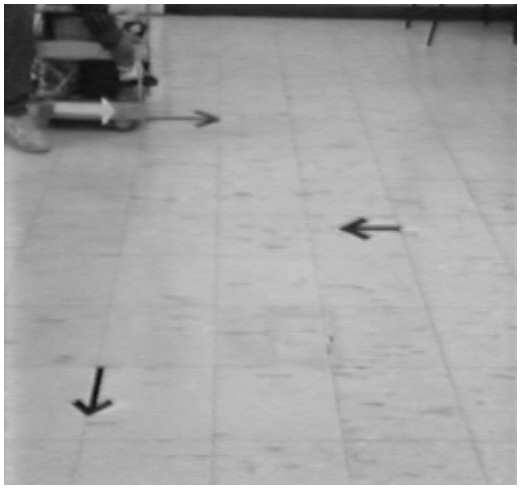
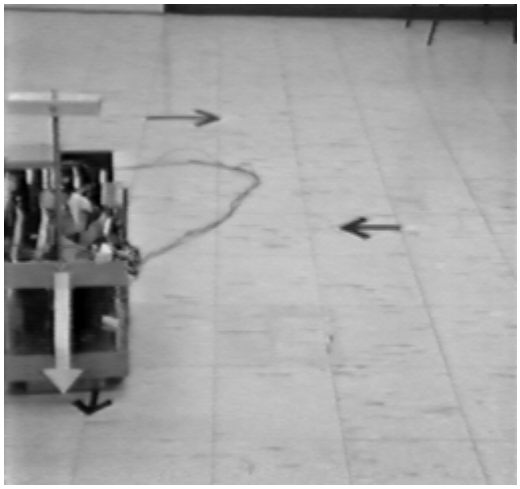


Figura 22- Trajetória #4, composta por dois *aim points*



a) Tempo = 0min, 0s.



b) Tempo = 1min, 57s.

Figura 23- Fotografias relativas à trajetória # 4, em tempo real, com dois *aim points*.

Tabela 2- Erros de estado final relativos à trajetória # 4

	Trajetória # 4		
ERRO DE ESTADO FINAL	X [m]	Y [m]	ϕ [°]
Sistema de Navegação	-0,099	0,023	2,1
Sistema de Guiagem	0,04	-0,085	3

qualquer modo, essas conclusões proporcionam uma boa aproximação dos erros reais exibidos pelo sistema de guiagem.

6 CONCLUSÕES

Um sistema para navegação e guiagem de robôs móveis autônomos foi proposto, implementado e avaliado neste trabalho. Para navegação utilizou-se integração, via filtro de Kalman, de dois sensores de baixo custo, odômetro com *encoders* e bússola digital. Esses sensores exibem períodos de amostragem diferentes, o que motivou a implementação em ambiente multitarefa, baseado no sistema operacional Linux.

A informação sobre posição e orientação obtida pelo sistema de navegação foi utilizada para se projetar um controlador tipo *fuzzy* para guiagem. A sintonização deste controlador é simples, não requer o conhecimento do modelo dinâmico do robô, e os resultados obtidos experimentalmente explicitam o bom desempenho obtido.

Para se elevar a autonomia do robô móvel e a precisão do rastreamento das trajetórias, faz-se necessário utilizar sensores mais sofisticados, como por exemplo a combinação INS+GPS. Resultados nesse sentido serão reportados oportunamente.

Agradecimentos:

Este trabalho foi em parte financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), via proc. 300158/95-5.

Os autores agradecem aos revisores pelas sugestões efetuadas.

7 REFERÊNCIAS BIBLIOGRÁFICAS

Aicardi, M.; G. Casalino; A. Bicchi and A. Balestrino (1995). Loop Steering of Unicycle-like Vehicles via Lyapunov Techniques. *IEEE Robotics and Automation Magazine*, Vol. 2, No.1, pp. 27-35.

Asami, S. (1994). Robots in Japan: Present and Future**.** *IEEE Robotics and Automation Magazine*, Vol. 1, No. 2, pp. 22-26.

Barshan, B. and H. F. Durrant-Whyte (1995). Inertial Navigation System for Mobile Robots*. IEEE Trans. Robotics and Automation*, Vol. 11, No. 3, pp. 328-342.

Borenstein, J. and L. Feng (1996). Measurement and Correction of Systematic Odometry Errors in Mobile Robots. *IEEE Trans. Robotics Automation*, Vol. 12, No. 6, pp. 869-880.

Canudas de Wit, C. and O.J. Sordalen (1992). Exponential Stabilization of Mobile Robots with Nonholonomic Constraints. *IEEE Trans. Automat. Control*, Vol. 37, Vol. 11, pp. 1791-1797.

Fuke, Y. and E. Krotkov (1996). Dead Reckoning for a Lunar Rover on Uneven Terrain. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, pp. 411-416.

Gomide, F.A.C. e R. R. Gudwin (1994). Modelagem, Controle, Sistemas e Lógica Fuzzy. *Revista Controle & Automação*, Vol. 4, No. 3, pp. 97-115.

Hemerly, E.M. e C.C. Rodrigues (1994). Guiagem de Veículos Autônomos Utilizando Sensor de Visão. *Anais do 10° Congresso Brasileiro de Automática*, Rio de Janeiro, pp. 873-878.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, Englewood Cliffs, NJ.

Lages, W.F.; E. M. Hemerly e L. F. A. Pereira (1996). Controle Linerizante de uma Plataforma Móvel Empregando Realimentação Visual. *Anais do XI Congresso Brasileiro de Automática*, São Paulo, pp. 537-542.

Mandow, A. ; J.M. Gomes-de- Gabriel; J.L. Martinéz; V.F. Muñoz; A. Ollero and A. García-Cerezo (1996). The Autonomous Mobile Robot AURORA for Greenhouse Operation. *IEEE Robotics and Automation Magazine*, pp. 18-28.

Mendel, J.M. (1995). Fuzzy Logic Systems for Engineering: A Tutorial. *Proceedings of the IEEE*, Vol. 83, No. 3, pp. 345-377.

Murata, S. and T. Hirose (1993). Onboard Locating System Using Real-Time Image Processing for a Self-Navigation Vehicle. *IEEE Trans. Industrial Electronics*, Vol. 40, No. 1, pp. 145-154.

Nascimento Jr., C.L. e T. Yoneyama (1997). *Inteligência Artificial em Automação e Controle***.** Pré-impressão, ITA, São José dos Campos-SP.

Pedrycz, W. (1989). *Fuzzy Control and Fuzzy Systems.* Research Studies, Press LTD.: Jhon Wiley & Sons Inc., Tanton, Somerset, England.

Sandi L., F.A; E. M. Hemerly e W. F. Lages (1997). Estimação em Tempo Real de Posição e Orientação de Robôs Móveis Utilizando Sensores com Diferentes Taxas de Amostragem. *Anais do 3° Simpósio Brasileiro de Automação Inteligente*, Vitória-ES, pp. 453-458.

Scherzinger, B.M.; J.J. Hutton and J.C. McMillan(1997). Low Cost Inertial/GPS Integrated Position and Orientation System for Marine Applications. *IEEE AES Systems Magazine*, May, pp. 15-19.

Schraff, R.D. (1994). Mechatronics and Robotics for Service Applications. *IEEE Robotics and Automation Magazine*, Vol. 1, No. 4, pp. 31-35.

Tanenbaum, A. (1992). *Modern Operating Systems.* Prentice-Hall, Englewood Cliffs.

Vaneck, T.W. (1997). Fuzzy Guidance Controller for an Autonomous Boat. *IEEE Control Systems Magazine*, pp. 43-51.

Vector 2X Compass Module (1996). *Application Notes*. Precision Navigation, Inc., Version 1.03.

Yamamoto, Y. and X. Yun (1994). Coordinating Locomotion and Manipulation of a Mobile Manipulador. *IEEE Trans. Automatic Control*, Vol. 39, No. 6, pp. 1326-1332.

Yodaiken, V. (1997). *Cheap Operating Systems Research and Teaching with Linux.* disponível *on-line* em http://luz.cs.nmt.edu/~rtlinux.