Tese apresentada à Pró-Reitoria de Pós-Graduação e Pesquisa do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Curso de Engenharia Eletrônica e Computação, Área de Sistemas e Controle

Cleiton Rodrigo de Oliveira

IMPLEMENTAÇÃO DE SISTEMA DE RASTREAMENTO PARA ROBÔS MÓVEIS AUTÔNOMOS

Tese aprovada em sua versão final pelos abaixo assinados:

Prof. Dr. Elder Moreira Hemerly

Orientador

Prof. Dr. Homero Santiago Maciel

Pró-Reitor de Pós-Graduação e Pesquisa

Campo Montenegro

São José dos Campos, SP - Brasil

2007

Dados Internacionais de Catalogação-na-Publicação (CIP)

Divisão Biblioteca Central do ITA/CTA

Oliveira, Cleiton Rodrigo de

Implementação de Sistema de Rastreamento para Robôs Móveis Autônomos / Cleiton Rodrigo de Oliveira.

São José dos Campos, 2007.

123f.

Tese de Mestrado – Curso de Engenharia Eletrônica e Computação. Área de Sistemas e Controle – Instituto Tecnológico de Aeronáutica, 2007. Orientador: Prof. Dr. Elder Moreira Hemerly. .

1. Robôs Móveis. 2. Controle. 3. Sensor Visual. I. Centro Técnico Aeroespacial. Instituto Tecnológico de Aeronáutica. Divisão de Sistemas e Controle. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

OLIVEIRA, Cleiton Rodrigo de. **Implementação de Sistema de Rastreamento para Robôs Móveis Autônomos**. 2007. 123f. Tese de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

Cleiton Rodrigo de Oliveira

Av. Nelso Dávila, 443

CEP 12.245-020 – São José dos Campos–SP

IMPLEMENTAÇÃO DE SISTEMA DE RASTREAMENTO PARA ROBÔS MÓVEIS AUTÔNOMOS

Cleiton Rodrigo de Oliveira

Composição da Banca Examinadora:

Prof. Dr.	Roberto Kawakami Harrop Galvão	Presidente	-	ITA
Prof. Dr.	Elder Moreira Hemerly	Orientador	-	ITA
Prof. Dr.	Marcelo Carvalho Minhoto Teixeira	Membro Externo	-	UNESP
Prof. Dr.	Jackson Paul Matsuura	Membro	-	ITA
Prof. Dr.	Cairo Lúcio Nascimento Jr	Membro	-	ITA

ITA

A todos aqueles que de alguma forma participaram da construção desse trabalho, seja com idéias profissionais renovadoras ou com palavras pessoais de otimismo e perseverança.

Agradecimentos

À Deus e seus enviados, pela ajuda invisível nos duros momentos de dificuldade e por serem a fonte de esperança para prosseguir.

À Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP) pelo auxílio concedido em forma de bolsa de Mestrado.

À minha família, por terem não só me mostrado o caminho mas também me ensinado a trilhá-lo.

À minha namorada, por ter sido, muitas vezes, o colo amigo que serviu de apoio para vencer o desânimo.

Aos meus mais que amigos, irmãos, que já há mais de sete anos não têm medo de desembainhar suas espadas e lutar ao meu lado em qualquer batalha.

E aos colegas de mestrado, pelas intermináveis resenhas que possibilitavam fugas da realidade diminuindo o peso diário do trabalho.

"Só pode reconhecer a luz
quem tiver conhecido a escuridão."
— P. Z.

Resumo

No presente trabalho apresenta-se o estudo de duas diferentes abordagens relacionadas a utilização de sensores visuais na robótica móvel. Na primeira delas as imagens são usadas para fornecer parâmetros que permitam determinar a postura do próprio veículo durante o rastreamento de uma trajetória preestabelecida. Na segunda abordagem, os parâmetros fornecidos pelo processamento das imagens permitem calcular dados sobre a posição relativa entre o objeto imageado e o robô com o objetivo de seguir alvos móveis.

No caso do rastreamento de trajetória, utiliza-se a lei de controle cinemático proposta em [1] em simulações realistas com os modelos cinemático e dinâmico do robô móvel Magellan Pro. Obtém-se as informações de postura do robô, com base na comparação entre dois conjuntos de imagens de três pontos em um plano de referência. O primeiro conjunto de imagens é formado por fotografias feitas durante a condução humana e representa a trajetória a ser rastreada. Já o segundo conjunto de imagens é adquirido durante o rastreamento, e cumpre o papel de fornecer ao sistema de controle a postura atual do robô.

Para obter as informações de posição e orientação, extraindo-as de ambos os conjuntos de imagens, utilizam-se técnicas de decomposição de homografias e reconstrução euclidiana propostas por [2]. Por fim, para aproximar mais da realidade os resultados das simulações

explora-se ainda o comportamento do controle na presença de ruído.

Com relação ao segundo tema abordado (rastreamento de alvos móveis), utiliza-se o algoritmo de detecção visual por retroprojeção de histogramas [3] e [4] em conjunto com duas leis de controle. Inicialmente, para garantir a acurácia da técnica de detecção visual, simula-se a mesma estratégia de controle tipo *fuzzy* implementada em [4]. Adicionalmente, para avaliar o funcionamento do algoritmo de detecção visual em conjunto com outras leis de controle, simulações foram realizadas utilizando o controle cinemático apresentado em [5], obtido por meio de técnicas baseadas na análise de estabilidade segundo Lyapunov.

Os bons resultados de simulação obtidos motivaram a implementação da lei de controle proposta em [5] no robô móvel Magellan Pro, em conjunto com a detecção visual baseada em histogramas. Para tanto, foram desenvolvidas rotinas em C++ executadas no veículo, contendo o processamento de imagens e o controle. Implementou-se também uma rotina remota para interagir com o usuário na etapa de seleção manual do alvo, necessária para inicializar o algoritmo de detecção visual. Os resultados obtidos comprovaram o bom funcionamento dessa lei de controle operando em conjunto com o processamento de imagens proposto.

Abstract

This work investigates two different approaches for using visual sensor in mobile robots. In the first one the images are used to provide parameters to determinate the own vehicle pose during a prerecorded trajectory tracking. In the second one, the parameters obtained by processing the images allows to calculate information about the relative position between the object and the robot.

In the trajectory tracking problem, the kinematic control law proposed in [1] is used to produce realistic simulation results using kinematic and dynamic Magellan Pro models. Information about robot pose are obtained by comparing two image sets of three points in a reference plane. The first image set is composed by prerecorded pictures taken during a human driven trajectory. This image set represents the reference trajectory. The second image set is acquired during the tracking, and is used to give the actual pose parameters to the control system.

The information about position and orientation, are extracted from both image sets, using homography decomposition and euclidian reconstruction proposed in [2]. More realistic results are obtained by using noise in the simulations.

In the second approach, concerning mobile target tracking, the visual detection algorithm based on histogram backprojection [3] and [4] is used in conjunction with two other

control laws. Initially, in order to evaluate if the visual detection technic implementation is correct, the fuzzy *control strategy in [4] is simulated. Aiming at exploring the behavior of this visual detection with other control laws, the Lyapunov-based one presented in [5] employed.*

The good simulation results motivated the real-time implementation, by combining the visual detection algorithm with the control law proposed in [5]. With this goal, C++ routines where developed to run in the vehicle, containing the image processing algorithm and the control law implementation. One remote routine is also developed, for interfaces with the user during the first interaction of the visual detection algorithm. The obtained results corroborate the performance of this control law when employing the proposed image processing.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O crescente interesse por robôs autônomos se deve à grande diversidade de atividades que podem ser realizadas pelos mesmos, tais como em tarefas de transporte em fábricas, em aplicações hospitalares, domésticas e até de vigilância [7], [8]. Para tanto, um grande esforço de pesquisa tem sido dedicado na última década para se permitir que robôs móveis tenham a capacidade de se mover de forma autônoma em ambientes pouco estruturados e sem a necessidade de supervisão.

Nesse contexto há dois temas básicos que são de especial interesse: a determinação da posição e orientação do veículo em um dado instante de tempo (postura), e o controle, que se preocupa com o rastreamento de uma trajetória (ou alvo móvel), ou com a regulação em torno de uma certa postura.

No que concerne à determinação de postura problemas associados ao *drift* odométrico têm estimulado a utilização de sensores adicionais (fusão de sensores), ou sensores que sozinhos possam fornecer maior quantidade de informações (sensores visuais). Da mesma forma, a dificuldade em se modelar determinadas dinâmicas desses veículos dificulta a utilização de técnicas de controle convencionais favorecendo assim o estudo daquelas que independam (lógica *fuzzy* e redes neurais), ou desprezem (controle cinemático) essas dinâmicas na etapa de projeto.

Nesse contexto, faz-se a seguir uma breve apresentação sobre algumas das abordagens apresentadas na literatura para a solução desses problemas, destacando aquelas que fazem uso de sensores visuais, aliados a controles tipo *fuzzy* ou controle cinemático.

1.1 Algumas Abordagens Para a Determinação de Postura de Robôs Móveis

O procedimento mais simples para se determinar a postura de um robô móvel consiste no uso do denominado sistema *dead reckoning* (odômetro), que basicamente equivale a se efetuar a integração dos pulsos provenientes dos encoders disponíveis nas rodas do robô [9]. Esta abordagem apresenta algumas deficiências, como por exemplo, a necessidade de se predefinir a trajetória e a existência de erros acumulativos (*drift*) na determinação da postura, devido à assimetria das rodas e eventuais escorregamentos.

As deficiências anteriores podem ser removidas mediante a utilização de sensores adicionais que compensem a característica de *drift* do odômetro. Em [10], por exemplo, é considerado o problema de estimação de postura integrando-se a informação de orientação fornecida por um odômetro com a informação de uma bússola digital mediante filtro de Kalman.

Outra abordagem possível, é a utilização de sensores que forneçam maior quantidade de informação. Nesse caso, um sensor muito relevante é o baseado em visão, embora tipicamente exija grande capacidade computacional no processamento das imagens. Isso porque, as imagens oriundas da câmera precisam ser processadas para fornecer os parâmetros necessários para o controle, estimulando portanto, a busca de técnicas que tornem esse

processamento mais rápido. Como exemplo, tem-se [11] onde é proposto um algoritmo baseado em filtro de Kalman para se estimar os coeficientes de uma guia de navegação, obtida via processamento de imagem. O método denominado KIF (*Kalman Information Filter*) é comparado com o método usual denominado WRLS (*Weighted Recursive Least Squares*) tendo sido mostrado que há um ganho considerável no tempo de processamento.

Ainda no que se refere ao uso de sensores visuais, duas formas distintas de utilização têm sido abordadas, quais sejam: câmera *off board*, onde o dispositivo visual observa externamente o comportamento do robô enviando informações para o sistema de controle, ou câmera *on board*, onde a câmera se move junto com o robô visualizando o caminho por onde o mesmo se move. Esse segundo tipo de abordagem é também conhecida como câmera *in hand*.

Uma aplicação relativamente recente do primeiro tipo de abordagem é apresentado em [12], onde apenas uma câmera é utilizada para fazer comparações entre imagens do robô em situações diferentes. O objetivo é fazer com que o robô repita uma trajetória previamente gravada, fazendo com que as imagens da trajetória pré-gravada e atual sejam iguais.

Já em [13], duas câmeras são utilizadas na construção de um sistema para rastreamento e predição de trajetória de um robô aéreo baseado em câmeras digitais de baixo custo. São utilizadas técnicas de processamento de imagens, além de algoritmos de filtragem e predição para determinar a localização e orientação de um dirigível em miniatura, durante vôos *indoor*. Tal localização é obtida por meio do rastreamento de um arranjo de emissores infravermelho, anexos ao dirigível, e que permitem a correspondência entre os pontos captados pelas duas câmeras em terra. Além disso, as incertezas do ambiente são compensadas via Filtro de Kalman.

Entretanto, para ambientes não estruturados a manutenção da autonomia dos robôs requer métodos que independam da prévia instalação de câmeras no ambiente onde o veículo vier a operar. Nesse tipo de aplicação torna-se mais interessante o estudo de técnicas de controle com câmera *in hand*. É o caso do trabalho apresentado em [14], onde é desenvolvido o controle de um robô móvel autônomo que servirá de protótipo para uma futura cadeira de rodas inteligente. As informações a respeito do ambiente são recebidas através de uma câmera digital e sensores infravermelho. O objetivo da câmera é identificar o ambiente e, principalmente, reconhecer quando o robô estiver em um corredor estreito.

Reiterando o que foi dito com relação a ambientes não estruturados, a utilização da abordagem camera *in hand* foi utilizada na concepção da sonda *Pathfinder* que esteve em Marte em 1997 [15] e de suas sucessoras mais recentes *Spirit* e *Opportunity* muito difundidas na mídia. Nesse sentido, um dos fatores limitantes era o tempo de atraso na comunicação com os robôs, além do alto custo dessas transmissões, sendo então desejável que o robô tivesse uma autonomia ainda mais elevada. Para tanto, [15] propõe um método de controle chamado *Mobile Camera-Space Manipulation* (MCSM), que necessita de apenas um ciclo de transmissão de dados para que o robô seja capaz de realizar as manobras necessárias ao posicionamento do manipulador. O planejamento da trajetória é feito com base nas coordenadas determinadas usando as informações visuais de duas câmeras embarcadas.

Outra abordagem no que concerne ao processamento de imagens é mostrada em [16] onde é desenvolvido o controle de robô autônomo empregando técnicas de inteligência artificial baseadas em comportamento. Através de diversos módulos o robô é capacitado a desenvolver tarefas e comportamentos em diversos níveis. A navegação é proporcionada por um sistema visual de reconhecimento de pontos de referência. A busca por esses pontos

é implementada com algoritmos genéticos de reconhecimento de padrões em imagens digitais.

Além desses, pode-se citar ainda [1], por exemplo, onde um processo semelhante a [12] é utilizado para rastrear uma trajetória pré-gravada, e [4], no qual é feita a implementação de um sistema de rastreamento de alvo para robótica móvel baseado em informações obtidas por sensor visual. Dentro desse contexto, percebe-se a relevância da continuação do estudo e desenvolvimento de técnicas de utilização de sensores visuais no controle de robôs autônomos.

1.2 Controle de Robôs Móveis - Rastreamento de Trajetória e de Alvos Móveis

Uma vez definidas a posição e orientação do robô móvel é necessária a ação de um controle para que os objetivos do veículo, de acordo com a aplicação, sejam alcançados. Dessa forma, a metodologia escolhida deverá levar em conta não só as características do veículo (robô), como também o objetivo do controle.

As técnicas tradicionais de controle geralmente precisam do conhecimento da dinâmica do processo a ser controlado, que nesse caso são tipicamente difíceis de modelar. Nesse caso uma possível solução é a utilização de controladores tipo *fuzzy*. É o caso, por exemplo, dos trabalhos apresentados em [4], [10], [14] e [16].

Para o desenvolvimento desse tipo de controlador, basicamente o que se requer é que os comandos de controle, com base na experiência heurística do operador humano, sejam expressos em variáveis lingüísticas, regras de produção e condições do tipo: se *condição 1*

e *condição 2*, então *ação*, [4], tornando-o relativamente simples de ser implementado.

Outra metodologia possível é a apresentada em [1] e [12], nos quais apenas o modelo cinemático do veículo é utilizado, desprezando-se as dinâmicas envolvidas. Isso equivale a considerar que o veículo pode ser tratado como um ponto material, abordagem que tem seu resultado tão melhor quanto essa consideração se aproximar da verdade. Nesses casos, utiliza-se o método de Lyapunov para desenvolver leis de controle que garantam a anulação dos erros de postura do veículo em relação a uma dada referência.

Em complemento a essa técnica, pode-se ainda utilizar o método implementado em [5], em que o controle cinemático é auxiliado por uma rede neural *feedforward* que "aprende" a compensar as dinâmicas não modeladas somando uma componente ao controle cinemático.

Já com relação ao objetivo do controle tem-se observado basicamente duas vertentes. Uma que visa o rastreamento de uma trajetória pré-determinada, [1] e [12], e outra que procura seguir uma trajetória determinada em "tempo real". É o caso de [15], no qual a trajetória é determinada de acordo com restrições do manipulador (braço robótico) do veículo e de [4], que se presta a seguir um alvo móvel.

Para esse tipo de problema, o controle deve levar em conta apenas a posição relativa entre o alvo e o robô, não sendo necessário portanto, o conhecimento da posição absoluta do veículo. Caso a lei de controle necessitasse dessa informação uma outra câmera (*off board*) ou outro sensor seriam necessários, sacrificando entretanto a questão da autonomia.

Nesse cenário, em que muitos outros trabalhos também poderiam ser citados tais como [17], [18], [19], [20],[21], [7],[22], [23],[24], e [25], o estudo de técnicas capazes de unir visão computacional com os controles mencionados parece bastante relevante. Nos capítulos seguintes será feita uma explanação mais aprofundada no que diz respeito às técnicas

de processamento de imagem e metodologias de controle utilizadas nesse trabalho com o objetivo de rastrear um alvo móvel. A figura 1.1 resume os principais aspectos o problema de robótica móvel discutidos nesta seção.

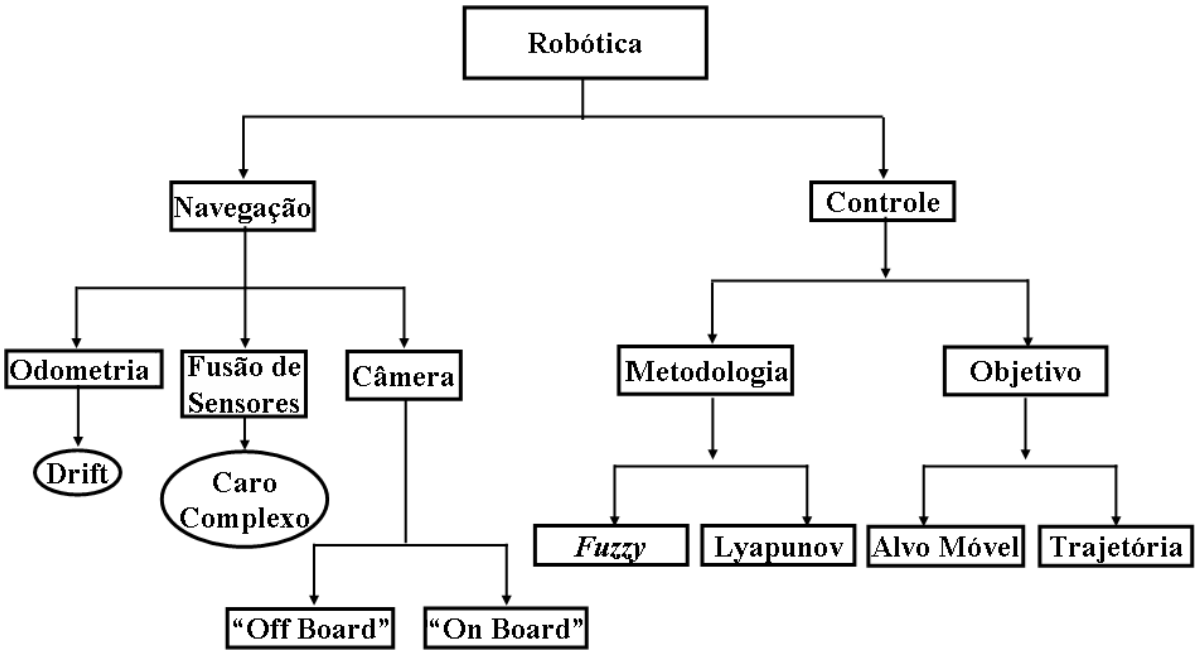

Figura 1.1 – Principais aspectos do problema de navegação e controle de robôs móveis.

Assim, as principais contribuições são.

1. Obtenção de resultados de simulação com modelo cinemático, utilizando um algoritmo de rastreamento de trajetórias baseado em informações de sensores visuais.

2. Aplicação dessa estratégia de controle ao modelo dinâmico do robô Magellan Pro obtendo resultados de simulação mais realistas, uma vez que tipicamente na literatura sobre tal problema apenas o modelo cinemático é utilizado;

3. Obtenção de resultados de simulação fundindo as técnicas de controle cinemático de [5] com a detecção visual por intersecção de histogramas [4], para o rastreamento de alvo móvel.

4. Obtenção de resultados práticos implementados no robô Magellan Pro, relativos ao rastreamento de alvos citado no item 3. Nesse caso foi avaliado o comportamento do controle em trajetórias de interesse, concluindo-se pelo bom desempenho da técnica;

2 Processamento de Imagem para Rastreamento de Alvos Móveis

A realimentação visual é capaz de fornecer uma imensa quantidade de informações que permitem compensar erros provenientes de outros sensores, tais como *drifts* odométricos. As imagens devem ser processadas para que as informações sejam extraídas e convertidas de tal forma a possibilitar a tomada de decisões para alcançar a referência desejada.

Assim, se estabelece uma estrutura como a mostrada na Figura 2.1, com base em [4], na qual a primeira etapa num processo de controle usando sensor visual é a aquisição da imagem. Nessa etapa, o principal "agente" é o sensor (câmera), responsável por capturar as imagens e disponibilizá-las para o processamento.

Na segunda etapa, o processador executa os algoritmos nele armazenados usando como entrada as imagens obtidas pela câmera, para que delas possam ser extraídas as informações relevantes. Particularmente no caso do presente trabalho, assim como em [4] o interesse é determinar a posição do alvo na imagem, informação que será usada pelo controlador para estabelecer a atitude do robô.

Obtidos os parâmetros da imagem, o controlador é acionado. Aqui, as limitações físicas do sistema e do controlador devem ser respeitadas na ocasião do projeto, uma vez que,

controles exageradamente grandes podem danificar o processo (queima de componentes, por exemplo) e pequenos demais podem não vencer possíveis atritos ou inércia do sistema [4].

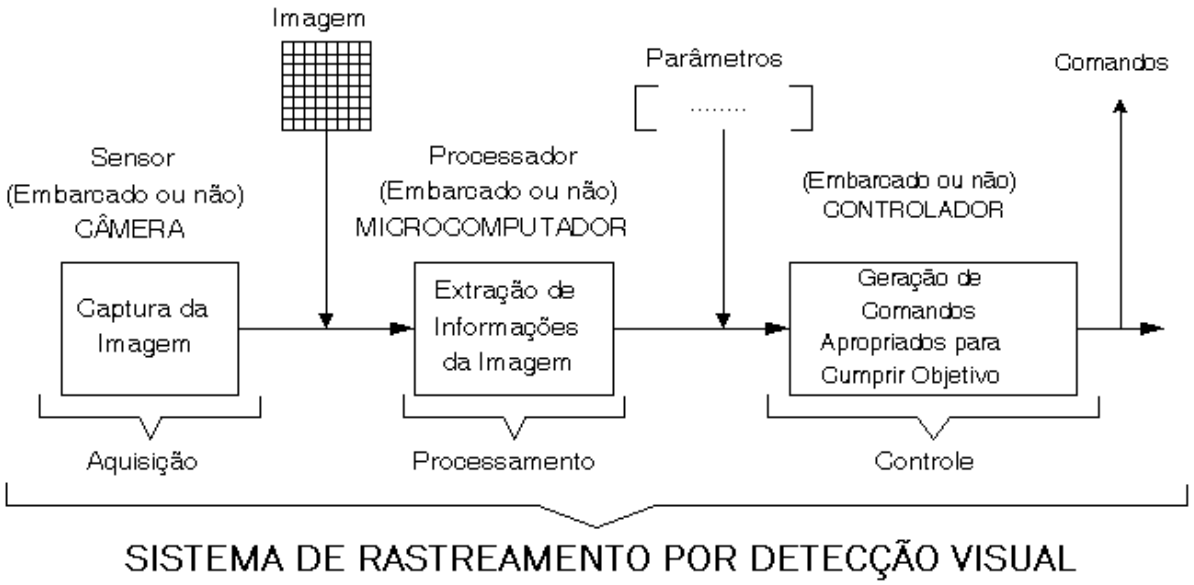

Figura 2.1 – Esquema geral de um sistema de rastreamento usando sensor visual de acordo com [4].

Dentro desse contexto, mostra-se nesse capítulo um breve relato a respeito de algumas técnicas de processamento de imagens utilizadas no rastreamento de alvos móveis, em especial aquelas que foram implementadas neste trabalho. Serão mostrados os conceitos de matrizes de rotação, de cores no sistema RGB e como o software MatLab trabalha com imagens. Por fim, apresenta-se o algoritmo de detecção por intersecção de histogramas.

2.1 Matrizes de Rotação

Uma vez que a imagem foi obtida pela câmera e a primeira etapa do processamento calcula a posição do alvo na cena imageada torna-se necessário converter essas coordenadas para o sistema do mundo real. Isso porque, a imagem obtida é apenas uma projeção da

cena real conforme mostra a Figura 2.2.

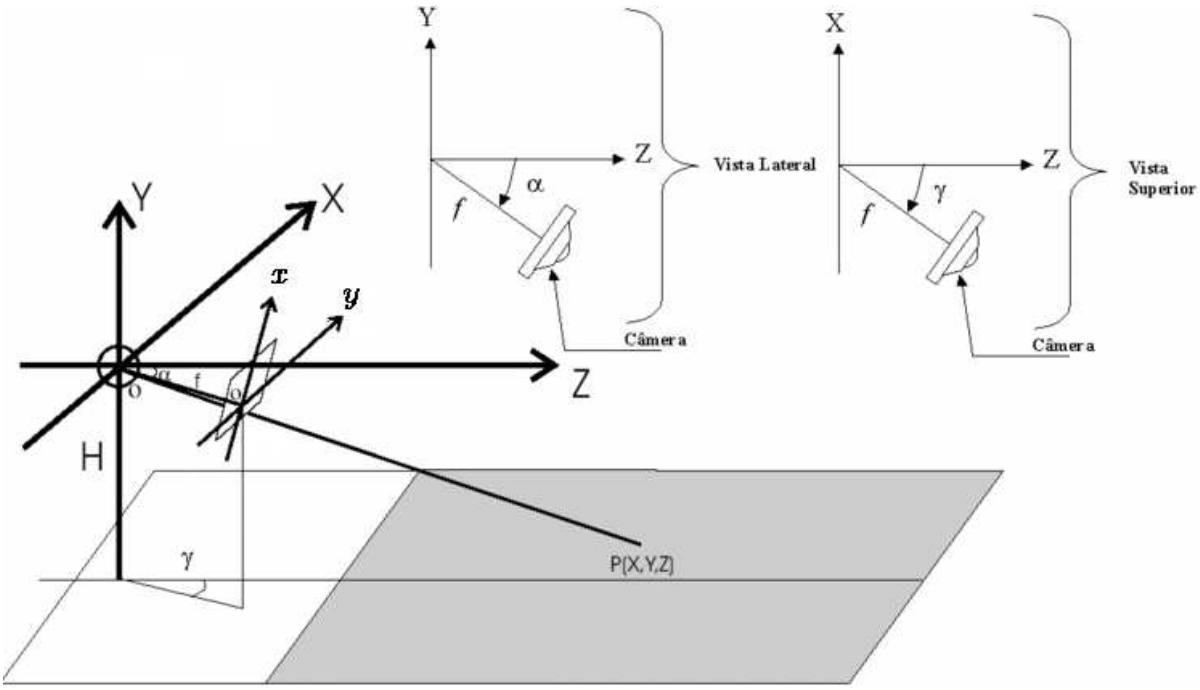

Figura 2.2 – Sistemas de coordenadas tipicamente utilizado em robótica móvel.

Dessa forma, considerando os graus de liberdade de movimento da câmera na vertical (*tilt*) e na horizontal (*pan*), e sob a hipótese do conhecimento da medida H (altura da câmera com relação ao solo), pode ser mostrado (vide [26] que a rotação entre os eixos da imagem e o mundo real é dada por

$$R = \begin{bmatrix} \cos(\gamma) & 0 & -\sin(\gamma) & 0 \\ \sin(\gamma)\sin(\alpha) & \cos(\alpha) & \sin(\alpha)\cos(\gamma) & 0 \\ \sin(\gamma)\cos(\alpha) & -\sin(\alpha) & \cos(\alpha)\cos(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

em que α é o ângulo de *tilt* e γ é o ângulo de *pan* da câmera.

Por meio da matriz R, pode-se então determinar as expressões diretas que dão a conversão entre as coordenadas do alvo na imagem para o mundo real. Pode ser mostrado

(vide [4]) que esse mapeamento é dado por

$$Z = Y\frac{y\sin(\alpha)\cos(\gamma) - x\sin(\gamma) + f\cos(\alpha)\cos(\gamma)}{y\cos(\alpha) - f\sin(\alpha)} \tag{2.2}$$

$$X = \frac{xY\sin(\alpha) - Z(x\cos(\alpha)\cos(\gamma) + f\sin(\gamma))}{-x\sin(\gamma)\cos(\alpha) + f\cos(\gamma)} \tag{2.3}$$

em que, x e y são as coordenadas do centro de massa do alvo na imagem; α é o ângulo de *tilt* da câmera; γ é o ângulo de *pan* da câmera; f é a distância focal da câmera e X, Y e Z são as coordenadas do alvo no mundo real.

Dessa forma, dados os valores de x e y do centro de gravidade do alvo na imagem, da distância focal da câmera f e além disso, conhecendo-se a altura da câmera em relação ao solo H (da Figura 2.2) é possível calcular X e Z, coordenadas reais do alvo em relação ao centro óptico da câmera do robô. Note-se que neste caso o valor H equivale à coordenada Y do alvo no mundo real e de acordo com os eixos coordenados mostrados na Figura 2.2 tem sinal negativo.

2.2 Sistema RGB

Os objetos no mundo real podem ser vistos por nossos olhos porque refletem a luz que recebem de fontes externas tais como as lâmpadas, velas ou mesmo o Sol. De acordo com a cor de cada objeto a parte dessa luz refletida varia ou dito de outra forma, diferentes comprimentos de onda de luz são refletidos por objetos de diferentes cores. As cores vermelha, verde e azul são então consideradas cores primárias e com base nelas pode-se formar todas as outras cores "misturando-se" os comprimentos de onda de cada uma.

Da mesma forma, é no momento que a luz refletida pelos objetos passa pela lente de uma câmera e chega até o sensor que as informações a respeito desse objeto podem ser obtidas, sendo a imagem, como dito anteriormente, tratada por um processador, a fim de que tais parâmetros possam ser calculados. Isso leva à necessidade de se definir uma maneira para tratar cores por meio de processadores digitais.

Nesse contexto, existem várias formas de representações de cores que podem ser utilizadas no processamento digital de imagens. Em especial, [4] utiliza o sistema RGB, onde as cores do mundo real são tratadas pelo computador como sendo vetores num espaço tridimensional de base (R,G,B), como mostrado na Figura 2.3. Assim, qualquer cor pode ser representada como combinação linear desses três vetores, projetando-se nesse modelo o comportamento da natureza.

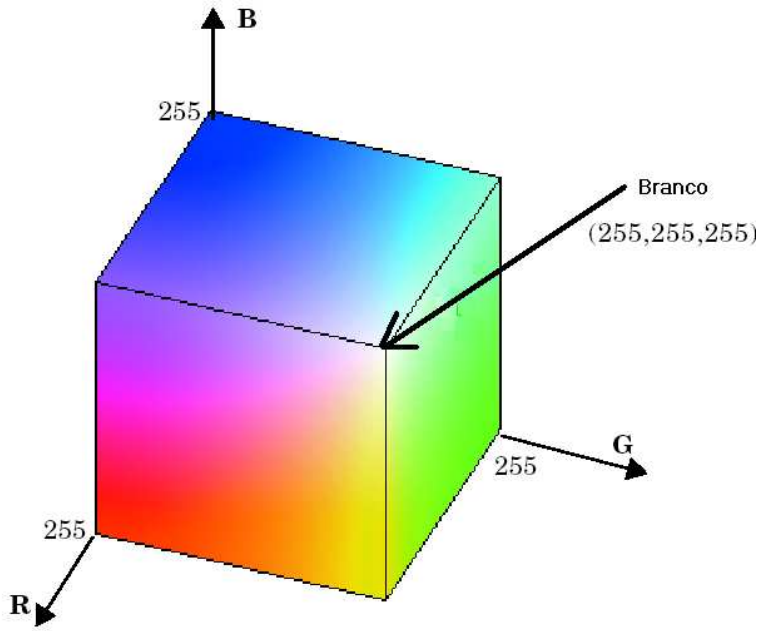

Figura 2.3 – Sistema RGB representado como um espaço vetorial.

Uma vez que se represente as cores no sistema RGB, a quantidade de cores reais que poderão ser reproduzidas pelo sistema de processamento será limitada pelo número de bits que podem ser tratados simultaneamente pelo processador. Isso porque utilizando

esse modelo, cada pixel de uma imagem é tratado como um vetor tridimensional. Nesse caso, um processador capaz de trabalhar com imagens de 24 bits, ou seja cores que vão de (0,0,0) a (255,255,255), estará capacitado a manipular 16.777.216 cores, sistema conhecido como *true color*.

2.2.1 Tratamento de Imagens usando o MatLab

O software MatLab, utilizado para a realização das simulações no presente trabalho, é capaz de operar com figuras em diversos formatos [6]. As imagens coloridas, em especial, podem ser tratadas de duas formas: como imagens indexadas ou como imagens RGB. No primeiro tipo, duas matrizes são geradas para representar a imagem. Uma delas, X por exemplo, tem na posição de cada pixel um número inteiro que representa uma linha da segunda matriz. Essa segunda matriz, a qual é chamada de *mapa*, por sua vez, é de dimensão m x 3 e cada uma de suas linhas representa uma cor no sistema RGB. Assim, diz-se que a matriz X mapeia as cores da imagem no *mapa*.

A segunda forma utilizada pelo MatLab para tratar imagens coloridas é o chamado tipo RGB. Nesse caso, cada imagem é tratada como uma matriz de dimensão m x n x 3, como mostrado na Figura 2.4. Nesse caso, o mapa não é necessário pois cada *pixel* já traz consigo a informação a respeito da intensidade de sua própria cor.

No presente trabalho o formato escolhido foi RGB, uma vez que, nesse caso, o algoritmo de detecção visual utilizado torna-se mais simples de ser implementado, devido a características construtivas da câmera. Na Seção 2.3 será mostrado como os conceitos expostos até aqui são utilizados na detecção de alvos móveis.

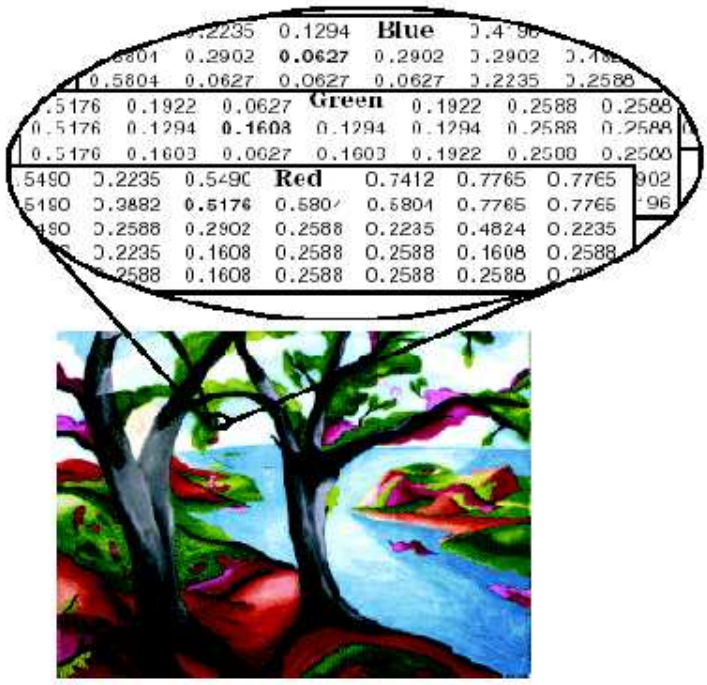

Figura 2.4 – Sistema RGB na forma como é utilizado no MatLab [6].

2.3 Algoritmo Adaptativo de Detecção Visual por Intersecção de Histogramas

O algoritmo de intersecção de histogramas para rastreamento de alvos móveis foi proposto por [3] e reproduzido com muito sucesso por [4]. A principal razão deste sucesso está na constante atualização do modelo do alvo que é capaz de compensar variações na iluminação do ambiente.

Sua utilização se justifica, em contrapartida a outras técnicas mais simples se for requerido que o sistema de visão trabalhe em cenas naturais, onde informações quanto a cores são usualmente mais realizáveis e flexíveis. Além disso, essa é uma técnica utilizada para reconhecimento de objetos coloridos, especialmente quando é necessária a localização de um alvo entre vários de uma base de dados [3].

O processo se inicia com a aquisição da imagem de uma cena que contenha o alvo, a

qual será chamada aqui de **imagem original**. Nessa imagem, o alvo deve ser selecionado manualmente pelo usuário, mostrando para o robô o que ele deverá seguir.

A imagem original é então percorrida, e calcula-se a intensidade de cor de cada pixel, utilizando para isso a Equação 2.4, em que N é o valor máximo que cada componente de cor pode atingir. Esse cálculo permite representar todas as intensidades de cores do sistema RGB em um gráfico unidimensional, diferentemente do que fora mostrado na Figura 2.3.

$$i = R + G * N + B * N^2 \tag{2.4}$$

A partir daí, o número de pixels que apresenta uma mesma intensidade de cor é contado e gera-se o histograma da imagem, ou seja, um gráfico unidimensional que contém informações a respeito da imagem original.

Em seguida, a imagem do alvo que havia sido selecionada manualmente passa por processo semelhante, porém, o histograma gerado é calculado com base na Equação 2.5 uma vez que segundo [3], para alvos monocromáticos coloridos sujeitos a poucas variações de iluminação o modelo de distribuição Gaussiana é uma boa aproximação. Esse histograma é chamado de histograma modelo.

$$h(V) = \frac{1}{(2\pi)^{-\frac{3}{2}}[det(P)]^{\frac{1}{2}}} e^{-\frac{1}{2}((V-M)^T P^{-1}(V-M))} \tag{2.5}$$

em que, $V = [r \quad g \quad b]^T$ é o vetor das cores para cada pixel, $M = [\mu_r \quad \mu_g \quad \mu_b]$ é o vetor das médias de cada cor na imagem e P é a matriz de covariâncias das cores na imagem em relação ao vetor das médias.

Neste ponto faz-se necessário explicar como a Equação 2.5 produz um gráfico bidimensional para ser comparado com o histograma produzido na primeira etapa do algoritmo. Para tanto utiliza-se a Equação 2.4 para indexar cada vetor V das cores dos pixels. O resultado final seria então melhor representado pela notação do Cálculo para funções compostas $H(i(V))$.

Terminada essa etapa tem-se dois histogramas, o da imagem original e o histograma modelo do alvo. Na etapa subseqüente tais histogramas são submetidos a um processo chamado intersecção de histogramas, que nada mais é do que uma divisão das raias do histograma modelo pelas equivalentes no histograma da imagem original.

Uma vez concluída a intersecção de histogramas, faz-se uma nova varredura na imagem calculando-se a intensidade de cor para cada pixel e em seguida, escolhe-se o mínimo entre o resultado dessa operação e 1, gerando uma nova matriz do mesmo tamanho da imagem, porém com valores de intensidade menores ou iguais a 1. Assim, o resultado obtido terá valores próximos de 1 para pixels cujas intensidades de cor, na imagem original, tenham maior probabilidade de pertencer ao alvo. Na verdade o que se faz é um "espalhamento" do resultado da intersecção sobre a imagem original, com o objetivo de reconstruir a posição do alvo. A matriz assim formada recebe o nome de **imagem retroprojetada** ($B(x, y)$).

A imagem retroprojetada é então convoluída com uma máscara unitária (matriz de uns) de mesmo tamanho da janela selecionada manualmente pelo usuário. O resultado dessa convolução (matriz $C(x, y)$) apresentará um pico aproximadamente sobre o centro de massa do alvo, terminando portanto a localização do mesmo. Nesse ponto, basta calcular as coordenadas Z na Equação 2.2 e em seguida X na Equação 2.3 para que se conheça a posição do alvo em relação ao robô.

Embora o alvo já tenha sido localizado deseja-se ainda um processo iterativo e autônomo.

Para isso o algoritmo propõe a adaptação do modelo do alvo, compensando assim variações que possam ocorrer na iluminação do ambiente. Isso é feito usando o fato de que pixels com maiores intensidades na matriz $C(x, y)$ têm maior probabilidade de pertencer ao alvo. Assim, pixels com intensidades maiores que um determinado valor limite T (em [3] é usado $T = 15$), escolhido de acordo com o nível de ruído a ser rejeitado são mapeados em 1 enquanto os demais são mapeados em 0.

Isso gera uma matriz $(D(x, y))$ que funciona como máscara para recalcular o modelo do alvo sem que seja necessário uma nova seleção manual pelo usuário. Os novos parâmetros do alvo M^+ e P^+ são calculados de acordo com

$$M^+ = \frac{1}{N^+} \sum_{x=1}^{X_{max}} \sum_{y=1}^{Y_{max}} D(x, y)C(x, y)f(x, y) \tag{2.6}$$

$$N^+ = \sum_{x=1}^{X_{max}} \sum_{y=1}^{Y_{max}} D(x, y)C(x, y) \tag{2.7}$$

$$D(x, y) = \begin{cases} 1, & se \quad C(x, y) > T \\ 0, & caso \quad contrrio \end{cases} \tag{2.8}$$

$$P^+ = \frac{1}{N^+ - 1} \sum_{x=1}^{X_{max}} \sum_{y=1}^{Y_{max}} [D(x, y)C(x, y)][f(x, y) - M^+][f(x, y) - M^+]^T \tag{2.9}$$

em que, V^+ é o total de pixels com maior probabilidade de pertencer ao alvo, M^+ é a nova média das cores do alvo, $C(x, y)$ é a imagem convoluída e $f(x, y)$ é a nova imagem adquirida no instante subseqüente ao primeiro.

Concluída essa etapa pode-se retomar a Equação 2.5 e recalcular o histograma modelo

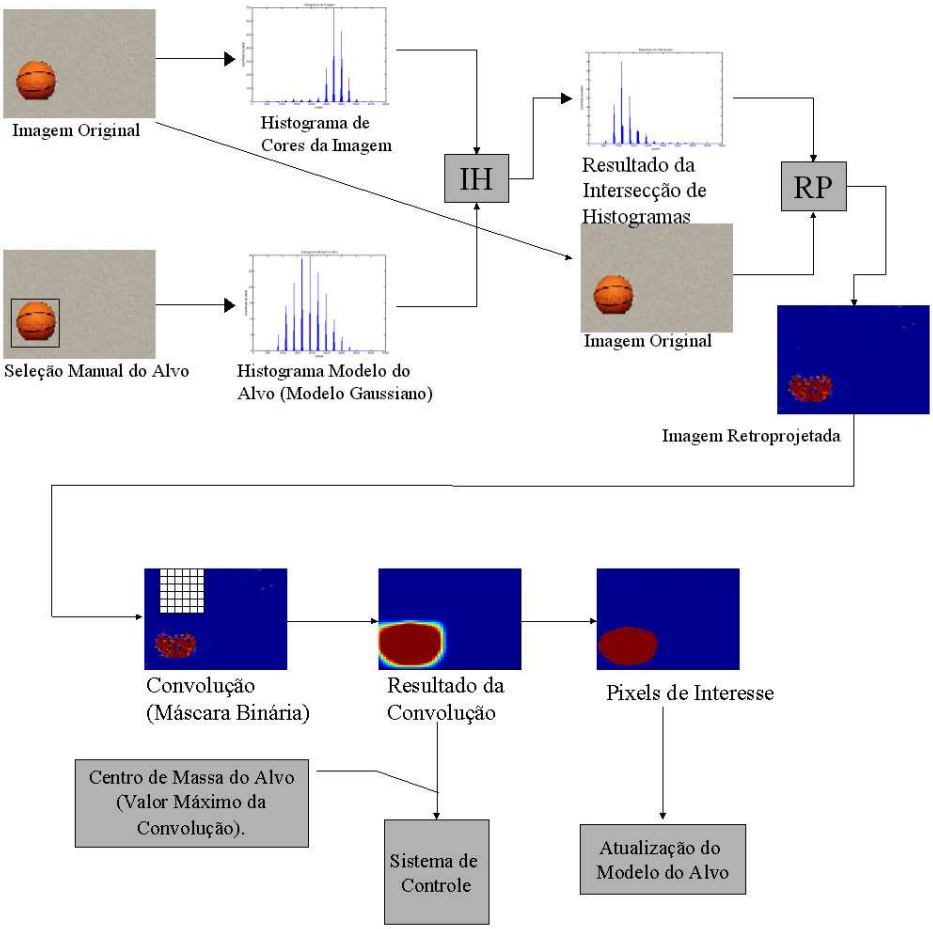

Figura 2.5 – Esquema representando as etapas do algoritmo de detecção visual por retroprojeção de histogramas. IH representa a operação de intersecção de histogramas e RP a retroprojeção.

fechando-se o ciclo e estabelecendo a recursão. A Figura 2.5 ilustra o algoritmo. Note em cada estágio a imagem resultante e o formato do histograma modelo seguindo exatamente uma Gaussiana.

3 Processamento de Imagem para Determinação da Posição e Postura de Robôs Móveis

No presente trabalho, além de simular e implementar técnicas de processamento de imagem e leis de controle com o objetivo de rastrear alvos móveis estudou-se e simulou-se ainda outras aplicações em robótica que também utilizam-se dos benefícios dos sensores visuais.

Uma dessas aplicações é utilizada em [1], no qual a técnica de processamento de imagem utilizada tem por objetivo determinar a posição absoluta do veículo em relação a um referencial fixo com o objetivo de rastrear uma trajetória preestabelecida.

Desse modo, nesse capítulo será apresentada a técnica de processamento para a extração da posição e postura do robô num dado instante de tempo, utilizada neste trabalho.

3.1 Reconstrução Euclidiana

Considere-se o cenário ilustrado na Figura 3.1. O sistema F se move junto com a plataforma do robô e tem o eixo x perpendicular ao eixo das rodas, o eixo y paralelo ao mesmo e o eixo z apontando para cima. A origem corresponde ao ponto central do eixo das rodas e ao foco da câmera. O eixo F_d corresponde ao eixo F durante a geração da trajetória de referência e os ângulos θ e θ_d são orientados em relação à horizontal (eixo x^*) conforme a Figura 3.1. O sistema F^* representa o referencial fixo de coordenadas ortogonais.

A velocidade linear do robô na direção x é denotada por $v_c(t)$ e a velocidade angular $\omega_c(t)$ é positiva no sentido anti-horário.

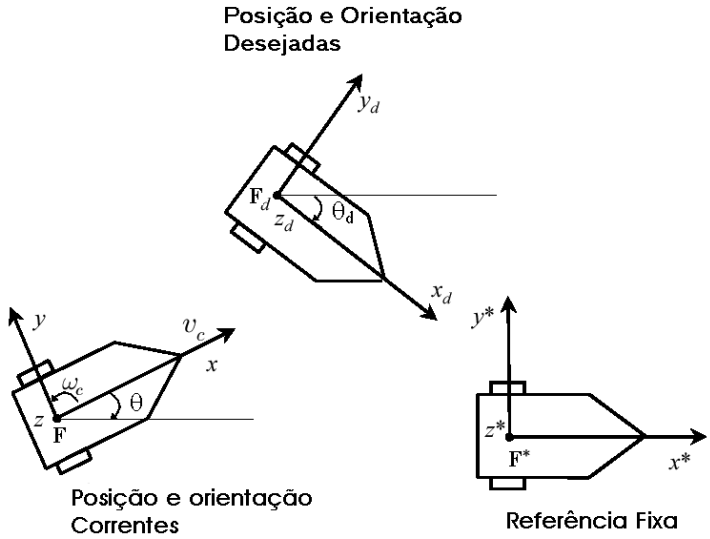

Figura 3.1 – Relações entre os eixos coordenados.

O que se deseja é obter as relações geométricas entre os eixos de coordenadas F, F_d, F^* e um plano de referência π definido por três pontos não colineares O_i, $(i = 1, 2, 3)$ observados pela câmera nas três situações ilustradas na Figura 3.1. As coordenadas Euclidianas desses pontos expressas em termos de F, F_d e F^* como sendo $\bar{m}_i(t)$, $\bar{m}_{di}(t)$ e $\bar{m}_i^* \in \mathbb{R}^3$, respectivamente, são definidas como segue, de acordo com a Figura 3.2.

$$\begin{aligned} \bar{m}_i(t) &\triangleq [x_i(t) \quad y_i(t) \quad z_i(t)]^T \\ \bar{m}_{di}(t) &\triangleq [x_{di}(t) \quad y_{di}(t) \quad z_{di}(t)]^T \\ \bar{m}_i^* &\triangleq [x_i^* \quad y_i^* \quad z_i^*]^T \end{aligned} \tag{3.1}$$

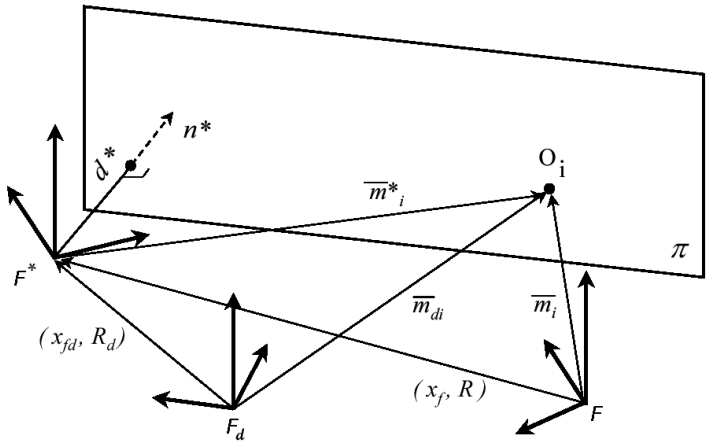

Figura 3.2 – Relações entre os eixos coordenados.

As distâncias das origens dos respectivos eixos de coordenadas até os pontos através do eixo de foco são positivas, ou seja, $x_i(t), \quad x_{di}(t) \quad e \quad x_i^* \geq \epsilon > 0$, onde ϵ é uma pequena constante positiva. A rotação entre F^* e F é denotada por $R(t) \in \mathbb{R}^{3x3}$, e a translação entre ambos é denotada por $x_f(t) \in \mathbb{R}^3$, onde $x_f(t)$ é expresso em termos de F. Da mesma forma, $R_d(t) \in \mathbb{R}^{3x3}$ representa a rotação desejada entre F^* e F_d, e $x_{fd}(t) \in \mathbb{R}^3$ denota a translação desejada entre F_d e F^*, onde $x_{fd}(t)$ é expresso em termos de F_d. Uma vez que o movimento do robô é limitado ao plano, $x_f(t)$ e $x_{fd}(t)$ são definidos por,

$$\begin{aligned} x_f(t) &\triangleq [x_{f1}(t) \quad x_{f2}(t) \quad 0]^T \\ x_{fd}(t) &\triangleq [x_{fd1}(t) \quad x_{fd2}(t) \quad 0]^T \end{aligned} \tag{3.2}$$

De acordo com a Figura 3.2 pode-se então relacionar os pontos $\overline{m}_i(t)$ e $\overline{m}_{di}(t)$ com $\overline{m}_i^*$ da seguinte forma,

$$\bar{m}_i = x_f + R\bar{m}_i^* \quad \bar{m}_{di} = x_{fd} + R_d\bar{m}_i^* \tag{3.3}$$

Em 3.3, $R(t)$ e $R_d(t)$ são definidas como,

$$R \triangleq \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

$$R_d \triangleq \begin{bmatrix} \cos(\theta_d) & -\sin(\theta_d) & 0 \\ \sin(\theta_d) & \cos(\theta_d) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

e ainda $\theta(t)$ e $\theta_d(t) \in \mathbb{R}$ são definidos conforme a Figura 3.1. Dessas definições percebe-se que,

$$\dot{\theta} = -\omega_c \quad \dot{\theta}_d = -\omega_{cd} \tag{3.5}$$

Assume-se ainda que os ângulos de rotação estão confinados nas seguintes regiões,

$$-\pi < \theta(t) < \pi \quad -\pi < \theta_d(t) < \pi \tag{3.6}$$

Para as relações geométricas mostradas na Figura 3.2, a distância $d^* \in \mathbb{R}$ de F^* a π na direção do vetor normal unitário n^* é dada por,

$$d^* = n^{*T}\overline{m}_i^* \tag{3.7}$$

em que, $n^* = [n_x^* \quad n_y^* \quad n_z^*]^T \in \mathbb{R}^3$ representa o vetor constante e unitário normal ao plano π. Da Equação 3.7 e da Figura 3.2, as relações na Equação 3.3 podem ser expressas como segue,

$$\overline{m}_i = \left(R + \frac{x_f}{d^*} n^{*T}\right) \overline{m}_i^* \quad \overline{m}_{di} = \left(R_d + \frac{x_{fd}}{d^*} n^{*T}\right) \overline{m}_i^* \tag{3.8}$$

Uma vez que as coordenadas de $\overline{m}_i(t)$, $\overline{m}_{di}(t)$ e $\overline{m}_i^*$ não podem ser diretamente medidas, coordenadas ditas normalizadas [1] podem ser reconstruídas relacionando as diversas imagens desses pontos. Essas coordenadas são definidas por,

$$\begin{aligned} m_i &\triangleq [1 \quad m_{iy} \quad m_{iz}]^T = \frac{\bar{m}_i}{x_i} \\ m_{di} &\triangleq [1 \quad m_{diy}(t) \quad m_{diz}]^T = \frac{\bar{m}_{di}}{x_{di}} \\ m_i^* &\triangleq [1 \quad m_{iy}^* \quad m_{iz}^*]^T = \frac{\bar{m}_i^*}{x_i^*} \end{aligned} \tag{3.9}$$

em que, $\bar{m}_i$, $\bar{m}_{di}$ e $\bar{m}_i^*$ foram definidos na Equação 3.1.

Adicionalmente, cada ponto O_i terá suas coordenadas nas imagens obtidas em F, F_d e F^*, respectivamente chamados de $p_i(t)$, $p_{di}(t)$ e p_i^*. Conforme visto na Seção 3.4, as coordenadas Euclidianas dos pontos são então relacionadas com os dados da imagem por meio da matriz de mudança de coordenadas associada ao modelo da câmera de acordo com,

$$p_i = Am_i \quad p_{di} = Am_{di} \quad p_i^* = Am_i^* \tag{3.10}$$

em que, $A \in \mathbb{R}^{3x3}$ é uma matriz constante e conhecida de calibração da câmera (vide Seção 2.1).

Assim, uma vez que $m_i(t)$, $m_{di}(t)$ e m_i^* podem ser determinados com base na Equação 3.10, a rotação e translação entre os sistemas de coordenadas podem ser agora obtidos

com base nas coordenadas normalizadas como segue,

$$m_i = \underbrace{\frac{x_i^*}{x_i}}_{\alpha_i} \quad \underbrace{\left(R + x_h n^{*T}\right)}_{H} \tag{3.11}$$

$$m_{di} = \underbrace{\frac{x_i^*}{x_{di}}}_{\alpha_{di}} \quad \underbrace{\left(R_d + x_{hd} n^{*T}\right)}_{H_d} \tag{3.12}$$

em que, $\alpha_i(t)$, $\alpha_{di}(t) \in \mathbb{R}$ são as taxas de profundidade, $H(t)$, $H_d(t) \in \mathbb{R}^{3x3}$ são as homografias euclidianas, e $x_h(t)$, $x_{hd}(t) \in \mathbb{R}^3$ são os vetores de translação escalonados definidos por,

$$\begin{aligned} x_h &\triangleq [x_{h1} \quad x_{h2} \quad 0]^T = \frac{x_f}{d^*} \\ x_{hd} &\triangleq [x_{hd1} \quad x_{hd2} \quad 0]^T = \frac{x_{fd}}{d^*} \end{aligned} \tag{3.13}$$

Dessa forma, usando-se as Equações 3.4 e 3.13, a matriz H na Equação (3.11) pode ser escrita como,

$$H = \begin{bmatrix} \cos\theta + x_{h1} n_x^* & -\sin\theta + x_{h1} n_y^* & x_{h1} n_z^* \\ \sin\theta + x_{h2} n_x^* & \cos\theta + x_{h2} n_y^* & x_{h2} n_z^* \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Percebe-se assim, que H é formada por argumentos que não podem ser diretamente medidos ($\theta(t)$, x_h(t) e n^*). Portanto, por meio das Equações (3.9), (3.11) e (3.14), nota-se

que para cada ponto serão geradas três equações com nove incógnitas conforme,

$$1 = \alpha_i(H_{11} + H_{12}m^*_{iy} + H_{13}m^*_{iz}) \tag{3.15}$$

$$m_{iy} = \alpha_i(H_{21} + H_{22}m^*_{iy} + H_{23}m^*_{iz}) \tag{3.16}$$

$$m_{iz} = \alpha_i m^*_{iz} \tag{3.17}$$

em que, os $H_{jk}(t)$ são os termos que compõem a matriz $H(t)$ na Equação 3.14.

Uma vez calculada a matriz $H(t)$, o próximo passo para a determinação da postura do robô móvel é a decomposição das homografias para a obtenção dos valores de $\theta(t)$ e $x_h(t)$. De forma semelhante, o procedimento anterior é seguido para o cálculo dos termos que compõem $H_d(t)$, e faz-se necessário decompor as homografias para obter $\theta_d(t)$ e $x_{hd}(t)$. O procedimento para cumprir esse objetivo foi proposto por [27], e será mostrado na Seção 3.2, para o caso de $H(t)$, sendo que para $H_d(t)$ basta seguir o mesmo procedimento.

3.2 Decomposição das Homografias

Inicialmente considera-se as seguintes definições,

$$k^2 \triangleq x_h^T x_h, \quad k > 0 \tag{3.18}$$

$$p \triangleq n^T x_h \tag{3.19}$$

Da Equação 3.11 pode-se mostrar que,

$$H^T H = I + n x_h^T + x_h^T n + k^2 n n^T \tag{3.20}$$

em que, I é a matriz identidade de ordem 3x3 e a notação $H(t)$ foi abandonada para simplificar as demonstrações. Note-se que devido à relações de álgebra, a matriz R,definida em 3.4, efetivamente não aparece na Equação 3.20.

Das definições de autovalor e autovetor é possível perceber, pela Equação 3.20 que o primeiro autovetor de $H^T H$ e seu correspondente autovalor são,

$$v_1 = x_h \times n, \quad \eta_1 = 1 \tag{3.21}$$

Uma vez que $H^T H$ é simétrica, de acordo com a Equação 3.20 os outros dois autovalores devem estar no plano perpendicular a $v1$. Daí pode-se concluir que os mesmos tenham a seguinte forma,

$$v_{2,3} = a x_h + b n \tag{3.22}$$

Então, da definição de autovalor chega-se à

$$H^T H(a x_h + b n) = \eta_{2,3}(a x_h + b n) \tag{3.23}$$

em que, $\eta_{2,3}$ são os autovalores correspondentes aos autovetores mostrados na Equação 3.22.

Dessa forma, substituindo a Equação 3.20 em 3.23 obtém-se os resultados mostrados nas equações de 3.24 a 3.27, de acordo com o valor de p.

para $p \neq -1$,

$$\eta_{2,3} = 1 + p + \frac{2k(p+1)}{-k \pm \sqrt{k^2 + 4(p+1)}} \tag{3.24}$$

$$v_{2,3} = \frac{-k \pm \sqrt{k^2 + 4(p+1)}}{2k(p+1)} x_h + n \tag{3.25}$$

para $p = -1$,

$$\eta_2 = k^2, \quad \eta_3 = 0 \tag{3.26}$$

$$v_2 = \frac{1}{k^2} x_h + n, \quad v_3 = xh \tag{3.27}$$

Neste ponto, para simplificar as notações seguintes, define-se as variáveis mostradas a seguir,

$$\gamma, \lambda \triangleq \frac{-k \pm \sqrt{k^2 + 4(p+1)}}{2k(p+1)} \tag{3.28}$$

Assim, com base nas soluções anteriores pode-se avaliar os diversos casos para solução dos autovalores. Para isso, algumas restrições precisam ser definidas.

Pela definição de k e p nas Equações 3.18 e 3.19, respectivamente, pode-se chegar às desigualdades,

$$k \geq p, \quad \|k\| \geq \|p\| \tag{3.29}$$

Sabe-se também que $H^T H$ é uma matriz simétrica e real, e portanto sempre tem

autovalores reais. Desse modo, a terceira restrição é tal que,

$$k^2 + 4(p+1) \geq 0 \tag{3.30}$$

Assim, baseado nas três restrições apresentadas nas Equações 3.29 e 3.30 pode-se mostrar que $\eta_2 \geq \eta_1 = 1$, e que $\eta_3 \geq \eta_1 = 1$.

Desse modo, para o caso geral em que x_h e n não estão alinhados valem as seguintes inequações, $\eta_2 > \eta_1 = 1 > \eta_3$. Assume-se então que os três valores singulares de H sejam $\rho_2 > \rho_1 > \rho_3$ e seus correspondentes autovetores ortonormais sejam $\pm u_2, \pm u_1, \pm u_3$. Uma vez que $(\rho_1)^2 = \eta_1$, então $\rho_1 = 1$.

Relembrando a relação entre valores singulares e autovalores, tem-se que $(\rho_{2,3})^2 = \eta_{2,3}$ o que leva a,

$$k = \rho_2 - \rho_3, p = \rho_2\rho_3 - 1 \tag{3.31}$$

Define-se então, com o objetivo de efetuar a normalização dos vetores $\|v_2\|$ e $\|v_3\|$, as grandezas mostradas a seguir,

$$\mu \triangleq \|v_2\|, \delta \triangleq \|v_3\| \tag{3.32}$$

Substituindo a Equação 3.25 em 3.31, obtém-se,

$$\pm u_2 = \frac{\gamma x_h + n}{\mu}, \pm u_3 = \frac{\lambda x_h + n}{\delta} \tag{3.33}$$

E finalmente, resolvendo o conjunto de Equações 3.33, tem-se o conjunto de quatro soluções mostrado a seguir,

$$x_h = \pm\frac{\gamma u_2 - \delta u_3}{\gamma - \lambda}, n = \pm\frac{\gamma\delta u_3 - \lambda\mu u_2}{\gamma - \lambda} \tag{3.34}$$

$$x_h = \pm\frac{\gamma u_2 + \delta u_3}{\gamma - \lambda}, n = \pm\frac{\gamma\delta u_3 + \lambda\mu u_2}{\gamma - \lambda} \tag{3.35}$$

Relembrando que as soluções devem ser coerentes com a realidade, de tal forma que $m_i \cdot n > 0$, duas das quatro soluções são facilmente removidas. Com base nas duas soluções remanescentes, a matriz de rotação R da Equação 3.4 pode ser obtida por,

$$R = H(I + x_h n^T)^{-1} \tag{3.36}$$

Com isso, os parâmetros de orientação e posição do robô (θ, x_{h1} e x_{h2}) são conhecidos e portanto a postura do robô está descrita. Para o caso em que os vetores n e x_h estiverem alinhados recomenda-se a leitura de [27], bem como para obtenção de maiores detalhes.

4 Controle para Rastreamento de Alvos Móveis

Neste capítulo são apresentadas as duas metodologias de controle utilizadas neste trabalho. A primeira delas, utiliza-se dos conceitos de lógica *fuzzy* para elaborar controladores de maneira rápida e razoável (atendendo ao menos a requisitos mínimos de especificação), de acordo com os conhecimentos acumulados por operadores humanos.

Já a outra faz uso das técnicas de análise de estabilidade segundo Lyapunov e do modelo cinemático de robôs móveis, para conduzir os erros de posição e postura para valores nulos.

Dessa forma, a seguir será feita uma breve explanação sobre a utilização dessas técnicas com aplicação no rastreamento de alvos móveis.

4.1 Controle tipo *fuzzy*

Devido à sua técnica de projeto empírica, o controle tipo *fuzzy* pode ser utilizado tanto para rastreamento de alvos móveis [4] quanto pra trajetórias pré gravadas [10], desde que os sensores sejam utilizados de modo apropriado para cada caso. A seguir, é descrito o

processo de desenvolvimento de um controlador para o primeiro caso, de acordo com o mostrado em [4], mais detalhes a respeito podem também ser encontrados em [3].

O diagrama de um sistema de controle fuzzy, mostrado na Figura 4.1, incorpora os seguintes elementos:

1. Base das regras (conjunto de regras Se-Então).

2. Mecanismo de inferência, que emula a decisão feita pelas interpretações lingüísticas e conclui a resposta sobre como melhorar o controle da planta.

3. Interface de *fuzificação*, que converte a entrada do controle em informações que o mecanismo de inferência possa facilmente usar para ativar as respectivas regras.

4. Interface de *defuzificação*, que converte as conclusões do mecanismo de inferência em atuação na entrada do processo.

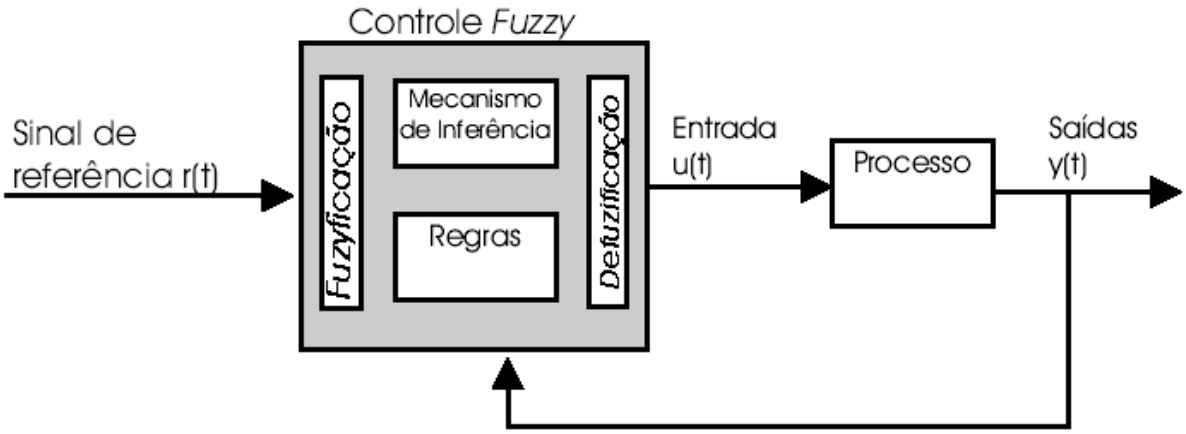

Figura 4.1 – Arquitetura de um Controlador *fuzzy*. [4]

Basicamente, o controlador *fuzzy* opera em um sistema malha fechada em tempo real adquirindo os dados de saída da planta y(t), comparando com a referência desejada r(t) e decidindo qual será a entrada da planta u(t).

Para a aplicação desejada o controlador *fuzzy* implementado possui duas entradas e uma saída conforme ilustrado na Figura 4.2. Pode-se perceber ainda que é bastante coerente admitir que essas entradas sejam o erro e sua derivada, o que dá ao controlador uma estrutura semelhante à de um controlador proporcional-derivativo (PD) em que regras *fuzzy* servirão para gerar o mapeamento não linear.

Figura 4.2 – Modelo do controlador *fuzzy* para rastreamento de alvos móveis.

Os controles de velocidade linear e angular são implementados separadamente, obedecendo o modelo cinemático utilizado em [5]. Além disso, deve-se estabelecer que o objetivo do controle é **manter uma distância fixa entre robô e alvo, além de garantir que o robô esteja sempre orientado na direção do mesmo.**

Assim convencionando-se: GN é grande negativo, MN meio negativo, PN pequeno negativo, ZZ zero, PS pequeno positivo, MP meio positivo, e GP grande positivo pode-se associar os seguintes conjuntos nebulosos à entrada do controlador.

Distância entre alvo e robô - $D_{ar} \Rightarrow \{GN, MN, PN, ZZ, PP, MP, GP\}$

Derivada da distância - $dD_{ar} \Rightarrow \{GN, MN, PN, ZZ, PP, MP, GP\}$

e o conjunto associado a variável de saída do controle de velocidade linear por:

Velocidade - $V_d \Rightarrow \{GN, MN, PN, ZZ, PP, MP, GP\}$

Da mesma maneira pode-se definir os conjuntos para a orientação do robô em relação ao alvo por

Ângulo entre alvo e robô - $\theta_{ar} \Rightarrow \{GN, MN, PN, ZZ, PP, MP, GP\}$

Derivada do ângulo - $d\theta_{ar} \Rightarrow \{GN, MN, PN, ZZ, PP, MP, GP\}$

e o conjunto associado a variável de saída do controle de velocidade angular por

Velocidade - $V_\omega \Rightarrow \{GN, MN, PN, ZZ, PP, MP, GP\}$

Com estas definições, o conjunto de regras de inferência utilizado pode ser representado de forma compacta através da Figura 4.3 onde, por exemplo, se o Erro é GP e a Derivada do Erro é ZZ, então a saída é GN.

		Derivada do Erro					
	GN	MN	PN	ZZ	PP	MP	GP
GN	GP	GP	GP	GP	MP	PP	ZZ
MN	GP	GP	GP	MP	PP	ZZ	PN
PN	GP	GP	MP	PP	ZZ	PN	GN
ZZ	GP	MP	PP	ZZ	PN	GN	GN
PP	MP	PP	ZZ	PN	GN	GN	GN
MP	PP	ZZ	PN	GN	GN	GN	GN
GP	ZZ	PN	GN	GN	GN	GN	GN

Erro

Figura 4.3 – Regras base para os controladores de velocidade linear e angular.

Uma vez definidos os conjuntos de variáveis nebulosas, é necessário 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 com os atuadores do sistema. À esse mapeamento entre valores numéricos e variáveis lingüísticas dá-se o nome de função de pertinência.

Para realizar o procedimento de inferência, é necessário conhecer, para cada valor de entrada, as funções de pertinência associadas a ele e os respectivos graus de pertinência. Como a lógica *fuzzy* é uma generalização da lógica convencional, é bastante coerente que as funções de pertinência, denotadas por, $\mu(\cdot)$, tenham variação entre 0 e 1 [2].

A Figura 4.4 apresenta as 7 funções de pertinência para o erro de distância entre o robô e alvo utilizando a caracterização triangular. A Figura 4.5 representa as funções de

pertinência para a derivada do erro e a Figura 4.6 as funções de pertinência para a saída do controlador.

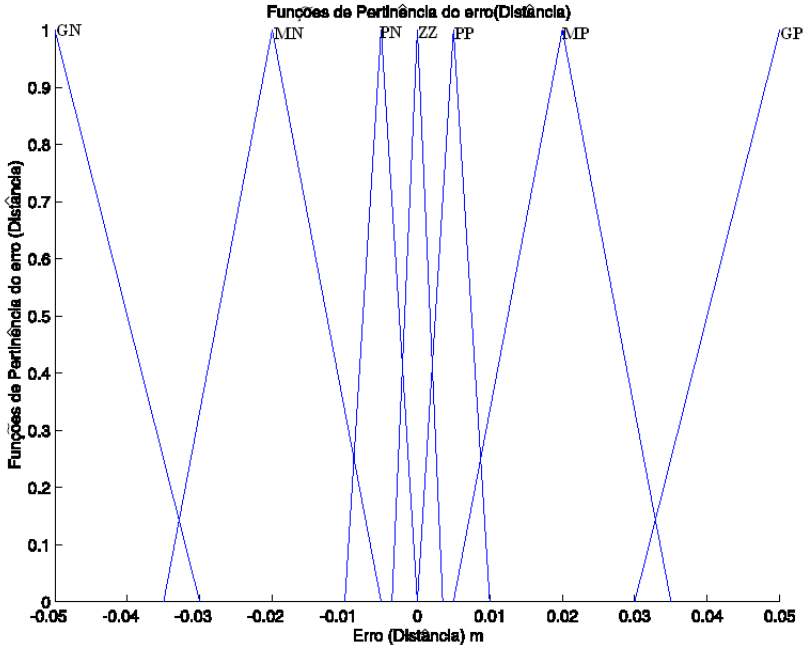

Figura 4.4 – Funções de pertinência para o erro de distância.

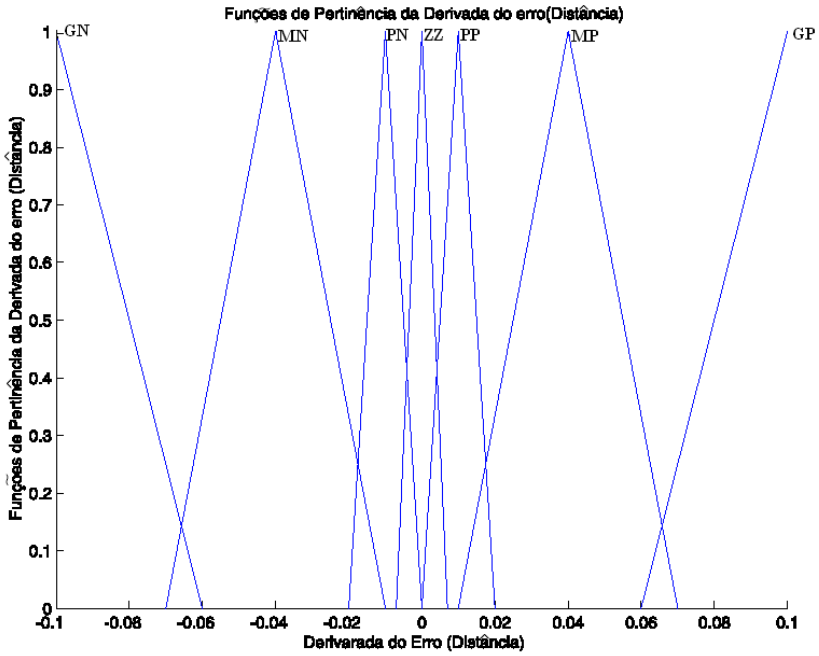

Figura 4.5 – Funções de pertinência para a derivada do erro de distância.

Da mesma forma, para o controlador angular são apresentadas as funções de pertinência para as entradas de erro angular (Figura 4.7), derivada do erro (Figura 4.8) e

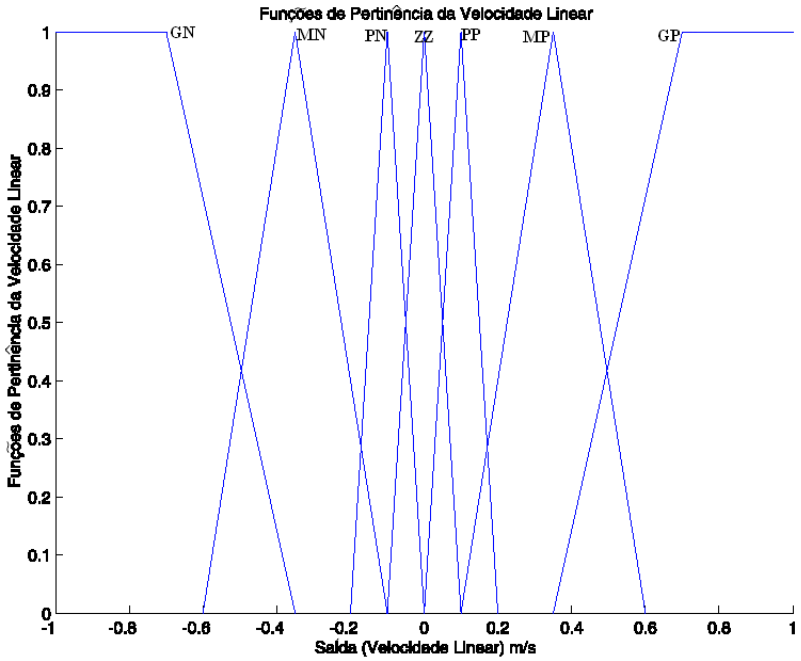

Figura 4.6 – Funções de pertinência para saída de velocidade linear.

para a saída de controle angular (Figura 4.9). Deve-se notar ainda, que devido ao fato de os controladores produzirem em suas saidas velocidades relativas (vide [3]) é necessário somar-se as velocidades correntes do robô para que o alvo seja corretamente rastreado. Isso é possível fazendo-se a contagem dos pulsos dos encoders existentes nas rodas do robô.

Finalmente, mostra-se nas Figuras 4.10 e 4.11 as superfícies de decisão obtidas na simulação (utilizando a ferramenta *fuzzy* do MatLab). dos controladores. Pode-se perceber aí a suavidade das transições obtidas devido à máquina de inferência.

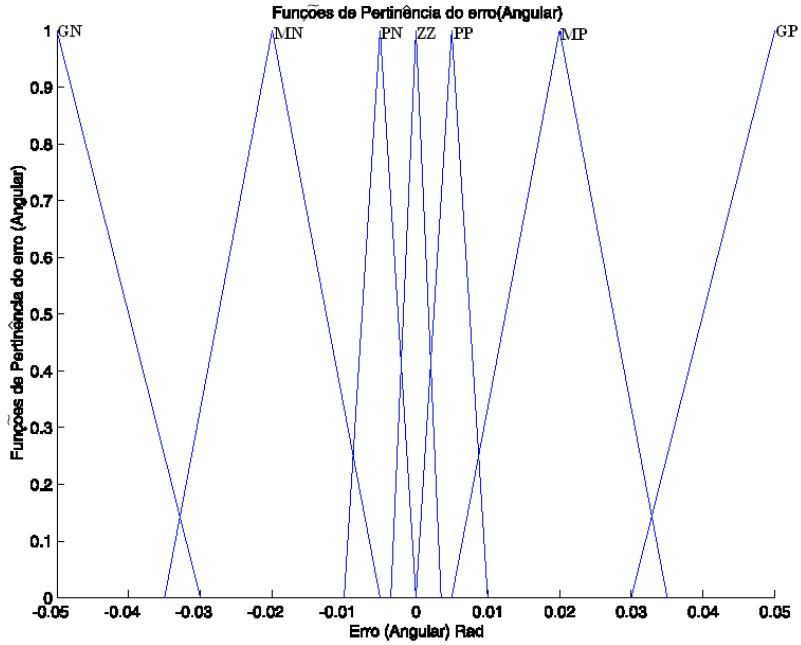

Figura 4.7 – Funções de pertinência para o erro angular entre alvo e robô.

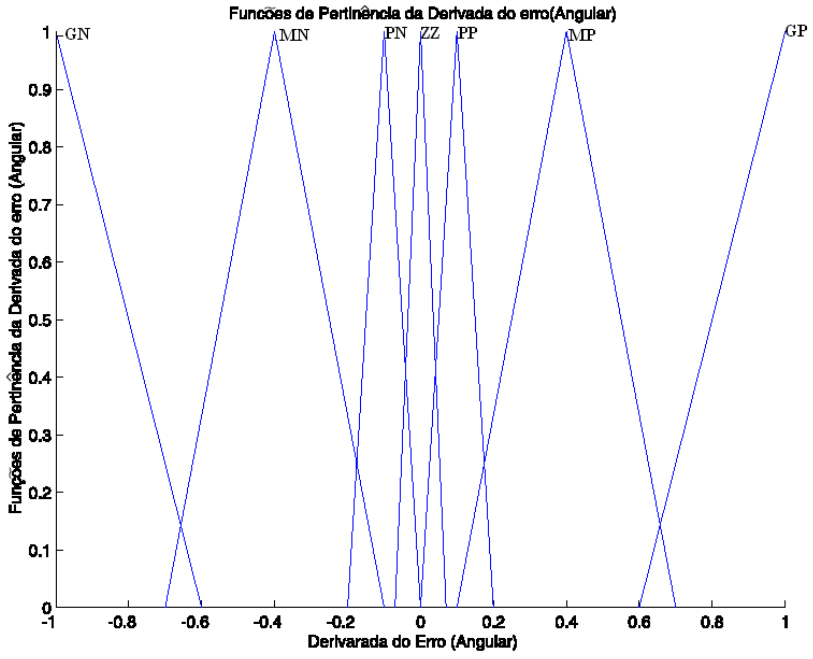

Figura 4.8 – Funções de pertinência para a derivada do erro angular.

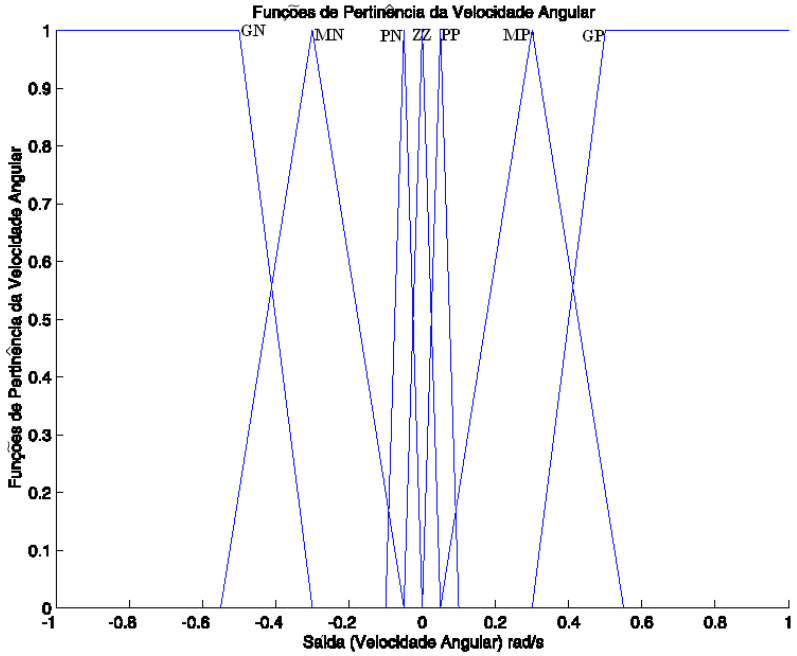

Figura 4.9 – Funções de pertinência para saída de velocidade angular.

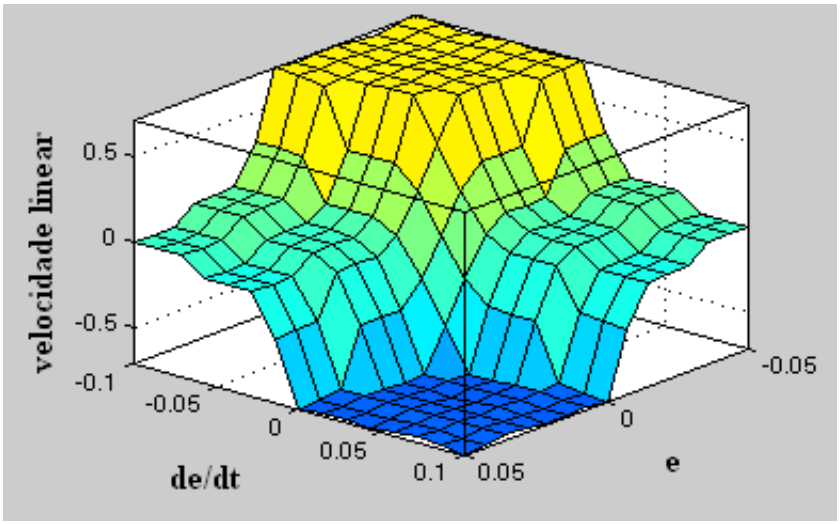

Figura 4.10 – Superfície de decisão para o controlador de velocidade linear(erro em m, velocidade em m/s).

4.2 Leis de Controle Baseadas na Análise de Estabilidade Segundo Lyapunov

Muito tem-se estudado com relação a leis de controle baseadas na análise de estabilidade segundo Lyapunov, sobretudo com aplicações que dependem de trajetórias previamente

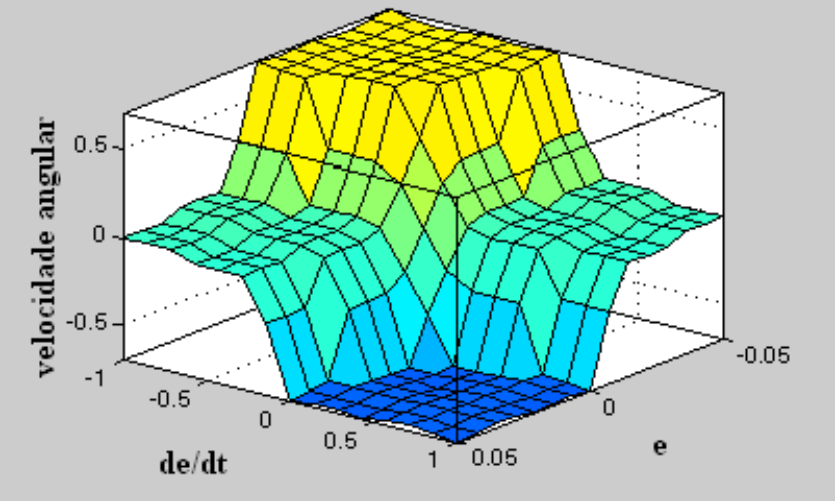

Figura 4.11 – Superfície de decisão para o controlador de velocidade angular (erro em rad, velocidade em rad/s).

determinadas. Em alguns casos, entretanto, esse tipo de abordagem pode não ser viável, como por exemplo no rastreamento de um alvo dotado de movimento. Nesse caso, a lei de controle obtida só poderá ser utilizada, com a aplicação de apenas um sensor, no caso de não depender de informações acerca da posição absoluta do robô em relação a um referencial estático.

Um caso que foi estudado nesse trabalho, é a lei de controle apresentada em [1], no qual, o problema apresentado é a reconstrução de uma trajetória pré gravada, e portanto, o sensor visual pode ser utilizado unicamente com o fim de determinar a posição absoluta do robô em relação ao referencial fixo F^* (vide Figura 3.2).

O objetivo do controle é assegurar que o eixo coordenado F rastreie a trajetória variante no tempo F_d, ou seja, $\bar{m}_i(t)$ rastreie $\bar{m}_{di}(t)$. Fica claro então, que este objetivo

será alcançado se o erro de rastreamento $e(t) \triangleq [e_1 \quad e_2 \quad e_3]^T \in \mathbb{R}^3$ definido por

$$\begin{aligned} e_1 &\triangleq x_{h1} - x_{hd1} \\ e_2 &\triangleq x_{h2} - x_{hd12} \\ e_3 &\triangleq \theta - \theta_d \end{aligned} \tag{4.1}$$

tender a zero. Pode-se provar então [1] que a lei de controle mostrada nas Equações 4.2 e 4.3 em conjunto com a lei de adaptação para um parâmetro de profundidade desconhecido $\hat{d}^*(t) \in \mathbb{R}$ mostrada na Equação 4.4, leva a $\|e(t)\| \to 0$, assintoticamente, desde que $\lim_{t\to\infty} \dot{x}_{hd1} \neq 0$.

$$v_c \triangleq k_{v1} e_1 - \bar{e}_2 \omega_c + \hat{d}^*(x_{h2}\omega_c - \dot{x}_{hd1}) \tag{4.2}$$

$$\omega_c \triangleq k_\omega e_3 - \dot{\theta}_d - \dot{x}_{hd1}\bar{e}_2 \tag{4.3}$$

em que, $\bar{e}_2 \triangleq k_{v2} e_2 - x_{hd1} e_3$.

$$\dot{\hat{d}}^* \triangleq \gamma_1(x_{h2}\omega_c - \dot{x}_{hd1}) \tag{4.4}$$

Entretanto, as grandezas $\dot{x}_{hd1}$ e x_{h2}, são proporcionais às coordenadas absolutas do robô durante a gravação da trajetória de referência e durante a execução da trajetória respectivamente, conforme visto no Capítulo 3. Nesse caso, a câmera fica comprometida com a visualização de pontos fixos no plano de referência π, não podendo portanto ser usada para determinar a posição do alvo.

Uma solução possível seria, então, a utilização de outros sensores, com o objetivo de determinar a posição absoluta do robô. Porém, tal abordagem sairia do escopo desse trabalho, pelos diversos motivos já mencionados nas seções anteriores. Outra alternativa também viável seria a procura de leis de controle que não tivessem essa limitação.

Nesse sentido, trabalhos anteriores, realizados no grupo de pesquisa do qual esse trabalho faz parte já capitalizaram no sentido de encontrar outras leis de controle para rastreamento de trajetórias pré-gravadas. A lei de controle apresentada em [5], por exemplo, depende apenas da posição e orientação de um robô de referência em relação ao robô móvel.

$$v_c = v_r \cos(e_3) + k_1 e_1 \tag{4.5}$$

$$\omega_c = \omega_r + k_2 v_r e_2 + k_3 v_r \sin(e_3) \tag{4.6}$$

Dessa forma, é possível utilizar as Equações 4.5 e 4.6 para rastrear um alvo móvel, desde que se possa determinar suas velocidades linear e angular **em relação ao referencial fixo** (v_r e ω_r respectivamente). Uma vez que o período de amostragem T é conhecido pode-se determinar, por meio do algoritmo de detecção visual apresentado na Seção 2.3, a variação de posição $\triangle s$ do alvo em relação ao robô entre dois quadros subseqüentes, e conseqüentemente estimar sua velocidade relativa.

Como a velocidade relativa entre alvo e robô segue a relação apresentada na Equação 4.7, basta conhecer-se a velocidade absoluta atual do robô para que se possa determinar a velocidade do alvo. Isso é possível usando os pulsos provenientes dos *encoders* de forma

semelhante ao que é feito em [4].

Nesse caso, o problema de *drift* não será tão grave, primeiramente porque esse sensor não será usado para determinação de posição e também pelo fato de que a câmera estará continuamente informando ao sistema de controle o erro atual real, ainda que as rodas patinem.

$$v_{relativa} = v_{alvo} - v_{robo} \tag{4.7}$$

De forma equivalente, é necessário apenas fazer a mesma análise para a velocidade angular de maneira a determinar ω_r na Equação 4.6. Vale ressaltar ainda que uma análise cuidadosa da estrutura de controle dada nas Equações 4.5 e 4.6 revela que no caso de os erros e_1, e_2 e e_3 tenderem a zero v_c e ω_c tenderão para v_r e ω_r, respectivamente.

5 Resultados de Simulação

Neste Capítulo são apresentados os resultados de simulação obtidos neste trabalho. Inicialmente apresenta-se os relativos ao sistema de detecção visual por intersecção de histogramas. Em seguida, compara-se os resultados de controle de trajetória usando a lei de controle mostrada na Seção 4.2 com os modelos cinemático e dinâmico, além da análise na presença de ruídos. Por fim, é feita uma comparação entre o controle tipo *fuzzy* e cinemático (projetado utilizando método de Lyapunov para estabilidade) no rastreamento de alvos móveis.

5.1 Algoritmo de Detecção Visual

Para testar o algoritmo de detecção visual gerou-se via software comum de imagens *bitmap*, um fundo monocromático simulando uma parede. Com base nesse fundo foram criadas duas imagens adicionando-se o alvo em posições diferentes. Essas imagens foram poluídas com ruído Gaussiano de média zero e variância 0.1, de tal sorte que se pudesse selecionar o alvo em uma delas e o algoritmo encontrasse o mesmo alvo na segunda imagem.

Na primeira iteração as rotinas geradas no software MatLab solicitam que o usuário selecione o alvo manualmente na primeira imagem, conforme pode ser visto na Figura 5.1.

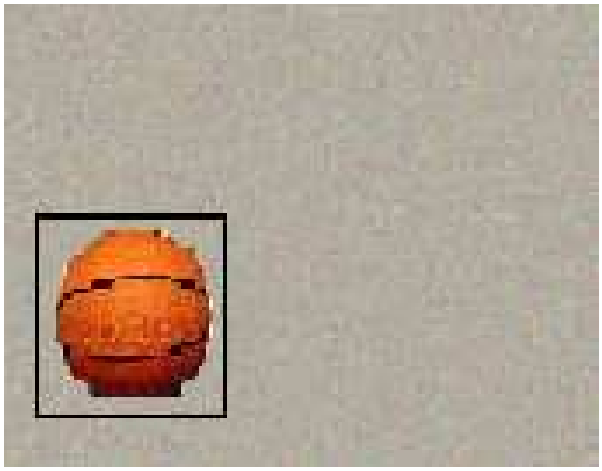

Figura 5.1 – Seleção manual do alvo.

Uma vez que o alvo tenha sido selecionado são gerados os histogramas da imagem (no segundo instante) e do modelo (selecionado manualmente), como mostrado nas Figuras 5.2 e 5.3, respectivamente.

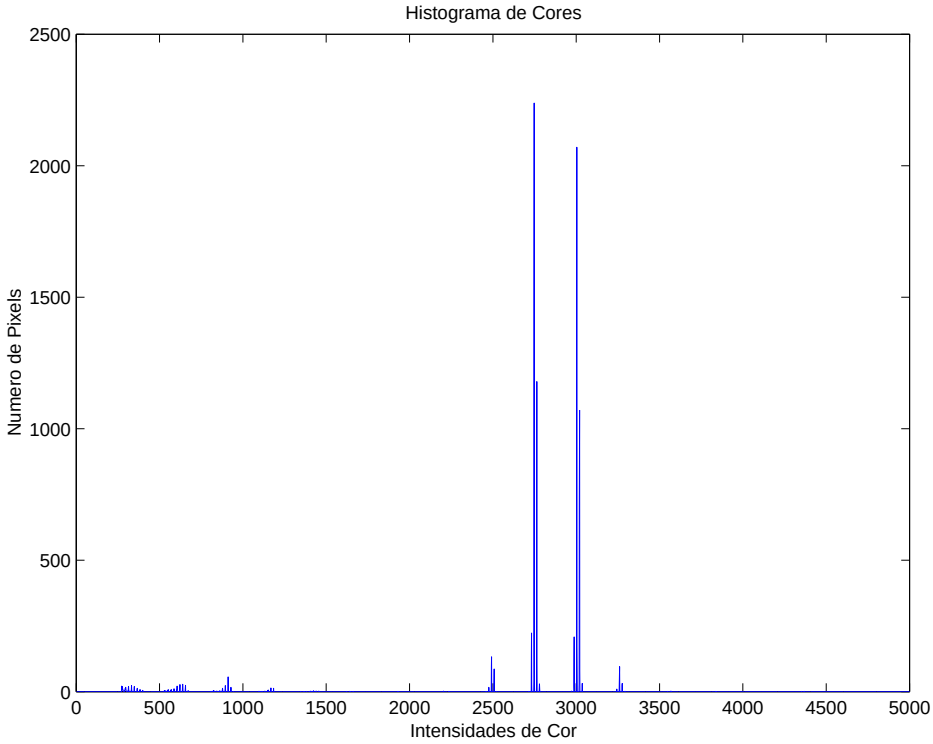

Figura 5.2 – Histograma da imagem original.

Em seguida, realiza-se a intersecção de histogramas, gerando o gráfico mostrado na Figura 5.4.

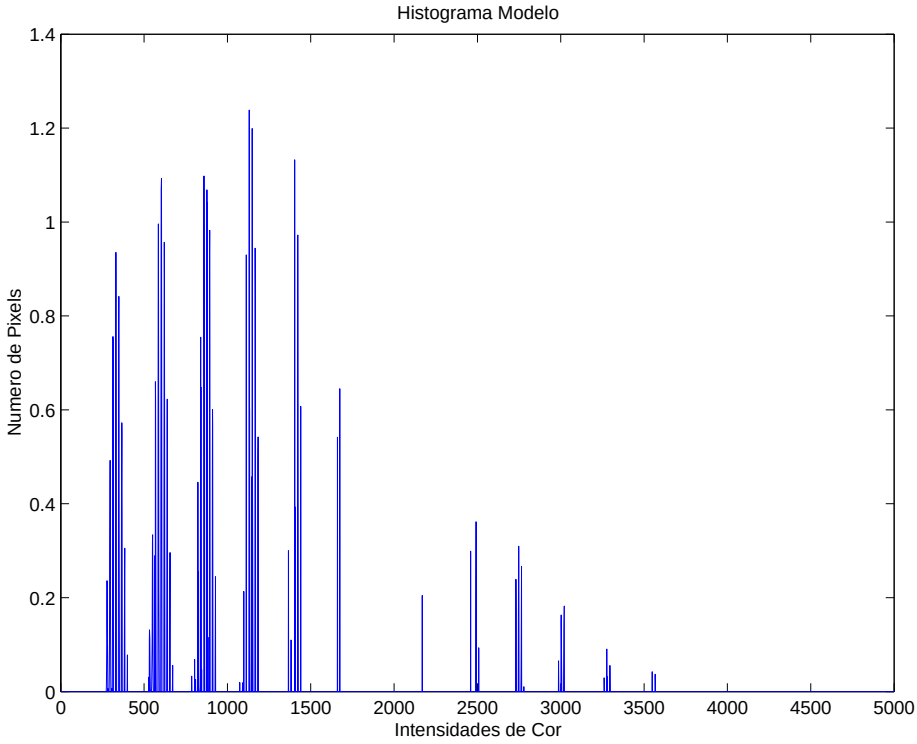

Figura 5.3 – Histograma modelo.

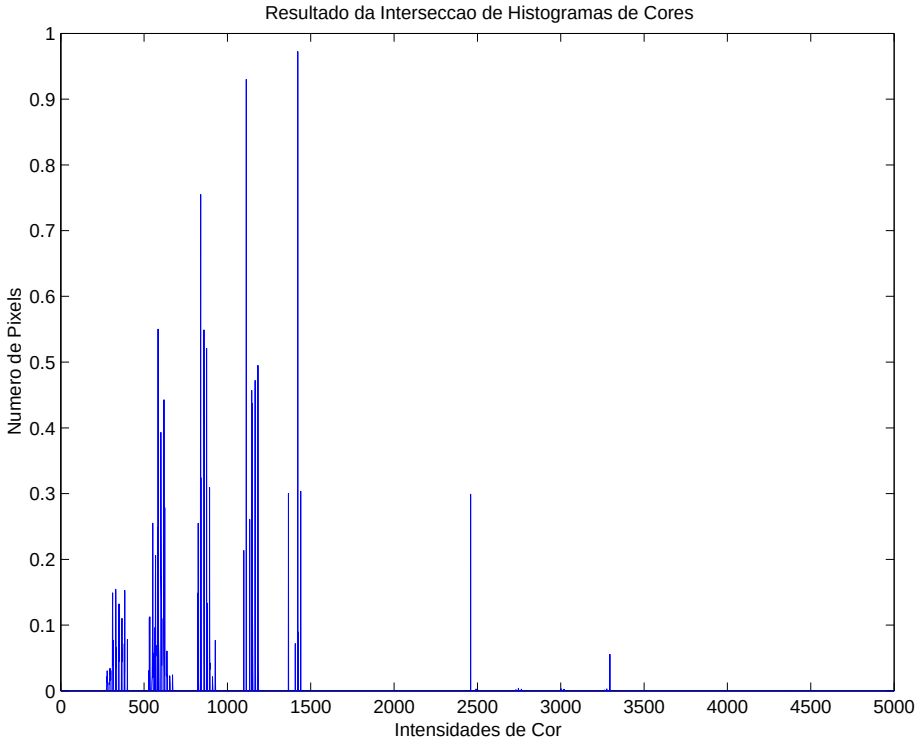

Figura 5.4 – Resultado da intersecção de histogramas.

Percorre-se então a imagem do instante 2, produzindo a imagem retroprojetada, mostrada na Figura 5.5.

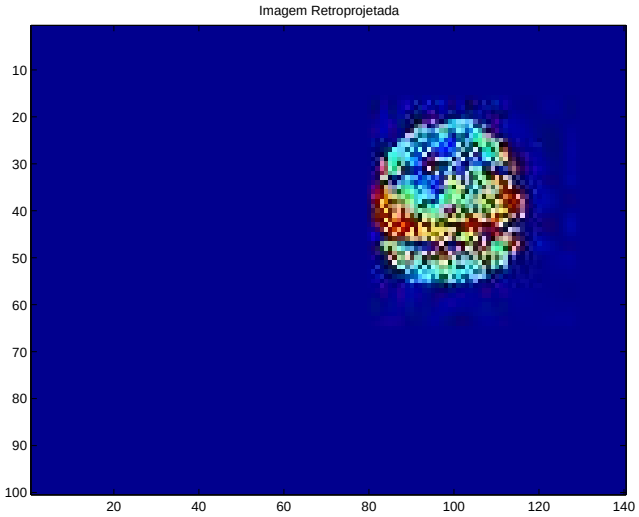

Figura 5.5 – Imagem retroprojetada da imagem no instante 2.

A imagem retroprojetada é convoluída com a máscara binária e gera a imagem mostrada na Figura 5.6.

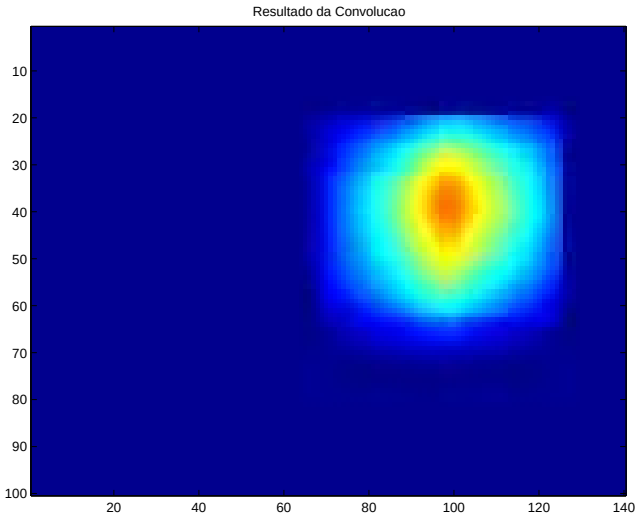

Figura 5.6 – Imagem resultante da convolução com uma máscara binária.

Aqui, encontrando-se o valor máximo da imagem convoluída determina-se o centro de gravidade do alvo no segundo instante. Feito isso, termina-se o processo de detecção do alvo e o programa seleciona-o na imagem, conforme mostrado na Figura 5.7. Porém, como dito anteriormente deseja-se um processo recursivo e para tanto é necessária a adaptação do modelo, de tal forma que o usuário não precise mais interferir selecionando o alvo.

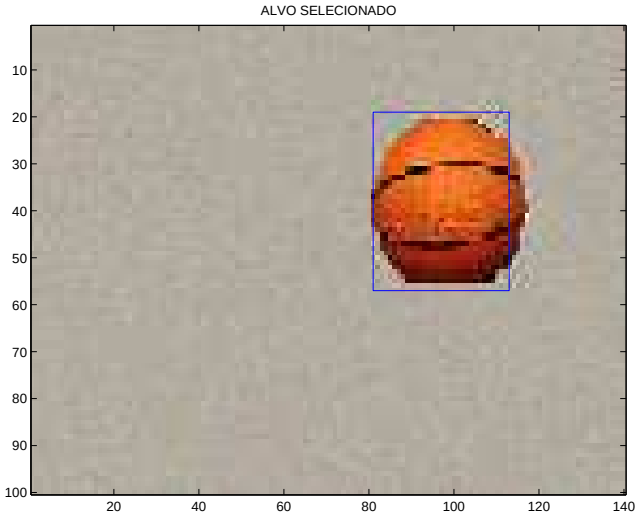

Figura 5.7 – Alvo localizado no instante 2.

Dessa maneira, seleciona-se os pixels de interesse, conforme proposto na Seção 2.3, para gerar uma imagem que sirva de máscara na identificação do alvo no instante subseqüente. A seleção dos pixels de interesse dá origem à imagem exibida na Figura 5.8 que é então, utilizada para atualizar o modelo do alvo e continuar o processo sem a interferência humana.

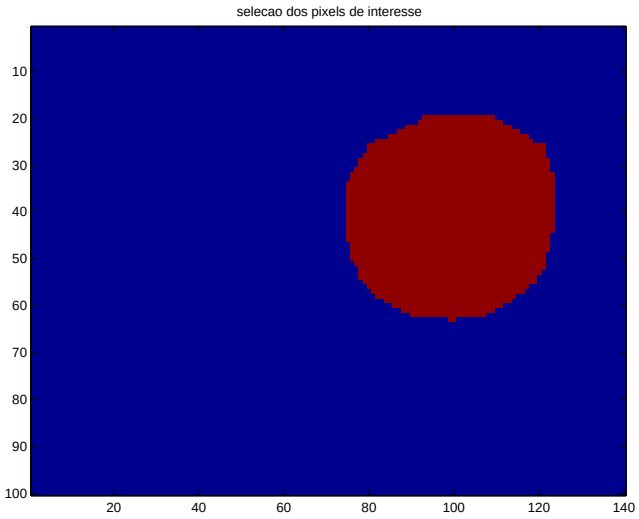

Figura 5.8 – Seleção dos pixels de interesse.

Para verificar o comportamento do algoritmo funcionando de forma recursiva o mesmo processo descrito até aqui foi repetido também emulando-se o movimento do alvo numa trajetória senoidal. O resultado obtido é mostrado na Figura 5.9 e exibe a capacidade do

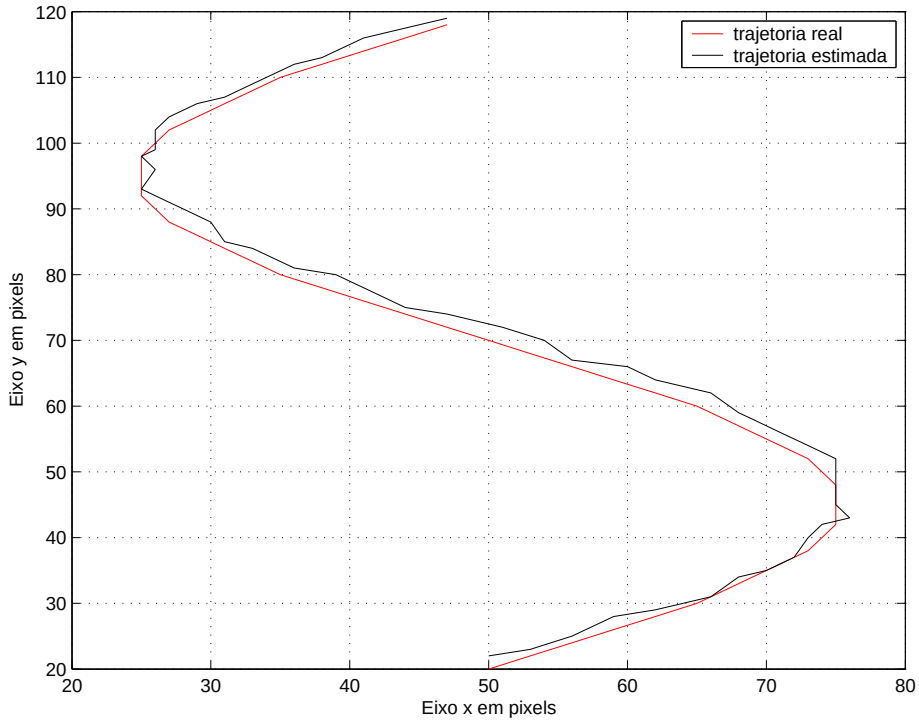

Figura 5.9 – Detecção da trajetória do alvo utilizando o algoritmo de intersecção de histogramas.

algoritmo em localizar o alvo.

Note-se que há sempre um pequeno erro entre a localização do alvo e a posição encontrada pelo algoritmo. Esse erro se deve à seleção inicial feita manualmente pelo usuário. Dessa forma, o resultado será tão melhor quanto a seleção inicial.

Por fim, as figuras 5.10 e 5.11 apresentam a detecção das coordenadas x e y do alvo, respectivamente, em função do tempo.

5.2 Estratégias de Controle

5.2.1 Controle para Rastreamento de Trajetória

Nessa seção são mostrados os resultados de simulação obtidos utilizando a lei de controle da Seção 4.2 (Equações 4.2 e 4.3).

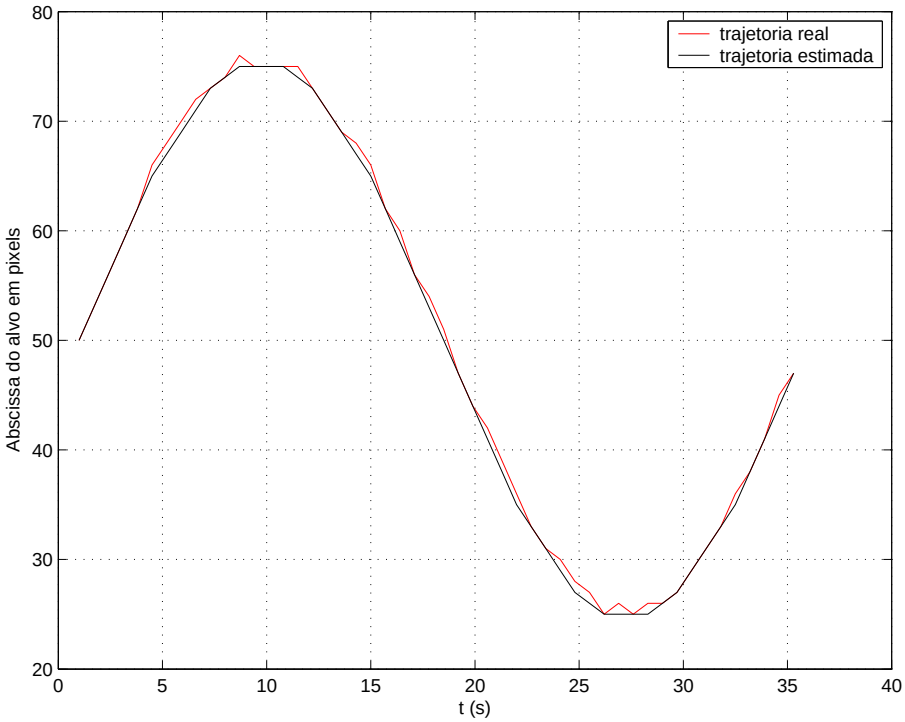

Figura 5.10 – Detecção da coordenada x do alvo em função do tempo.

5.2.1.1 Utilizando Modelo Cinemático

Para a realização destas simulações foi utilizado o modelo cinemático apresentado em [5] - mostrado na Equação 5.1 - com dados reais do robô Magellan Pro. O parâmetro d, que nesse caso vale 0.04 m, é a distância entre o ponto de intersecção do eixo de simetria com o eixo das rodas e o centro de massa do veículo.

$$\begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5.1}$$

Assim, de acordo com o algoritmo descrito na Seção 3.1 escolheu-se três pontos não colineares , definidos na Equação 5.2, para gerar o plano de referência π.

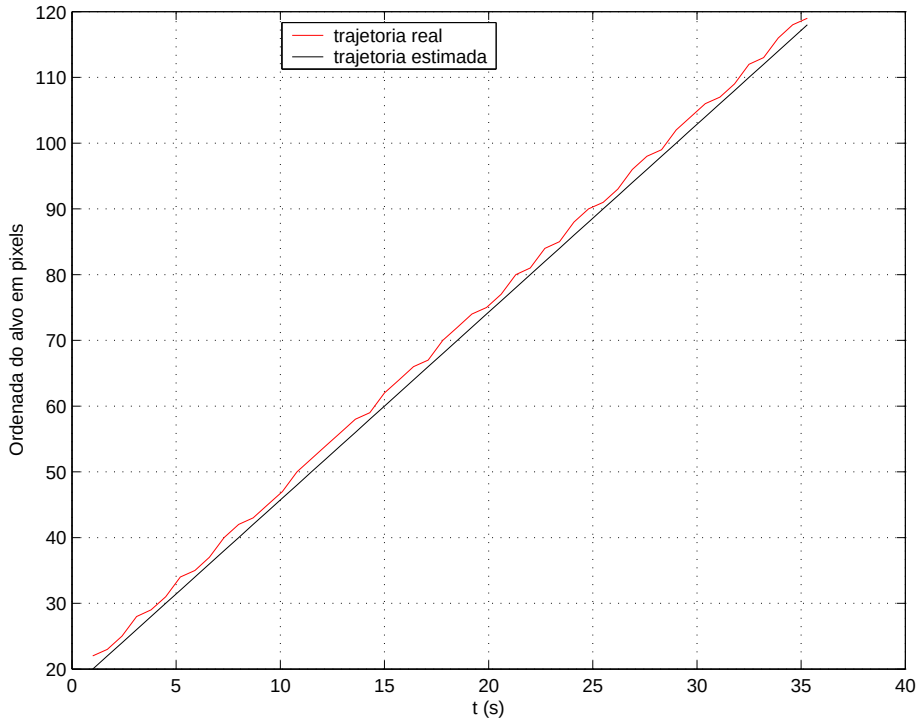

Figura 5.11 – Detecção da coordenada y do alvo em função do tempo.

$$\begin{aligned} \bar{m}_1^* &= [1 \quad 2 \quad 5]^T \\ \bar{m}_2^* &= [1 \quad 1 \quad 1]^T \\ \bar{m}_3^* &= [7 \quad 3 \quad 2]^T \end{aligned} \tag{5.2}$$

Com base nesses pontos um programa MatLab foi escrito de forma a gerar artificialmente a trajetória pré-gravada, com $v_{cd} = 0.2$ m/s e $\omega_{cd} = 0.1\sin(0.3t)$ rad/s. A freqüência de amostragem utilizada para os quadros foi de 50 ms, e a função *spline* do software MatLab foi usada para interpolar os valores de x_{hd1} e x_{hd2}, uma vez que os controles mostrados nas Equações 4.2 e 4.3 são todos contínuos. Ajustou-se então, por tentativa e erro os ganhos k_{v1}, k_{v2}, k_ω e γ_1, resultando os valores apresentados na Equação 5.3. Para esse ajuste usou-se a sistemática de produzir movimentos dos alvos em apenas

um dos eixos por vez, facilitando assim a determinação de cada uma das constantes.

$$k_{v1} = k_{v2} = 20, \quad k_\omega = 1, \quad \gamma_1 = 0.09 \tag{5.3}$$

Um erro inicial em relação à trajetória desejada foi adicionado. Tais ajustes levaram aos resultados mostrados nas Figuras 5.12, 5.13, 5.14 e 5.15.

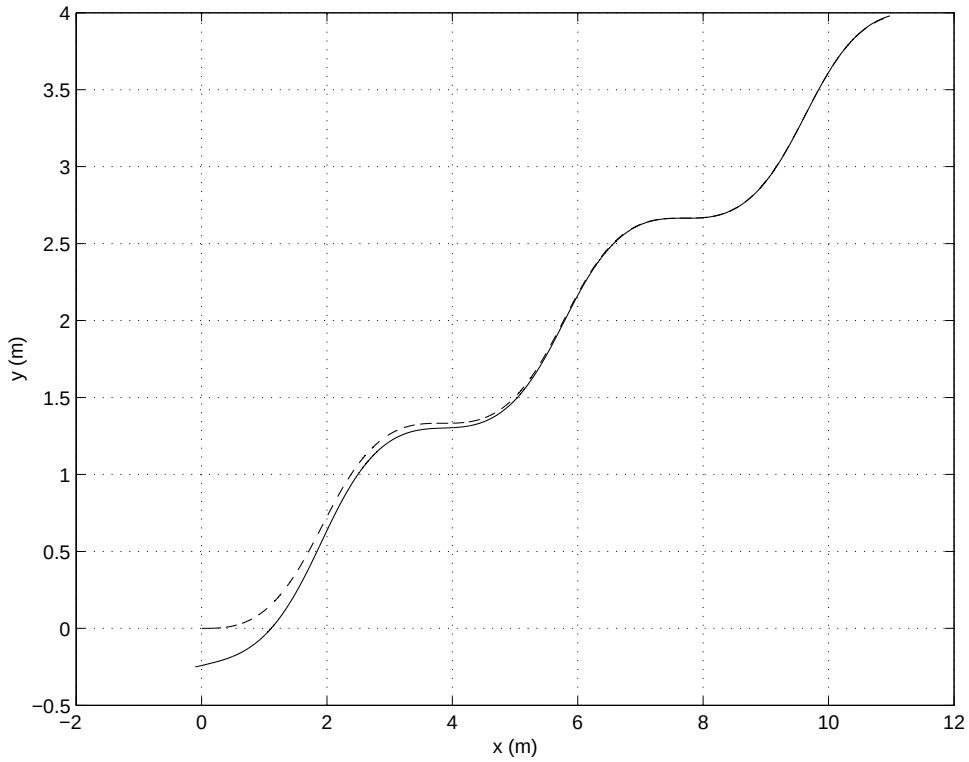

Figura 5.12 – Trajetória pré-gravada (pontilhada) e realizada (linha cheia).

Percebe-se ainda, na Figura 5.15, que os controles foram relativamente suaves, exceto nos primeiros segundos de simulação onde o erro inicial gerado era maior.

5.2.1.2 Utilizando Modelo Dinâmico

Para conseguir mais realismo nas simulações o mesmo procedimento descrito anteriormente foi repetido utilizando agora o modelo dinâmico proposto por [28] e com os dados

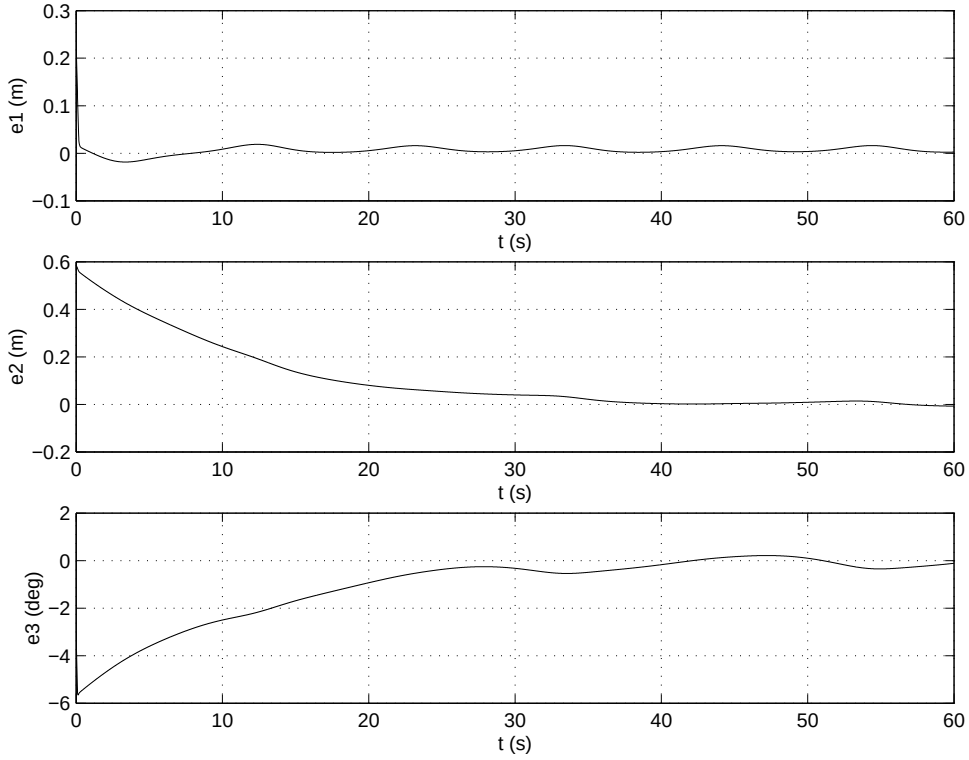

Figura 5.13 – Erros e_1, e_2 e e_3 para simulação usando modelo cinemático.

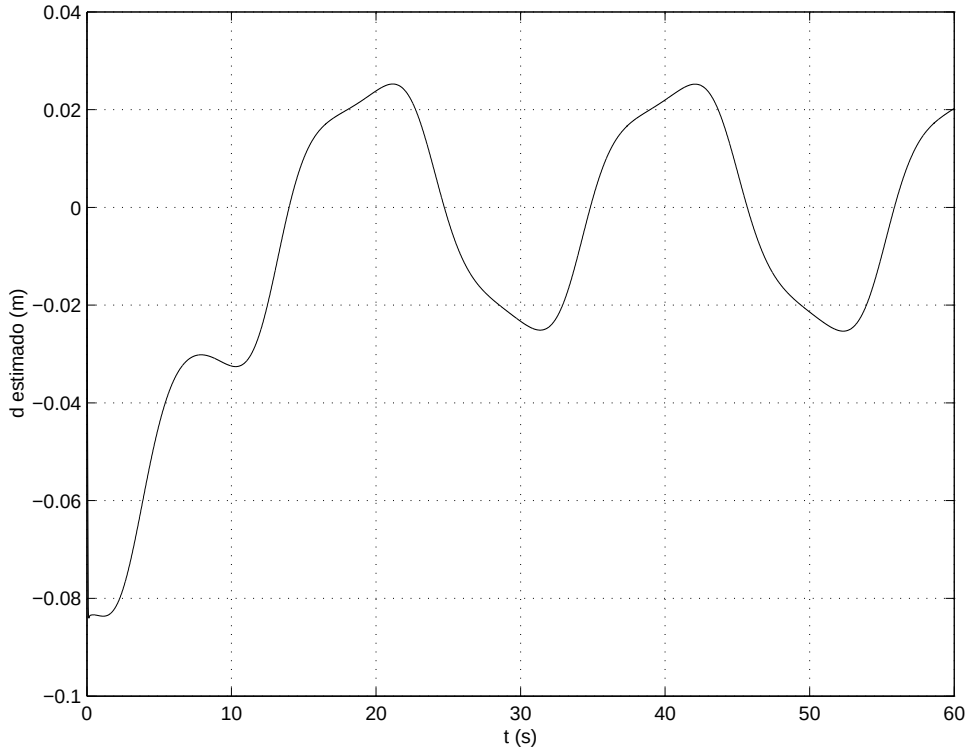

Figura 5.14 – Comportamento de $\hat{d}$ usando modelo cinemático.

reais do robô Magelan-ISR.

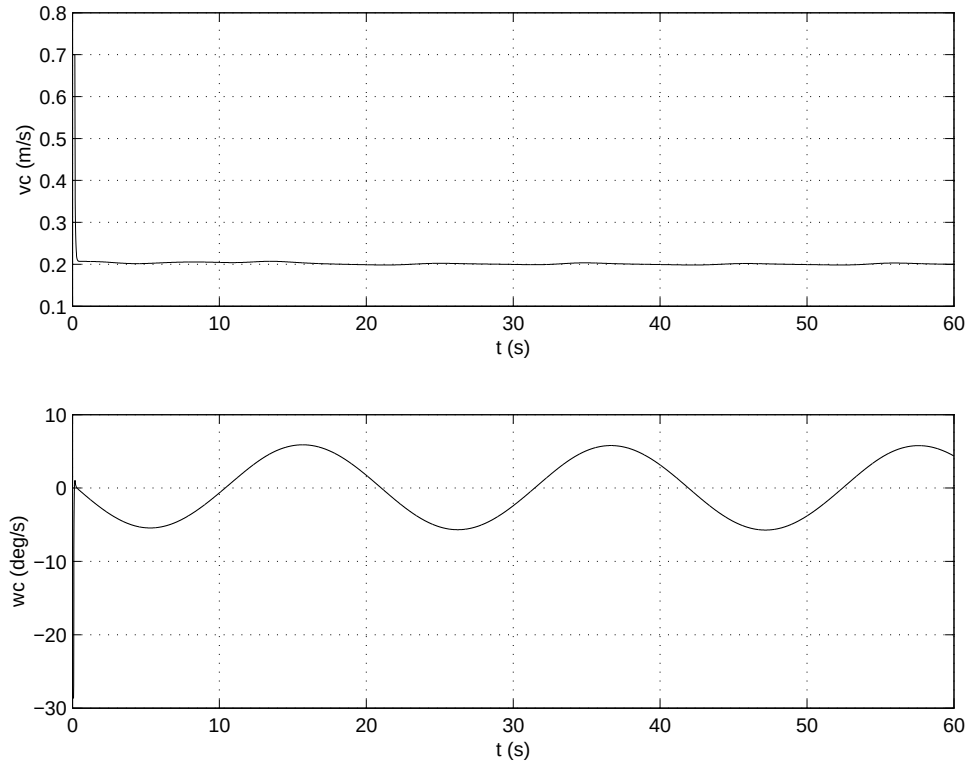

Figura 5.15 – Sinal dos atuadores usando modelo cinemático.

Neste caso, foi necessário projetar novamente os ganhos para se obter resultados satisfatórios, chegando-se aos valores mostrados na Equação 5.4.

$$k_{v1} = k_{v2} = 40, \quad k_\omega = 1.5, \quad \gamma_1 = 0.0009 \tag{5.4}$$

Assim, obteve-se os resultados mostrados nas Figuras 5.16, 5.17, 5.18 e 5.19.

Novamente nota-se que os resultados são satisfatórios, e a saturação dos controles ocorre apenas nos instantes iniciais devido ao erro propositalmente inserido. No entanto, para o caso com modelo dinâmico a tarefa de rastrear a trajetória é mais difícil, pois as velocidades não são instantaneamente atingidas, como ocorre no caso com modelo cinemático.

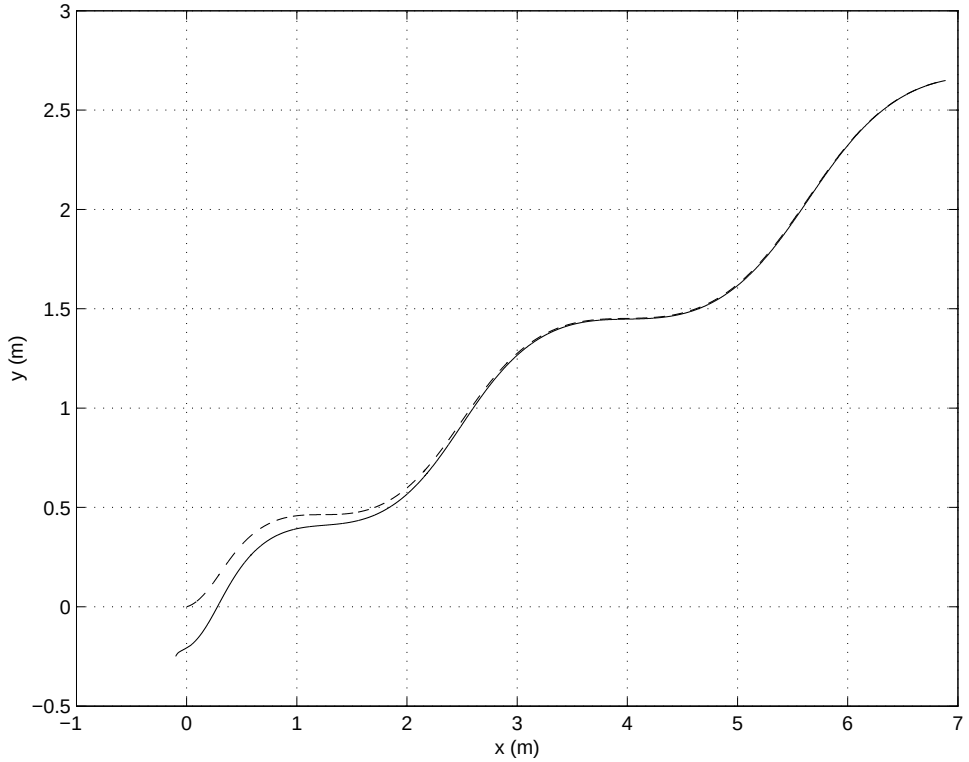

Figura 5.16 – Trajetória pré-gravada (pontilhada) e realizada (linha cheia)usando modelo dinâmico.

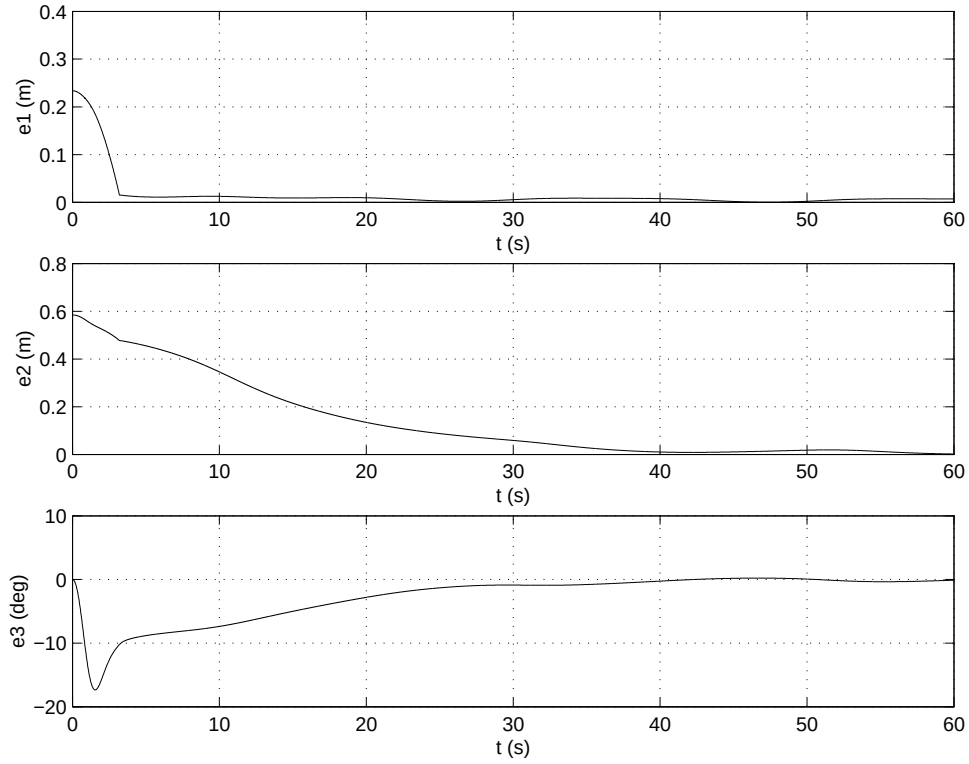

Figura 5.17 – Erros e_1, e_2 e e_3 para simulação usando usando modelo dinâmico.

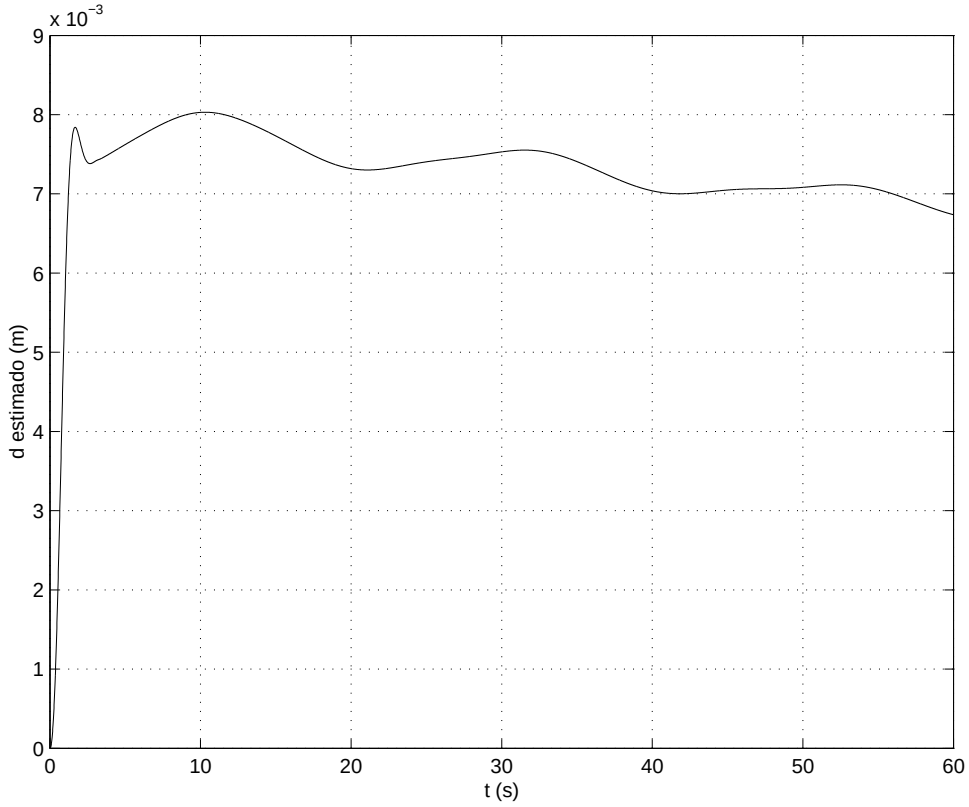

Figura 5.18 – Comportamento de $\hat{d}$ usando usando modelo dinâmico.

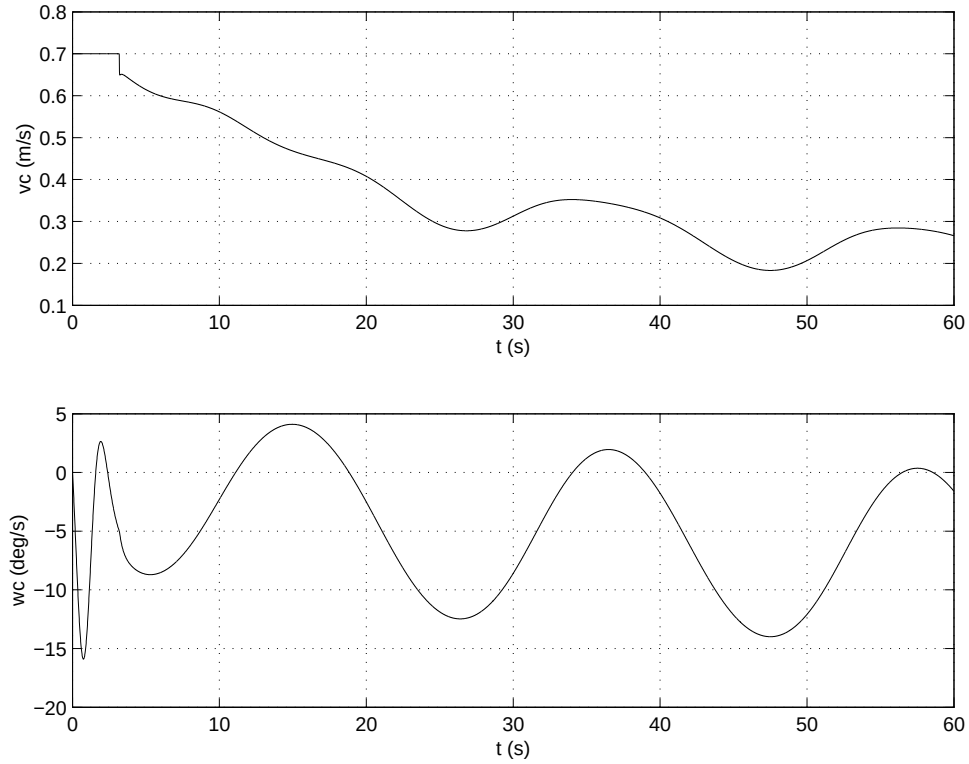

Figura 5.19 – Sinal dos atuadores usando usando modelo dinâmico.

Isso explica, por exemplo, a necessidade de se projetar novamente os parâmetros do controlador, deixando-o mais agressivo para ter sensibilidade às variações de velocidade ocorridas no período transitório.

5.2.1.3 Análise do Desempenho na Presença de Ruído

Para a análise do desempenho do controle de trajetória na presença de ruído um número pseudo-aleatório foi somado às leituras de posição (x_c e y_c) e postura (θ) do alvo, simulando possíveis incertezas na leitura da câmera.

Para as variáveis de posição foi utilizado ruído normal de média 0 e variância 10^{-3}, ou seja, desvio padrão de aproximadamente 3 cm, respeitando uma relação sinal ruído coerente. No caso da variável de postura utilizou-se um ruído também normal, de média 0 e variância 10^{-6}, desvio padrão de aproximadamente 0.06 grau. As simulações foram feitas utilizando-se o modelo cinemático, com as constantes do controle dadas na Equação 5.9. Os resultados para esse caso são mostrados nas Figuras 5.20, 5.21, 5.22 e 5.23.

Embora os resultados sejam razoáveis quando comparados com o caso sem ruído, percebe-se que os atuadores saturam devido, muito provavelmente, à sensibilidade do controle às altas freqüências do ruído. Para diminuir esse efeito foram utilizados filtros nas entradas do sistema, produzindo os resultados mostrados nas Figuras 5.24, 5.25, 5.26 e 5.27.

Dessa forma, pode-se observar que os resultados se mantiveram razoáveis, porém, agora os controles não saturam como aconteceu no caso sem filtro. De qualquer forma, para ter-se um índice numérico quanto ao desempenho do controle na presença de ruído com e sem a utilização do filtro foi feita uma análise via método de Monte Carlo.

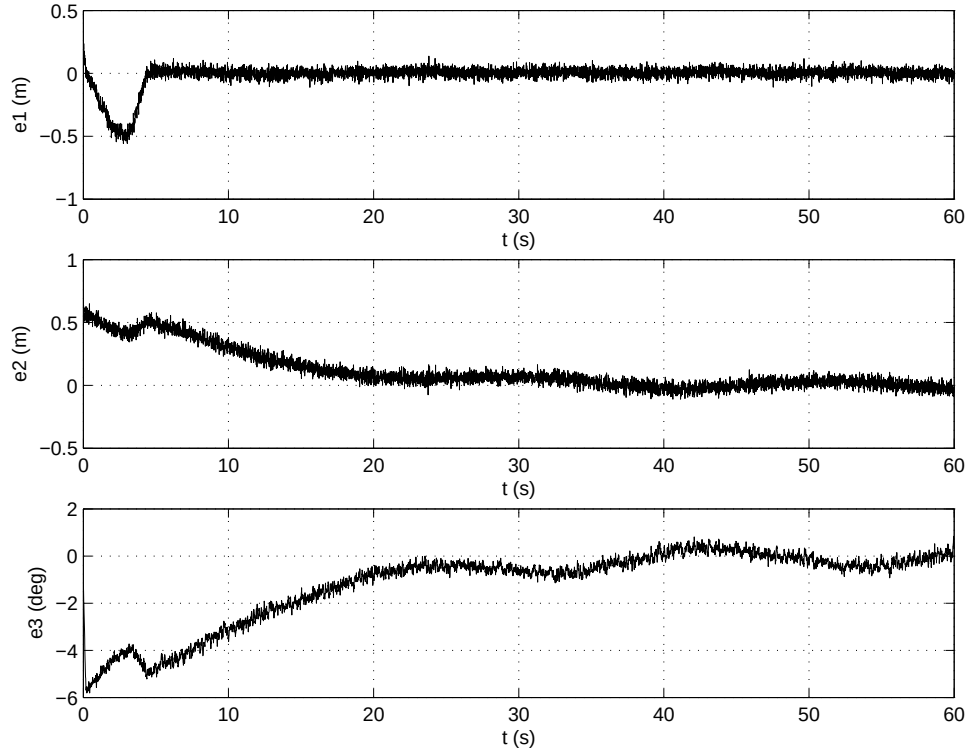

Figura 5.20 – Comportamento dos erros na presença de ruído.

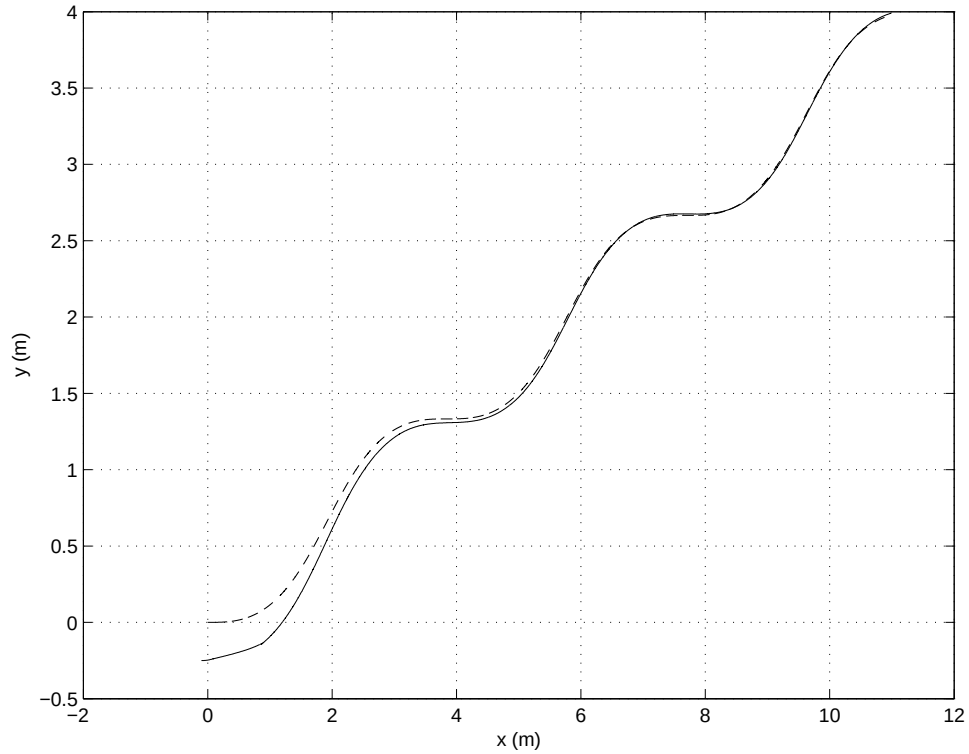

Figura 5.21 – Trajetória pré-gravada (pontilhada) e realizada(linha cheia) para o caso ruidoso.

Assim, gerou-se um programa MatLab que executou 100 simulações com diferentes sementes de ruído - para as mesmas características de ruído descritas acima. Em cada

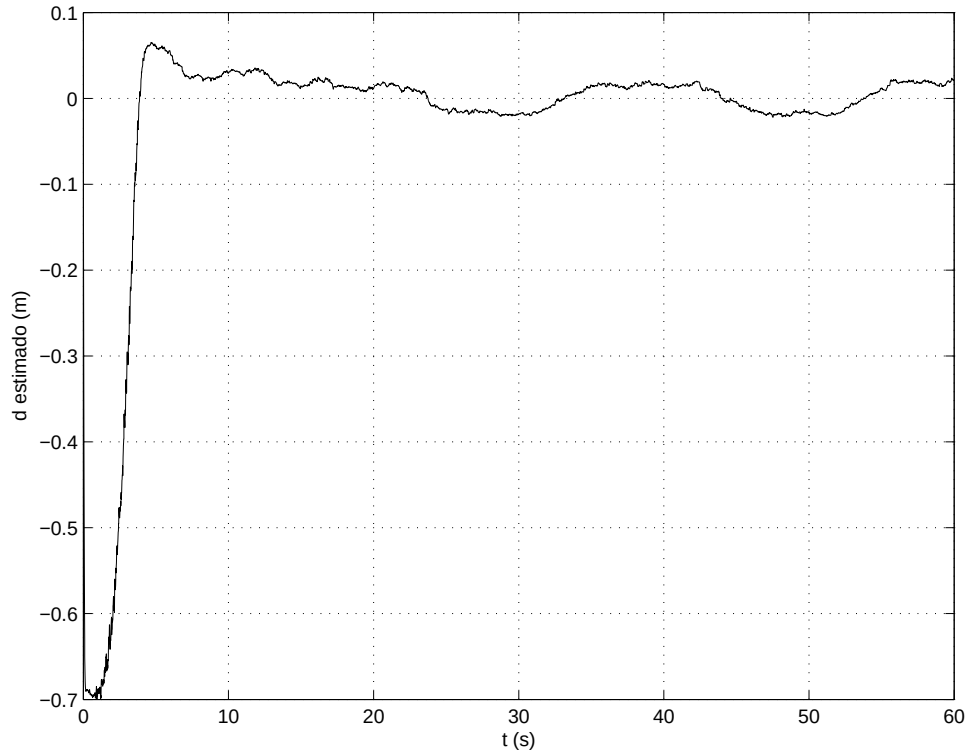

Figura 5.22 – Comportamento de $\hat{d}$ para o caso ruidoso.

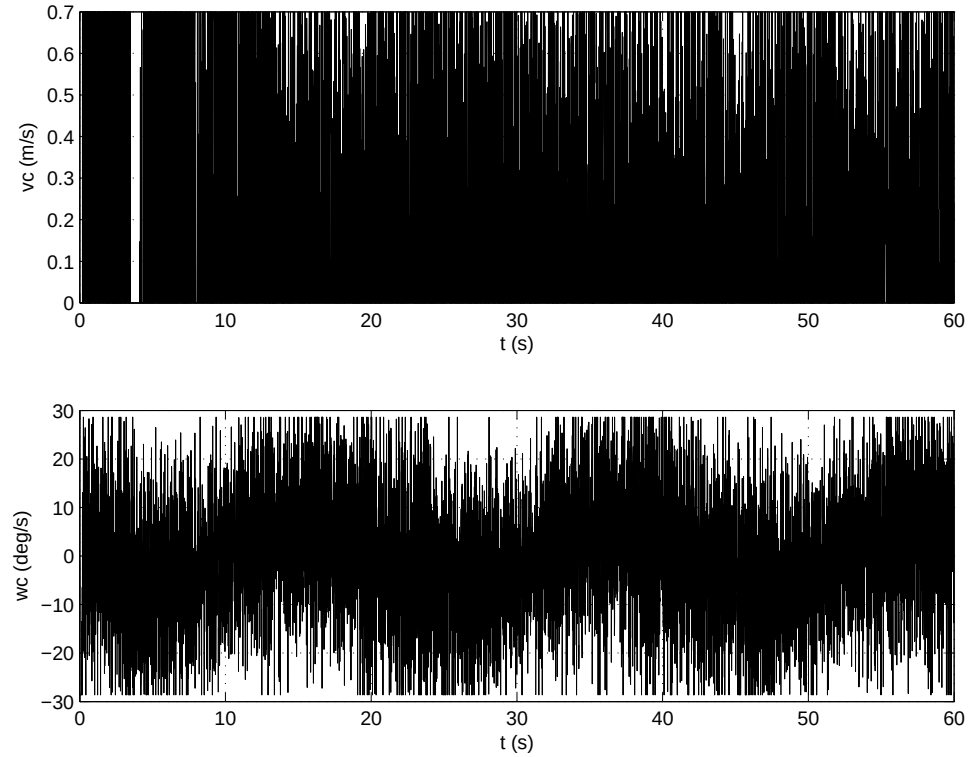

Figura 5.23 – Sinal dos atuadores para o caso ruidoso.

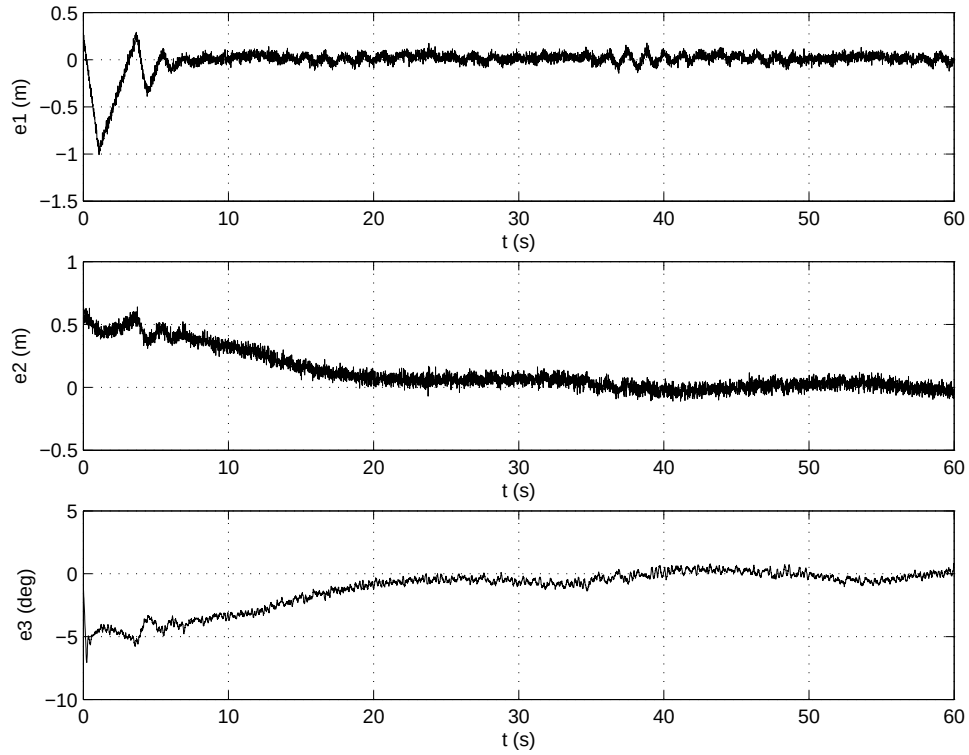

Figura 5.24 – Comportamento dos erros na presença de ruído com filtro.

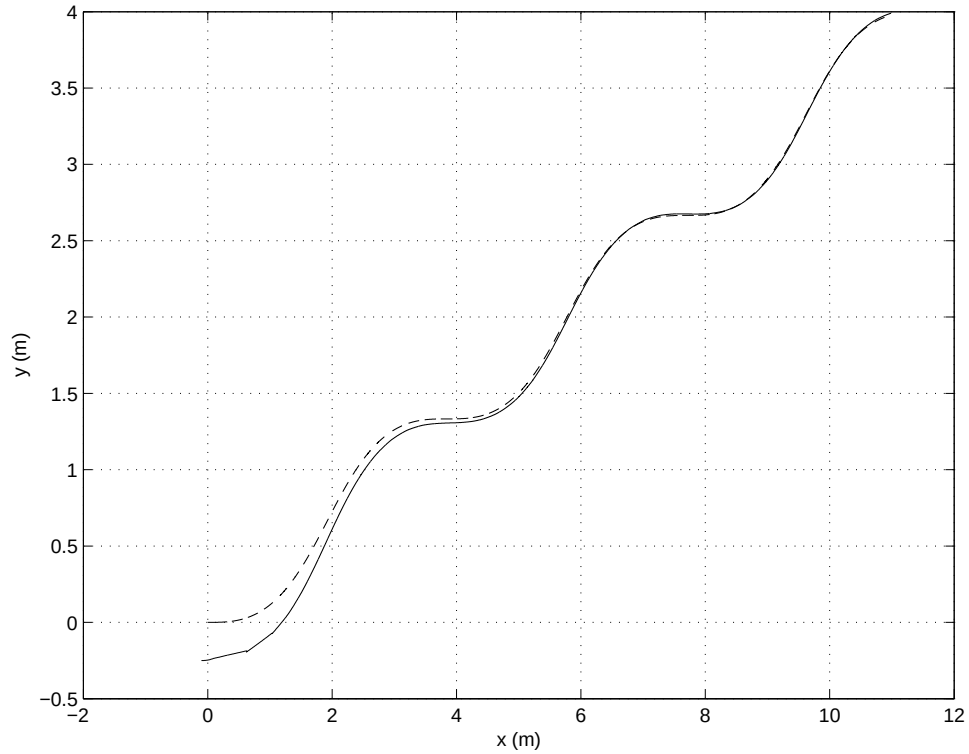

Figura 5.25 – Trajetória pré-gravada (pontilhada) e realizada(linha cheia) para o caso ruidoso com filtro.

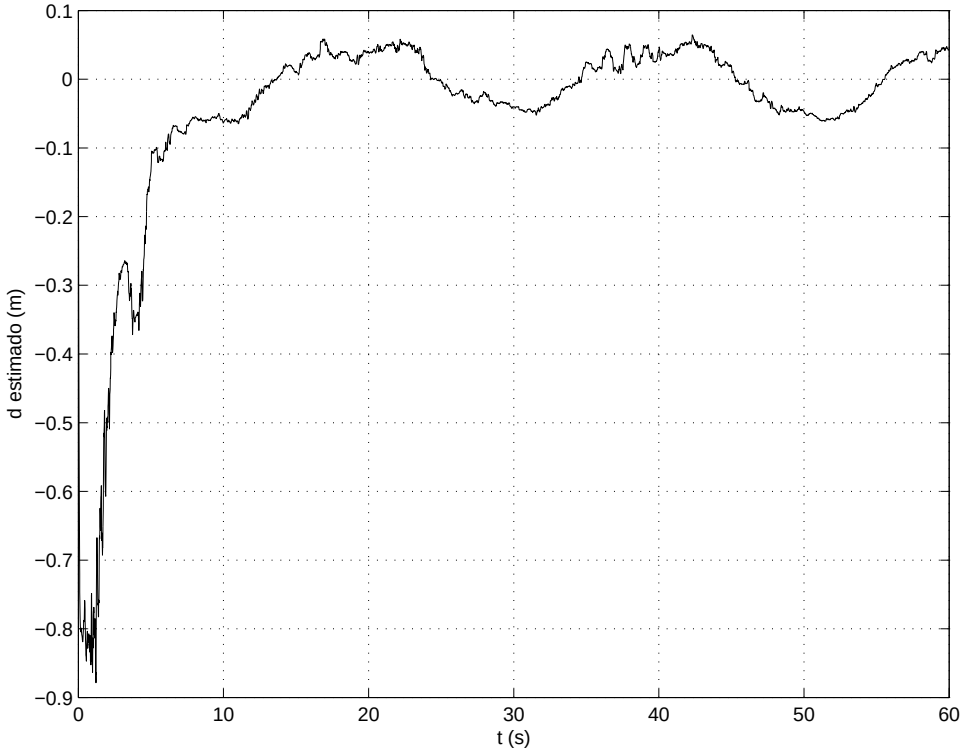

Figura 5.26 – Comportamento de $\hat{d}$ para o caso ruidoso com filtro.

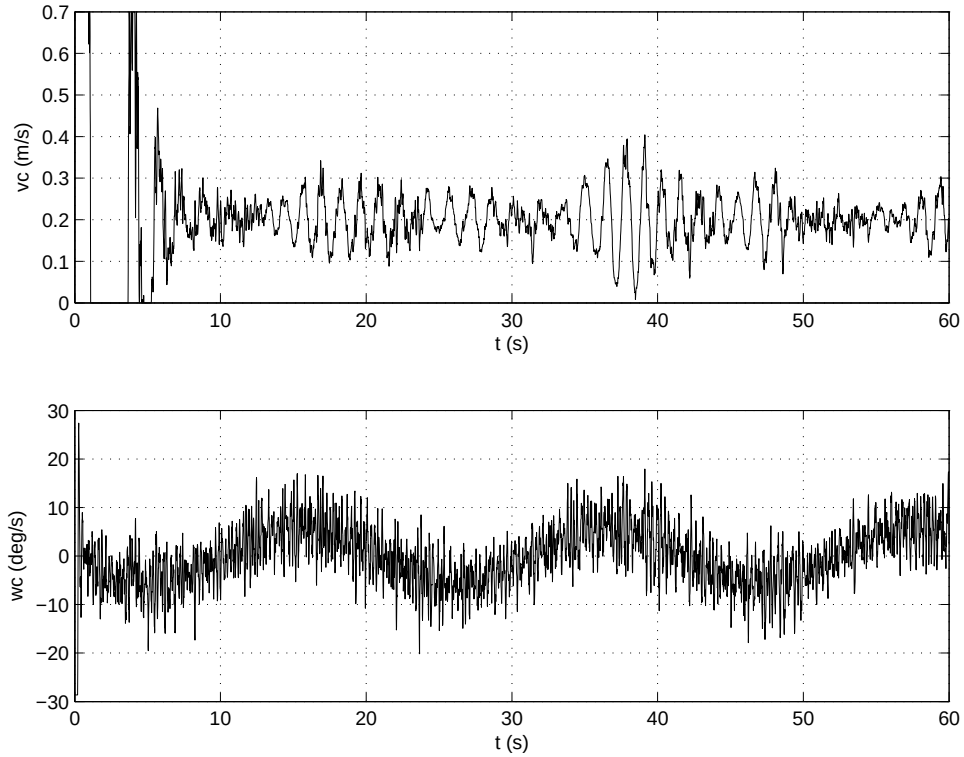

Figura 5.27 – Sinal dos atuadores para o caso ruidoso com filtro.

uma delas foram calculados os erros médios quadráticos de posição e postura, assim como os erros finais em x, y e θ, gerando-se um gráfico de distribuição. Os gráficos para cada um dos erros são mostrados nas Figuras 5.28, 5.29, 5.30, 5.31 e 5.32.

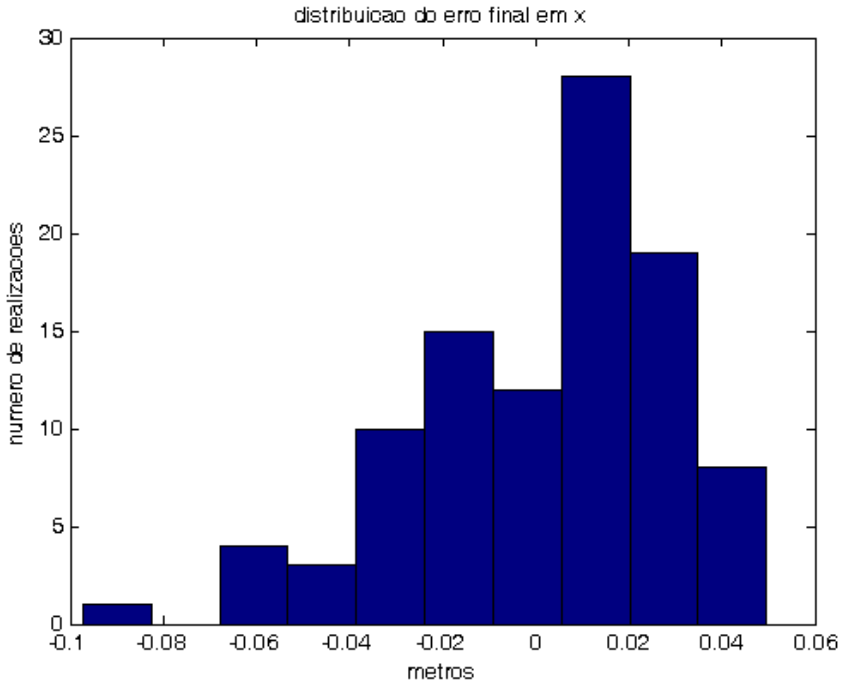

Figura 5.28 – Distribuição do erro final de posição no eixo x em função das realizações.

Em seguida, o mesmo procedimento foi seguido e realizou-se 100 simulações, porém agora com filtros, obtendo-se os gráficos de erros mostrados nas Figuras 5.33, 5.34, 5.35, 5.36 e 5.37.

Assim, percebeu-se que em geral a inserção dos filtros causa uma piora na distribuição dos erros, aumentando a média e principalmente a variância das distribuições. De certa forma, isso já era esperado uma vez que os controles ficam mais suaves devido à presença dos filtros. De qualquer forma, nesse caso, há saturação apenas no início do processo, assim como ocorria no caso livre de ruídos.

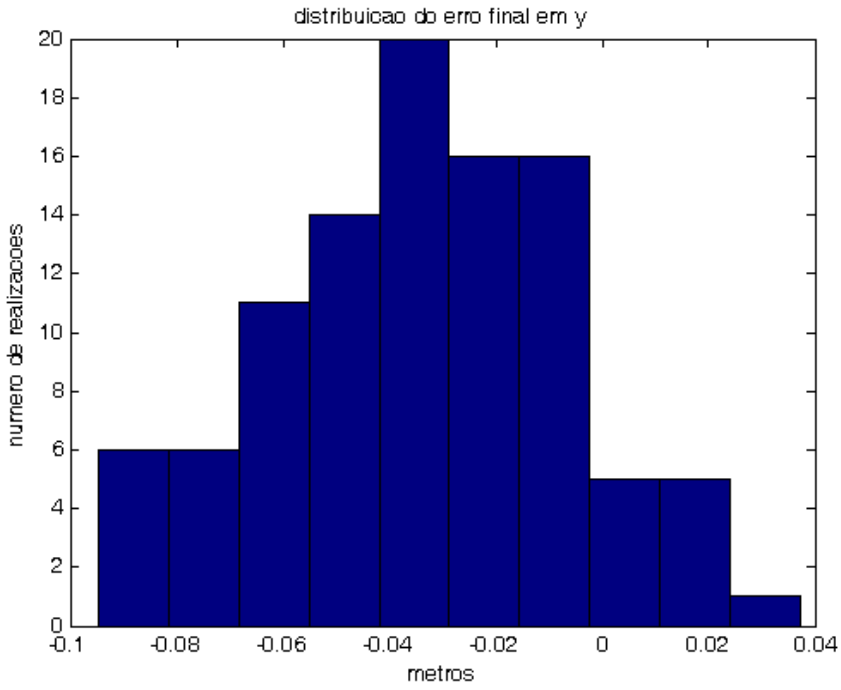

Figura 5.29 – Distribuição do erro final de posição no eixo y em função das realizações.

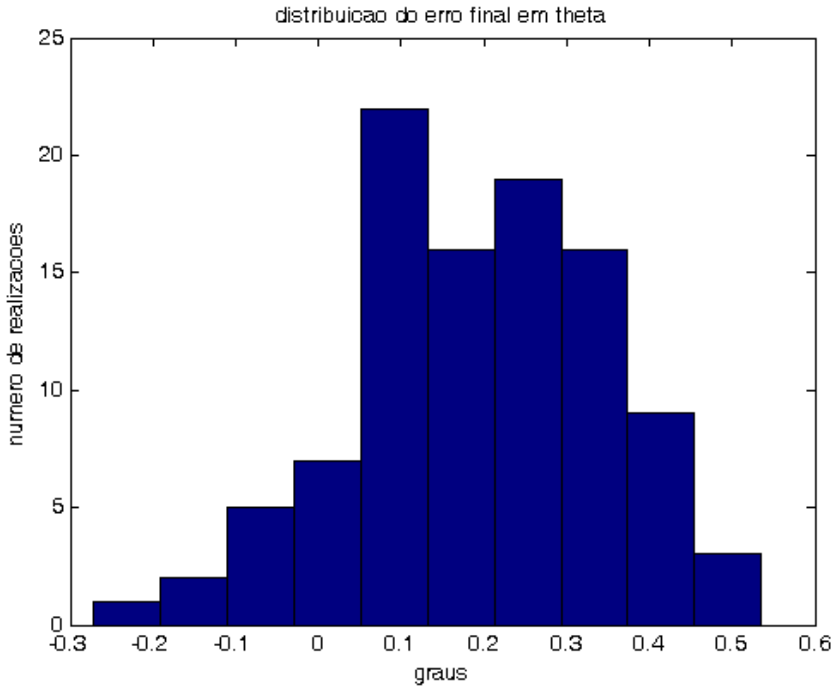

Figura 5.30 – Distribuição do erro final de orientação em função das realizações.

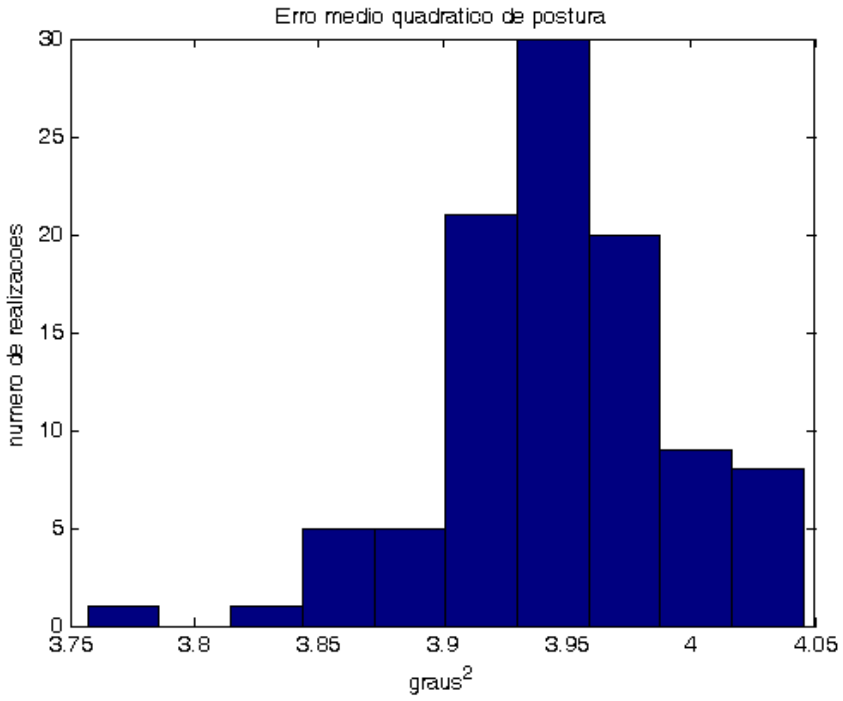

Figura 5.31 – Distribuição do erro médio quadrático de orientação em função das realizações.

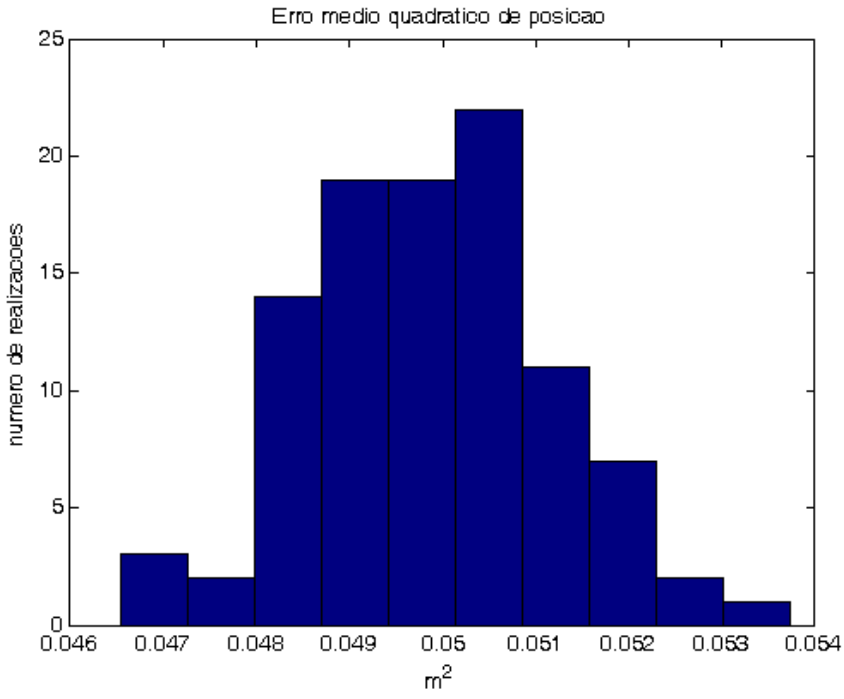

Figura 5.32 – Distribuição do erro médio quadrático de posição em função das realizações.

5.2.2 Rastreamento de Alvo Móvel

Nessa seção são apresentados os resultados de simulação obtidos, para o rastreamento de alvos móveis, usando controle tipo *fuzzy* mostrado na Seção 4.1, e o controle cinemático

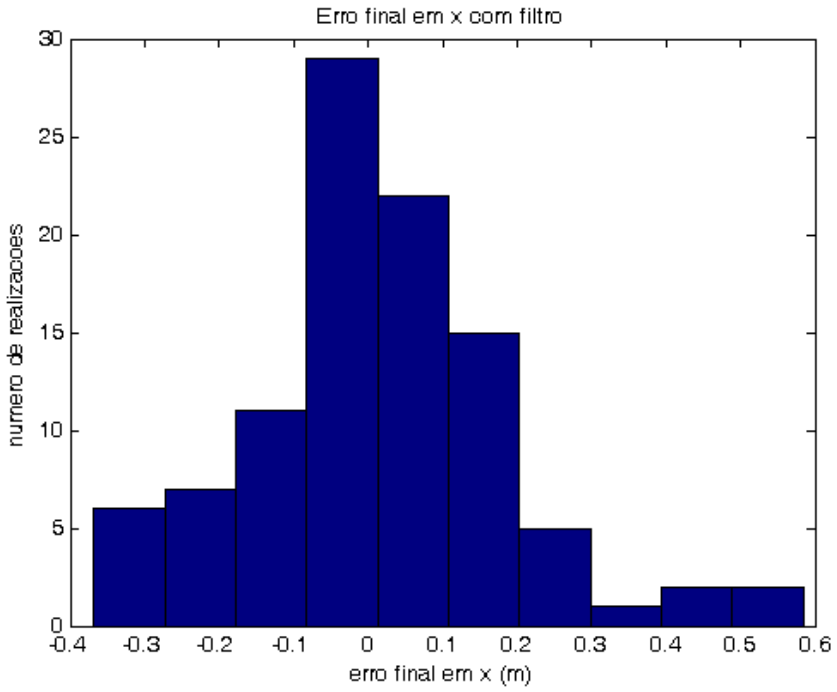

Figura 5.33 – Distribuição do erro final de posição no eixo x em função das realizações com filtro.

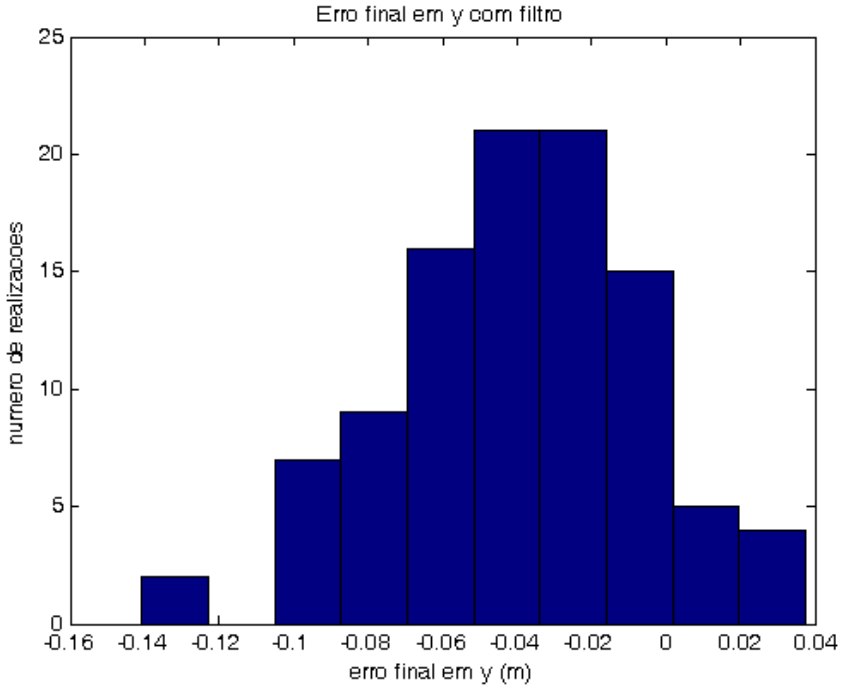

Figura 5.34 – Distribuição do erro final de posição no eixo y em função das realizações com filtro.

mostrado nas Equações 4.5 e 4.6.

Para a realização das simulações escolheu-se duas trajetórias. Na primeira o alvo

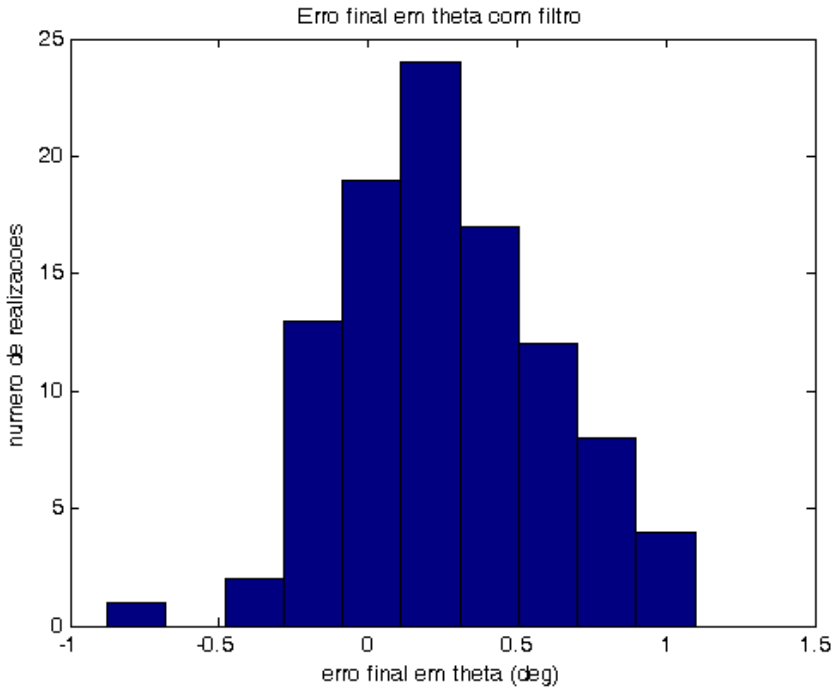

Figura 5.35 – Distribuição do erro final de orientação em função das realizações com filtro.

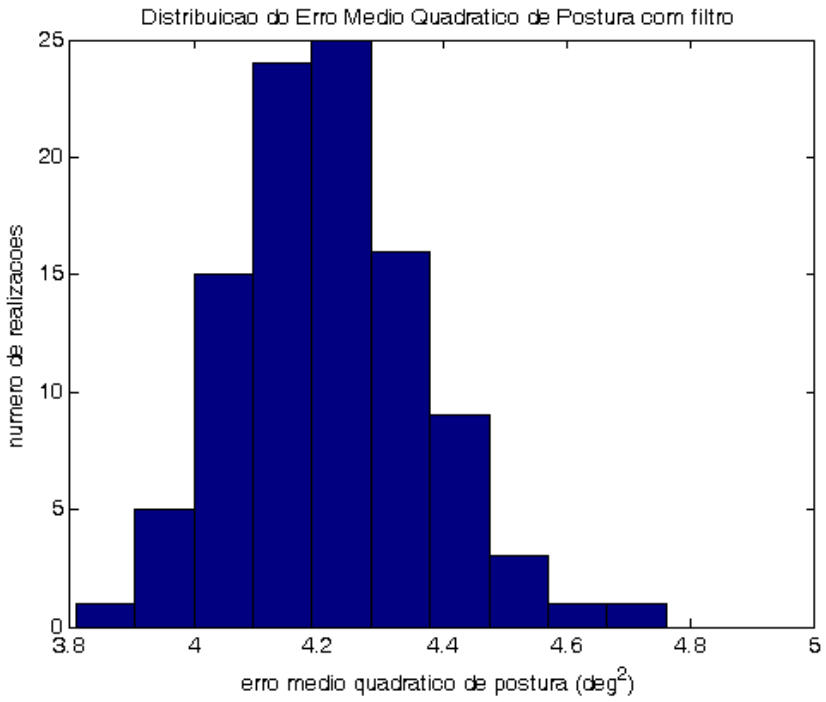

Figura 5.36 – Distribuição do erro médio quadrático de orientação em função das realizações.

move-se em linha reta com velocidade constante por uma distância de 3 m. Na outra, o alvo se desloca por 3 metros em linha reta e em seguida entra em uma curva circular

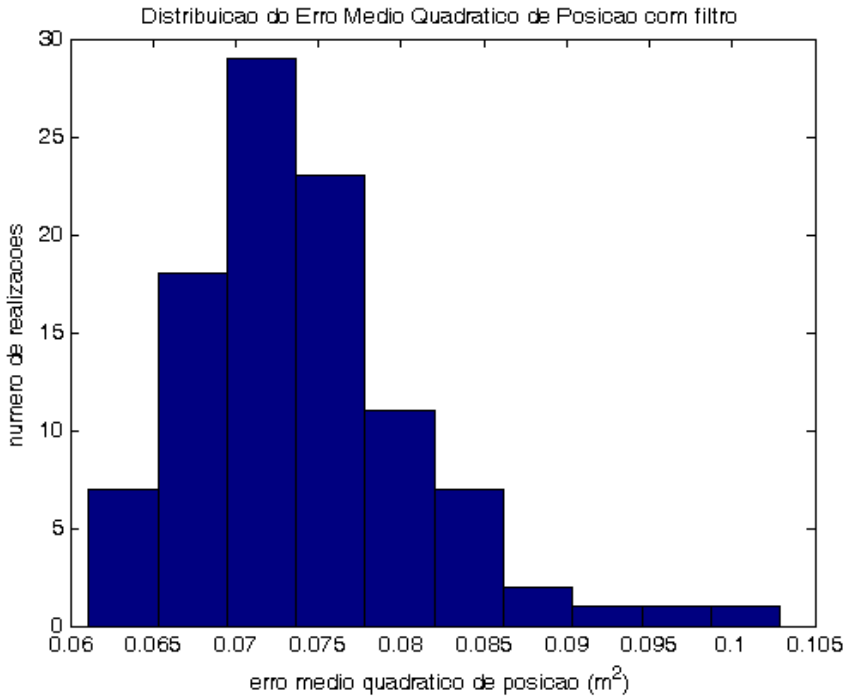

Figura 5.37 – Distribuição do erro médio quadrático de posição em função das realizações com filtro.

assim como proposto em [4]. Em ambos os casos a velocidade do alvo é mantida constante e igual a 3 m/s. A distância inicial entre robô e alvo é de 0.5 m e o ângulo entre eles é de 25º. Deseja-se que o robô mantenha uma distância fixa de 0.20 m e defasagem angular nula. Por fim, com o objetivo de comparar os resultados, os erros médios quadráticos de posição e orientação foram calculados.

5.2.2.1 Utilizando controle *fuzzy*

Para a trajetória em linha reta obteve-se o resultado mostrado na Figura 5.38. Os erros médios quadráticos foram de 0.0625 m^2 e 0.4118 $graus^2$.

Já para a trajetória mista (linha reta + curva circular) o resultado obtido é mostrado na figura 5.39. Para esse caso os erros médios quadráticos foram de 0.0581 m^2 para posição e 3.2330 $graus^2$ para orientação.

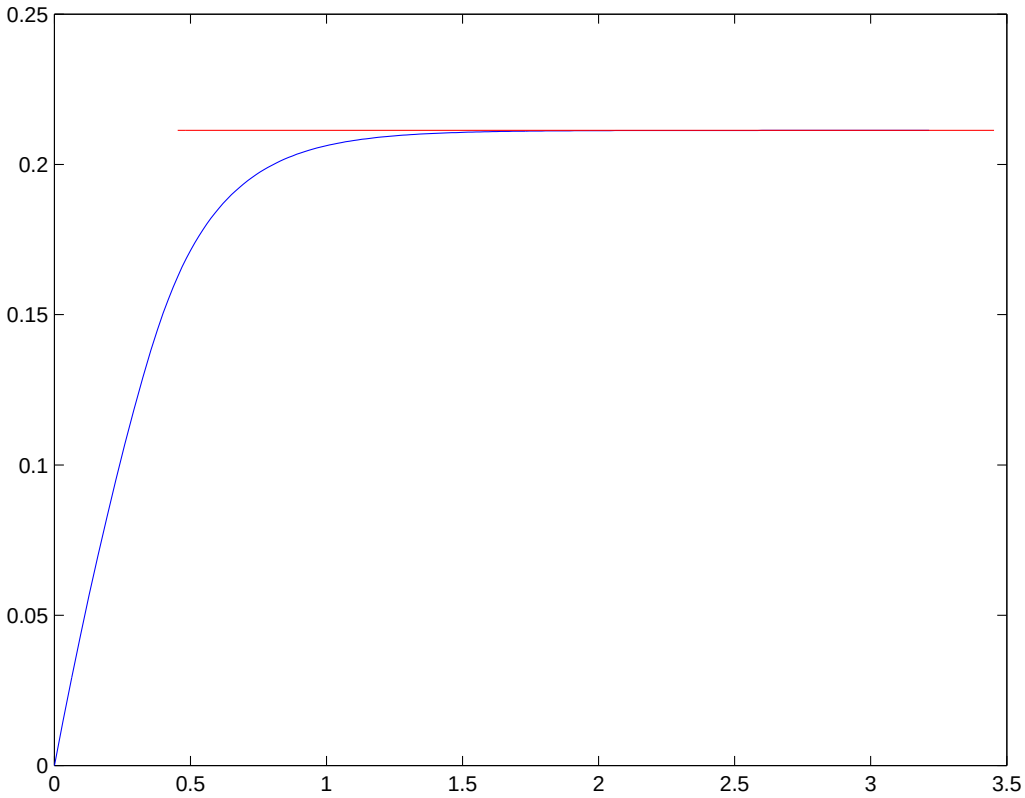

Figura 5.38 – Comportamento do alvo (em vermelho) e do robô (em azul). Eixos em metros.

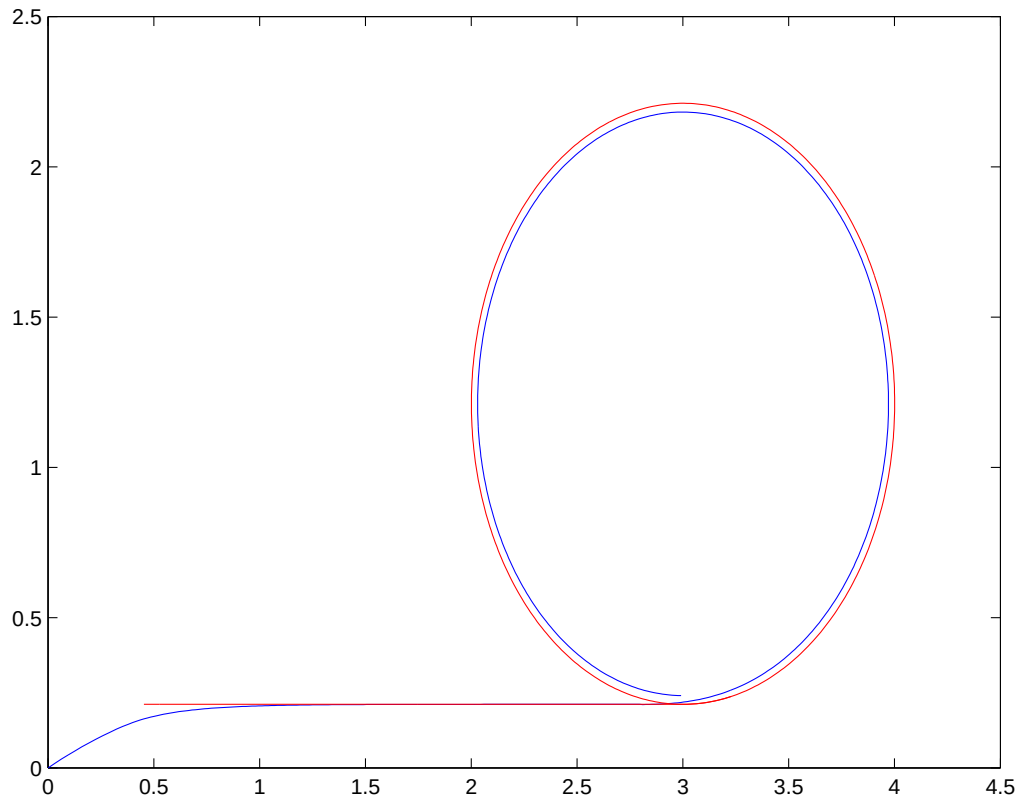

Figura 5.39 – Comportamento do alvo (em vermelho) e do robô (em azul).Eixos em metros.

Além disso, são mostrados também, nas Figuras 5.40 e 5.41 os gráficos dos controles aplicados e do comportamento dos erros de distância e orientação relativa.

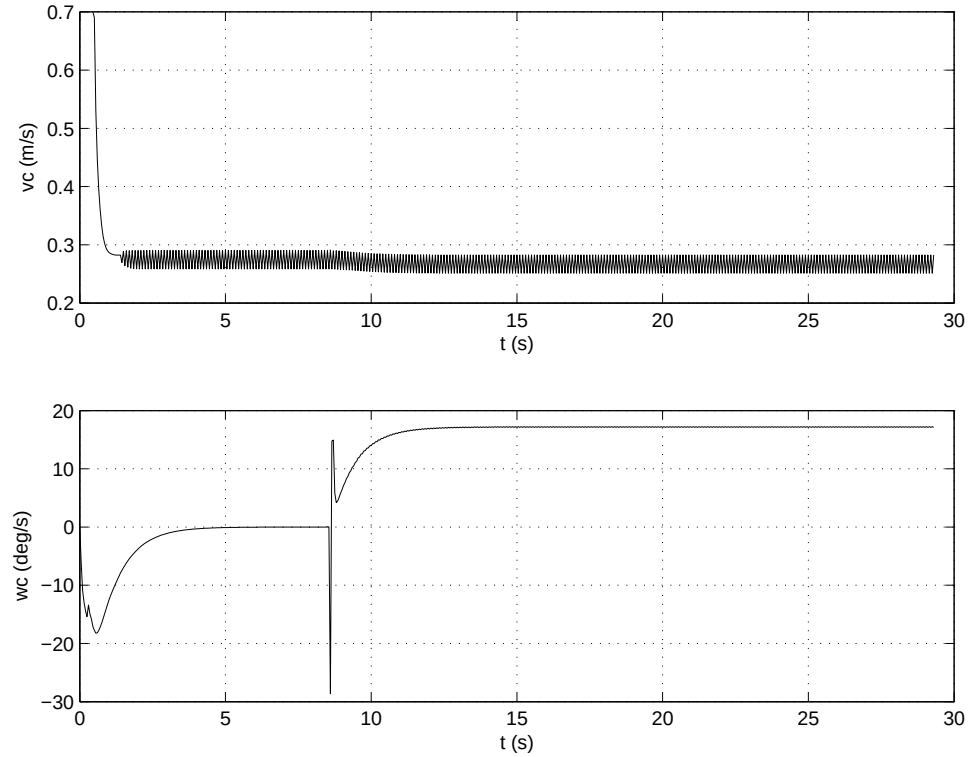

Figura 5.40 – Comportamento dos sinais de controle de velocidade linear (acima) e angular (abaixo).

A seguir, mostra-se os resultados obtidos utilizando o controle cinemático e faz-se as comparações entre ambos.

5.2.2.2 Utilizando controle cinemático

Uma vez que a lei de controle mostrada nas Equações 4.5 e 4.6 está em coordenadas cartesianas e o problema de interesse nesse trabalho é descrito em coordenadas polares, algumas pequenas modificações foram necessárias. Além disso, essa mesma lei de controle foi desenvolvida para levar os erros para zero, e aqui deseja-se que o robô siga o alvo a uma determinada distância.

Para tanto a distância D_{ar} mostrada na Figura 5.42, que será fornecida pela câmera

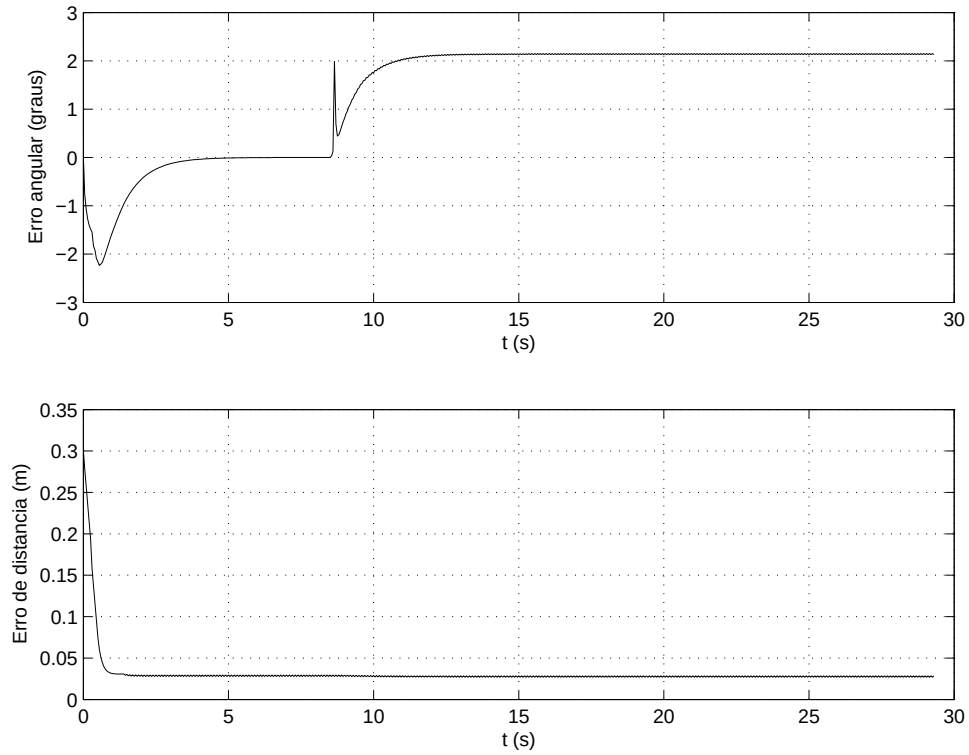

Figura 5.41 – Comportamento dos erros angular (acima) e de distância (abaixo).

deve ser usada para determinar a distância d que será informada ao sistema de controle, já que deseja-se que o robô se mantenha a d_{des} do alvo.

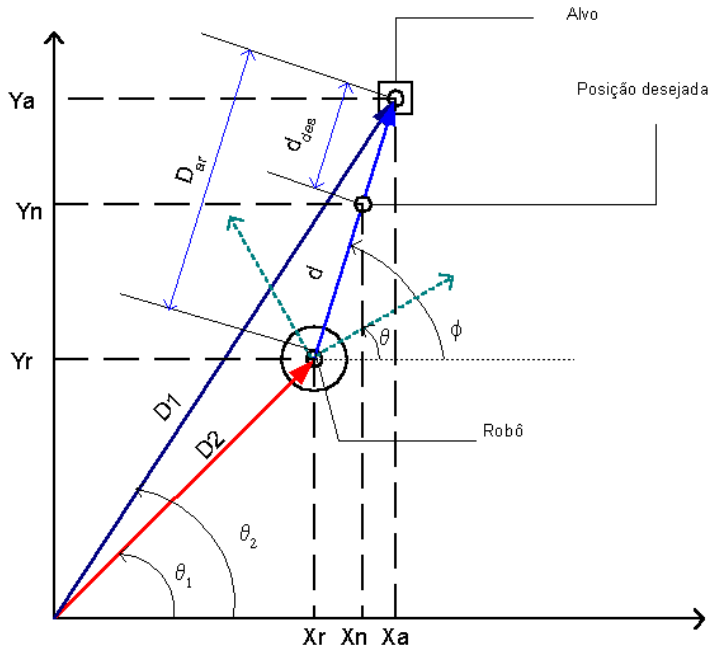

Figura 5.42 – Sistema de coordenadas para determinação dos erros de posição e postura relativos entre alvo e robô.

Desse modo, percebe-se que

$$d = D_{ar} - d_{des} \tag{5.5}$$

Da mesma forma, a Figura 5.42 pode ser usada para mostrar o resultado exibido nas Equações 5.6 e 5.7.

$$X_n = X_r + d\cos(\phi) \tag{5.6}$$

$$Y_n = Y_r + d\sin(\phi). \tag{5.7}$$

Entretanto, interessa para o sistema de controle a distância relativa entre alvo e robô, que aliás, será o dado disponível no caso prático, uma vez que a câmera só é capaz de fornecer essa informação conforme já foi visto na Seção 3.1. Dessa forma, os erros e_1, e_2 e e_3 podem ser expressos como na Equação 5.8.

$$\begin{aligned} e_1 &= X_n - X_r = d\cos(\phi) \\ e_2 &= Y_n - Y_r = d\sin(\phi) \\ e_3 &= \theta - \phi \end{aligned} \tag{5.8}$$

Note-se portanto, que conforme era desejado, apenas as grandezas relativas e_1, e_2 e e_3, precisam ser conhecidas. Na prática, a câmera fornecerá parâmetros para se determinar D_{ar}, permitindo assim a utilização dessa mesma sistemática.

Uma vez feitas essas considerações foram efetuadas simulações para as mesmas condições

descritas para o controle *fuzzy*, usando-se as Equações 4.5 e 4.6 com as constantes mostradas na Equação 5.9 determinadas por tentativa e erro.

$$k_1 = 10, \quad k_2 = 12, \quad k_3 = 46 \tag{5.9}$$

Para a trajetória em linha reta obteve-se o resultado mostrado na figura 5.43. Os erros médios quadráticos foram de 0.0503 m^2 e 0.4438 $graus^2$.

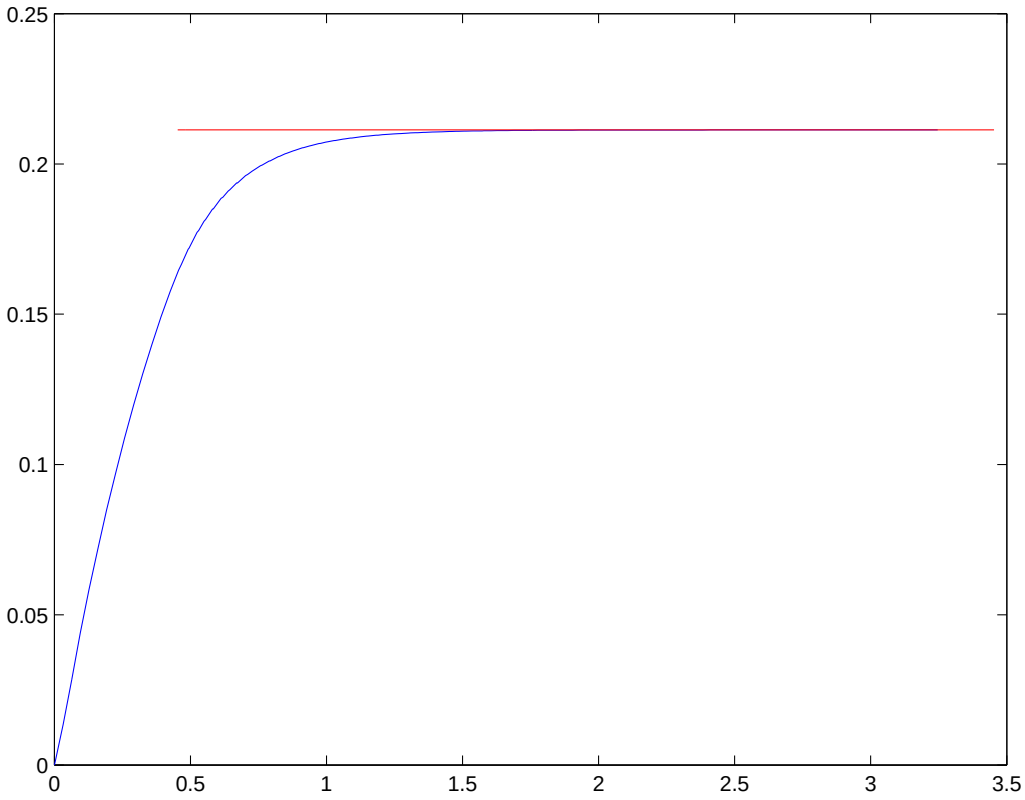

Figura 5.43 – Comportamento do alvo (em vermelho) e do robô (em azul). Eixos em metros.

Já para a trajetória mista o resultado obtido é mostrado na Figura 5.44. Para esse caso os erros médios quadráticos foram de 0.0539 m^2 para posição e 3.2625 $graus^2$ para orientação.

Além disso, são mostrados também, nas Figuras 5.45 e 5.46 os gráficos dos controles

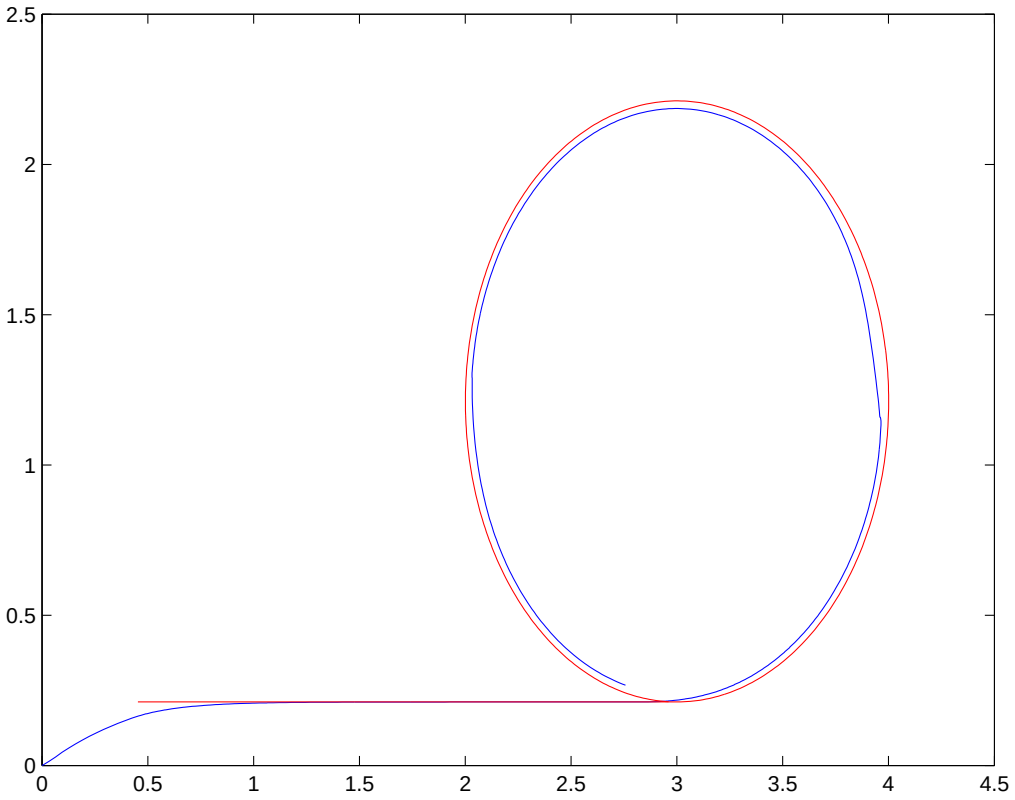

Figura 5.44 – Comportamento do alvo (em vermelho) e do robô (em azul). Eixos em metros.

aplicados e do comportamento dos erros de distância (d na Figura 5.42) e orientação relativa.

Por fim, a Tabela 5.1 mostra uma comparação entre os erros médios quadráticos para ambas as estratégias de controle em cada tipo de trajetória. Nesse caso, pode-se perceber que as diferenças são muito pequenas não sendo possível portanto concluir que qualquer dos controladores seja melhor que o outro.

Com relação ao controle cinemático uma observação interessante é em relação ao resultado mostrado na Figura 5.44, onde pode-se notar uma pequena "quebra" durante a circunferência. Isso se deve a um problema, já descrito em [5] para o caso em que todos os erros vão para zero. Nesse caso o controle angular perde a sensibilidade aos erros, e só volta a recuperá-la após os mesmos atingirem um valor maior.

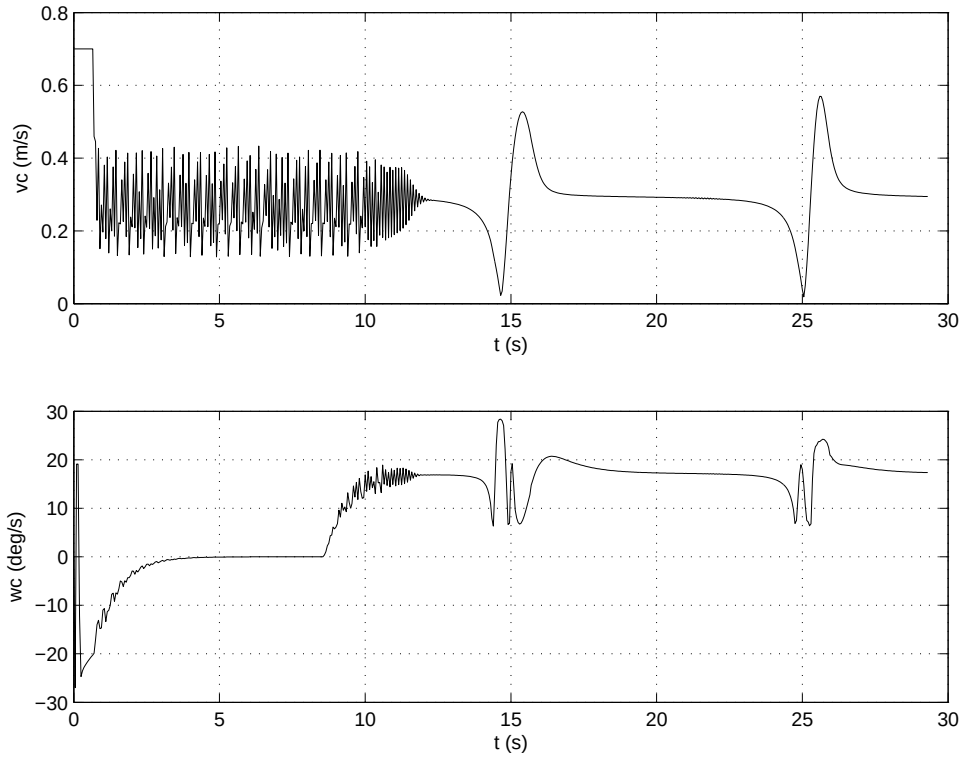

Figura 5.45 – Comportamento dos sinais de controle de velocidade linear (acima) e do angular (abaixo).

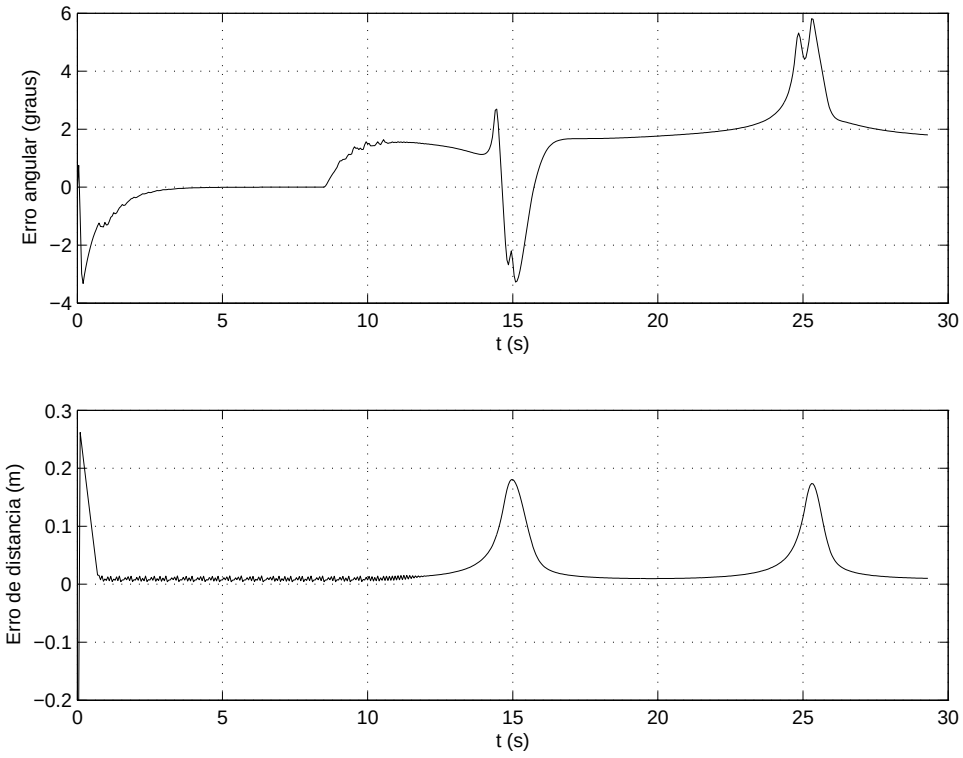

Figura 5.46 – Comportamento dos erros de distância (acima) e angular (abaixo).

Tabela 5.1 – Comparação entre os erros médios quadráticos para as duas estratégias de controle.

Erro Médio Quadrático (Trajetória em Linha Reta)			
Fuzzy		Controle Cinemático (Lyapunov)	
Angular	0.4118 (graus^2)	Angular	0.4438 (graus^2)
Linear	0.0625 (m^2)	Linear	0.0503 (m^2)
Erro Médio Quadrático (Trajetória Mista)			
Fuzzy		Controle Cinemático (Lyapunov)	
Angular	3.2330 (graus^2)	Angular	3.2625 (graus^2)
Linear	0.0581 (m^2)	Linear	0.0539 (m^2)

Note-se que isso faz com que os erros angulares sejam sempre maiores para o caso do controle cinemático desenvolvido via Lyapunov do que para o controle tipo *fuzzy*.

6 Sistemática de Implementação

6.1 Rotinas em C++

Para a implementação e execução em tempo real do algoritmo de detecção visual [4] e da lei de controle [5] simulados no Capítulo 5 fez-se necessária a tradução para C++ de todas as rotinas escritas em MatLab.

Essas rotinas foram agrupadas em dois programas instalados um no computador remoto e o outro no computador embarcado do robô. O primeiro deles, escrito em C++ usando o compilador *Borland C Builder* ®, serve para automatizar o processo de seleção manual do alvo na primeira iteração e exibir as imagens feitas pelo robô em tempo real conforme mostra a Figura 6.1. O outro, compilado no próprio robô utilizando o compilador do Linux, é responsável por adquirir a imagem atual e exibi-la na tela do computador remoto (via servidor de *ftp*) até que o usuário faça a seleção do alvo e inicie o processo de rastreamento (Figura 6.2).

A utilização de ferramentas gráficas no desenvolvimento do programa remoto permitiu chegar-se ao *lay out* mostrado na Figura 6.3. Assim, uma vez feita a seleção do alvo na imagem corrente, basta um *click* no botão OK para que o programa que roda no robô inicie a localização do alvo na imagem atual além do cálculo e aplicação do controle.

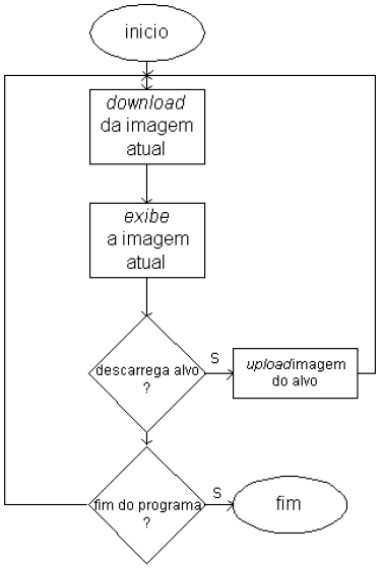

Figura 6.1 – Fluxograma simplificado do programa implementado no computador remoto usando Borland C Builder.

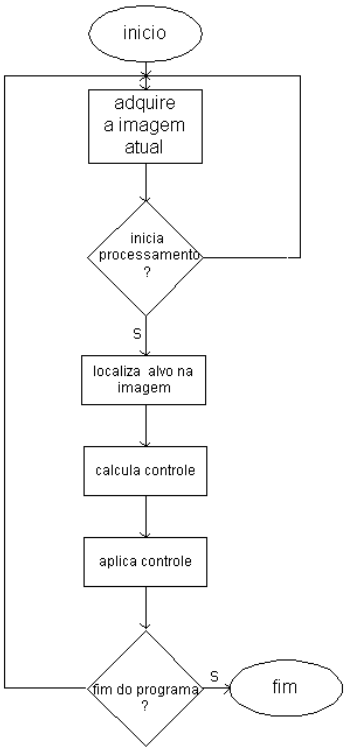

Figura 6.2 – Fluxograma simplificado do programa implementada no robô Magellan Pro.

O processo só é interrompido quando qualquer tecla do computador remoto é pressionada, parando assim o veículo. Os demais botões mostrados na Figura 6.3 têm a função de iniciar a conexão com ftp do robô ("conect"), reinicializar as variáveis que definem a seleção do alvo ("refresh") e cancelar a conexão ("cancel"). A caixa "dados lidos" exibe os

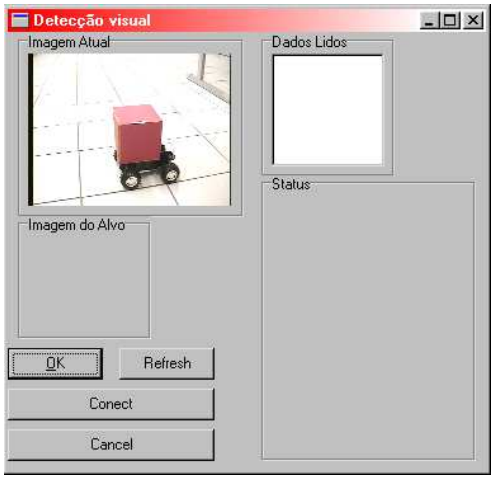

Figura 6.3 – Tela do programa remoto utilizado para selecionar o alvo e acompanhar o alvo "visto" pelo robô.

limites superior esquerdo e inferior direito do retângulo de seleção do alvo, "Imagem do alvo", exibe o recorte da imagem atual correspondente à seleção do alvo e "Status" exibe o andamento dos processos de conexão e troca de dados entre o computador remoto e o robô.

6.2 Experimentos em Tempo Real

Depois de implementadas todas as rotinas descritas na seção anterior, partiu-se para a execução dos experimentos em tempo real. Vale ressaltar aqui que problemas com o link via rádio que permite a conexão com o robô consumiram um tempo considerável, sendo que a solução encontrada foi a conexão do robô via cabo de rede normal, o que limitou o tamanho das distâncias percorridas pelo mesmo.

Além disso, percebeu-se também que os valores de ganhos obtidos por simulação levavam a respostas muito bruscas do controle que poderiam danificar a estrutura do robô, efeito não previsível pelos modelos utilizados. Com isso, foi necessário reduzir essas constantes e também a velocidade do alvo para evitar danos.

Para os experimentos três trajetórias foram escolhidas, seguindo o que foi feito nas etapas de simulação. A primeira delas, com o objetivo de testar o funcionamento da lei de controle angular, foi produzida movendo-se o alvo em frente ao robô evitando mudar a distância entre ambos. O controle linear nesse caso foi desabilitado pois pretendia-se avaliar o funcionamento do controle angular.

No segundo experimento, com ambos os controles habilitados fez-se com que o alvo andasse em linha reta por aproximadamente 6m, para avaliar o desempenho do controle linear e do controle angular na compensação de possíveis escorregamentos das rodas.

Por fim, realizou-se uma trajetória onde o robô anda em linha reta por aproximadamente um metro e depois entra em uma curva circular à direita, diferentemente da trajetória simulada em que a curva era pra esquerda. Essa mudança foi necessária devido a presença do cabo de rede já mencionado, evitando-se assim que o mesmo sofresse danos.

7 Resultados dos Experimentos em Tempo Real

Neste capítulo serão apresentados os resultados obtidos nos experimentos em tempo real utilizando o robô móvel Magellan Pro. Para cada um dos experimentos são mostrados os gráficos de erro de distância, erro angular e sinais de controle aplicados (linear e angular) além de fotos seqüenciais obtidas por câmera externa ao robô mostrando seu comportamento durante as trajetórias. A função das fotografias é fornecer informações sobre a posição e orientação globais do robô no ambiente de trabalho, uma vez que a estratégia de controle utiliza apenas informações relativas ao robô.

Para que as explicações subseqüentes sejam mais facilmente entendidas ambas as leis de controle mostradas na Seção 4.2 estão reescritas nas Equações 7.1 e 7.2.

$$v_c = v_r \cos(e_3) + k_1 e_1 \tag{7.1}$$

$$\omega_c = \omega_r + k_2 v_r e_2 + k_3 v_r \sin(e_3) \tag{7.2}$$

Para todos os experimentos o *sampling time* utilizado foi de 0.7s, limitado pelo alto

tempo de processamento das imagens. A aquisição das imagens foi feita na mesma taxa.

Para a transformação de perspectiva, conforme mostrado no Capítulo 2, os valores de foco e ângulos de *pan* e *tilt* são mostrados na Tabela 7.1.

Tabela 7.1 – Valores dos parâmetros utilizados para transformação de perspectiva.

pan	0^o
tilt	-25^o
Foco	5.4mm

Como pode-se perceber, esses valores são os próprios limites que são apresentados no Apêndice A e extraídos do manual da câmera [29]. Essa estratégia foi escolhida pois na plataforma Mobility não foi encontrada nenhuma rotina que permitisse obter os ângulos da câmera por meio de seus *encoders*, ou mesmo fazer a leitura do foco. A validação dessa abordagem foi feita colocando-se o alvo a distâncias conhecidas e executando o algoritmo de detecção visual, tendo-se obtido um erro médio de 5%, considerado satisfatório.

Caso as informações mostradas na Tabela 7.1 não tivessem sido encontradas um procedimento de calibração da câmera teria sido necessário, como por exemplo o apresentado em [30].

Como alvo foi utilizado um veículo acionado por controle remoto via rádio com um cubo vermelho anexo, conforme mostrado na Figura 7.1, de aproximadamente 15cm de aresta escolhido por ser monocromático e garantir visualização ainda que em perspectiva.

Figura 7.1 – Alvo utilizado nos experimentos em tempo real.

7.1 Experimento 1

Como foi descrito no Capítulo 6, o primeiro experimento realizado teve como objetivo verificar o funcionamento do controle angular, independente do controle linear. As constantes utilizadas foram obtidas por tentativa e erro, em um processo norteado pelos resultados das simulações, porém agora procurando-se obter respostas de controle mais lentas para evitar danos ao robô uma vez que o mesmo bate a parte dianteira inferior no chão quando há variações bruscas de velocidade, efeito não previsível pelos modelos.

As constantes obtidas são mostradas na Equação 7.3, na qual a constante k_1 foi omitida já que o controle de velocidade linear foi desabilitado para esse experimento.

$$k_2 = 1.5, \quad k_3 = 10 \tag{7.3}$$

O procedimento adotado foi o de mover o alvo ao redor do robô tentando manter a distância entre ambos aproximadamente constante, como mostrado na Figura 7.2.

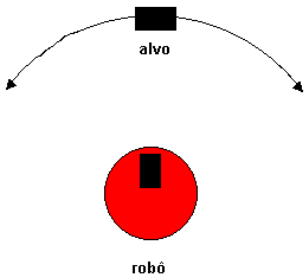

Figura 7.2 – Figura exemplificando o experimento realizado.

Desse modo os resultados obtidos para o erro angular são mostrados na Figura 7.3 e o sinal de controle correspondente é mostrado na Figura 7.4. Pode-se perceber nas figuras um grande erro inicial propositalmente gerado levando o alvo bastante longe de

sua posição original. Nesses gráficos apenas metade do tempo do experimento é mostrada, já que a outra metade foi praticamente igual.

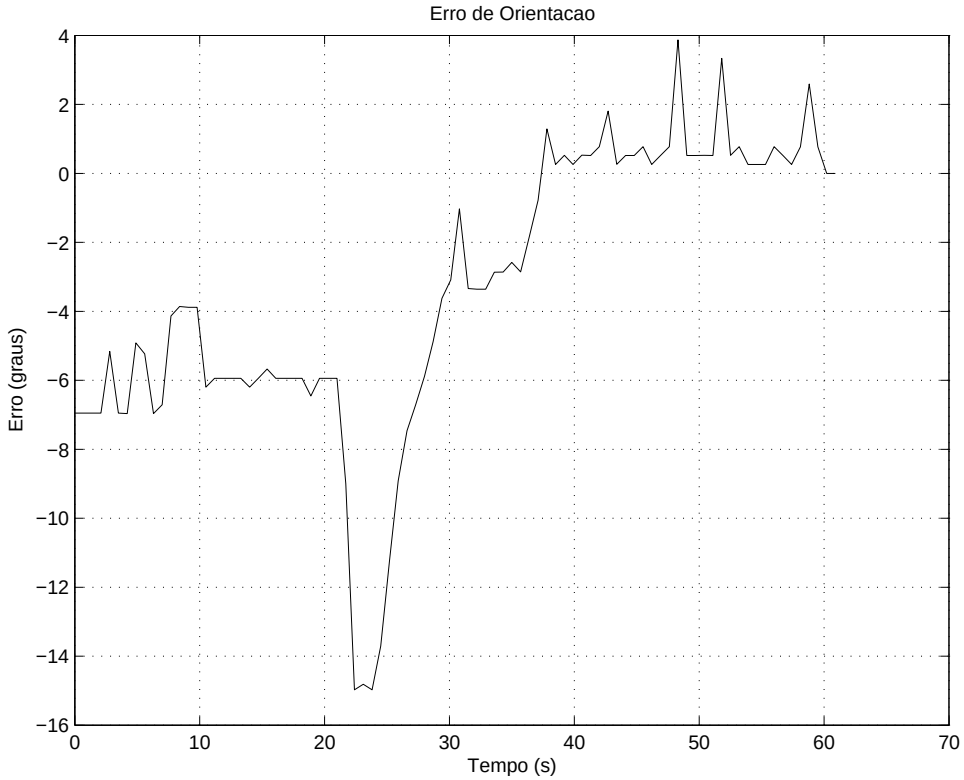

Figura 7.3 – Comportamento do erro angular para o experimento 1.

Finalmente, na Figura 7.5 é apresentada uma seqüência de imagens feita por uma câmera externa ao experimento, na qual pode-se notar o movimento do robô acompanhando o alvo.

7.2 Experimento 2

Nesse experimento o controle de velocidade linear foi novamente habilitado. A distância desejada entre alvo e robô foi definida com o valor de 1m. Para cumprir esse objetivo os seguintes valores de constantes foram obtidos.

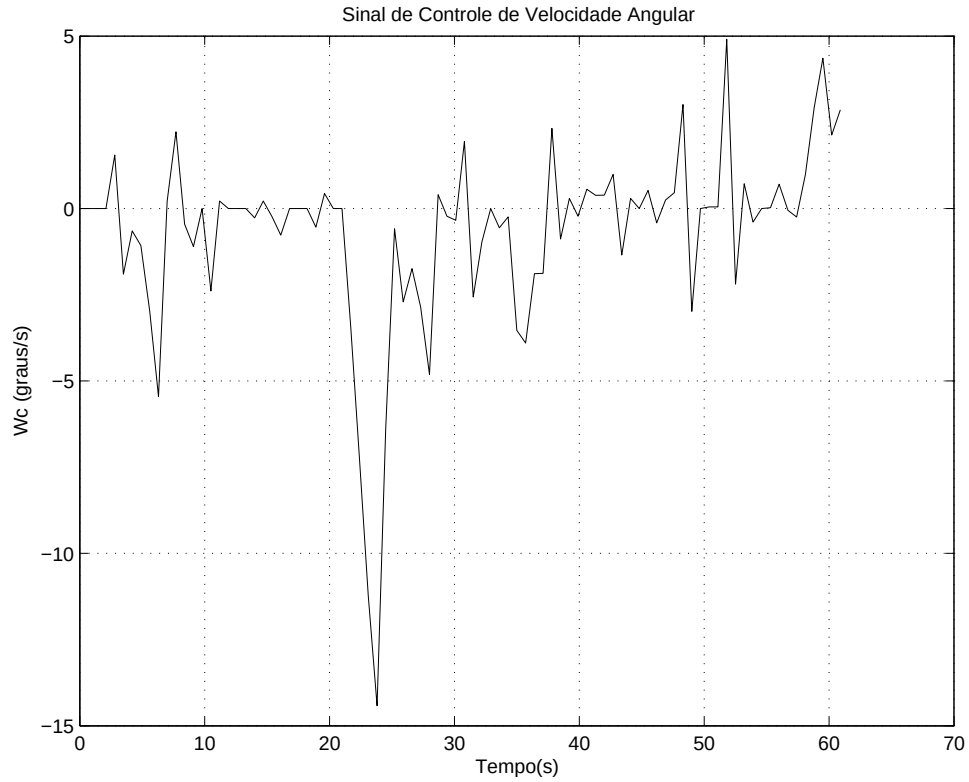

Figura 7.4 – Sinal de Controle angular calculado para o experimento 1.

$$k_1 = 1.0 \quad k_2 = 1.6 \quad k_3 = 12 \tag{7.4}$$

As Figuras 7.6 e 7.7 mostram os resultados obtidos para o erro de distância e erro de orientação, respectivamente. Já as Figuras 7.8 e 7.9 mostram os sinais de controle aplicados.

Finalmente, a Figura 7.10 mostra o conjunto de imagens feito por câmera externa acompanhando o movimento do robô. Nessas imagens pode-se perceber a aproximação do robô em relação ao alvo pelo quadriculado formado pelo ladrilho do piso do laboratório.

Figura 7.5 – Imagens captadas por câmera externa para o experimento 1.

7.3 Experimento 3

Nesse experimento fez-se com que o alvo andasse aproximadamente 1m em linha reta e depois entrasse em uma curva circular (desenhou-se uma circunferência a lápis no chão para guiar o condutor do alvo). A distância desejada entre alvo e robô foi selecionada para 0.6m. Para esse caso as constantes obtidas são mostradas na Equação 7.5.

$$k_1 = 1.0 \quad k_2 = 1.6 \quad k_3 = 12 \tag{7.5}$$

Dessa maneira as figuras 7.11 e 7.12 mostram os resultados obtidos para o erro de distância e erro de orientação, respectivamente. Já as Figuras 7.13 e 7.14 mostram os

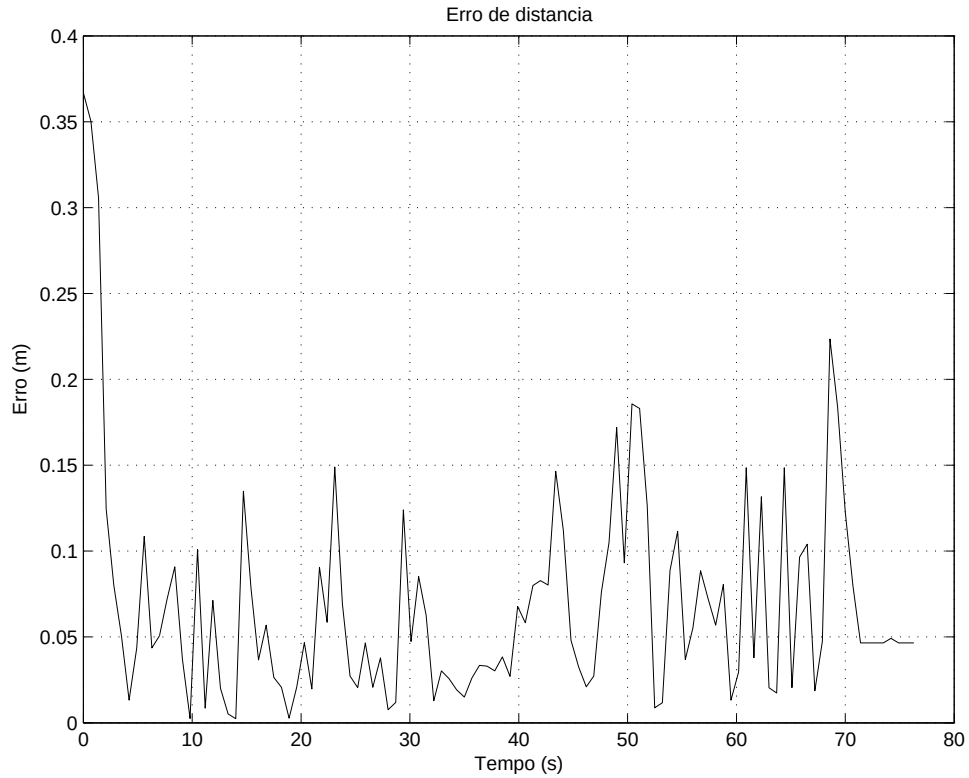

Figura 7.6 – Comportamento do erro de distância para o experimento 2.

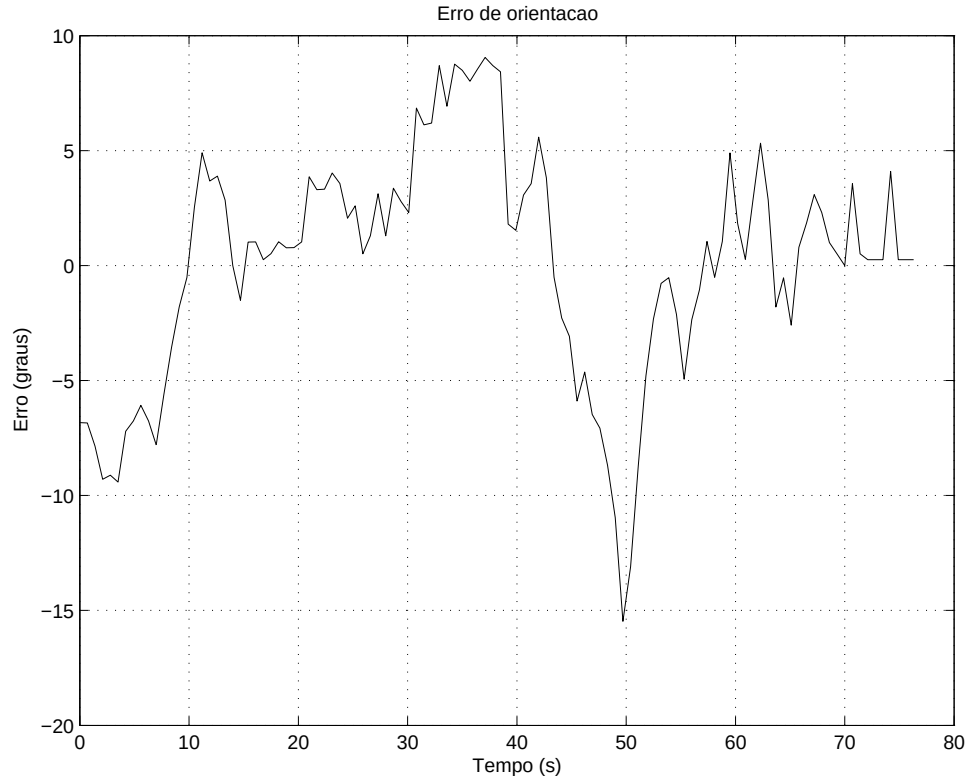

Figura 7.7 – Comportamento do erro de orientação para o experimento 2.

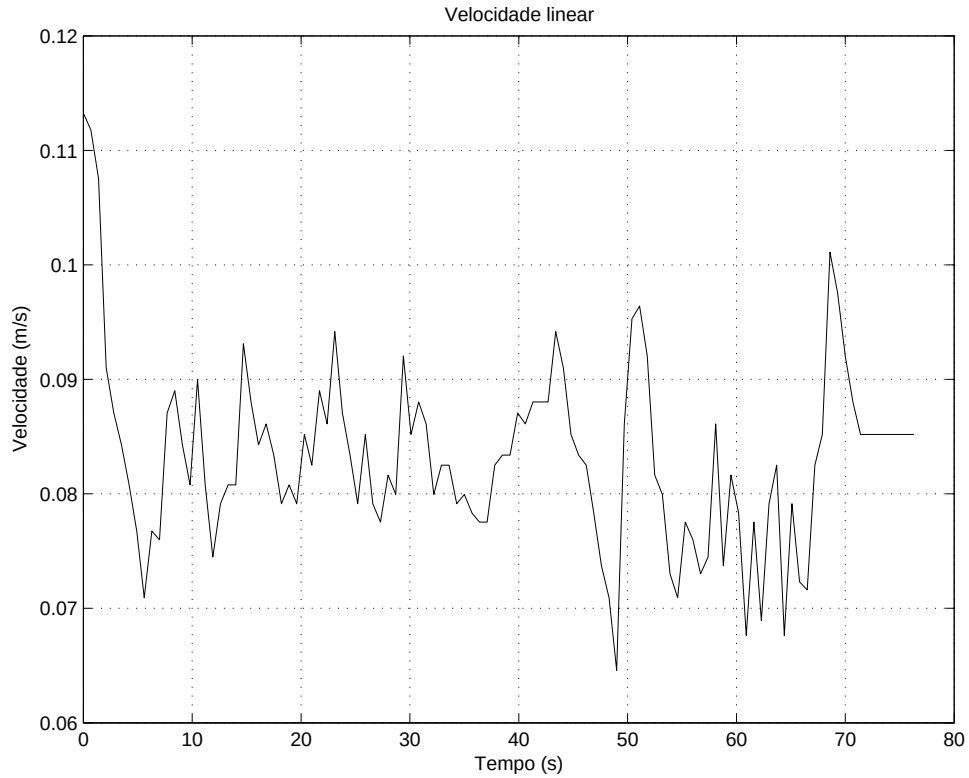

Figura 7.8 – Sinal de controle de velocidade linear para o experimento 2.

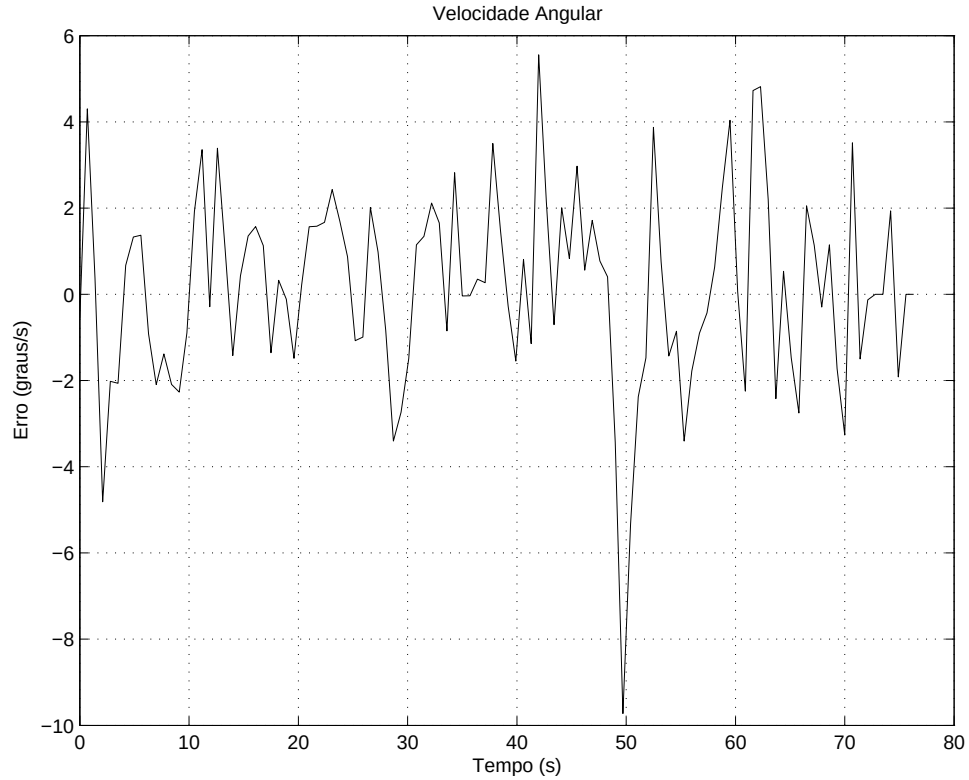

Figura 7.9 – Sinal de controle de velocidade angular para o experimento 2.

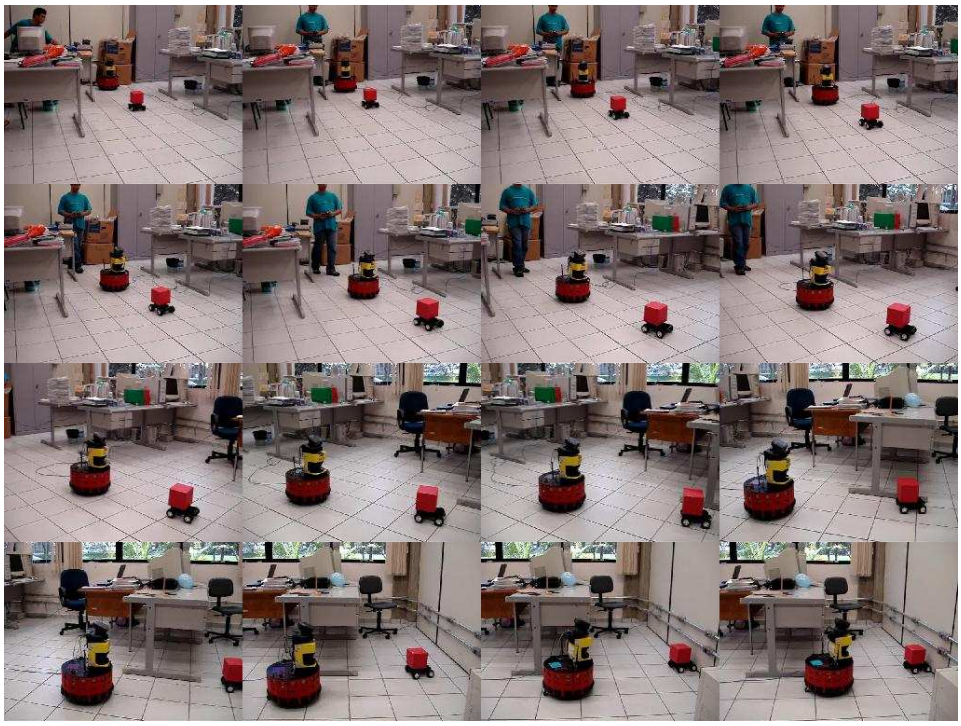

Figura 7.10 – Conjunto de imagens feitas por câmera externa para o experimento 2.

sinais de controle aplicados.

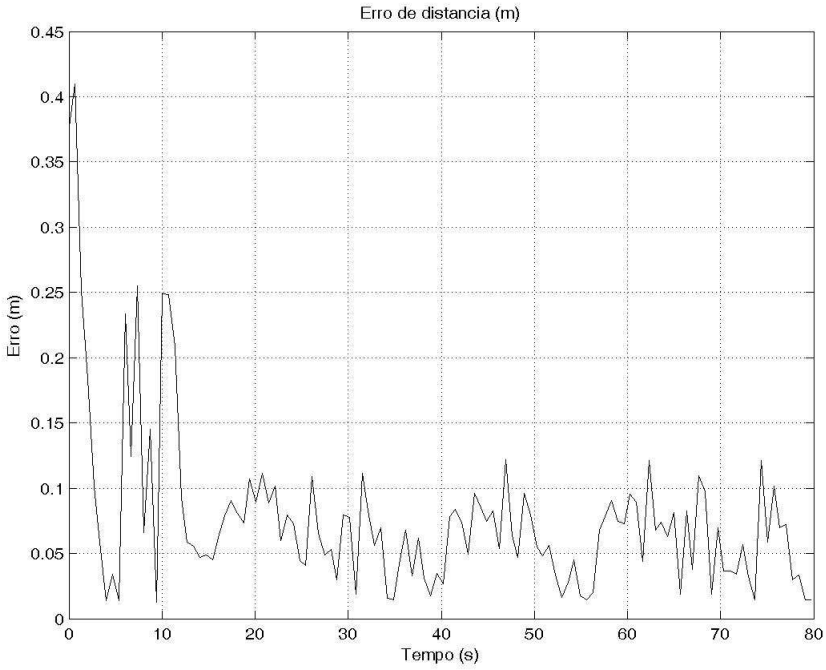

Figura 7.11 – Comportamento do erro de distância para o experimento 3.

Por fim, a Figura 7.15 mostra um conjunto de fotos tiradas por câmera externa ex-

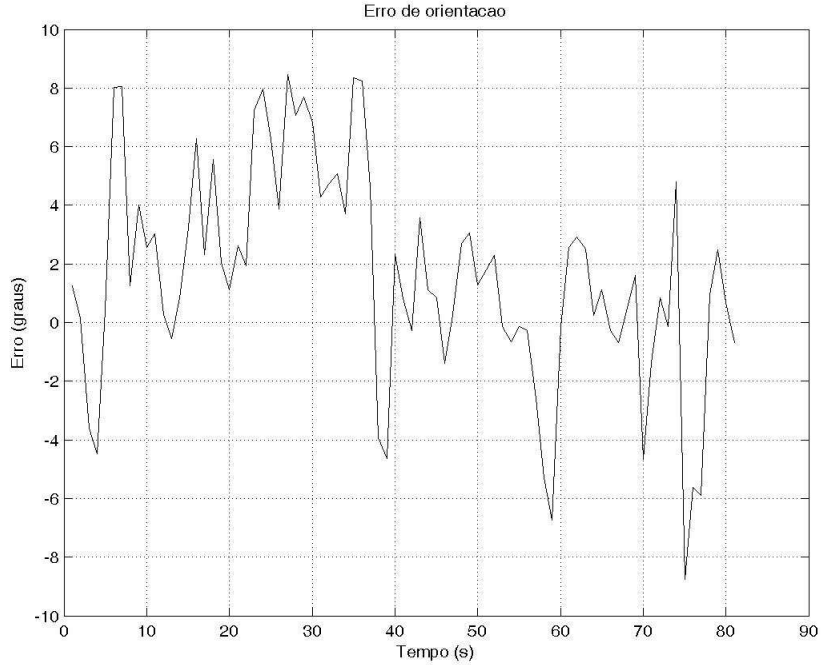

Figura 7.12 – Comportamento do erro de orientação para o experimento 3.

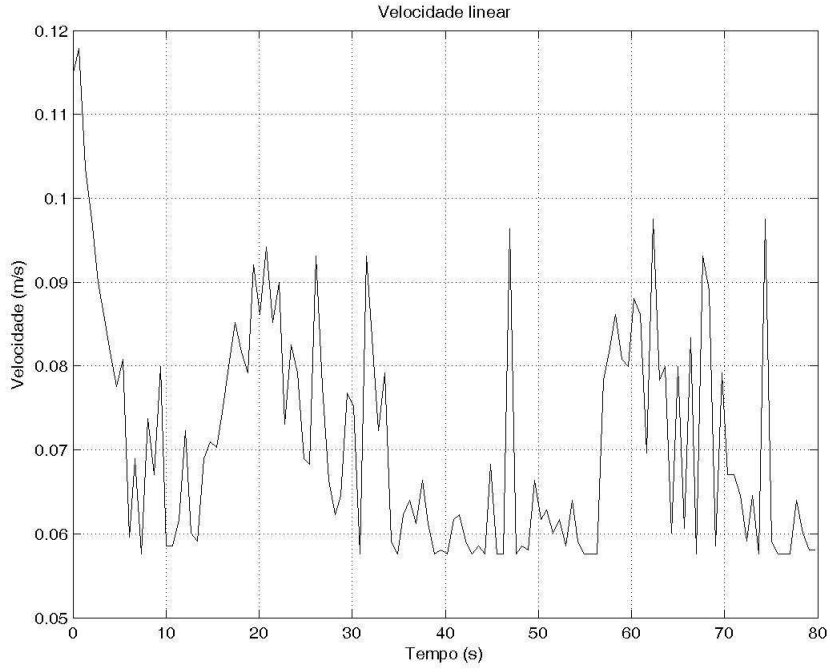

Figura 7.13 – Sinal de controle de velocidade linear para o experimento 3.

ibindo o movimento do robô enquanto rastreia o alvo.

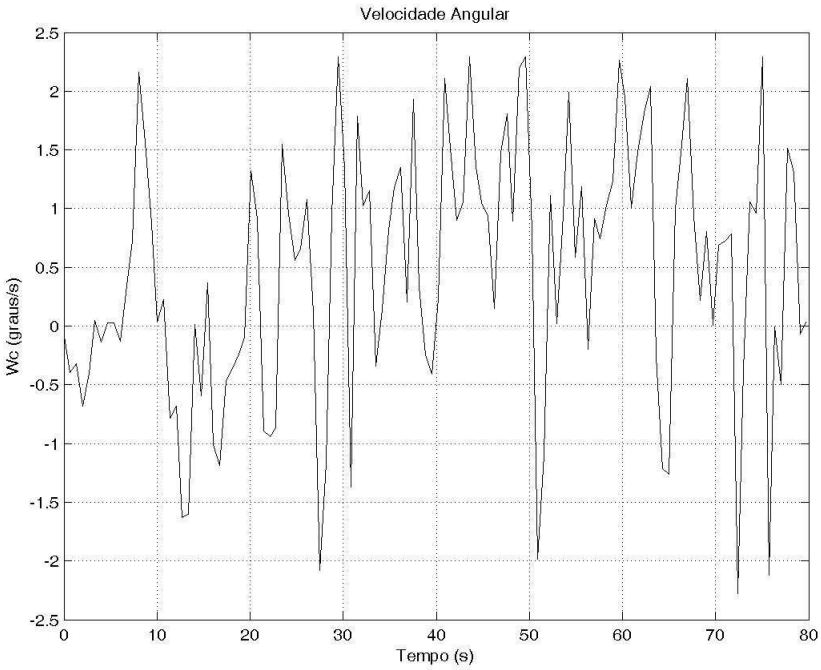

Figura 7.14 – Sinal de controle de velocidade angular para o experimento 3.

Figura 7.15 – Conjunto de imagens feitas por câmera externa para o experimento 3.

8 Discussão dos Resultados dos Experimentos em Tempo Real

Os resultados apresentados no Capítulo 7 foram considerados satisfatórios, uma vez que o rastreamento do alvo foi obtido com sucesso, embora com algumas diferenças em relação aos obtidos em simulação.

Essas diferenças são atribuídas principalmente a dois motivos. O primeiro deles se refere à forma de controle do alvo móvel, que não permite ajuste de velocidade preciso nem comportamento suave durante sua aceleração. Em segundo lugar, o tempo dos experimentos ficou bastante limitado devido à presença do cabo de rede que não permitiu o alcance de distâncias maiores que 8 metros.

Isso fez com que os gráficos apresentados nas figuras do Capítulo 7 apresentassem apenas uma pequena parte do período de regime permanente do rastreamento. Mesmo assim, em todos os experimentos é possível perceber o comportamento de convergência dos erros, como era de se esperar.

Com relação ao comportamento ruidoso dos sinais apresentados dois motivos foram detectados. Primeiro com relação ao alvo, que conforme fora dito não permite movimentos suaves, e portanto produz mudanças bruscas de distância e erro angular. Em segundo

lugar, mudanças na iluminação e erros associados ao próprio processo de detecção visual, já levantados em [4], parecem ser os motivos mais relevantes.

Para atenuar esses problemas, a derivada numérica proposta na Seção 4.2, para determinar a velocidade do alvo, foi substituída por um filtro passa-faixa reduzindo bastante o efeito do ruído.

Além disso, o problema detectado na lei de controle das Equações 4.5 e 4.6 comentado na Seção 5.2.2.2, não ocorreu durante os experimentos em tempo real, já que o ruído do sensor não permite que os erros se anulem totalmente.

Finalmente, com relação ao rastreamento de trajetórias, a técnica apresentada na Seção 3.1, embora tenha sido explorada por meio de simulações, cujos resultados foram apresentados na Seção 5.2.1, não foi implementada em tempo real por dois principais motivos: 1 - A técnica mostrada na Seção 3.2 demandaria um tempo maior que o disponível para ser implementada em C++ e 2 - O tempo de processamento do algoritmo de detecção visual já elevado, ficaria ainda maior no caso de se detectar três pontos.

Ainda assim, o modelo dinâmico do robô foi utilizado para produzir resultados mais próximos da realidade. Assim, propõe-se que após feitas melhorias no código da detecção visual, que diminuam o tempo em questão, possa-se fazer as implementações o que ficaria como sugestão para trabalhos futuros.

9 Conclusões

O presente trabalho apresentou o estudo de alguns problemas relacionados à visão computacional em robótica móvel.

O primeiro deles, a detecção visual de alvos, foi explorada utilizando o algoritmo de intersecção de histogramas, cujos resultados de simulação obtidos foram apresentados na Seção 5.1, onde pôde-se perceber um bom desempenho. Da mesma forma, o algoritmo foi implementado também em C++ e resultados satisfatórios foram obtidos em conjunto com a lei de controle das Equações 4.5 e 4.6.

O segundo problema explorado diz respeito ao rastreamento de trajetórias. Nesse caso, resultados de simulação foram obtidos, usando a técnica de determinação de postura do robô, por sensor visual baseada em reconstrução euclidiana e homografias. Com os resultados obtidos, também satisfatórios, acredita-se ter obtido duas contribuições, quais sejam, as simulações usando modelo cinemático e principalmente as simulações utilizando o modelo dinâmico.

Finalmente, o terceiro problema abordado diz respeito ao rastreamento de alvos móveis. Simulações foram realizadas procurando comparar o controlador tipo *fuzzy* com aqueles baseados em provas de estabilidade segundo Lyapunov, além de agrupar as técnicas de detecção visual por intersecção de histogramas com leis de controle desse segundo tipo.

Dessa forma, acredita-se que três contribuições foram dadas, quais sejam, a comparação por simulação entre as leis de controle mencionadas, a simulação da fusão entre as técnicas de detecção visual e controle via Lyapunov e finalmente a implementação e obtenção de resultados práticos com essa fusão.

Desse modo, três estratégias de controle foram simuladas, duas estratégias de utilização de sensor visual foram exploradas, além de problemas intermediários tais como a transformação entre as coordenadas da imagem e coordenadas do mundo real, cujos resultados não foram apresentados mas sem o qual nenhum dos demais teria sido obtido. Esses resultados permitiram a produção de um artigo [31] já publicado, baseado na exploração do segundo problema descrito acima, e de um outro a ser submetido, relativo à implementação mostrada no Capítulo 7 (terceiro problema).

Acredita-se assim, que os objetivos propostos foram cumpridos de maneira satisfatória, e para a continuidade das linhas de pesquisa abordadas propõe-se os seguintes temas para trabalhos futuros.

1. Desenvolvimento de técnicas que permitam a seleção automática do alvo pelo próprio robô, tal como aquelas baseadas em detecção de movimento, por exemplo;

2. Investigação e desenvolvimento de técnicas que auxiliem no projeto dos ganhos para as leis de controle exploradas;

3. Implementação em tempo real da técnica de rastreamento de trajetórias simulada neste trabalho.

Referências Bibliográficas

1 CHEN, J.; DIXON, W. E.; DAWSON, D. M.; MCINTYRE, M. Homography-based visual servo tracking control of a wheeled mobile robot. **IEEE Transactions on Robotics**, v. 2, p. 1814–1819, Outubro 2003. 7, 9, 24, 25, 39, 43, 58, 59, 124

2 ZADEH, L. A. The role of fuzzy logic in the management of uncertainty in expert systems. **Fuzzy Sets and Systems**, v. 11, p. 199 – 227, 1983. 7, 9, 53, 124

3 LUO, R. C.; CHEN, T. M. Autonomous mobile target tracking system based on grey-fuzzy control algorithm. **IEEE Transaction on Industrial Eletronics**, v. 47, n. 4, p. 920–931, Agosto 2000. 8, 9, 34, 35, 37, 51, 55, 124

4 SILVA, . A. A. da. **Sistema de Rastreamento de Alvos para Robôs Móveis Baseado em Sensor Visual e Controle Fuzzy**. Dissertação (Tese de Mestrado) — Instituto Tecnológico da Aeronáutica, São José dos Campos, 2004. 8, 9, 10, 14, 24, 25, 26, 28, 29, 31, 32, 34, 50, 51, 61, 85, 94, 111, 124

5 SILVEIRA, A. V. J. **Controle Cinemático e Dinâmico de Robôs Móveis**. Dissertação (Tese de Mestrado) — Instituto Tecnológico de Aeronáutica, Julho 2003. 8, 10, 25, 26, 52, 60, 68, 91, 94, 124

6 MATHWORKS. **Image Processing Toolbox Users Guide**. [S.l.], 2003. 14, 33, 34

7 SCRAFF, . R. D. Mechatronics and robotics for service applications. **IEEE Robotics and Automation Magazine**, v. 1, n. 4, p. 31–35, 1994. 20, 25

8 MANDOW, . A.; GABRIEL, J. M. G. de; MARTINÉZ, J. L.; MUÑOZ, V. F.; OLERA, A.; GARCIA-CEREZO, A. The autonomous mobile robot aurora for greenhouse operation. **IEEE Robotics and Automation Magazine**, p. 18–28, 1996. 20

9 HEMERLY, . E. M.; RODRIGUES, C. C. Guiagem de veículos autônomos utilizando sensor de visão. **Anais do CBA**, p. 873–878, Setembro 1994. 21

10 LORA, . F. A. S.; HEMERLY, E. M.; LAGES, W. F. Sistema para navegação e guiagem de robôs móveis autônomos. **Controle e Automação**, v. 9, n. 3, p. 107–118, 1998. 21, 24, 50

11 PIMENTEL, . J. C.; HEMERLY, E. M. An algorithm to detect lanemarks for mobile robots navigation. **Proc. of the IV Industry Applications Conference- INDUSCON'2000**, p. 114–119, novembro 2000. 22

12 CHEN, J.; DIXON, W. E.; DAWSON, D. M.; BEHAL, A. Adaptive homography-based visual servo tracking. **Proc. of the IEEE International Conference on Intelligent Robots and System**, p. 230–235, Outubro 2003. 22, 24, 25

13 HÄCKER, . J.; KRÖPLIN, B. H. An experimental study of visual flight trajectory tracking and pose prediction for the automatic computer control of a miniature airship. **SPIE AeroSense**, p. 21–25, April 2003. 22

14 ONO, Y.; UCHIYAMA, H. A.; POTTER, W. A mobile robot for corridor navigation: A multi-agent aproach. **ACM Southeast Regional Conference archiev Proceedings of the 42nd Annual Southeast Regional Conference**, p. 379–384, 2004. 23, 24

15 SEELINGER, . M.; YOUDER, J. D.; BAUMGARTNER, E. T.; SKAAR, S. B. High precision visual control of mobile manipulators. **IEEE Transactions on Robotics and Automation**, v. 18(6), p. 957–965, 2002. 23, 25

16 LI, H.; YANG, S. X. A bahavior-based mobile robot with a visual landmark-recognition system. **IEEE/ASME Transaction on Mechatronics**, v. 8, n. 3, p. 390–400, Setembro 2003. 23, 24

17 LORA, . F. A. S.; HEMERLY, E. M.; LAGES, W. F. Estimação em tempo real de posição e orientação de robôs móveis utilizando sensores com diferentes taxas de amostragem. **Anais do III Simpósio Brasileiro de Automação Inteligente**, p. 453–458, Setembro 1997. 25

18 RIBEIRO, . C. H.; HEMERLY, E. M. Autonomous learning based on cost assumptions: Theoretical studies and experiments in robot control. **Int. Journal of Neural Systems**, v. 10, n. 1, p. 71–78, 2000. 25

19 JUNIOR, . C. S.; HEMERLY, E. M. Adaptive control of mobile robots using a neural network. **Int. Journal of Neural Systems**, v. 11, n. 3, p. 211–218, 2001. 25

20 JR, . C. S.; HEMERLY, E. M.; GALVÃO, R. K. H. Adaptive control for mobile robots using wavelet networks. **IEEE Transactions on Man, Systems and Cybernetics-B**, Accepted for publication in dec/2001 and scheduled to appear in Volume 32, Part B, n. 4, issue of Aug 2002. 25

21 ASAMI, . S. Robots in japan: Present and future. **IEEE Robotics and Automation Magazine**, v. 1, n. 2, p. 22–26, 1994. 25

22 LORA, . F. A. S. **Navegação e Guiagem de Robôs Móveis**. Dissertação (tese de mestrado) — ITA, Dezembro 1997. 25

23 PIMENTEL, . J. C. **Técnicas de Visão Computacional e de Controle para Aplicação em Robótica Móvel**. Dissertação (Mestrado) — ITA, Dezembro 2000. 25

24 LAGES, . W. F. **Controle e Estimação de Posição e Orientação em Robôs Móveis**. Dissertação (tese de doutorado) — ITA, Dezembro 1998. 25

25 LUO, R. C.; CHEN, T. M. Autonomous mobile target tracking system based on grey-fuzzy control algorithm. **IEEE Trans Ind. Electr.**, v. 47, n. 4, p. 920 931, Agosto 2000. 25

26 JUNIOR, . C. S.; HEMERLY, E. M. Neural network-based controllers for mobile robots. **Proc. of the VIth Brazilian Symposium on Neural Networks**, p. 50–55, novembro 2000. 30

27 ZHANG, Z.; HANSON, A. R. Scaled euclidian 3d reconstruction based on externally uncalibrated cameras. **IEEE Symp. on Computer Vision**, p. 37–42, 1995. 45, 49

28 FIERRO, R.; LEWIS, F. L. Control of a nonholonomic mobile robot using neural networks. **IEEE Transactions On Neural Networks**, v. 9, n. 4, p. 589–596, July 1998. 70

29 SONY SA. **Intelligent Communication Color Video Camera EVI D30/31 User Guide**. [S.l.]. 99, 118

30 TSAI, R. Y. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. **IEEE JOURNAL OF ROBOTICS AND AUTOMATION**, RA-3, n. 4, p. 323 – 344, august 1987. 99

31 OLIVEIRA, C. R. de; HEMERLY, E. M. Rastreamento de trajetória para robôs móveis utilizando sensor visual e homografias. **XVI Congresso Brasileiro de Automática**, Agosto 2006. 113

Apêndice A - Plataforma Magellan Pro-Rwi e Classe Mobility

A utilização do robô móvel Magellan Pro-Rwi, disponível no Laboratório do Grupo de Pesquisa NCROMA (Navegação e Controle de Robôs Móveis Autônomos) é relativamente simples, porém para aqueles pouco familiarizados com a mesma o início pode ser bastante trabalhoso devido à escassez de documentação sobre seu uso.

Para facilitar esse duro início os pesquisadores pioneiros do grupo deixaram alguns trabalhos que podem nortear os primeiros contatos com o veículo. Entretanto, esses são trabalhos internos do instituto de pesquisa e não estão disponíveis ao acesso geral, além de não incluírem informações importantes com relação à utilização da câmera, por exemplo.

Por essa razão, acredita-se que a documentação sobre o acesso do robô móvel Magellan PRO-Rwi tem certa relevância e portanto resolveu-se incluir o presente apêndice.

A.1 Características Gerais do Robô Móvel Magellan Pro

O robô Magellan Pro é uma plataforma com base de formato circular produzida pela *iRobot Corporation* equipada com 16 sensores de contato ("*bumpers*"), 16 sensores infra vermelho, 16 sonares e 1 odômetro, além de um suporte extra para fixação de um scanner lazer e uma câmera conforme mostrado na Figura A.1.

Figura A.1 – Robô Móvel Magellan PRO.

A câmera da marca *Sony* (modelo EVI-D30) possui sensor CCD e adquire imagens de 160x120 pixels (resolução limitada por software pelo fabricante da plataforma robótica), podendo variar o ângulo de *tilt* de -25º até 25º em relação ao plano horizontal, e de *pan* de -100º a 100º em relação ao plano vertical [29]. Além disso, seu zoom óptico variável pode chegar a 12x variando o foco entre 5.4mm e 64.8mm

O robô possui ainda comunicação via rádio *ethernet*, servidor *telnet* e ftp, podendo ser acessado remotamente por qualquer computador ligado à rede por meio de seu número de IP fixo. A CPU possui processador Pentium-III, 400 MHz com 32MB de memória RAM, rodando sistema operacional Linux em sua distribuição Red Hat 5.2 vindo de fábrica. A operação em tempo real é então garantida por um circuito lacrado chamado *rFlex Control System* de configuração não divulgada pelos fabricantes.

Seu acesso é feito via seletor de função e *display*, na parte superior da base do robô (Figura A.2) ou via classes CORBA integrantes da chamada plataforma Mobility.

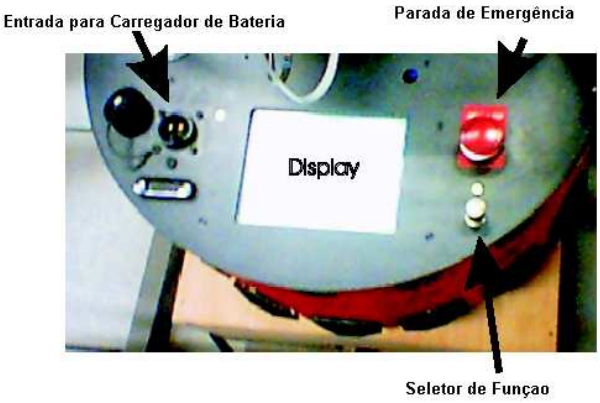

Figura A.2 – Vista Superior do Robô Móvel Magellan PRO.

A.2 Classe Mobility

A Plataforma Integrada Mobility é um conjunto de rotinas orientadas a objeto para a construção de softwares de acesso a robôs para tarefas individuais ou colaborativas (grupos de robôs), desenvolvida pela *IS Robotics*.

Essa plataforma oferece um conjunto de ferramentas de alto nível para acessar os diversos sensores e os atuadores do robô móvel Magellan Pro, facilitando assim o desenvolvimento de programas para seu controle.

A.2.1 Acesso à Classe Mobility

Para acessar a plataforma Mobility o usuário deve seguir os passos,

1. Estabelecer conexão remota com o robô via *Telnet*;

2. Ativar o serviço de identificação do robô;

3. Inicializar os servidores de todos os sensores e atuadores a serem utilizados;

4. Escrever e compilar um programa em linguagem C++ (Linux) não se esquecendo de inicializar as classes necessárias à utilização de sensores e atuadores.

Com relação ao primeiro ítem, a conexão deve ser feita via *prompt* de comando no MS-Windows digitando-se <telnet "IP fixo do robô">. É importante observar que deve-se fazer tantas conexões quantas necessárias, possibilitando assim rodar todos os processos requeridos à execução do programa.

No caso do presente trabalho três conexões eram necessárias, uma para o servidor da câmera, outra para o servidor da base e a terceira para rodar o programa contendo o processamento de imagens e o cálculo da lei de controle. Cada conexão abre no computador remoto uma tela de comandos do sistema operacional Linux, que roda no robô.

Para ativar o serviço de identificação e inicializar os servidores, relativos aos items 2 e 3, os seguintes comandos devem ser digitados.

name -i - Serviço de identificação (deve ser digitado em todas as telas);

base - Inicialização do servidor dos *bumpers*, sensores infra-vermelho, sonares, odômetro e atuadores;

framegrab0 - Inicialização do servidor da câmera

Uma vez concluídos os passos anteriores o usuário deve compilar seu programa no robô (item 4), modificando o *Makefile* contido em seu diretório raiz para este fim. O programa escrito em C++ para Linux, pode ser editado no computador remoto utilizando qualquer editor de textos comum e posteriormente copiado para as pastas do robô por meio do servidor de *ftp*. Um cuidado especial a ser tomado na elaboração das rotinas é que elas devem conter as inicializações das classes para utilização dos sensores e atuadores, tal como segue.

Inicialização da Classe do Odômetro:

```
MobilityActuator::ActuatorState_var pOdo;
MobilityActuator::ActuatorData odo_current;
```

Inicialização da Classe dos Atuadores:

```
MobilityActuator::ActuatorState_var pDriveCommand;
MobilityActuator::ActuatorData OurCommand;
```

Inicialização da Classe da Câmera:

```
MobilityImage::ImageState_var pImage;
MobilityImage::ImageData_var pImageData;
```

Feitas as inicializações das classes, basta acessar os dados lidos (no caso dos sensores), ou determinar um comando de velocidade (no caso dos atuadores), da maneira exemplificada a seguir.

Leitura de posição usando o Odômetro:

```
pOdo->update_sample(0,OdoInfo);
X = odo_current.position[0];
Y = odo_current.position[1];
Theta = odo_current.position[2];
```

Leitura de velocidade usando o Odômetro:

```
pOdo->update_sample(0,OdoInfo);
V = odo_current.velocity[0];
W = odo_current.velocity[1];
```

Comandos de velocidade:

```
OurCommand.velocity[0]=vr;
OurCommand.velocity[1]=wr;
pDriveCommand->new_sample(OurCommand,0);
```

Leitura da Câmera:

```
pImageData = pImage->get_sample(0);
```

Para o caso específico da camera, a imagem é retornada na estrutura apontada por *pImageData* cujo protótipo é tal que,

```
struct FImageData{
AreaSpec area;
long organazationFlags;
MobilityBase::TimeStampData timestamp;
MobilityBase::FloatSeq data;
};
```

Nesse caso, *AreaSpec* é outra estrutura com protótipo,

```
struct AreaSpec{
unsigned long width;
unsigned long height;
unsigned long orgX;
usngined long orgY;
AreaType type;
};
```

Os dados adquiridos são retornados no campo "data" da estrutura *FImageData*, este campo é um vetor de dimensão 1xM onde M é o valor de três vezes o tamanho da imagem

(largura x altura), uma vez que os três tons *Red*, *Green* e *Blue* (RGB) são separados, como mostrado na Figura A.3. Na estrutura *AreaSpec*, os campos *width* e *height* informam o tamanho da imagem, dados necessários para que se possa montar novamente a imagem no formato correto para ser exibida na tela do computador remoto.

vetor data =

Pixels	B	G	R	B	G	R	...
Posição	0	1	2	3	4	5	6

Figura A.3 – Composição do Vetor data, que contém as informações obtidas pela câmera.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO Nº	4. Nº DE PÁGINAS
TM	30 de outubro de 2007	CTA/ITA-IEE/TM-018/2007	123

5. TÍTULO E SUBTÍTULO:

Implementação de Sistema de Rastreamento para Robôs Móveis Autônomos

6. AUTOR(ES):

Cleiton Rodrigo de Oliveira

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Eletrônica – ITA/IEE

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Robôs Móveis; Controle; Sensor Visual

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Dinâmica de robôs; Controle automático; Sensores inerciais; Visão por computadores; Processamento de imagens; Rastreamento (posição); Trajetórias; Simulação; Controle

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos, 2007, 123 páginas

11. RESUMO:

No presente trabalho apresenta-se o estudo de duas diferentes abordagens relacionadas a utilização de sensores visuais na robótica móvel. Na primeira delas as imagens são usadas para fornecer parâmetros que permitam determinar a postura do próprio veículo durante o rastreamento de uma trajetória preestabelecida. Na segunda abordagem, os parâmetros fornecidos pelo processamento das imagens permitem calcular dados sobre a posição relativa entre o objeto imageado e o robô com o objetivo de seguir alvos móveis.

No caso do rastreamento de trajetória, utiliza-se a lei de controle cinemático proposta em [1] em simulações realistas com os modelos cinemático e dinâmico do robô móvel Magellan Pro. Obtém-se as informações de postura do robô, com base na comparação entre dois conjuntos de imagens de três pontos em um plano de referência. O primeiro conjunto de imagens é formado por fotografias feitas durante a condução humana e representa a trajetória a ser rastreada. Já o segundo conjunto de imagens é adquirido durante o rastreamento, e cumpre o papel de fornecer ao sistema de controle a postura atual do robô.

Para obter as informações de posição e orientação, extraindo-as de ambos os conjuntos de imagens, utilizam-se técnicas de decomposição de homografias e reconstrução euclidiana propostas por [2]. Por fim, para aproximar mais da realidade os resultados das simulações explora-se ainda o comportamento do controle na presença de ruído.

Com relação ao segundo tema abordado (rastreamento de alvos móveis), utiliza-se o algoritmo de detecção visual por retroprojeção de histogramas [3] e [4] em conjunto com duas leis de controle. Inicialmente, para garantir a acurácia da técnica de detecção visual, simula-se a mesma estratégia de controle tipo *fuzzy* implementada em [4]. Adicionalmente, para avaliar o funcionamento do algoritmo de detecção visual em conjunto com outras leis de controle, simulações foram realizadas utilizando o controle cinemático apresentado em [5], obtido por meio de técnicas baseadas na análise de estabilidade segundo Lyapunov.

Os bons resultados de simulação obtidos motivaram a implementação da lei de controle proposta em [5] no robô móvel Magellan Pro, em conjunto com a detecção visual baseada em histogramas. Para tanto, foram desenvolvidas rotinas em C++ executadas no veículo, contendo o processamento de imagens e o controle. Implementou-se também uma rotina remota para interagir com o usuário na etapa de seleção manual do alvo, necessária para inicializar o algoritmo de detecção visual. Os resultados obtidos comprovaram o bom funcionamento dessa lei de controle operando em conjunto com o processamento de imagens proposto.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **CONFIDENCIAL** ( ) **SECRETO**

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo