Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica

Identificação e controle de micro-robôs móveis

Guilherme Augusto Silva Pereira

Dissertação de mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Mário Fernando Montenegro Campos

Co-orientador: Prof. Walmir Matos Caminhas

Belo Horizonte, fevereiro de 2000

Resumo

Este trabalho propõe um controlador de posição para robôs móveis não-holonômicos baseado em técnicas clássicas de controle. O projeto dos controladores é baseado em modelos lineares, determinados através de métodos estocásticos de identificação que se utilizam de dados experimentais. Além disso, estes modelos são utilizados como preditores para compensação de tempo morto e conseqüente melhoria no desempenho do sistema. O trabalho propõe ainda um desacoplamento entre as variáveis de saída do robô, permitindo o uso de controladores monovariáveis independentes. As metodologias propostas foram validadas em uma plataforma de futebol de robôs, desenvolvida por alunos de graduação na UFMG, que se constitui basicamente de um computador que controla, por rádio, três micro-robôs observados por um sistema de visão computacional. Resultados experimentais mostraram que, apesar das dificuldades impostas pelos aspectos construtivos deste sistema, o controle proposto apresenta bons resultados e é uma boa alternativa às técnicas complexas encontradas na literatura. A proposta foi implementada considerado-se que o sistema será usado como plataforma de testes para cooperação em tempo real de agentes robóticos em jogos de futebol de robôs.

Abstract

This work proposes a position controller for nonholonomic mobile robots based on classic techniques of control. The controllers project is based on linear models, determined through stochastic methods of identification based on experimental data. Besides, these models are used as predictors for compensation of dead time and consequent improvement in the response of the system. The work still proposes, a decoupling among the robot output variables, yielding independent and monovariable controllers. The suggested methodology was validated using a platform of robots' soccer, developed by under-graduate students at UFMG. The setup is constituted basically by a computer that controls, via radio, three robots observed by a computer vision system. Experimental results showed that, in spite of the difficulties imposed by the constructive aspects of this system, the proposed control presents good results and it is an adequate alternative to the complex techniques found in the literature. The proposal was implemented considered that the system will be used as platform of tests for real time cooperation of three agents playing robot soccer.

A Mamãe,
por tudo que ela tem feito por mim
durante toda a minha vida.

Agradecimentos

Agradeço em primeiro lugar aos professores Mário Campos, pela confiança, paciência e orientação, Walmir Caminhas, pela orientação acadêmica e Luis Aguirre pelas opiniões e discussões.

Gostaria de fazer um agradecimento especial à Cinthia pelo apoio, companheirismo e paciência e ainda a todos os meus familiares e amigos que de alguma forma contribuíram para realização deste trabalho.

Não poderia de deixar de agradecer aos colegas de "equipe": Anício, Boc, Bené, Caixeta, Danilo, Hartmann, Marco Aurélio, Valder e principalmente ao André, pelo sistema de visão e Mangini, pelo sistema no LINUX. Além destes, sem os quais o projeto MIneiROSOT não teria sido possível, gostaria de agradecer o pessoal do Laboratório de Robótica: Alexei, Chaimo, Denilson, Lúcio, Paulo, Piti, Rafael, Raquel, Saulo e os "calouros", Bruno, Daniel, Fábio, Rodrigo e Thiago pelo bom ambiente de trabalho.

Queria agradecer aos colegas Flister, Adriano e Léo, pelo companheirismo além das discussões técnicas e filosóficas durante os nossos poucos momentos de folga.

Merecem os meus agradecimentos também todos os meus professores de graduação e em especial os Profs. Fábio Jota, José Carlos, Ronaldo Pena, Eduardo Mota e Constantino Seixas cujos ensinamentos foram muito úteis durante a execução desta dissertação.

Finalmente, gostaria de agradecer o apoio financeiro da CAPES e dos patrocinadores do projeto: Engetron, IMPEX e DCC.

Sumário

Lista de Figuras

Capítulo 1

Introdução

> Se todo instrumento pudesse realizar seu próprio trabalho, obedecendo ou antecipando o desejo dos outros... se a lançadeira pudesse tecer, e o pico tocasse a lira, sem uma mão para guiá-los, os chefes não precisariam de criados nem os mestres de escravos.
>
> *Aristóteles (384–322 a.C.)*

OS robôs móveis ou veículos autônomos têm sido muito estudado nos últimos anos. Em particular, os robôs móveis terrestres, também chamados de AGV's (*autonomous ground vehicles*), são sistemas motorizados, equipados com rodas e controlados por computadores digitais que operam sem a intervenção humana. Para interagir com seu ambiente de trabalho, estes robôs podem ter diversos tipos de sensores que incluem sensores de distância, posição, velocidade, força e câmeras que são sensores visuais. Em relação à localização, os sensores podem ser internos ou externos. O sensores internos são aqueles localizados no próprio robô e por isso se movem juntamente com o mesmo. Em oposição, os sensores externos são fixos ao ambiente ou em algum outro sistema móvel e observam o movimento do veículo. Além desta classificação, os sensores podem fornecer um conhecimento local, que é usado para determinação das características internas do robô (ex.: velocidade e posição dos motores), ou um conhecimento global que provê informação dos estados do robô em relação ao ambiente (ex.: velocidade e posição do

robô) [Abidi and Gonzales, 1992].

Os sensores visuais [Horn, 1986] são em geral sensores globais e podem ser internos ou externos. Quando as câmeras estão localizadas no robô, este usa conhecimentos prévios e marcas visuais para se localizar no ambiente [Campos and de Souza Coelho, 1999, Ma et al., 1999]. De maneira similar, as câmeras externas são usadas para observar o movimento do robô e informar a sua localização. A principal diferença neste caso, é que o processamento da imagem não está no veículo, mas em um segundo computador que toma algumas decisões e as transmite ao robô, muitas vezes chamado de agente. O controle dos robôs é então, feito de forma externa e quando não existe além do sistema visual, um processamento e sensoriamento local, este se torna uma tarefa não trivial. Atualmente, esta configuração tem sido largamente encontrada em aplicações onde se deseja obter robôs de baixo custo ou que não podem conter carga, como aquele mostrado em [Zhang and Ostrowski, 1999] e robôs usados em competições de futebol de robôs.

O futebol de robôs tem se apresentado, por si só, um ambiente de testes para diversas áreas como cooperação de multi-agentes, robótica, controle, desenvolvimento mecânico, visão computacional e computação distribuída. Apesar do objetivo principal desta competição estar na modelagem e teste de comportamentos autônomos e cooperativos [Kim et al., 1997, Werger, 1998], várias pesquisas em robótica e visão têm usado este cenário, já que o mesmo, proporciona um ambiente para a implementação rápida de novas abordagens e tecnologias [McKerow, 1998].

Por estes motivos, as várias categorias de futebol de robôs têm despertado o interesse de pesquisadores de diversos centros de pesquisa, universidades e empresas em todo o mundo. Em geral estas categorias estão divididas entre

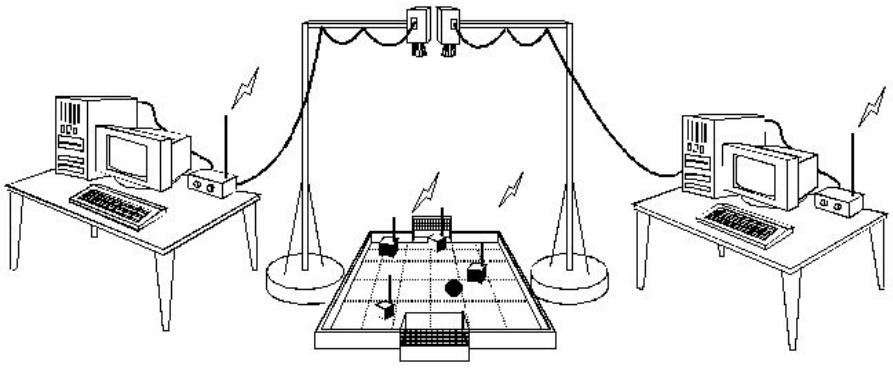

Figura 1.1: Cenário do Futebol de Robôs.

duas associações: FIRA [Kim et al., 1997] e ROBOCUP [Kitano et al., 1997]. A principal diferença entre as associações está no tamanho dos robôs e número de jogadores dos times. A categoria MIROSOT (*Micro Robot Soccer Tornament*) da FIRA, por exemplo, possuem times de 3 jogadores, menores e mais velozes que os da categoria equivalente da ROBOCUP cujos times possuem 5 jogadores. Normalmente, os robôs usados nos jogos se comunicam com um computador central através de comunicação sem fio. Este computador "visualiza" e coordena as jogadas com o auxilio de uma ou mais câmeras de vídeo situadas a uma determinada altura do campo (Figura 1.1). O número de câmeras, computadores, equipamentos em geral, dimensões do campo e ainda as formas de comunicação sem fio são determinados pelas regras da competição.

1.1 Objetivos

O principal objetivo deste trabalho é conceber, projetar e implementar um controlador para agentes robóticos externamente controlados, sem nenhum tipo de processamento local e observados por visão, usados em jogos de futebol de robôs. As metodologias propostas serão aplicadas a robôs usados na disputa da categoria MIROSOT. Como será mostrado, estes robôs são constituídos basicamente por atuadores, sendo que o controle é totalmente executado em computadores externos baseado em informações visuais fornecidas pela câmera. Deseja-se ainda, através de um conhecimento mais profundo do sistema, obtido através de técnicas de modelagem e análise de sistemas, sugerir melhorias que possam contribuir para um bom desempenho em competições de futebol de robôs.

1.2 Relevância e Contribuições

Apesar do controle de robôs ser largamente estudado por pesquisadores de diversas partes do mundo os resultados obtidos neste trabalho se diferenciam dos demais pois o controle se aplica a robôs com "pouca inteligência" que não possuem conhecimento sobre o seu próprio estado. Desta forma, a principal dificuldade está no fato de que algumas constantes de tempo rápidas que são importantes para o controle dos robôs não são percebidas pelos sensores visuais, na sua maioria mais lentos que o necessário. Outra importante contribuição está na modelagem estocástica de agentes robóticos que tem sido pouco explorada na literatura e que pode ser utilizada com sucesso em predição e controle, principalmente em ambientes não controlados como é o caso do futebol de robôs. A partir dos modelos estimados, foram ainda propostos preditores para compensação de atrasos no sistema. Além disso, em oposição

à maioria dos trabalhos recentes que utilizam técnicas de inteligência computacional ou outros algoritmos complexos, os sistemas de controle propostos no trabalho são baseados em controladores lineares clássicos.

1.3 Organização da Dissertação

O restante do texto está organizado da seguinte forma: o Capítulo 2 faz uma revisão bibliográfica do assunto mostrando as principais publicações na área de modelagem e controle de robôs móveis. O Capítulo 3 descreve teoricamente as metodologias usadas no trabalho para o controle dos veículos autônomos. No Capítulo 4 é feita uma descrição do sistema de futebol de robôs utilizado no trabalho para testar e validar as metodologias propostas no capítulo anterior. É feita a identificação estocástica do sistema e o modelo obtido é utilizado no Capítulo 5 para proporcionar o controle de trajetórias dos robôs. O Capítulo 6 faz uma síntese da dissertação apontando os principais pontos e discutindo algumas conclusões. São mostradas também as deficiências da metodologia empregada e propostos novos trabalhos que venham contribuir para os resultados obtidos. O Apêndice A mostra a construção de um simulador usado para testar as metodologias propostas antes da sua implementação prática. No Apêndice B são mostrados os algoritmos estendidos de mínimos quadrados usados como estimadores de parâmetros.

Capítulo 2

Trabalhos Relacionados

> Uma coisa só é impossível até que alguém duvide e prove o contrário.
>
> *Albert Einstein (1879–1955)*

EM geral, os sistemas de controle de robôs móveis autônomos possuem dois níveis principais como mostrado na Figura 2.1 [Kang et al., 1999]. O nível superior, ou *planejamento de trajetórias*, é responsável pela escolha de um melhor caminho a ser seguido pelo robô. O problema do planejamento de trajetórias pode ser resumido como [Wang et al., 1994]: dado um robô móvel com localização e orientação iniciais, uma posição e orientação alvo e um conjunto de obstáculos localizados no espaço, encontre um caminho contínuo para o robô do estado inicial até o objetivo sem que haja colisão com os obstáculos ao longo do percurso. A trajetória gerada indica, além das posições e orientações do robô no espaço, os perfis de velocidades e acelerações com que o robô deve se deslocar ao percorrer estas trajetórias. Apesar de aparentemente simples, este problema não está completamente solucionado. Alguns trabalhos que tratam do assunto são [Murray and Sastry, 1993, Wang et al., 1994, Jagannathan et al., 1994, Gorinevsky et al., 1996]. O problema torna-se ainda mais complicado quando o ambiente é não controlado e dinâmico, ou seja, os obstáculos e o alvo se movem. Este cenário tem sido largamente estudado por diver-

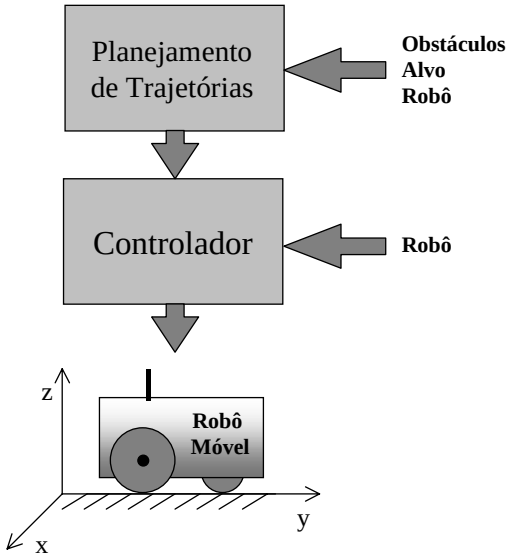

Figura 2.1: Níveis de controle de um robô autônomo

sos autores no contexto do futebol de robôs como em [Kim et al., 1996, Fukuda and Kubota, 1997, Sargent et al., 1997, Lee and Bautista, 1998, Kim et al., 1998, Kim et al., 1999, Veloso et al., 1999, la Rosa et al., 1999].

Na Figura 2.1, o nível inferior, ou *controlador*, é responsável por garantir que o robô siga a trajetória escolhida pelo nível superior. Em outras palavras, o planejamento de trajetórias pode ser visto como um gerador de referências para o controlador. Por isso, muitas vezes, o controlador é considerado como um nível de rastreamento de trajetórias [Caracciolo et al., 1999]. Exceto em alguns casos como em [Kang et al., 1999], onde os dois níveis são calculados na mesma taxa, o controlador tem em geral um período de amostragem menor que o nível de planejamento de trajetórias. O rastreamento de trajetórias está praticamente dominado quando se trata de robôs manipuladores [Fu et al., 1987, Ozaki et al., 1991, Stemmer et al., 1997, Teixeira et al., 1998]. A maior dificuldade deste controle em robôs autônomos está no fato de os mesmos serem,

em geral, sistemas multivariáveis, não-holonômicos e não-lineares. A característica multivariável é evidente quando sabe-se que os robôs são acionados por dois ou mais atuadores e têm como saída uma posição e orientação no espaço ou no plano.

Uma definição formal para a característica não-holonômica é: "Um sistema é chamado não-holonômico, quando a velocidade de um sistema mecânico satisfaz uma condição de igualdade que não pode ser escrita como uma condição equivalente de posição" [Wen, 1995]. Condições não-holonômicas podem surgir de restrições físicas como o puro rolar de uma roda ou de leis de conservação físicas como a conservação de impulso angular de um corpo livre flutuante. Desta forma, alguns robôs móveis possuem restrições não-holonômicas porque suas rodas somente podem girar em uma direção do espaço, impedindo que eles se movam em todas as direções apesar disso ser fisicamente possível se os mesmos fossem considerados massas. As não-linearidades aparecem principalmente pelo mesmo motivo, mas também pelas características dos sensores, atuadores e do próprio ambiente de trabalho do robô. Devido a estes problemas, diversas estratégias têm sido encontradas na literatura para o controle de trajetórias de robôs móveis. Estas estratégias podem ser divididas em duas categorias distintas:

1) Estratégias em malha aberta que buscam encontrar um conjunto de entradas de controle para conduzir o veículo de uma posição inicial a alguma posição arbitrária. Se os robôs são localmente controlados este método é factível e proposto em [Lafferriere and Sussmann, 1991]. Em geral estas estratégias são estudadas em conjunção com o planejamento do movimento ou trajetória.

2) Estratégias em malha fechada que consistem no desenvolvimento de malhas de realimentação para posicionar o veículo em um determinado pon-

to. Mesmo se o robô for localmente controlável e observável é mostrado em [Samson, 1995] e [Bloch et al., 1992] que não é possível estabilizar este tipo de sistema com uma realimentação de estados linear e invariante no tempo. Por este motivo, controladores não lineares são propostos em [de Wit and Sordalen, 1992, Walsh et al., 1994, Lee et al., 1999]. Em [Yang et al., 1998] e [Yang and Kim, 1999] é proposto um controlador em modos deslizantes no controle de trajetória do robô. Outros autores propõem estratégias de linearização do sistema como em [Park et al., 1999], para depois controlá-lo com controladores convencionais. Técnicas como essa já haviam sido utilizadas em manipuladores através de um método conhecido como "*computed torque*" [Asada and Slotine, 1985]. Zhang e Ostrowski [1999] usaram esta metodologia para controlar um dirigível autônomo. Outra metodologia que está sendo muito empregada é o uso de controladores variantes no tempo [Jiang and enk Nijmeijer, 1997, M'Closkey and Murray, 1997, Samson, 1995] ou preditivos [Normey-Rico et al., 1998]. Em seu artigo, M'Closkey e Murray [1997] mostram que realimentações suaves invariantes no tempo poderiam ser utilizadas se fossem feitas transformações de coordenadas não suaves no sistema. Estas transformações são propostas por Aicardi et al. [1995] e generalizadas por Hemerly [1998] para todas as classes de robôs móveis com rodas.

Um problema que pode dificultar o controle de trajetórias é a falta de medição das grandezas que devem ser controladas como velocidades e acelerações. Neste caso alguns trabalhos propõem o uso de estimadores de estado como o observador de Luenberger [Luenberger, 1971] em [Arimoto et al., 1994] ou de maneira mais robusta, filtros de Kalman [Kalman, 1960] em [Cortesao et al., 1998]. Outras alternativas são propostas em [Lizarralde and Wen, 1996] e [Caccavale et al., 1999] que fazem o

controle sem a medição ou estimação da velocidade.

Apesar do grande número de publicações referentes ao controle de trajetória de robôs autônomos, é importante ressaltar que na grande maioria delas a saída do controlador é composta por referências de velocidade para outros controladores implementados em *hardware* capazes de garantir que estas velocidades sejam alcançadas. Esta é principal diferença entre este trabalho e os demais, pois a saída do controlador projetado aqui é proporcional à própria tensão de alimentação dos motores do robô.

Pela bibliografia, vê-se que a maioria das estratégias propostas para controle de robôs autônomos, tanto no nível de planejamento como no controlador propriamente dito, utilizam-se de algum conhecimento do processo, ou seja, da cinemática ou dinâmica do robô. Em [la Rosa et al., 1999], por exemplo, modelos dinâmicos para robôs de futebol são usados até mesmo para selecionar qual jogador deve participar de uma jogada. Neste caso, o modelo é usado em um nível de estratégia de jogo que poderia ser um nível acima do planejamento de trajetórias. Desta forma, muitas vezes são necessários modelos completos e precisos para os veículos a serem controlados. Em [Campion et al., 1996], um artigo básico no que diz respeito a modelos cinemáticos e dinâmicos, são feitas várias observações sobre as diversas estruturas de robôs móveis com rodas. Apesar disso, não se têm encontrado muitas publicações referentes à modelagem, principalmente dinâmica, de robôs móveis. Em [Gomes and Ramos, 1998] é mostrada a modelagem pela física do processo de um dirigível autônomo. Em [Asada and Slotine, 1985] a mesma metodologia é utilizada para robôs manipuladores. Métodos de identificação de modelos, baseados na observação experimental do comportamento dos robôs, são, entretanto, pouco difundidos na prática. Para manipuladores robóticos encontram-se alguns artigos

como [Olsen and Bekey, 1986, Kozlowski and Dutkiewicz, 1996] o que não pode ser dito para robôs autônomos. Neste caso, Elnagar e Gupta [1998] utilizam modelos autoregressivos na predição do movimento de objetos móveis para facilitar o planejamento de trajetórias. Técnica semelhante é utilizada no presente trabalho na predição do movimento dos próprios robôs.

Capítulo 3

Metodologia

> A frase mais deliciosa que se pode ouvir na ciência, aquela que todos os cientistas adoram, não é "Eureka" mas "que engraçado"...
>
> *Issac Asimov (1920–1992)*

COMO foi mencionado no Capítulo 2, é sabido que controladores multivariáveis com uma realimentação de estados suave não estabiliza um sistema com restrições não-holonômicas. Entretanto, em [Bloch et al., 1992] é provado que se o sistema for controlado por partes a estabilidade pode ser alcançada. Este tipo de controlador consiste no projeto de sistemas de controle independentes para algumas variáveis. Neste capítulo será mostrado o projeto de um controlador para robôs autônomos não-holonômicos baseado neste princípio. Além disso, transformações de variáveis serão feitas para facilitar o controle, tornando simples o projeto dos controladores. Assume-se que as trajetórias a serem seguidas pelo robô são pré-especificadas por um nível de planejamento executado em malha aberta.

3.1 Definição do problema

O robô móvel considerado aqui é um veículo com rodas como mostrado na Figura 3.1. Duas rodas fixas são controladas por motores independentes

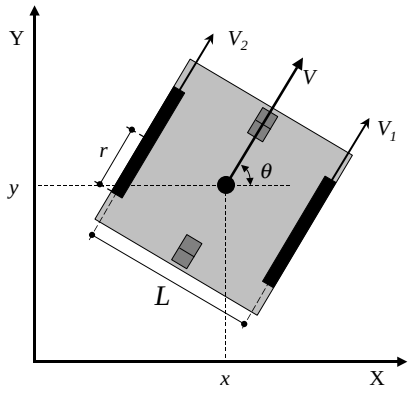

Figura 3.1: Estrutura geral do robô

e duas rodas centralizadas fazem o apoio do veículo tornando-o equilibrado durante o movimento. O robô é um sistema com dois graus de liberdade (*2-DOF*) cujas saídas são representadas pela sua posição (x, y) e orientação θ. Considera-se ainda que:

$$V_1 = r\,\omega_1 \qquad \text{e} \qquad V_2 = r\,\omega_2 \ , \tag{3.1}$$

onde V_i é a velocidade linear da roda i, ω_i é a velocidade angular do respectivo eixo e r é o raio das rodas. Sabe-se ainda que:

$$V = \frac{V_1 + V_2}{2} \tag{3.2}$$

$$\omega = \frac{V_1 - V_2}{L} \ , \tag{3.3}$$

onde V é a velocidade linear do centro do robô, ω é a velocidade angular do robô e L é a distância entre as rodas. Além disso, a velocidade linear V, pode ser decomposta ao longo dos eixos X e Y respectivamente como:

$$V_X = V\,\cos(\theta) \qquad \text{e} \qquad V_Y = V\,\text{sen}(\theta) \ , \tag{3.4}$$

onde θ é o angulo mostrado na Figura 3.1. Então, um modelo cinemático para o robô pode ser escrito como:

$$\begin{aligned} \dot{x} &= V_X \\ \dot{y} &= V_Y \\ \dot{\theta} &= \omega \ . \end{aligned} \tag{3.5}$$

Substituindo as equações para V_X, V_Y e ω em (3.5) tem-se que:

$$\begin{aligned} \dot{x} &= \frac{V_1 + V_2}{2} \cos(\theta) \\ \dot{y} &= \frac{V_1 + V_2}{2} \operatorname{sen}(\theta) \\ \dot{\theta} &= \frac{V_1 - V_2}{L} \ . \end{aligned} \tag{3.6}$$

A restrição não-holonômica do sistema pode ser representada pela seguinte equação:

$$\dot{x} \operatorname{sen}(\theta) - \dot{y} \cos(\theta) = 0 \ , \tag{3.7}$$

que especifica a direção tangente a qualquer caminho possível para robô e um limite de curvatura para este caminho.

3.1.1 Controlador

Os sistemas de controle de trajetórias tradicionalmente utilizados para controlar robôs móveis podem ser representados por um diagrama de blocos semelhante ao da Figura 3.2. Esta é uma estratégia de controle conhecida como "cascata", onde a saída de um controlador é a referência para ou-

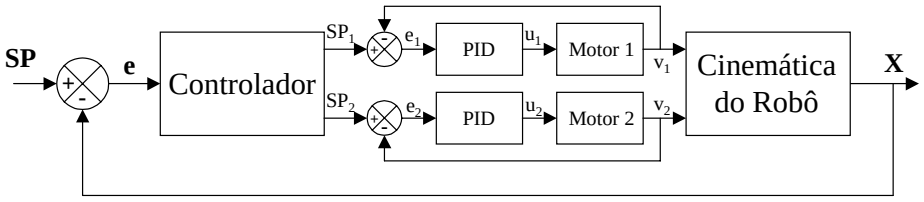

Figura 3.2: Sistema de controle tradicional.

tro [McMillan, 1994]. Os controladores internos (PID[1]) são chamados de "secundários" ou "escravos" e o externo é conhecido como "primário" ou "mestre". De uma maneira geral, este tipo estratégia é usada para melhorar o desempenho do sistema por meio de uma rápida correção dos distúrbios na malha interna. O aumento do desempenho pode ser muito grande se o tempo morto da fonte de distúrbios for pequeno em relação às variáveis controladas pelos controladores internos e as constantes de tempo dos controladores secundários forem pequenas em relação à do controlador primário, considerando-os assim reguladores. Este tipo de estratégia é largamente utilizada em veículos autônomos, onde os controladores escravos são geralmente construídos em *hardware* através de microcontroladores digitais que aumentam a velocidade de processamento. Se as condições anteriores forem cumpridas, os controladores internos podem ser desconsiderados durante o projeto do bloco "controlador" (Figura 3.2), pois a sua dinâmica tende a ser desprezível em relação à dinâmica total do robô. Na Seção 3.3.2 será mostrado que os controladores escravos, neste caso específico, têm ainda, a função de desacoplamento entre as variáveis de saída do sistema.

Geralmente, o controle de trajetórias para robôs autônomos consiste em fazer o robô seguir referências de posição e orientação, (x, y, θ), e velocidades, (V, ω). Em alguns casos (por exemplo [Yang et al., 1998]), algumas destas

[1]Proporcional+Integral+Derivativo [Ogata, 1993].

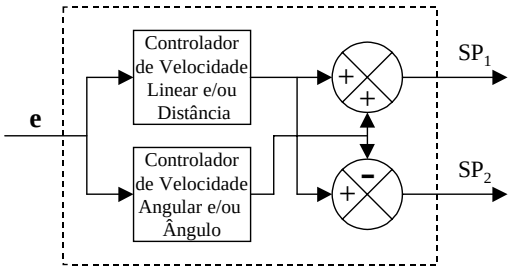

Figura 3.3: Diagrama de blocos de controlador.

variáveis, como V e ω podem ser consideradas livres. Assim, o controlador deve ser projetado como uma função que gere um par de sinais de referência (*setpoints*), SP_1 e SP_2, que minimize o vetor de erros $\mathbf{e} = [e_x, e_y, e_\theta, e_V, e_\omega]'$, onde o apóstrofo significa transposição. Como o sistema possui restrições não-holonômicas e não pode ser controlado por um único controlador multivariável, uma boa alternativa é a mudança de variáveis e o uso de controladores independentes para cada nova variável ou para um conjunto destas variáveis. Um controlador como este pode ser visto na Figura 3.3. Nesta figura, houve uma mudança de variáveis no sentido de que as variáveis x e y não são diretamente controladas e há o aparecimento da variável "distância", que pode ser a distância entre o alvo e o robô. Desta forma a distância poderia, por exemplo, substituir as variáveis e_x e e_y no vetor de erros.

O controlador da Figura 3.3 supõe que as "variáveis lineares" posição e velocidade linear podem ser controladas de forma independente das "variáveis angulares", orientação e velocidade angular. Além disso, é considerado que cada uma dessas variáveis pode ser controlada por um único controlador monovariável, já que a única saída de cada um dos controladores é usada como parte de duas variáveis manipuladas. O projeto de cada controlador pode, então, ser feito de forma independente, considerando que o sistema contém apenas uma variável manipulada que é a soma das variáveis reais,

V_1 e V_2. Assim, pelas Equações (3.2) e (3.3), considera-se que as novas variáveis manipuladas são proporcionais a V e ω. A Equação (3.3) explica também o sinal negativo da saída SP_2. Desta forma, o problema de controle foi simplificado e pode ser resumido como a determinação de uma função que minimize o erro de uma única variável medida a partir de uma única variável manipulada. Apesar deste ser um problema relativamente simples, conhecido e facilmente solucionado por controladores lineares [Ogata, 1993, Philips and Nagle, 1995, Kuo, 1995] a grande dificuldade do projeto está no fato que o robô não possui os controladores locais[2] de velocidade, como será visto nas próximas seções.

3.2 Modelo Dinâmico

O modelo cinemático trata da geometria do movimento, relacionando posição, velocidade, aceleração e tempo, sem referência às suas causas. O modelo dinâmico, por sua vez, trata das relações entre as forças que agem no robô e o seu movimento [Beer and Johnston Jr., 1991].

Nesta seção, um modelo dinâmico, útil no projeto e análise de controladores, será derivado com base na cinemática mostrada na Seção 3.1. Como os controladores dos robôs são implementados em computadores digitais, um modelo discreto para o sistema é uma forma mais apropriada. Assim, utilizando uma aproximação primeira ordem para as derivadas, a Equação (3.5)

[2]Os controladores PID de velocidade são chamados locais porque, em geral, estes estão localizados no robô próximos às rodas.

pode ser rescrita como:

$$\begin{aligned} x_k &= x_{k-1} + V_{X\,k-1}\,T \\ y_k &= y_{k-1} + V_{Y\,k-1}\,T \\ \theta_k &= \theta_{k-1} + \omega_{k-1}\,T \quad , \end{aligned} \tag{3.8}$$

onde x_k indica o valor de x no tempo $k \cdot T$ e x_{k-1} indica o valor desta variável um período de amostragem, T, antes. A aproximação para as derivadas é representada por [Philips and Nagle, 1995]:

$$\frac{d}{dt}y(t) \approx \frac{y(t) - y(t-T)}{T} \quad .$$

Para "adicionar dinâmica" ao modelo (3.8), basta considerar uma relação dinâmica para as velocidades do robô. Assim:

$$\begin{aligned} V_{1\,k} &= f_1(V_{1\,k-1}, \cdots, V_{1\,k-n},\, u_{1\,k-1}, \cdots, u_{1\,k-m}) \\ V_{2\,k} &= f_2(V_{2\,k-1}, \cdots, V_{2\,k-n},\, u_{2\,k-1}, \cdots, u_{2\,k-m}) \quad , \end{aligned}$$

onde $f_1(\cdot)$ e $f_2(\cdot)$ são funções quaisquer e u_1 e u_2 são as entradas dos motores direito e esquerdo respectivamente. Consequentemente,

$$\begin{aligned} V_k &= f(V_{k-1}, \cdots, V_{k-m},\, u_{1\,k-1}, \cdots, u_{1\,k-m},\, u_{2\,k-1}, \cdots, u_{2\,k-n}) \\ \omega_k &= g(\omega_{k-1}, \cdots, \omega_{k-m},\, u_{1\,k-1}, \cdots, u_{1\,k-m},\, u_{2\,k-1}, \cdots, u_{2\,k-n}) \quad , \end{aligned}$$

Então, inicialmente pode-se considerar que um modelo dinâmico para as

velocidades do robô é da forma:

$$\begin{aligned} V_k &= V_{k-1} + a\,T \\ \omega_k &= \omega_{k-1} + \alpha\,T \ , \end{aligned} \tag{3.9}$$

onde a e α são as acelerações linear e angular do robô respectivamente. Estas acelerações podem ser encontradas a partir da segunda lei de Newton que pode ser escrita como:

$$\sum \vec{F} = m \cdot \vec{a} \tag{3.10}$$

$$\sum \vec{T} = I \cdot \vec{\alpha} \ , \tag{3.11}$$

onde m é massa do robô, I é o momento de inércia em torno do centro de massa G, $\vec{T}$ é conjugado aplicado em relação a G e $\vec{F}$ representa um vetor de força. A Figura 3.4, mostra o diagrama das forças aplicadas ao robô. Considerando que o centro de massa está exatamente no centro geométrico do robô (isto não é necessariamente verdade) tem-se as seguintes equações:

$$F_1 + F_2 + F_{a1} + F_{a2} = m\,a \tag{3.12}$$

$$F_1 \frac{L}{2} - F_2 \frac{L}{2} = I\,\alpha \ , \tag{3.13}$$

onde F_{a1} e F_{a2} são os módulos das forças de atrito na linha de contato entre o solo e as rodas de apoio. Estas equações são possíveis porque todas as forças têm a mesma direção. Substituindo as equações anteriores em (3.9) tem-se:

$$\begin{aligned} V_k &= V_{k-1} + \frac{T}{m}\left(F_{1\,k-1} + F_{2\,k-1} + F_{a1\,k-1} + F_{a2\,k-1}\right) \\ \omega_k &= \omega_{k-1} + \frac{T}{2I}\left(F_{1\,k-1} - F_{2\,k-1}\right) \ . \end{aligned} \tag{3.14}$$

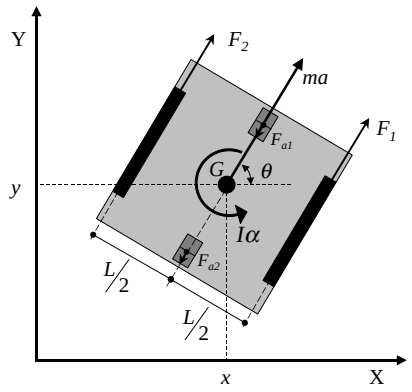

Figura 3.4: Diagrama de forças aplicadas ao robô

Caso o centro de massa não coincida com o centro geométrico, tem-se um conjugado associado às forças de atrito do suporte de equilíbrio. Assim, deve-se adicionar este conjugado na equação de velocidade angular em (3.14).

Supõe-se ainda, que as forças F_1 e F_2, que no modelo (3.14) representam as entradas conhecidas, são linearmente relacionadas com as entradas dos motores u_1 e u_2. Assim:

$$\begin{aligned} F_{1\,k} &\approx a_1\, u_{1\,k-d} \\ F_{2\,k} &\approx a_2\, u_{2\,k-d} \quad . \end{aligned} \tag{3.15}$$

onde d representa o atraso ou tempo morto entre a aplicação dos sinais de controle e o aparecimento das respectivas forças.

Um modelo geral para o robô pode ser obtido substituindo as Equações (3.14) e (3.15) no modelo cinemático (3.8). Desta forma:

$$\begin{aligned} x_k &= x_{k-1} + \left[V_{k-1}\, T + (a_1\, u_{1\,k-d} + a_2\, u_{2\,k-d} + F_{a1\,k-1} + F_{a2\,k-1}) \frac{T^2}{m} \right] \cos(\theta_{k-1}) \\ y_k &= y_{k-1} + \left[V_{k-1}\, T + (a_1\, u_{1\,k-d} + a_2\, u_{2\,k-d} + F_{a1\,k-1} + F_{a2\,k-1}) \frac{T^2}{m} \right] \operatorname{sen}(\theta_{k-1}) \\ \theta_k &= \theta_{k-1} + \omega_{k-1}\, T + (a_1\, u_{1\,k-d} - a_2\, u_{2\,k-d}) \frac{T^2}{2I} \quad . \end{aligned} \tag{3.16}$$

3.2.1 Parametrização do modelo

O modelo (3.16) descreve fisicamente o sistema. O maior problema deste modelo é que alguns termos, como as forças de atrito, são de difícil obtenção. Outra grande dificuldade é que em geral os termos em velocidade não são diretamente medidos pelos sensores, devendo ser estimados. Para evitar estes problemas, o modelo pode ser parametrizado para que as incógnitas sejam estimadas através de dados experimentais. O modelo físico (3.16) pode então, ser rescrito como:

$$\begin{aligned} x_k &= x_{k-1} + c_1\, V_{X\,k-1} + [c_2\, u_{1\,k-d} + c_3\, u_{2\,k-d}]\, \cos(\theta_{k-1}) \\ y_k &= y_{k-1} + c_4\, V_{Y\,k-1} + [c_5\, u_{1\,k-d} + c_6\, u_{2\,k-d}]\, \mathrm{sen}(\theta_{k-1}) \\ \theta_k &= \theta_{k-1} + c_7\, \omega_{k-1} + c_8\, u_{1\,k-d} + c_9\, u_{2\,k-d} \quad . \end{aligned} \tag{3.17}$$

É importante notar que a massa, m (I na equação de θ), o período de amostragem, T, e as forças de atrito, F_{ai}, foram agrupadas nos parâmetros c_i.

A componente em X da velocidade, V_X, a princípio, pode ser calculada de forma simples a partir de medições de x como:

$$V_{X\,k-1} = \frac{x_{k-1} - x_{k-2}}{T} \quad . \tag{3.18}$$

Usando esta equação, a primeira linha do modelo (3.17) pode ser representada como um modelo do tipo autoregressivo com entrada externa [Ljung, 1987] do tipo:

$$x_k = a_1\, x_{k-1} + a_2\, x_{k-2} + [b_1\, u_{1\,k-1} + b_2\, u_{2\,k-1}]\, \cos(\theta_{k-1}) \quad .$$

O mesmo procedimento pode ser aplicado para y e θ. Assim, o modelo dinâmico completo do robô pode ser representado de forma paramétrica pelas

seguintes equações:

$$\begin{aligned} x_k &= a_1^X\, x_{k-1} + a_2^X\, x_{k-2} + \left[b_1^X\, u_{1\,k-d} + b_2^X\, u_{2\,k-d}\right] \cos(\theta_{k-1}) \\ y_k &= a_1^Y\, y_{k-1} + a_2^Y\, y_{k-2} + \left[b_1^Y\, u_{1\,k-d} + b_2^Y\, u_{2\,k-d}\right] \sin(\theta_{k-1}) \\ \theta_k &= a_1^\Theta\, \theta_{k-1} + a_2^\Theta\, \theta_{k-2} + b_1^\Theta\, u_{1\,k-d} + b_2^\Theta\, u_{2\,k-d} \quad . \end{aligned} \tag{3.19}$$

É importante notar que nas duas primeiras linhas do modelo (3.19), funções não-lineares de uma das saídas do modelo multiplicam outros regressores. Isso resulta em não-linearidades "fracas" no sentido de que apesar, de ter regressores não-lineares, o modelo é ainda linear nos parâmetros. Assim, estritamente falando, as duas primeiras equações do modelo são do tipo NARX (*nonlinear autoregressive with exogenous inputs*) [Leontaritis and Billings, 1985, Aguirre et al., 1998] enquanto a última é uma equação linear ARX (*autoregressive with exogenous inputs*) [Ljung, 1987, van den Bosch and van der Klauw, 1994]. De fato, as não-linearidades das duas primeiras equações de (3.19) aparecem como conseqüência de um relação estática não-linear entre os sinais envolvidos [Aguirre and Jácome, 1998]. Apesar disso, o modelo é totalmente linear nos parâmetros e estimadores lineares bem conhecidos, como os métodos de mínimos quadrados [Ljung, 1987] podem ser usados na determinação dos parâmetros desconhecidos, como será discutido a seguir.

3.2.2 Estimação dos Parâmetros

O modelo (3.19) é uma representação matemática do sistema a ser controlado. Para que este modelo possa ser utilizado no projeto de controladores, seus parâmetros devem ser estimados. Uma maneira eficiente é utilizar métodos numéricos de otimização que, a partir de um conjunto de dados

reais do sistema, permite a estimação dos parâmetros desconhecidos de uma equação. Como foi mencionado, o modelo é linear nos parâmetros e por isso a família de métodos de mínimos quadrados lineares pode ser aplicada.

Como este tipo de método se baseia em dados de entrada e saída do sistema, deseja-se neste caso um conjunto de dados de entrada u_1 e u_2 que foram causas das saídas x, y e θ. A aquisição destes dados demanda uma série de considerações:

1. Sempre que possível, os dados devem ser coletados em malha aberta para evitar a correlação entre as entradas e os possíveis ruídos de medição, o que poderia causar polarização dos parâmetros [Aguirre, 1999];

2. O sistema deve ser corretamente excitado de forma que seja possível extrair a informação necessária dos dados;

3. Desde que o modelo (3.19) foi derivado considerando as leis físicas e assumindo algumas aproximações, um número de fenômenos reais como não linearidades entre as variáveis de controle e as forças aplicadas, podem não ter sido representados. Para reduzir o efeito destes fenômenos não modelados o robô deve ser excitado de forma a proporcionar pequenas variações em torno de determinado ponto de operação.

O item 1, é difícil de ser executado experimentalmente, porque, em geral, a área de atuação do robô é limitada. Para evitar que o robô colida com paredes ou obstáculos à sua volta, podendo causar além de descontinuidades nos dados, danos à sua estrutura, os sinais de excitação devem ter suas amplitudes escolhidas de forma a diminuir ou evitar as colisões. Caso isso não seja possível, aquisições em malha fechada são permitidas se a correlação cruzada entre o ruído e o sinal de entrada não for significativa em relação à variância do sinal de entrada [Aguirre, 1999]. Para tanto, pode-se adicionar

sinais aleatórios de alto espectro de freqüências às variáveis manipuladas do sistema. Estes sinais, que garantem também o item 2, podem ser PRBS (*Pseudo Random Binary Signals*) [Ljung, 1987] ou sinais similares. Este tipo de sinal é usualmente empregado em identificação de sistemas porque é de fácil obtenção e o espectro de freqüência correspondente é similar ao do ruído branco para uma determinada gama de freqüências. Para reduzir o efeito das não linearidades, pequenas amplitudes da PRBS podem ser utilizadas, juntamente com uma escolha adequada do seu período. Existe então um compromisso, já que pequenas amplitudes podem provocar uma pequena correlação cruzada entre as entradas e as saídas comprometendo a identificação. Para compensar este problema pode-se então adquirir um número suficientemente grande de pontos, o que por sua vez aumenta a probabilidade de colisões. Por estes e outros motivos, vê-se que a tarefa de coleta de dados dinâmicos não é muito simples e requer alguns cuidados.

Uma vez coletados os dados, pode-se aplicar, por exemplo, o método estendido de mínimos quadrados (EMQ) descrito no Apêndice B para a estimação dos parâmetros do modelo (3.19). Para estimação do atraso (tempo morto) pode-se utilizar entradas em degrau e fazer a estimação através da observação da resposta do sistema ou, se possível, métodos estocásticos de correlação cruzada [van den Bosch and van der Klauw, 1994].

3.3 Projeto do controlador

Nesta seção os modelos encontrados serão utilizados no projeto de controles lineares para o robô. Diferente do controlador da Figura 3.2 onde existem controladores locais de velocidade o controlador proposto é mostrado na Figura 3.5. Apesar de ser conhecido que os controladores em cascata permitem

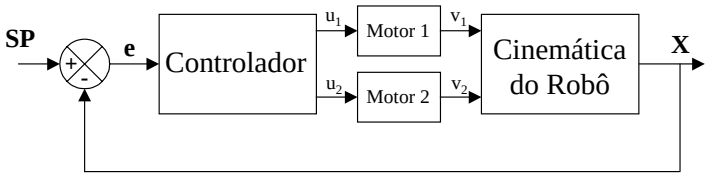

Figura 3.5: Sistema de controle proposto.

que o controle seja feito de forma mais eficiente, estes não serão utilizados pois deseja-se considerar que o robô é limitado e não possui nenhum processamento local.

3.3.1 Análise do sistema

Adotando o mesmo princípio utilizado na Equação (3.18) pode-se considerar que:

$$
\begin{aligned}
u^X_{1\,k-d} &= u_{1\,k-d}\ \cos\theta_{k-1} \\
u^Y_{1\,k-d} &= u_{1\,k-d}\ \operatorname{sen}\theta_{k-1} \\
u^X_{2\,k-d} &= u_{2\,k-d}\ \cos\theta_{k-1} \\
u^Y_{2\,k-d} &= u_{2\,k-d}\ \operatorname{sen}\theta_{k-1}\ .
\end{aligned}
$$

Note que o índice das entradas de controle é $k-d$ enquanto o índice de θ nesta equação é $k-1$, indicando o atraso mínimo de 1 período de amostragem [Philips and Nagle, 1995] entre o valor de θ medido e a saída do sistema e um atraso, d, entre o envio do sinal do controle e o seu efeito.

Baseado nas relações anteriores, o modelo (3.19) pode ser rescrito como:

$$\begin{aligned} x_k &= a_1^X\, x_{k-1} + a_2^X\, x_{k-2} + b_1^X\, u_{1\,k-d}^X + b_2^X\, u_{2\,k-d}^X \\ y_k &= a_1^Y\, y_{k-1} + a_2^Y\, y_{k-2} + b_1^Y\, u_{1\,k-d}^Y + b_2^Y\, u_{2\,k-d}^Y \\ \theta_k &= a_1^\Theta\, \theta_{k-1} + a_2^\Theta\, \theta_{k-2} + b_1^\Theta\, u_{1\,k-d} + b_2^\Theta\, u_{2\,k-d} \ . \end{aligned} \tag{3.20}$$

Pela Equação (3.5) vê-se que o sistema possui uma integração ou um pólo na origem do plano S [Ogata, 1993]. O mesmo pode ser observado em (3.8) que possui um pólo em $z = 1$ no plano Z [Philips and Nagle, 1995]. Estendendo esta observação para o modelo (3.20) vê-se que $a_1^i + a_2^i = 1$ ou $a_2^i = 1 - a_1^i$ [Philips and Nagle, 1995] onde i representa as variáveis do modelo. Desta forma, aplicando a transformada $\mathcal{Z}$ a cada uma das equações de (3.20) tem-se:

$$\begin{aligned} X &= \frac{b_1^X\, U_1^X\, z^{-(d-1)}}{(z-1)(z+a_2^X)} + \frac{b_2^X\, U_2^X\, z^{-(d-1)}}{(z-1)(z+a_2^X)} \\ Y &= \frac{b_1^Y\, U_1^Y\, z^{-(d-1)}}{(z-1)(z+a_2^Y)} + \frac{b_2^Y\, U_2^Y\, z^{-(d-1)}}{(z-1)(z+a_2^Y)} \\ \Theta &= \frac{b_1^\Theta\, U_1\, z^{-(d-1)}}{(z-1)(z+a_2^\Theta)} + \frac{b_2^\Theta\, U_2\, z^{-(d-1)}}{(z-1)(z+a_2^\Theta)} \ , \end{aligned} \tag{3.21}$$

onde as variáveis em letras maiúsculas, representam a transformada $\mathcal{Z}$ daquelas representadas anteriormente em minúsculas. Então $X = X(z) = \mathcal{Z}\{x\}$ onde o (z) foi omitido por simplicidade. Vê-se que não existem funções de transferência explícitas entre as entradas e as saídas do sistema pois cada saída depende de duas entradas do sistema, como já era previsto. Entretanto, para facilitar o uso dos métodos de análise e projetos de controladores comumente utilizados, deseja-se colocar estas equações em forma semelhante às funções de transferência. Considerando então, que o sistema é linear e

utilizando-se o teorema da superposição[3] [Ogata, 1993], tem-se:

$$G_X(z) = G1_X(z) + G2_X(z)$$
$$G_Y(z) = G1_Y(z) + G2_Y(z)$$
$$G_\Theta(z) = G1_\Theta(z) + G2_\Theta(z) \ ,$$

onde $G1_X(z) = X/U_1^X$, que pode ser estendido para os demais termos.

3.3.2 Efeito dos controladores de velocidade

Até este momento, considerou-se que os controladores de velocidade do tipo PID mostrados na Figura 3.2 tinham o objetivo de apenas melhorar o controle. Na verdade, sem a existência destes controladores, alguns cuidados devem ser tomados em relação às observações feitas anteriormente. Intuitivamente, tudo o que foi falado sobre o projeto de controladores independentes no controle de trajetórias de robôs só está correto se for garantido que as velocidades V_1 e V_2 sejam cumpridas pelo robô ou se a malha de controle for rápida o suficiente a ponto de rejeitar as perturbações nos motores. Como geralmente a velocidade do controlador externo não é muito grande, pois envolve sensores lentos como visão ou GPS[4], é necessário garantir que as velocidades dos motores estejam corretas. Este problema pode ser melhor explicado através de um exemplo. Se controlador de ângulo ou velocidade angular for desligado momentaneamente, o controlador de distância ou velocidade linear fornecerá dois sinais de referência idênticos para os motores; se neste caso, os motores não forem idênticos e as perturbações nulas, o que na

[3]Neste caso o teorema só pode ser utilizado matematicamente para efeito de análise, pois sabe-se que o robô não se deslocaria se um dos motores fosse desligado.

[4]*Global Positioning System* – Sensor usado para medir posição absoluta [Kobayashi et al., 1994].

prática é impossível, o robô certamente terá uma componente de velocidade angular (ver Equação (3.3)). Assim, é fácil notar que existe um acoplamento entre os controladores, já que a saída de um controlador depende também do sinal de controle do outro, tornado o projeto do controlador de velocidade angular dependente do projeto do controlador de distância e *vice-versa*.

Para uma prova matemática destas afirmações considere o diagrama de blocos da Figura 3.6. Nesta figura vê-se que há dois controladores digitais independentes, D_1 e D_2, cujas saídas estão ligadas como mostrado na Figura 3.3. Estes controladores poderiam ser por exemplo, um controlador de velocidade e um controlador de ângulo. Se for este o caso, as funções de transferência $G_3(z)$ e $G_4(z)$ correspondentes ao ângulo poderiam ser respectivamente $G1_\Theta(z)$ e $G2_\Theta(z)$ e aquelas correspondentes à velocidade, alguma combinação das funções de X e Y. Sabe-se que:

$$\begin{aligned} Y_1(z) &= [S_1(z) + S_2(z)]\, G_1(z) + [S_1(z) - S_2(z)]\, G_2(z) \\ Y_2(z) &= [S_1(z) + S_2(z)]\, G_3(z) + [S_1(z) - S_2(z)]\, G_4(z) \quad . \end{aligned} \tag{3.22}$$

Então, para haver um completo desacoplamento entre as duas malhas de controle basta que:

$$G_2(z) = G_1(z) \qquad \text{e} \qquad G_4(z) = -G_3(z) \quad . \tag{3.23}$$

Neste caso, existem funções de transferência explícitas entre as referências e as saídas que serão calculadas a seguir. Pela Figura 3.6:

$$E_1(z) = R_1(z) - Y_1(z) \qquad \text{e} \qquad E_2(z) = R_2(z) - Y_2(z) \quad .$$

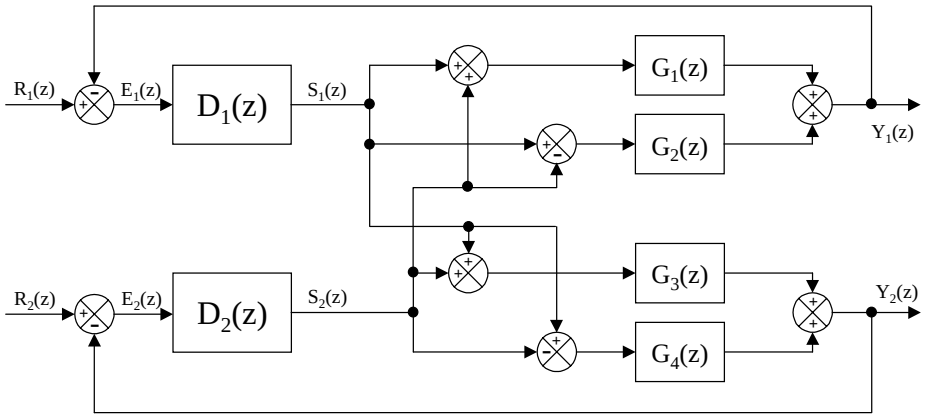

Figura 3.6: Diagrama de blocos de um controlador genérico sem os controladores em cascata.

A saída dos controladores é dada por:

$$S_1(z) = E_1(z)\ D_1(z) \qquad \text{e} \qquad S_2(z) = E_2(z)\ D_2(z)\ .$$

Substituindo estas equações na Equação (3.22), e fazendo as considerações de desacoplamento, após algumas manipulações algébricas tem-se:

$$\begin{aligned} \frac{Y_1(z)}{R_1(z)} &= \frac{2D_1(z)G_1(z)}{1 + 2D_1(z)G_1(z)} \\ \frac{Y_2(z)}{R_2(z)} &= \frac{2D_2(z)G_3(z)}{1 + 2D_2(z)G_3(z)} \end{aligned} \ ,$$

que são funções que relacionam as entradas de referências com as respectivas saídas de forma isolada, ou seja, sem nenhuma interferência entre as variáveis controladas. As condições de desacoplamento (3.23) são satisfeitas, se o ganho ou o numerador das funções de transferência G_1 e G_2, e G_3 e $-G_4$ forem iguais, já que os seus denominadores o são (ver equações (3.21))[5]. Quando existem controladores de velocidade locais devidamente ajustados

[5]Aqui os ganhos representam as funções de transferência totais dos motores que têm suas dinâmicas desprezadas em relação às constantes de tempo do robô.

e suficientemente rápidos para terem suas dinâmicas desprezadas, a relação estática entre os sinais de controle e o seu efeito é sempre constante. Como os controladores têm ainda as mesmas características, além de constantes estes ganhos são iguais[6]. Isto não acontece quando o sistema está em malha aberta pois, mesmo que os ganhos sejam constantes no tempo, o que normalmente não ocorre devido a problemas como diminuição da tensão de alimentação, eles são diferentes para cada subsistema (motor). Com isto, prova-se que uma outra função para os controladores locais de velocidade é o desacoplamento entre os controladores de orientação e posição. Como este trabalho pressupõe a ausência destes controladores, uma nova metodologia deverá ser abordada para que a relação requerida para os ganhos seja satisfeita. Mais à frente, será mostrado que há uma segunda dependência em relação as velocidades de resposta dos controladores que deverão ser observadas para melhorar o desempenho do sistema.

3.3.3 Desacoplamento entre os controladores

Nesta seção será mostrado que, mesmo sem os controladores locais de velocidade é possível o desacoplamento entre os dois controladores que formam o controlador de trajetória do robô. Como foi visto na seção anterior, há um perfeito desacoplamento se as funções de transferências relacionadas a cada um dos motores forem iguais. Portanto, antes de mostrar um método para torná-las iguais, é necessário mostrar estas funções e identificar a diferença entre elas. Para isto, considera-se a princípio que o controlador do robô é formado por duas malhas distintas: uma para minimizar a distância do robô

[6]Esta afirmação somente pode ser considerada verdadeira se existir uma integração no controlador e a mudança das referências de velocidade for suficientemente lenta para poder ser aproximada por um conjunto de degraus [Ogata, 1993, Kuo, 1995].

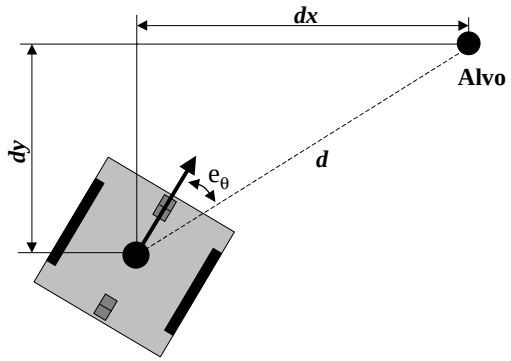

Figura 3.7: Função do controlador.

a um determinado ponto e outra para minimizar o ângulo ou a orientação. Na verdade, a função destes controladores é levar o robô de um ponto a outro já que o ângulo é calculado em tempo real e é função da posição do veículo. A Figura 3.7 mostra esta situação. Então, a entrada do primeiro controlador é a distância entre o robô e o alvo e a do controlador de ângulo é a diferença entre o ângulo que leva o robô em direção ao alvo e sua orientação atual. A distância entre o robô e o alvo pode ser calculada como:

$$d = \sqrt{dx^2 + dy^2} \ .$$

Como esta equação depende de x e y as funções de transferência envolvidas são aquelas mostradas nas duas primeiras linhas da Equação (3.21). Da mesma forma, o ângulo é obtido pela última função de (3.21). Assim, baseado no raciocínio da seção anterior, para que haja um completo desacoplamento entre os controladores de ângulo e distância as seguintes condições devem ser satisfeitas:

$$\begin{aligned} k_1^d &= k_2^d \\ k_1^\Theta &= -k_2^\Theta \ . \end{aligned}$$

onde k_1^d e k_2^d são os ganhos das funções de transferência de distância e k_1^Θ e k_2^Θ são os ganhos das funções de Θ. Para garantir que haja desacoplamento, pode-se então multiplicar o sinal de entrada de cada função de transferência por um determinado valor constante, de forma a satisfazer as condições anteriores.

Matematicamente, os ganhos são as variáveis independentes que multiplicam os sinais de controle. Por isso de (3.19),

$$k_1^\Theta = b_1^\Theta \ ; \qquad k_2^\Theta = b_2^\Theta \ .$$

Assim, considerando-se que u_1^Θ e u_2^Θ são os sinais de entrada das funções relativas a θ, uma das condições de desacoplamento é satisfeita se:

$$b_1^\Theta \, u_1^\Theta = b_2^\Theta \, u_2^\Theta$$

Para que esta relação seja válida, o lado esquerdo da equação pode ser multiplicado por b_2^Θ / b_1^Θ, que pode ser considerada a constante de desacoplamento para a variável θ. Fisicamente, o uso desta constante faz com que o robô gire parado quando recebe variáveis de controle de mesma amplitude e sinais opostos. Note que o sinal negativo mostrado anteriormente é inerente a um dos ganhos da função de θ, como pode ser visto na Equação (3.16).

As condição de desacoplamento para a distância (ou movimento de translação) não pode ser facilmente explicitada através das funções de transferência do sistema, pois envolvem operações não-lineares dos submodelos de x e y. Entretanto, uma boa aproximação para esta constante, pode ser obtida através do modelo de θ. Isto ocorre, porque, apesar das funções de transferência serem obtidas de forma independente, fisicamente elas foram geradas pelo mesmo sistema, ou seja, os motores responsáveis por todas as

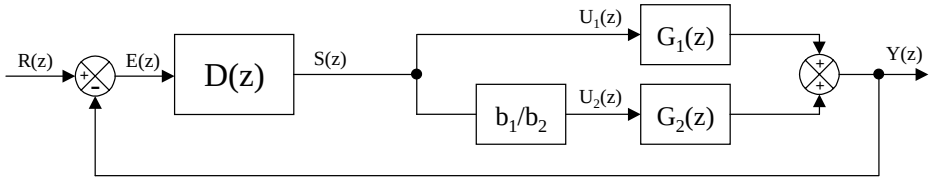

Figura 3.8: Diagrama de blocos do controlador de θ com desacoplamento.

variáveis são os mesmos. Como a função de desacoplamento representa basicamente a razão entre os ganhos dos motores, é de se esperar que esta razão seja a mesma, tanto para a distância quanto para o ângulo. Assim, a relação b_2^Θ/b_1^Θ pode ser utilizada para multiplicar um dos sinais de controle e tornar iguais os ganhos dos dois motores. Como no caso de θ, fisicamente, o uso da constante anterior, faz com que o robô ande em linha reta quando recebe sinais de controle iguais. Desta forma, nota-se que há realmente um desacoplamento entre as saídas do processo, pois consegue-se isolar cada uma delas[7]. Por esta análise, vê-se que, com o uso das constantes de desacoplamento, existe uma lei para o comportamento do robô: os sinais de modo comum provocam translação e os sinais de modo diferencial provocam rotação. Isto não poderia ser dito antes do uso de tais constantes.

A Figura 3.8 mostra o uso da constante de desacoplamento b_1^Θ/b_2^Θ no controlador de ângulo. O mesmo pode ser feito para o controlador de distância. Observe, que a função de desacoplamento é dependente de parâmetros do modelo, que podem ser encontrados através métodos de estimação tradicionais como foi mostrado na Seção 3.2.2 ou em tempo real como será mostrado a seguir.

[7]Considera-se que existem duas saídas: movimento translacional e rotacional.

3.3.4 Identificação em tempo real

A grande vantagem de se obter em tempo de execução as funções de desacoplamento entre os controladores é que neste caso possíveis mudanças no processo como diminuição da tensão de alimentação e desgaste das rodas e engrenagens do robôs que fatalmente alterarão os ganhos de velocidade dos motores serão detectadas. Uma forma de encontrar os ganhos em tempo real durante o controle é utilizar algoritmos recursivos de estimação de parâmetros sobre o modelo paramétrico desenvolvido na Seção 3.2. Assim, como as variáveis de desacoplamento são funções destes ganhos, elas podem ser calculadas a cada intervalo de amostragem. O Apêndice B mostra um algoritmo de mínimos quadrados que pode ser utilizado na identificação em tempo real.

Na estimação de parâmetros dois cuidados principais devem ser considerados. Em primeiro lugar, o fator de esquecimento, que indiretamente determina o número de dados a serem usados na estimação dos parâmetros, deve ser corretamente ajustado. Se poucos dados forem usados, os parâmetros podem variar muito rapidamente pois estão mais sujeitos ao ruído causando resultados insatisfatórios. Além disso, o uso de um fator de esquecimento pequeno pode causar uma não-convergência do estimador já que a parte "visível"[8] do sinal pode não ser suficiente para a estimação, devido às próprias características dos sinais de entrada [Aguirre, 1999]. Ao mesmo tempo, a utilização de muitos dados pode tornar o sistema muito conservativo o que causaria uma variação muito lenta nos parâmetros. Por isso, se o sistema for considerado linear, uma análise das constantes de tempo das possíveis variações do robô

[8] A expressão "visível" é utilizada porque o fator de esquecimento, λ , pode estar relacionado a uma janela temporal assintótica definida em [Harris and Billings, 1995] como: $JTA = 1/(1-\lambda)$.

ajudariam a escolher o valor do fator de esquecimento. Como o sistema não é linear, uma outra função da estimação recursiva é identificar os ganhos em cada região de trabalho do robô. Por isso, a escolha do fator de esquecimento deve levar em consideração também, a velocidade de variação das entradas de controle.

Em segundo lugar, deve-se prestar atenção a todo momento se a estimação está sendo feita corretamente. Estimações de parâmetros erradas podem acontecer se o sinal de controle não for suficiente para excitar o sistema [Åström and Wittenmark, 1995] ou se acontecerem problemas físicos, como por exemplo uma colisão. Este tipo de problema pode ser identificado se as variâncias dos parâmetros forem monitoradas. Se estas variâncias estiverem aumentando, problemas podem estar ocorrendo. Nestes casos, o estimador recursivo deve ser desligado e os últimos valores estimados usados como parâmetros do sistema. Para detectar o momento de desligamento do estimador de parâmetros, métodos heurísticos podem ser desenvolvidos ou, de forma mais elegante, podem ser aplicadas técnicas de detecção e diagnóstico de falhas em sistemas dinâmicos [Frank and Seliger, 1991, Caminhas, 1997].

3.3.5 Estrutura do controlador

Uma vez proposta uma metodologia para desacoplar os controladores, pode-se encontrar uma estrutura para o sistema de controle geral do robô. Como proposto anteriormente, a malha de controle pode ser formada por um controlador para o ângulo e um outro sistema para minimizar a distância entre o robô e o alvo. Para o ângulo, pode-se utilizar um controlador PI digital como a função $D(z)$ da Figura 3.8, seguido de uma estimação em tempo real da constante de desacoplamento. Este controlador deve ser projetado para ser o mais rápido possível e para que haja pouco ou nenhum sobresinal.

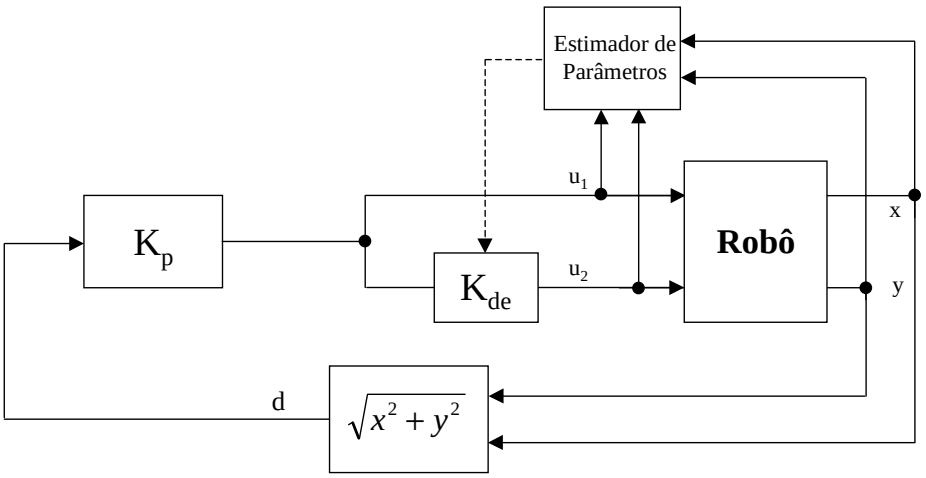

Figura 3.9: Diagrama de blocos do controlador de distância.

Para a malha de distância, poder-se-ia adotar a mesma estratégia, mas como a função desta malha é sempre tornar a distância até o alvo nula e o sistema já possui integração, por simplicidade de projeto, um controlador proporcional com estimação em tempo real da função de desacoplamento pode ser uma boa solução. Esta configuração é mostrada na Figura 3.9. Se um controlador proporcional puro for utilizado, o valor do ganho proporcional está diretamente relacionado com a velocidade linear do robô. Assim, nesta configuração, o robô estará mais rápido se estiver longe do alvo e se torna mais lento à medida que se aproxima dele. Desta forma, o controlador de distância poderia facilmente ser substituído por um controlador de velocidade linear. O maior problema então seria a estimação da velocidade, já que esta pode não estar disponível no sistema de medição.

Antes de iniciar o processo de determinação do controlador, uma característica de simetria do robô pode ser utilizada para diminuir o seu tempo de resposta. A princípio, pode-se considerar que não existe diferença construtiva entre a frente e as costas do robô, permitindo com que este se mova da mesma forma em ambos os sentidos. Assim, um método simples pode

ser utilizado em tempo de processamento para determinar se o robô deve andar para frente ou para trás. Este procedimento, evitará que o robô tenha que girar ângulos acima de $\pi/2$ radianos economizando tempo. Para tanto, o seguinte algoritmo deve ser aplicado aos erros, antes de "entregá-los" aos controladores:

while $e_\theta > \pi$ **do** {normalização entre $-\pi$ e π}
  $e_\theta = e_\theta - 2\pi$
end while
while $e_\theta < -\pi$ **do**
  $e_\theta = e_\theta + 2\pi$
end while

if $e_\theta < -\pi/2$ **then** {muda a frente do robô}
  $e_\theta = e_\theta + \pi$
  $d = -d$
else if $e_\theta > \pi/2$ **then**
  $e_\theta = e_\theta - \pi$
  $d = -d$
end if

onde e_θ e d são os erros de orientação e a distância ao alvo respectivamente. A primeira parte do algoritmo, serve para tornar o robô simétrico em relação ao ângulo, fazendo com que a referência de ângulo seja atingida pelo menor caminho possível. A segunda parte, torna o robô simétrico em relação ao eixo perpendicular à sua roda, já que muda a frente do robô de acordo com a posição do alvo. Esta operação é ilustrada na Figura 3.10.

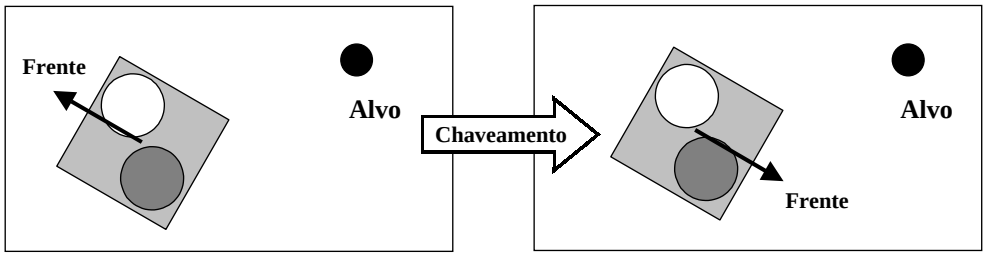

Figura 3.10: Utilização da simetria do robô para diminuir o tempo de resposta.

3.3.6 Projeto por Lugar das Raízes

Como foi proposto, como controlador de ângulo pode ser usado um controlador "proporcional+integral" digital. Neste controlador a parcela proporcional é utilizada para melhorar o regime transitório e a parcela integral é utilizada para melhorar a resposta em regime permanente do sistema com o acréscimo de um pólo dominante em $z = 1$. Desta forma, um sistema estável que já possui integração, como é o caso do robô, pode alcançar um erro nulo até mesmo para entradas em rampa [Ogata, 1993]. Uma possível equação de diferenças para um PI digital clássico é [Philips and Nagle, 1995]:

$$u_k = u_{k-1} + K_p\left(e_k - e_{k-1}\right) + \frac{K_i\,T}{2}(e_k + e_{k-1}) \ , \tag{3.24}$$

onde a K_p é o ganho proporcional, K_i é o ganho integral, e é o erro e u é a saída do controlador. A função de transferência deste controlador pode ser escrita como:

$$\frac{U(z)}{E(z)} = D(z) = K_p + K_i\,\frac{T}{2}\left(\frac{z+1}{z-1}\right) \ , \tag{3.25}$$

ou de forma mais compacta como:

$$D(z) = K_D \left(\frac{z - z_o}{z - 1} \right) \ , \tag{3.26}$$

onde:

$$K_D = K_p + K_i\, T/2 \ ,$$
$$z_o = \frac{K_p - K_i\, T/2}{K_p + K_i\, T/2} \ .$$

Desta forma, o projeto do controlador consiste em encontrar as constantes K_D e z_o que dão origem aos ganhos K_p e K_i respectivamente como:

$$K_p = K_D\, \frac{1 + z_o}{2} \ , \tag{3.27}$$
$$K_i = K_D\, \frac{1 - z_o}{T} \ . \tag{3.28}$$

No projeto do controlador pode-se usar o "princípio do dipolo" [Kuo, 1995] que se baseia na escolha do zero da função $D(z)$ muito perto do pólo localizado em $z = 1$. Assim, o ganho efetivo proveniente do controlador é essencialmente igual a K_D na Equação (3.26) e a estabilidade relativa do sistema é mantida.

O controlador pode então ser projetado usando-se o método do lugar das raízes. Assim, a terceira equação de (3.21), após o desacoplamento e com um atraso mínimo pode ser rescrita como:

$$G_\Theta(z) = \frac{K}{(z - 1)(z + a_2^\Theta)} \ , \tag{3.29}$$

onde $K = 2\, b_1^\Theta$. Com a ajuda de um diagrama de lugar das raízes, é possível escolher um novo ganho K de forma que o sistema tenha as características

de projeto requeridas para a malha fechada. Como o controlador utiliza o princípio do dipolo, o ganho K_D deriva da relação $K_D = K/(2\,b_1^{\Theta})$, onde K é o ganho escolhido no gráfico de lugar das raízes. A escolha do novo valor de K pode ser feita traçando-se as regiões do gráfico onde os parâmetros de projeto ζ e ω_n são válidos e encontrando a interseção destas regiões com o lugar das raízes da função. A Figura 3.11 mostra um mapa no plano z com algumas linhas onde estes parâmetros são constantes. O parâmetro ζ, conhecido como coeficiente de amortecimento (*damping ratio*), está diretamente relacionado com o tempo de acomodação (*settling time*) e com o máximo sobresinal do sistema realimentado. O parâmetro ω_n, ou freqüência natural, juntamente com ζ, está também relacionado com o tempo de acomodação da malha. Estas relações podem ser expressas por [Ogata, 1993]:

$$\zeta = \frac{-\ln(Os/100)}{\sqrt{\pi^2 + (\ln(Os/100))^2}} \tag{3.30}$$

$$\omega_n = \frac{4}{\zeta\, T_s}\, T \ , \tag{3.31}$$

onde Os é o sobresinal (*overshoot*) percentual e T_s é o tempo de acomodação do sistema. A escolha do tempo de acomodação pode ser feita em função da constante de tempo, τ, do sistema original que pode ser calculada a partir da definição da transformada $\mathcal{Z}$, como:

$$\tau = \frac{-T}{\ln(|a_2^{\Theta}|)} \ . \tag{3.32}$$

A função de transferência de ramo direto do sistema controlado pode ser

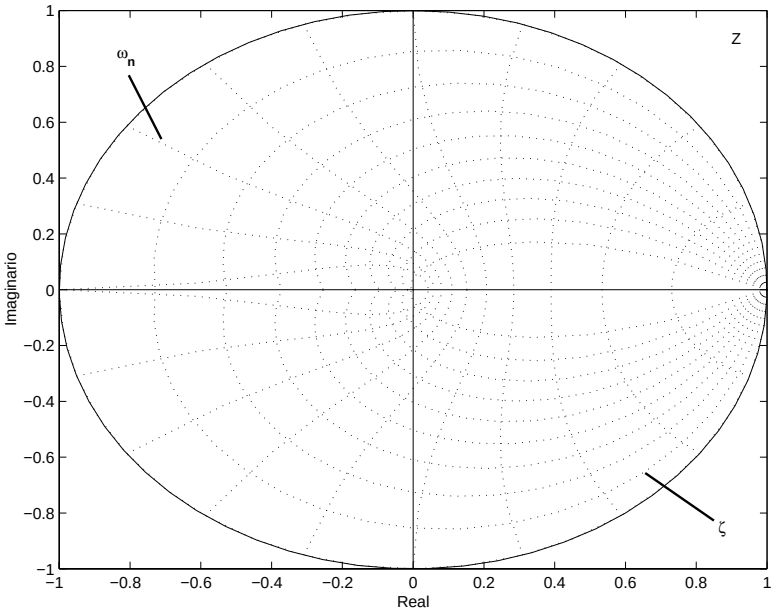

Figura 3.11: Localização dos parâmetros ζ e ω_n no plano z.

escrita como:

$$D(z)G_{\Theta}(z) = \frac{K_D\,(z - z_o)}{(z-1)^2(z - a_2^{\Theta})} \ , \tag{3.33}$$

onde z_o é um zero estável muito perto de 1 (em geral $0,990 \leq z_o \leq 0,999$) e K_D é determinado pelo gráfico de lugar das raízes [Ogata, 1993]. Determinados estes valores a implementação do controlador de ângulo pode então ser feita através da equação de diferenças (3.25) onde os ganhos são determinados por (3.27) e (3.28).

Para o projeto do controlador de distância a mesma metodologia poderia ser utilizada. Entretanto, como as funções de transferência neste caso são muito não-lineares e complicadas, o projeto é não-trivial. Como este controlador é formado apenas por um ganho e seu valor não é muito crítico uma boa aproximação é considerar a função de transferência de distância uma

função linear semelhante àquela de ângulo onde o pólo diferente de 1 pode ser escolhido entre um dos pólos das funções de x e y, ou alguma combinação dos dois. Com esta função, pode-se então projetar o valor do ganho K_p da Figura 3.9 como foi feito para o ângulo. A principal diferença é que neste caso, o sistema deve ser superamortecido com $\zeta > 1$ [Ogata, 1993] para que o robô atinja o alvo monotônicamente. Além disso, este controlador deve ser mais lento que aquele para o ângulo fazendo com que o ângulo "siga" ou se torne escravo da distância e o robô se estabilize rapidamente na trajetória ou alvo, que em geral é uma posição no espaço.

3.3.7 Efeito do atraso

Na seção anterior, o atraso foi considerado como sendo igual a unidade ou seja, o mínimo possível para sistemas amostrados. Na verdade, o atraso, em geral não pode ser desprezado pois a sua presença tem o efeito do aparecimento de pólos em $z = 0$ como é mostrado na Figura 3.12(b), onde foi desenhado o lugar das raízes para uma função semelhante à Equação (3.29) com o pólo $a_2^{\Theta} = 0,6$ e um tempo morto $d = 5$. Em oposição, a Figura 3.12(a) mostra o mesmo sistema com o atraso mínimo $d = 1$. Vê-se que em relação ao lugar dos pólos comuns entre as funções, há uma grande diferença entre os dois gráficos. Por isso, a não ser que haja uma compensação dos novos pólos de atraso ou que seja tolerado um erro de projeto, o atraso deve ser considerado durante a escolha dos ganhos do controlador.

As cruzes (+) na Figura 3.12(b) mostram onde estariam todas as raízes para um determinado ganho. Este ganho foi escolhido como estando no limite de estabilidade (dado pelo círculo de raio unitário) [Philips and Nagle, 1995] provocado pelos pólos originais do sistema. Por esta análise, para este tipo de sistema específico, pode ser visto que mesmo no limite, os pólos derivados do

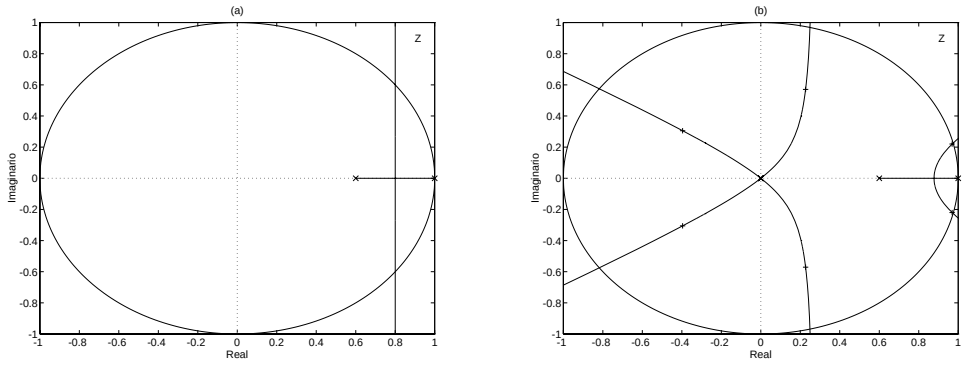

Figura 3.12: Efeito do atraso no lugar das raízes: (a) - lugar das raízes do sistema sem atraso ou com atraso simples; (b) - lugar das raízes do sistema com atraso igual a 5 intervalos de amostragem.

atraso continuam estáveis, indicando que mesmo sem compensação, o sistema ainda pode ser estável. Esta informação dá um indício de que a utilização do diagrama da Figura 3.12(b) no projeto do controlador, pode ainda manter o sistema estável, se forem utilizados valores de ζ bem próximos ou maiores que 1. Apesar disso, além do risco de se projetar controladores com especificações diferentes das desejadas, o não-conhecimento ou desprezo do atraso, pode resultar também, em malhas de controle instáveis. Isto ocorre, devido à grande diminuição da margem de ganho provocada pelo "curvamento" do lugar das raízes. Com a redução desta margem, a gama de controladores que estabilizam a malha fica reduzida e muitas vezes, não é possível encontrar um ganho que respeite as especificações de projeto, principalmente em relação ao tempo de acomodação. Assim, o principal efeito do atraso nos sistemas de controle em geral é empobrecimento da resposta transitória do sistema através de um aumento da oscilação (podendo chegar à instabilidade) e/ou um aumento no tempo de resposta. Para evitar que ocorra este tipo de problema, pode ser aplicada uma compensação do atraso que geralmente não é muito simples. Uma solução usando preditores é mostrada a seguir.

3.3.8 Compensação do atraso

Para compensar o atraso ou tempo morto, o uso de preditores pode ser empregado [Harris and Billings, 1995]. Tradicionalmente, uma das técnicas mais utilizadas é o Preditor de Smith [McMillan, 1994] que cancela o tempo morto da função de transferência para mudanças de referências através da adição do sinal de controle filtrado e atrasado à variável medida. Apesar de apresentar bons resultados em comparação com os controladores convencionais, o principal problema desta configuração é que o tempo morto em relação aos distúrbios de carga não é cancelado. Outro problema é que o Preditor de Smith pode alterar o comportamento dos controladores dificultando o seu projeto além de ser restrito a uma gama limitada de processos. Em oposição, pode-se utilizar preditores que atuem no caminho de realimentação dos controladores. Tenta-se então, a partir dos valores atuais, fornecer valores futuros para saídas do sistema. Assim, o erro na entrada dos controladores é calculado entre as referências e as saídas dos preditores, e não mais entre essas e as saídas do sistema. Esta configuração é mostrada na Figura 3.13 para o controle de ângulo. O maior problema, é que obviamente, aos preditores estão associados erros de predição que podem causar respostas insatisfatórias. Entretanto, se os preditores forem confiáveis e precisos, eles podem melhorar significativamente a resposta do sistema. A maior vantagem desta configuração é que o controlador pode ser projetado simplesmente desconsiderando-se[9] o atraso puro de tempo. Além disso, como o preditor está ligado diretamente às saídas, o atraso é compensado tanto para as referências quanto para as perturbações.

Para antecipar o comportamento do processo, o uso de modelos dinâmicos

[9] Algumas vezes o atraso não pode ser totalmente compensado e é apenas diminuído. Nestes casos deve-se então, considerar o atraso resultante.

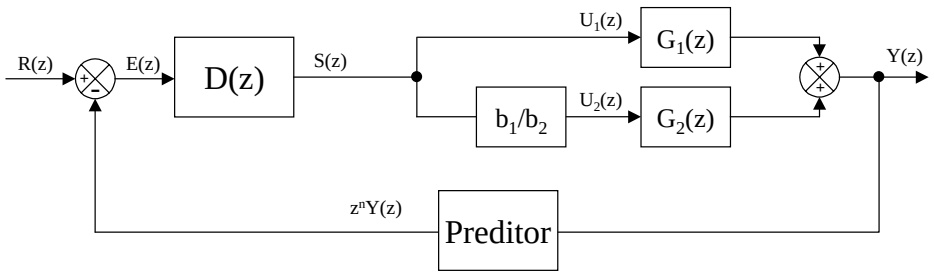

Figura 3.13: Uso de um preditor para compensação do atraso.

como os mostrados na Seção 3.2 podem apresentar bons resultados. Para tanto, pode-se utilizar ainda, modelos com atualização paramétrica em tempo real aumentando a sua exatidão. Se nesta atualização forem utilizados os métodos de mínimos quadrados recursivos, como foi proposto na Seção 3.3.4, é garantido que a predição de um passo á frente é a melhor possível para um determinado nível de ruído [Ljung, 1987]. O problema surge então, quando o atraso é maior que um intervalo de amostragem e o modelo deve ser realimentado com suas próprias predições. Neste caso, o preditor somente terá bom desempenho se o modelo realmente representar o processo. Em sistemas com integração, como é o caso do robô, este problema se torna ainda mais acentuado, pois há uma acumulação dos erros de cada passo de predição, gerando um deslocamento (*offset*) nas saídas preditas. Como é difícil corrigir este problema, pode-se fazer um teste anterior para verificar se o erro é significativo ou pode ser desprezado. Como já foi mencionado, no caso de estimação de parâmetros em tempo real (*on-line*), deve-se ainda, tomar o cuidado de verificar se a mesma está sendo feita corretamente, evitando assim o comprometimento do modelo e consequentemente da predição.

Nos dois próximos capítulos, as metodologias propostas aqui serão testadas na prática para controlar micro-robôs comandados por um computador externo e observados por um sistema de visão computacional.

Capítulo 4

Resultados Experimentais: Modelagem

Não basta conquistar a sabedoria, é preciso usá-la.

Marcus Tullius Cícero (106–43 a.C.)

NESTE capítulo e no próximo serão implementadas em simulação e na prática as metodologias propostas no capítulo anterior. Para tanto, serão utilizados os robôs do time MIneiROSOT [Campos et al., 1998] construído por alunos de graduação da UFMG para disputar os campeonatos mundiais de futebol de robôs de 1998 e 1999, na categoria MIROSOT.

Este capítulo está dividido da seguinte forma: a Seção 4.1 faz uma descrição das características construtivas do sistema e a Seção 4.2 determina um modelo dinâmico para o sistema através da estimação dos parâmetros da estrutura determinada no capítulo anterior. O Capítulo 5 mostra, então, o projeto do controlador e os resultados encontrados a partir deste modelo.

4.1 Descrição do sistema

Existem várias abordagens na construção de sistemas para futebol de micro-robôs, [Sargent et al., 1997, Achim et al., 1996]. Em relação ao lugar em que a "inteligência" está concentrada este desenvolvimento tem sido

divido em três categorias principais: modelos centralizados, distribuídos e híbridos.

No modelo distribuído, cada agente tem um processador que controla suas próprias ações, baseado nas informações do seu próprio conjunto de sensores. Cada robô comunica com o outro através de uma rede sem fio. Este modelo é o mais geral e permite várias oportunidades de desenvolvimento.

O modelo híbrido assume a existência de um processador central para algum controle e estratégia, mas algumas destas funções podem ser realizadas localmente por cada robô. É assumido também, a existência de um sistema dedicado mais poderoso, que ficará encarregado de processar os sensores locais e a comunicação com cada robô. O processador central pode ter acesso a um ou mais sensores globais, como visão.

O modelo de inteligência centralizada, assume a existência de um processador responsável por tudo relacionado com a estratégia e controle em diversos níveis. Os agentes (robôs) restringem seu conhecimento a seus próprios dados como posição e velocidade. Na maioria dos casos onde a inteligência centralizada é utilizada, os robôs tendem a ser muito simples e em geral eles possuem poucos ou nenhum sensor local. Por isso, no modelo centralizado, o processador deve ter toda a informação de todos os agentes e as decisões são tomadas baseadas em sensores globais, como visão. Em alguns casos, os agentes podem ser equipados com outras modalidades de sensores, mas na maioria das vezes, eles enviam a informação para o computador central para serem processadas.

O sistema do MIneiROSOT segue a tendência da maioria dos times que disputam o MIROSOT e usa o modelo de inteligência centralizada. Desta forma os robôs são controlados por um conjunto de algoritmos que rodam em um único computador. Um outro computador é usado para processar

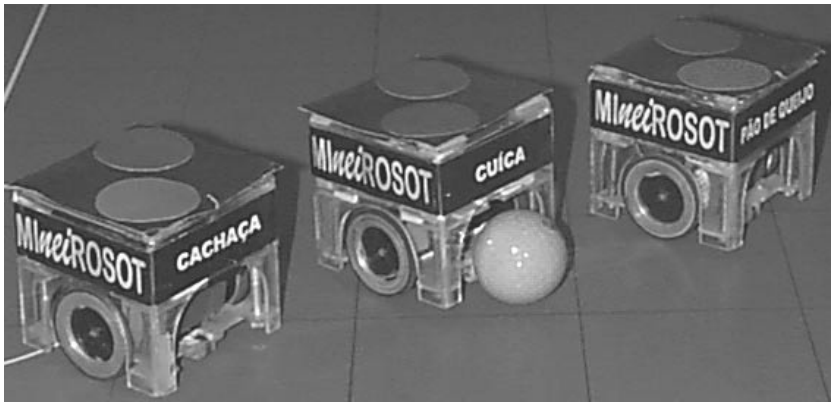

Figura 4.1: O time MIneiROSOT.

a informação de visão, como será discutido a seguir. O sistema completo é então constituído de três micro-robôs, mostrados na Figura 4.1, dois computadores e uma câmera de vídeo. A seguir serão detalhadas cada uma das partes constituintes do sistema.

4.1.1 Robô

O *hardware* do sistema envolve basicamente as áreas de eletrônica e mecânica. Nesta seção será feita uma descrição da parte eletrônica de controle e comunicação bem como da estrutura mecânica do robô. A principal linha do projeto era obter robôs de baixo custo e fáceis de se construir. Como será descrito a seguir, uma das principais dificuldades é a restrição de dimensões imposta pelas regras da FIRA [FIRA, 1999], onde cada robô não pode ocupar um volume maior que um cubo de $7,5\,\text{cm}$ de lado.

Eletrônica

O robô foi construído baseado em componentes de aeromodelos remotamente controlados (R/C). Estes incluem microservos, receptores e transmissores. O sistema é muito similar àquele descrito em [Hong et al., 1997].

Esta abordagem possui algumas vantagens:

- Utiliza componentes R/C padrões, facilmente encontrados comercialmente;
- Os servos são facilmente modificáveis, como será mostrado a seguir, para funcionar como servos de velocidade;
- O sistema de rádio é muito confiável e robusto e existem várias freqüências disponíveis.

O sistema de transmissão funciona da seguinte forma: a informação de velocidade (posição no sistema R/C original) é codificada como um pulso que pode variar de 1 ms a 2 ms. O zero está em $1,5$ ms. A 1 ms o servo está na velocidade máxima em uma direção, e a 2 ms, na velocidade máxima na direção oposta. Para controlar diversos servos é formada uma seqüência de pulsos separados por um pulso de sincronismo de $0,3$ ms. A seqüência é repetida em intervalos de 20 ms. Este trem de pulsos é então modulado por um transmissor na freqüência portadora.

Nos sistema R/C típicos, a largura de cada pulso é ajustada com a ajuda de um *joystick*. Entretanto, para proporcionar o controle dos servos via computador, é necessário a construção de uma interface que deve receber como entrada um vetor com as velocidades desejadas para cada servo e converter este vetor em pulsos com a largura correta a ser enviada para o transmissor R/C. Esta interface entre o computador e o transmissor foi implementada com um PIC16F84 [Microchip, 1996] e outros poucos componentes [Tavares Filho, 1997a]. A maioria destes transmissores é equipada com uma entrada que é chamada "entrada de instrutor". Esta entrada permite a conexão de um cabo do controle do instrutor ao controle do aluno, permitindo que esse tome o controle do aeromodelo em algumas situações.

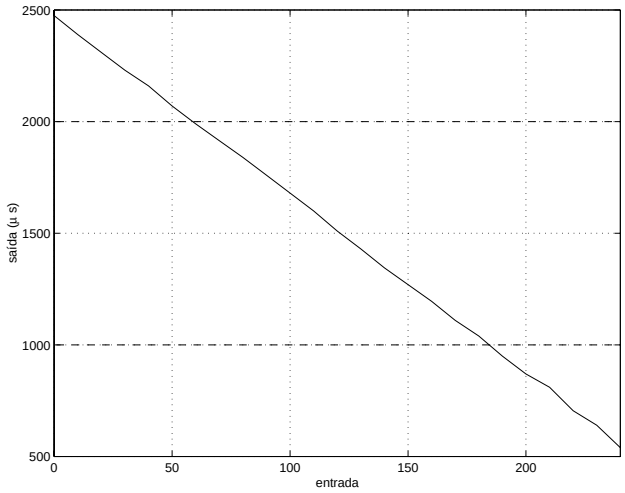

Figura 4.2: Relação entre a variável enviada pelo computador e a largura do pulso no receptor.

As informações de velocidade são então enviadas ao robô por esta entrada que deve ser sempre mantida no modo instrutor.

O transmissor R/C é capaz de prover seqüências de pulsos continuas. Entretanto, devido a restrições de tempo de microcontrolador utilizado, o computador quantiza as larguras de pulso em apenas 256 valores[1]. No receptor, um circuito extrai os pulsos da portadora e os disponibiliza para cada servo. A relação existente entre o sinal mandando pelo computador e o valor do pulso medido no receptor é mostrado na Figura 4.2. O conjunto receptor/servos é alimentado por dois conjuntos de duas baterias níquel-cádmio (NiCd) ligados em série, totalizando uma tensão de $4,8$ volts.

Mecânica

Como já foi mencionado, a parte mecânica do robô se baseia em micro-servos, modificados para prover altas velocidades com o maior conjugado

[1]Apesar desta quantização, a Figura 4.2 mostra que, efetivamente, apenas 128 valores são utilizados já que para os servos apenas os valores entre 1 e 2 ms têm algum significado.

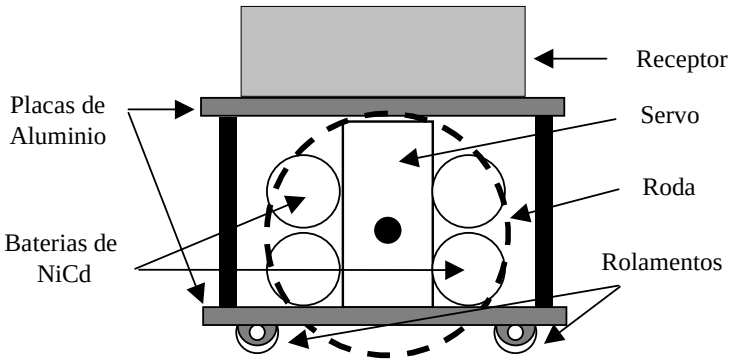

Figura 4.3: Estrutura mecânica do robô.

possível [Tavares Filho, 1997b]. O modelo de servo utilizado foi o FP-S3101 fabricado pela Futaba [Futaba, 1999]. Estes foram escolhidos principalmente pelo seu tamanho reduzido. Originalmente os servos têm um conjugado de saída de $2,5\,\mathrm{kgf}\cdot\mathrm{cm}$ e velocidade de $64,8\,\mathrm{rpm}$. Como esta velocidade é muito baixa para o futebol de robôs foi necessária a remoção de uma das engrenagens da caixa de redução original de cada servo. O novo conjugado foi então de $0,186\,\mathrm{kgf{\cdot}cm}$ e a velocidade passou para $892\,\mathrm{rpm}$. Outra modificação do servo, foi a substituição de um potenciômetro de realimentação de posição por resistores fixos o que possibilitou que o sinal do receptor representasse velocidade.

Depois os servos foram acomodados em uma estrutura juntamente com a bateria e o receptor como é mostrado na Figura 4.3. Esta estrutura é formada por duas placas de alumínio quadradas com $7,5\,\mathrm{cm}$ de lado, conectadas e espaçadas por dois parafusos longos. As rodas foram construídas de plástico coladas a um anel de alumínio coberto por borracha. O raio final foi de $1,85\,\mathrm{cm}$, o que permite uma velocidade máxima de $172,8\,\mathrm{cm/s}$. Na parte inferior do robô existem dois pequenos rolamentos para favorecer o equilíbrio do robô. O robô é protegido por uma cobertura de acrílico facilmente removível.

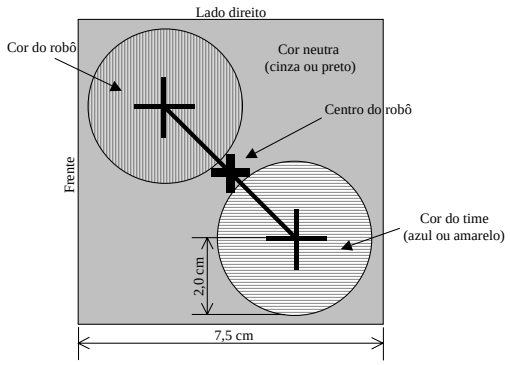

Figura 4.4: Sistema de identificação dos robôs.

4.1.2 Visão

O sistema de visão é constituído de uma câmera NTSC convencional e um computador equipado com um placa de aquisição de imagens com a capacidade de adquirir 60 quadros/s. O computador utilizado é um Pentium II, 330MHz rodando Windows98. Um programa foi desenvolvido em linguagem de baixo nível para processar as imagens em tempo real e permitir o rastreamento da trajetória de diversos objetos baseado nas suas cores. Assim, cada robô tem uma cor própria e mais uma cor, comum ao time, que permite a sua identificação. A Figura 4.4 mostra o sistema de identificação dos robôs, onde, a partir do centro de cada cor é possível determinar a posição e orientação dos agentes.

A saída do sistema de visão é um vetor de caracteres, que é transmitido via cabo de rede ao computador responsável pelo controle, contendo a posição e orientação de cada robô, dos adversários e da bola. Este sistema é bastante estável e muito robusto a variações de iluminação o que não é muito comum a sistemas do mesmo tipo. O processamento das imagens é feito a 40 quadros/s a uma resolução de 352×240 *pixels*. A Figura 4.5 mostra a característica dos dados de um robô parado na saída do sistema de visão. Note que as

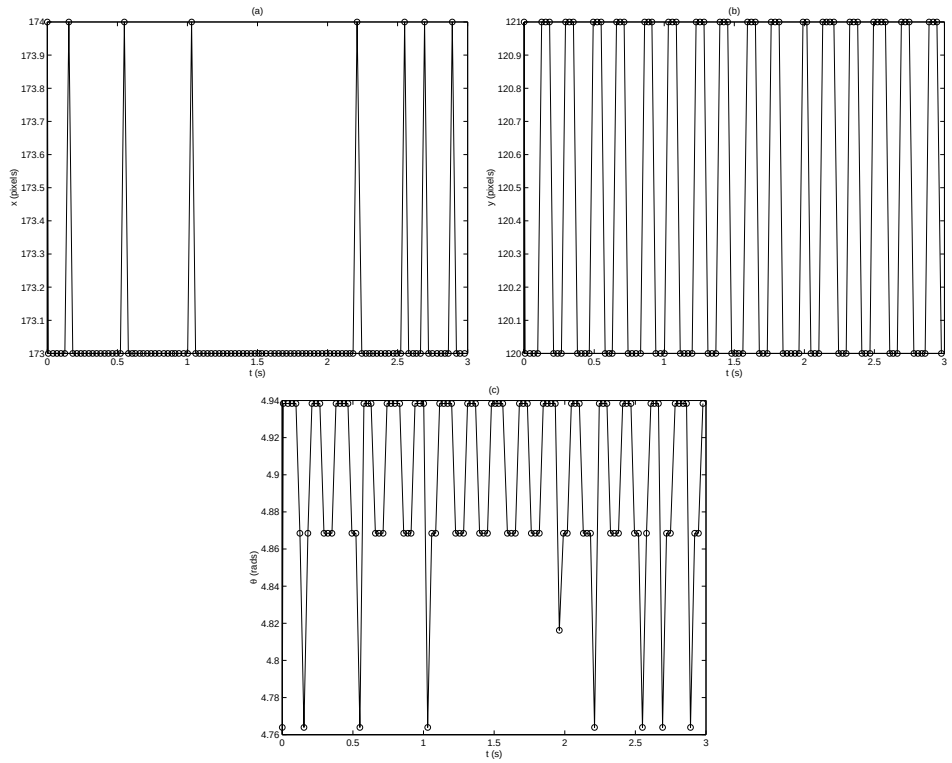

Figura 4.5: Saída do sistema de visão: (a) – coordenada x; (b) – coordenada y; (c) – orientação.

coordenadas de posição são números inteiros (*pixels*) e por isso se robô está entre dois inteiros existe a presença de um ruído. Por esse motivo, em geral a coordenada y é mais ruidosa devido a menor resolução. A orientação, apesar de ser mostrada em radianos, é também quantizada. Para diminuir o *overhead* de transmissão esta variável é transmitida pela rede em graus inteiros e depois convertida para radianos. Os desvios padrões das variáveis x, y e θ são respectivamente $0,26$ e $0,50$ *pixels* e $0,05$ radianos.

4.1.3 Controle

O sistema de controle é desenvolvido em um micro computador dedicado que usa o sistema operacional LINUX. Este computador, um Pentium

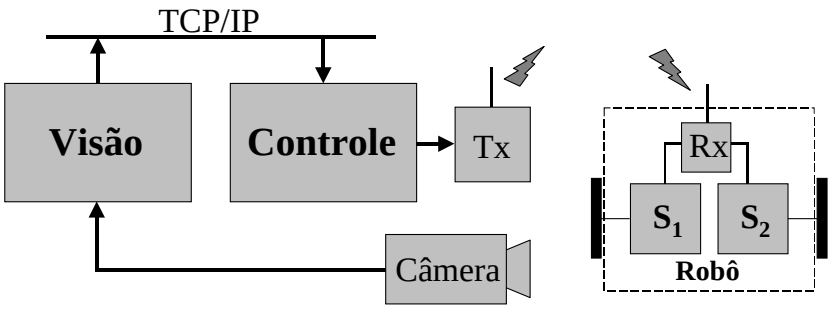

Figura 4.6: Arquitetura do sistema.

de 133MHz, comunica com o sistema de visão através de um protocolo assíncrono de comutação de pacotes do tipo datagrama [Stallings, 1996]. Os dados de velocidade dos robôs são enviados para a interface com o transmissor pela porta serial (RS232) também de forma assíncrona a uma taxa de 4800 bps. O sistema foi desenvolvido de forma que o único código de programa a ser modificado está neste computador sendo que o restante do sistema é considerado fixo. Assim, a única forma de interação com o usuário é feito por esta máquina através de programas que podem ser desenvolvidos em linguagens de alto nível como C e C++ [Kerningan and Ritchie, 1978, Stroustrup, 1986]. É nesta máquina é que, por exemplo, são feitas as aquisições de dados do sistema.

A arquitetura completa do time MIneiROSOT é mostrada através de um diagrama de blocos na Figura 4.6. Este sistema é montado conforme é mostrado na Figura 1.1, onde o campo de jogo possui dimensões de 150×130 cm e a altura da câmera é escolhida de forma que todo campo seja visível[2].

[2]Geralmente esta altura é de aproximadamente 2 m em relação ao campo

4.2 Modelagem

Nesta seção serão aplicados os algoritmos de estimação de parâmetros mostrados no Apêndice B às estruturas propostas na Seção 3.2.1 para determinação de um modelo dinâmico para os robôs do time MIneiROSOT. Para a utilização dos algoritmos, optou-se por obter inicialmente um conjunto de dados capaz de representar bem o robô em uma região de trabalho cuja média é a situação parado. Para tanto, foram adquiridos alguns conjuntos de dados independentes, dentre os quais alguns se constituíam por respostas a degraus e os demais por entradas binárias aleatórias com características muito similares às PRBS, todos executados em malha aberta.

A Figura 4.7 mostra a resposta do sistema a degraus nas entradas de controle. Para a obtenção destes dados o robô foi inicializado com uma orientação de aproximadamente 135 graus (diagonal do campo) e foram aplicados sinais de controle iguais a ambos os motores. Nesta figura, pode ser observado que o sistema possui uma ação integral, como era previsto pelo modelo (3.5). Por esta figura, é possível ainda fazer uma estimação do tempo morto do sistema que é de aproximadamente 215 ms.

A primeira análise mostrou que o sistema de medição (visão) fornece medidas de orientação na faixa de 0 a 2π radianos e por isso é necessário um pré-processamento para garantir consistência para esta variável durante a estimação. Em relação às variáveis x e y, optou-se por trabalhar diretamente com a informação em *pixels*, já que uma transformação para unidades métricas seria altamente dependente da altura e posicionamento da câmera e também de suas próprias características construtivas. O intervalo de amostragem do processo é limitado pelas características dos sistemas de visão, comunicação e controle e seu valor mínimo é de 27 ms. Pode-se notar, que deste tempo, 25 ms (40 quadros/s) são gastos com a visão e os outros

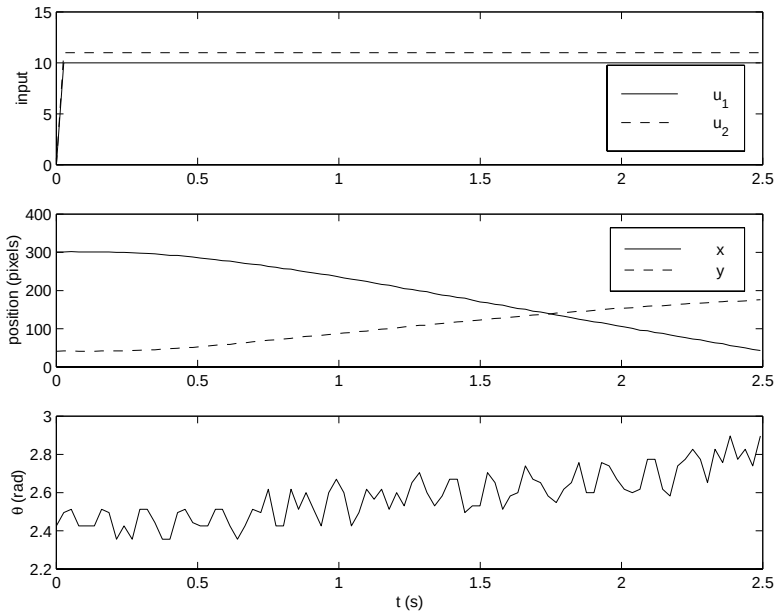

Figura 4.7: Reposta ao degrau do sistema.

2 ms com o restante do processo. Um procedimento simples de análise, sugerido em [Aguirre, 1995] e baseado na análise da auto-correlação entre as saídas, confirma que este valor é suficiente e adequado para a estimação de parâmetros do modelo dinâmico do robô. Na realidade, pela Figura 4.8, que mostra a função de auto-correlação (FAC) para a derivada da saída x a partir de entradas PRBS, vê-se que o período de amostragem poderia sofrer pequenas variações em torno do valor atual sem provocar prejuízo à estimação dos parâmetros, já que o procedimento de verificação sugere que existam entre 5 e 25 pontos antes do primeiro mínimo da FAC. A análise foi feita sobre a derivada das saídas, para tentar eliminar a ação integral do sistema e analisar, realmente, a dinâmica mais rápida do processo.

Na Figura 4.9 é mostrada a resposta do sistema a entradas binárias aleatórias. O período dos sinais de entrada bem como sua amplitude e média foram escolhidos de forma a favorecer o processo de aquisição e ao

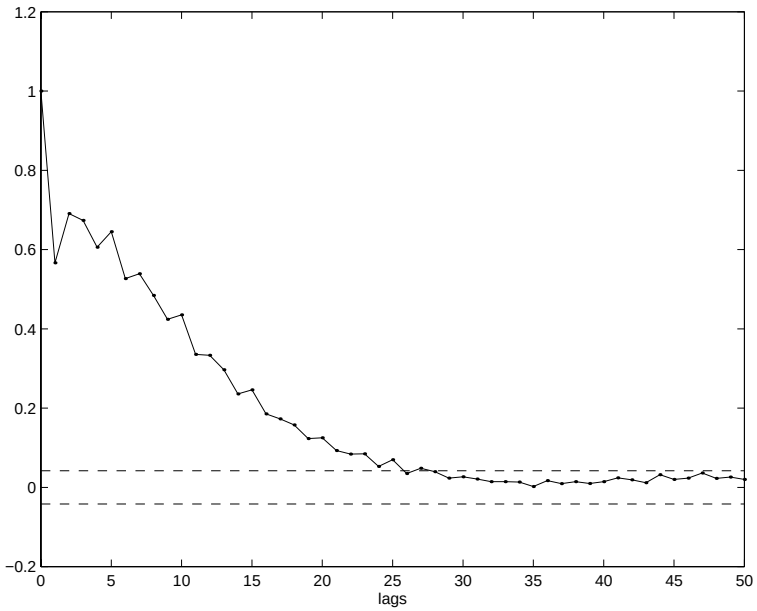

Figura 4.8: Função de auto-correlação da derivada da x.

mesmo tempo permitir a estimação de parâmetros como foi comentado na Seção 3.2.2. Desta forma, o período[3] dos sinais de entrada variou entre 10 e 15 intervalos de amostragem e a amplitude do sinal entre 8 e 15 níveis ou *bits* (o sinal é quantizado). A média escolhida para o sinal, subtraída antes da estimação, fazia com que o robô ficasse parado. Apesar de tomados todos os cuidados, durante os experimentos um pequeno número de breves impactos com as paredes do campo foi observado. Como o número de pontos no conjunto de dados para trajetórias normais do robô é muito maior que aqueles onde houve colisão, considerou-se que o efeito destes impactos pode ser desconsiderado durante a estimação.

[3]Aqui, entende-se por período o menor intervalo de tempo de variação do sinal.

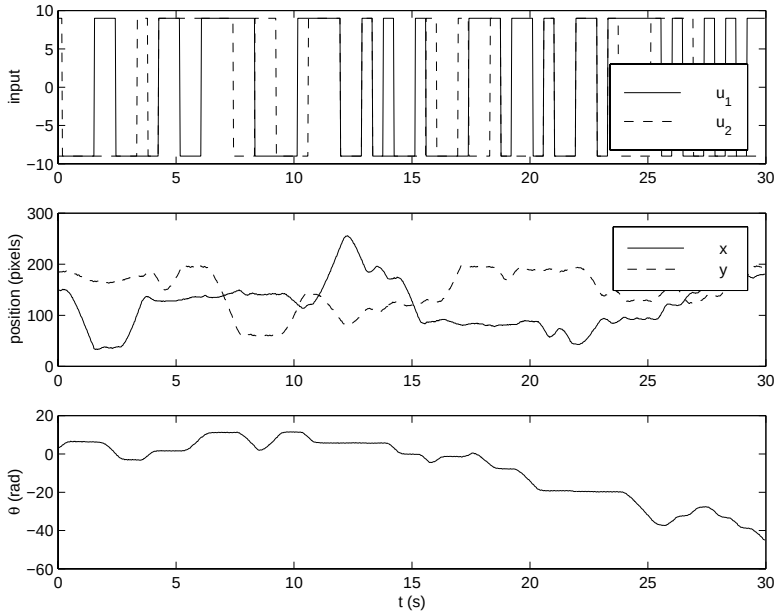

Figura 4.9: Parte dos dados usados na estimação de parâmetros

4.2.1 Estimação de Parâmetros

Para reduzir a polarização causada pelo ruído foram adicionados termos de média móvel do ruído a cada submodelo de (3.19). Em seguida, o método estendido de mínimos quadrados, mostrado no Apêndice B, foi aplicado de forma independente a cada uma das equações. Assim, a estrutura do modelo usada durante a estimação de parâmetros foi:

$$
\begin{aligned}
x_k &= a_1^X\, x_{k-1} + a_2^X\, x_{k-2} + \left[b_1^X\, u_{1\,k-d} + b_2^X\, u_{2\,k-d} + d_1^X\right]\cos(\theta_{k-1}) + \sum_{i=1}^{10} c_i^X e_x(k-i) \\
y_k &= a_1^Y\, y_{k-1} + a_2^Y\, y_{k-2} + \left[b_1^Y\, u_{1\,k-d} + b_2^Y\, u_{2\,k-d} + d_1^Y\right]\mathrm{sen}(\theta_{k-1}) + \sum_{i=1}^{10} c_i^Y\, e_y(k-i) \\
\theta_k &= a_1^\Theta\, \theta_{k-1} + a_2^\Theta\, \theta_{k-2} + b_1^\Theta\, u_{1\,k-d} + b_2^\Theta\, u_{2\,k-d} + d_1^\Theta + \sum_{i=1}^{10} c_i^\Theta e_\theta(k-i)\ . \qquad (4.1)
\end{aligned}
$$

Os parâmetros d_1 foram adicionados por que se mostraram úteis durante o processo de estimação. Estes parâmetros servem para "absorver" a média do ruído presente no sinal e assim como os 10 termos MA evitar polarização.

Por isso, eles serão ignorados após a estimação de parâmetros e todos as simulações mostradas neste trabalho são obtidas a partir de estruturas semelhantes a (3.19). O número de termos MA foi escolhido empiricamente de forma a obter o melhor modelo possível sem alterar o restante da estrutura. Para o tempo morto estimado e período de amostragem escolhido, vê-se que $d = 8$.

Após 10 iterações do algoritmo estendido de mínimos quadrados nos dados mostrados na Figura 4.9, o seguinte modelo foi obtido:

$$\begin{aligned} x_k &= 1,6824\, x_{k-1} - 0,6824\, x_{k-2} + [0,0441\, u_{1\,k-8} + 0,0403\, u_{2\,k-8}] \cos(\theta_{k-1}) \\ y_k &= 1,7150\, y_{k-1} - 0,7152\, y_{k-2} + [0,0336\, u_{1\,k-8} + 0,0318\, u_{2\,k-8}] \operatorname{sen}(\theta_{k-1}) \\ \theta_k &= 1,5640\, \theta_{k-1} - 0,5640\, \theta_{k-2} - 0,0063\, u_{1\,k-8} + 0,0065\, u_{2\,k-8} \ . \end{aligned} \tag{4.2}$$

Análises dos resíduos do modelo (4.2) revelaram que os erros de predição de um passo a frente são linearmente brancos. Em outras palavras, não há correlação estatisticamente significante nos resíduos, o que sugere que os parâmetros não estão polarizados. As FACs para os resíduos do modelo são mostradas na Figura 4.10.

Apesar do modelo ter sido obtido com a aproximadamente 2200 conjuntos de pontos de entrada e saída, aplicando-se o método recursivo de mínimos quadrados aos mesmos dados, vê-se que não são necessários todos estes pontos para obtenção dos mesmos parâmetros. A Figura 4.11 mostra a variância normalizada, calculada a cada passo do algoritmo recursivo, de um dos parâmetros do modelo de x (parâmetro que mais varia). Esta figura indica que depois de 800 pontos não existe praticamente mais nenhuma variação no parâmetro, indicando que este número é suficiente para a sua estimação. Para geração desta figura foi utilizado o método estendido de mínimos quadrados recursivo (EMQR) com fator de esquecimento igual a 1.

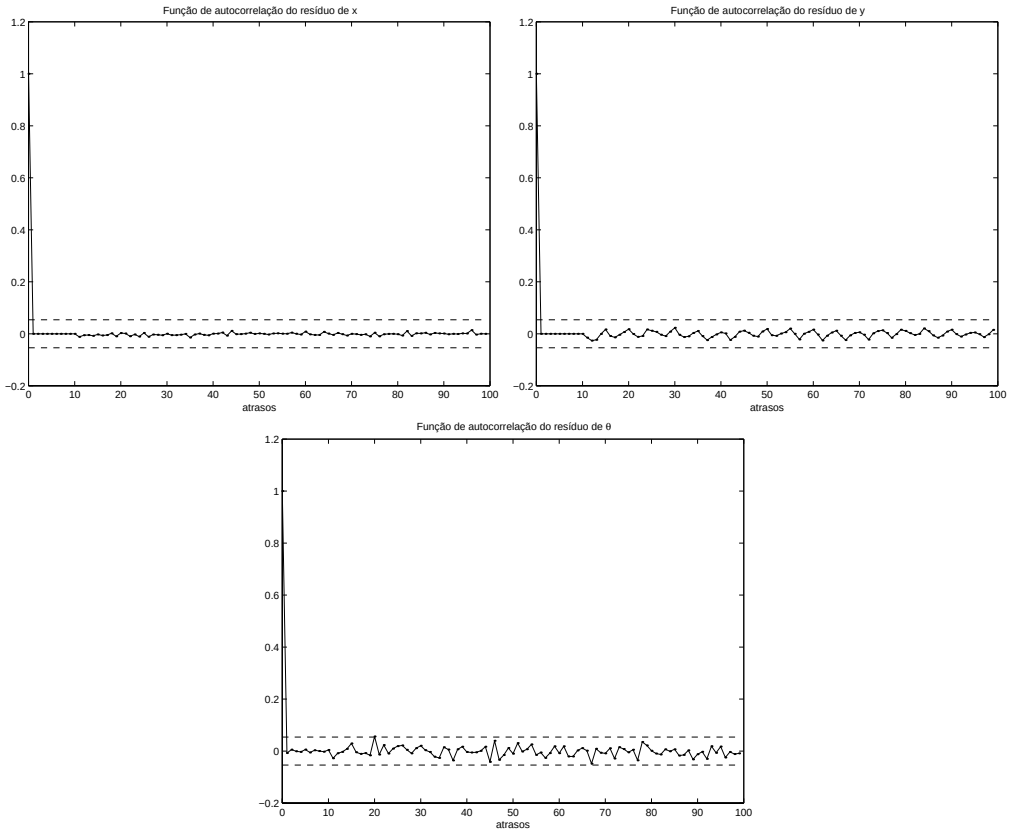

Figura 4.10: Funções de autocorrelação dos resíduos. As linhas tracejadas delimitam o intervalo de confiança de 95%.

Note que apesar do método recursivo ser em geral utilizado em tempo real, neste caso ele foi utilizado em dados coletados *a priori*. Na próxima seção é mostrado um resultado de predição do modelo onde o algoritmo recursivo foi aplicado em tempo real.

Como proposto pela Equação (3.21) o modelo anterior pode ser escrito como:

$$\begin{aligned} X &= \frac{0,0441\,U_1^X\ z^{-7}}{(z-1)(z-0,6824)} + \frac{0,0403\,U_2^X\ z^{-7}}{(z-1)(z-0,6824)} \\ Y &= \frac{0,0336\,U_1^Y\ z^{-7}}{(z-1)(z-0,7152)} + \frac{0,0318\,U_2^Y\ z^{-7}}{(z-1)(z-0,7152)} \\ \Theta &= \frac{-0,0063\,U_1\ z^{-7}}{(z-1)(z-0,5640)} + \frac{0,0065\,U_2\ z^{-7}}{(z-1)(z-0,5640)}\ , \end{aligned} \tag{4.3}$$

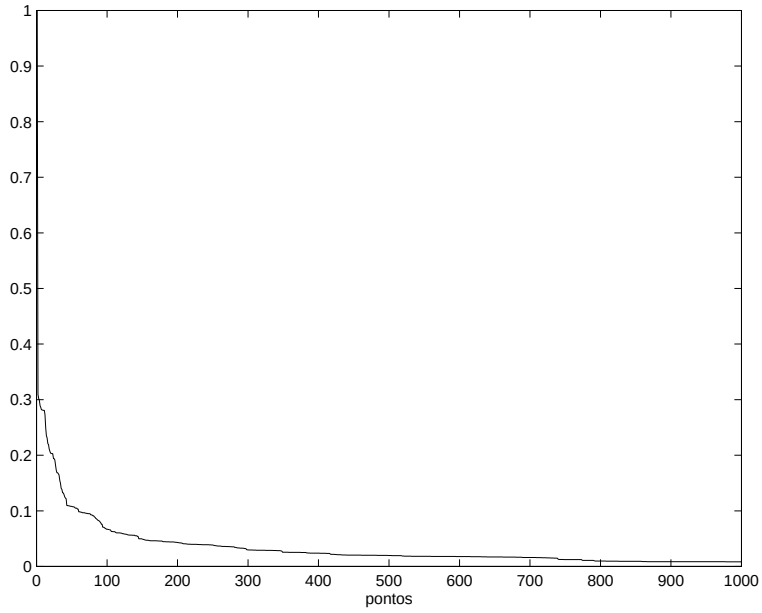

Figura 4.11: Variância de um dos parâmetros do modelo em função do número de pontos utilizados na estimação, obtida através da aplicação do método recursivo estendido de mínimos quadrados.

que é uma representação no domínio Z para o modelo do sistema. Note que há uma pequena diferença entre os ganhos das funções de transferência de cada saída, indicando que existe diferença entre os motores do robô. Devido às características de simetria do robô, era esperado que houvesse grande similaridade entre os modelos para X e Y. Entretanto, principalmente devido à diferença de resolução da câmera nas duas coordenadas estes modelos diferem um pouco, principalmente em relação aos ganhos, já que os pólos são muito parecidos[4]. O problema talvez fosse passível de ser resolvido se a diferença de resolução tivesse sido levado em consideração antes da estimação de parâmetros.

Como era esperado, o modelo encontrado indica que o sistema possui

[4]A diferença entre os ganhos de Y e X é de 25% e a diferença entre os pólos é de 5%.

integração. Isto pode ser verificado pela presença de um pólo em $z = 1$. Um fato interessante observado durante a estimação de parâmetros, é a influência do período de amostragem no modelo final. Se este período diminui, os pólos das funções de transferência tendem para a unidade. Se este período aumenta, ocorre o inverso, ou seja, o pólo tende a se tornar nulo. Não é possível mostrar o resultado com uma diminuição do período de amostragem devido às restrições do sistema mas um modelo para X onde $T = 54\,\text{ms}$, ou seja, o dobro do valor original é:

$$X = \frac{0,1895\,U_1^X\ z^{-7}}{(z-1)(z-0,1733)} + \frac{0,1658\,U_2^X\ z^{-7}}{(z-1)(z-0,1733)}$$

Uma possível explicação para o fato é a seguinte: O sistema (não o modelo) tem dois pólos sendo que um está em $z = 1$ e o outro é um pólo estável ($z \leq 1$). O primeiro é um integrador e, por definição, é o pólo dominante. À medida que o tempo de amostragem é aumentado passa-se a perder a informação dinâmica de alta freqüência (dinâmica rápida) e só é possível enxergar a dinâmica lenta, determinada pelo integrador. Por isso, para T grande o coeficiente de x_{k-1} tende a 1 e o outro tende a zero, ou seja, há uma convergência para um modelo com apenas um pólo (um integrador). É claro que se há um aumento muito grande de T, pode-se perder toda a informação.

Em oposição, diminuindo-se T passa-se a enxergar a dinâmica rápida. Nesse caso precisa-se de mais um pólo e o algoritmo estima valores estatisticamente significativos para x_{k-2}. Entretanto, o integrador não sumiu e precisa ser mantido no modelo. Isso é "garantido" pelo algoritmo fazendo com que $a_1 + a_2 \approx 1$. Com isso tem-se o seguinte resultado: Apesar do integrador ser o pólo dominante no modelo do robô, parece que ele sozinho não é suficiente para explicar a dinâmica observada nos dados. Um segundo pólo é necessário.

4.2.2 Validação do modelo

Apesar da análise de correlação de resíduos mostrar que os parâmetros do modelo não estão polarizados, o modelo deve ser validado com dados diferentes aqueles usados na estimação de parâmetros. Nesta seção, o modelo será validado tanto para a predição de alguns passos à frente como usando-se predição livre, ou seja sem nenhuma informação do sistema real durante a predição. Na Figura 4.12 é mostrada a validação do modelo para um conjunto de dados diferente daquele usado na estimação. Esta validação é feita de forma independente, ou seja, os valores de ângulo usados para validar os modelos de posição são aqueles medidos. Note que a boa predição do início do intervalo se deteriora com o aumento do tempo, devido ao acúmulo de erros. Este problema é característico dos sistemas com integração e difícil de ser eliminado. Entretanto, para verificar se a dinâmica do sistema foi realmente modelada, as tendências existentes na Figura 4.12 foram eliminadas e os resultados são mostrados na Figura 4.13. Note que neste caso os erros são bem menores que os da figura anterior indicando que o modelo representa com certa precisão a dinâmica do sistema.

A Figura 4.14 mostra a predição de 10 passos à frente para o modelo completo do robô descrevendo uma trajetória circular. Nesta figura, o campo e o robô foram desenhados em escala para tornar os erros facilmente visíveis. Na Figura 4.15 é mostrado o desvio padrão normalizado para a predição de k passos à frente em função de k para um novo conjunto de dados. Como esperado, quanto maior o horizonte de predição maior é o desvio padrão em relação à média do erro, que foi pequena e praticamente constante para todas as simulações.

Na Figura 4.16 a predição está sendo realizada em tempo real com o uso de um modelo que também está sendo atualizado em tempo de execução.

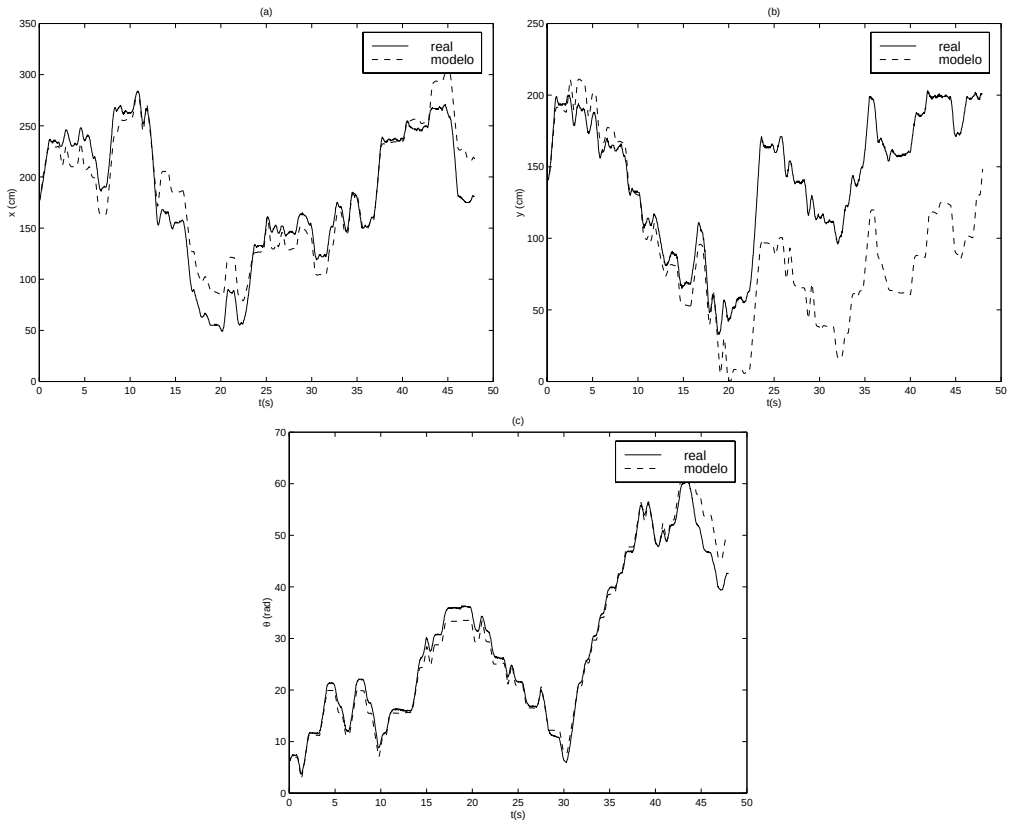

Figura 4.12: Validação independente do modelo (4.2) para um conjunto de dados diferente daquele usado na estimação. (a) - submodelo de x, (b) - submodelo de y, (c) - submodelo de θ

Para tanto, o algoritmo EMQR com fator de esquecimento igual a $0,99$ foi inicializado com os parâmetros do modelo (4.2) encontrados na identificação em batelada. Este resultado, somente é válido para o início do processo de predição, quando os parâmetros ainda não foram muito alterados pelo algoritmo recursivo de identificação. Isto ocorre, porque a trajetória circular não é suficientemente excitante para manter a estimação de parâmetros correta por muito tempo. Na próxima seção, será mostrado um método de detecção de possíveis falhas durante a estimação, quando o algoritmo deve ser desligado e os parâmetros mantidos constantes. Apesar disso, é importante ressaltar que a validação, mostrada nesta seção, provou que o modelo

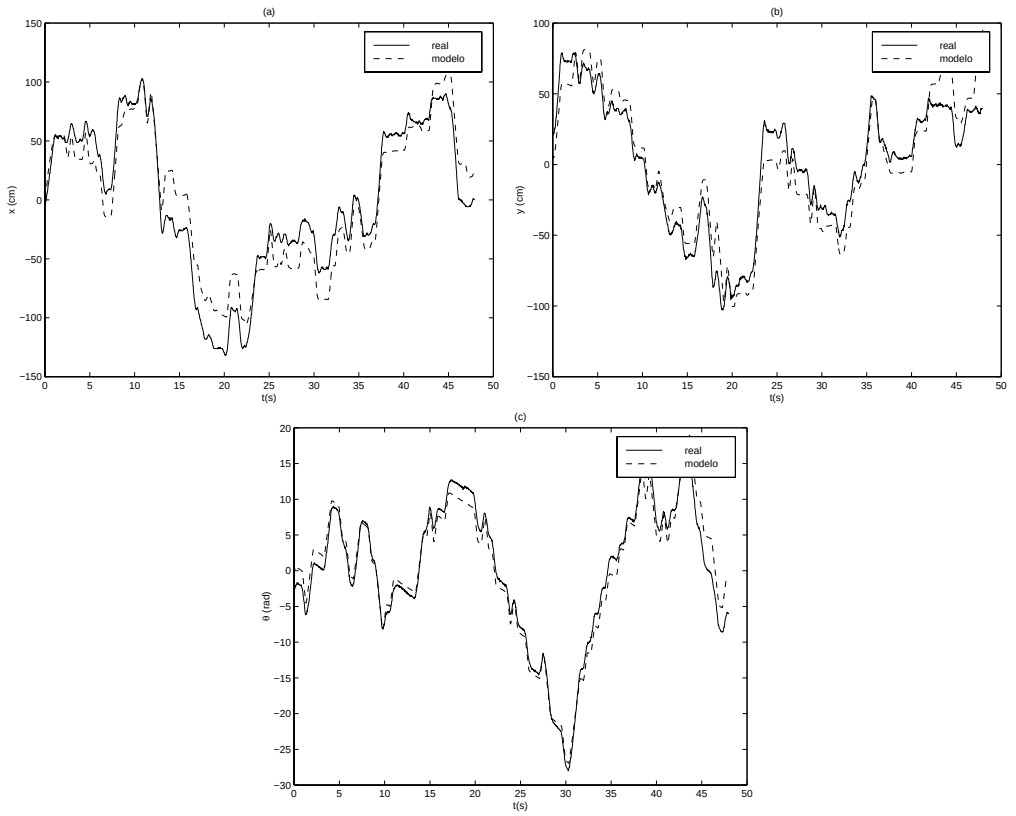

Figura 4.13: Eliminação da tendência nos dados de validação independente do modelo (4.2). (a) - submodelo de x, (b) - submodelo de y, (c) - submodelo de θ

representa com alguma precisão o processo real podendo assim, ser usado no projeto de controladores e como preditor do sistema.

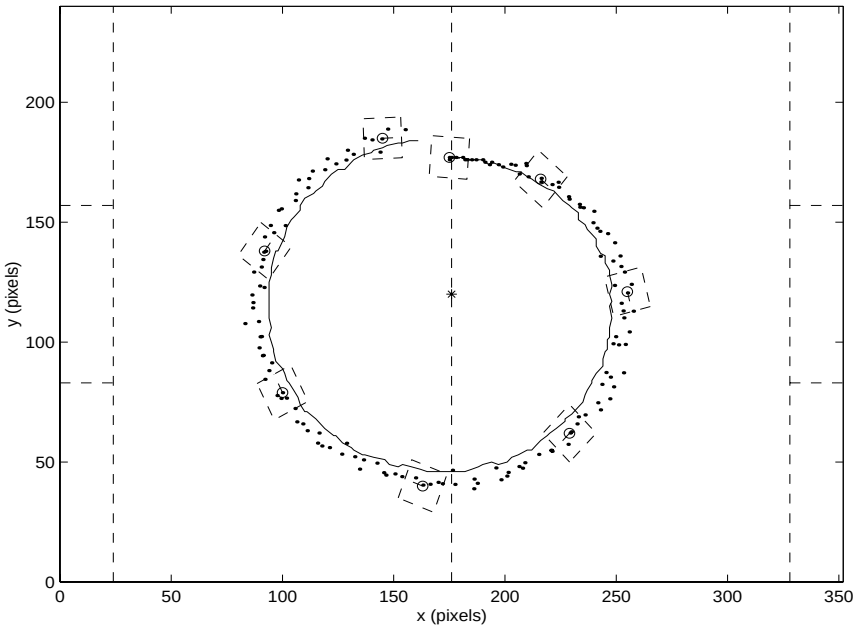

Figura 4.14: Predições obtidas usando o modelo completo (4.2). A linha contínua mostra a posição medida pelo sistema de visão e a linha pontilhada representa a predição 10 passos à frente.

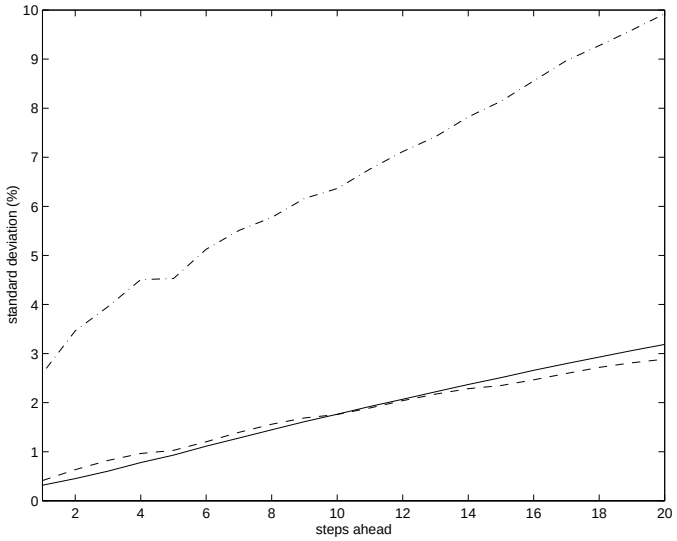

Figura 4.15: Validação do modelo: desvio padrão para predição de n passos à frente. A linha contínua se refere a x, a linha tracejada a y e a linha traço ponto a θ.

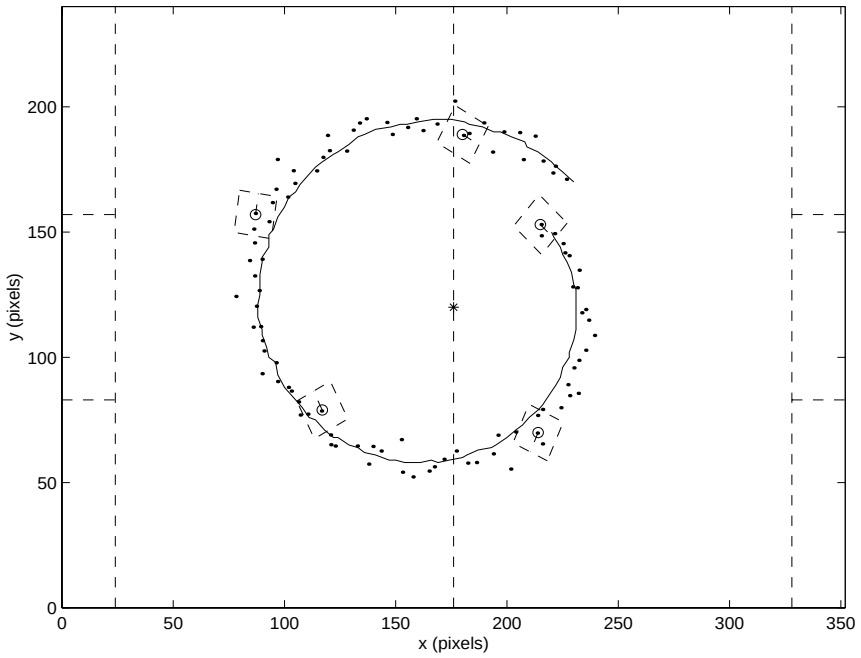

Figura 4.16: Predições obtidas usando o modelo completo com estimação em tempo real. A linha contínua mostra a posição medida pelo sistema de visão e a linha pontilhada representa a predição 10 passos à frente.

4.2.3 Estimação de parâmetros em tempo real

As Figuras 4.12 e 4.14 mostram que a predição usando o modelo identificado através do processo de estimação de parâmetros em batelada, possui erros aceitáveis. Para melhorar este comportamento e reduzir os erros de predição, a estimação em tempo real pode ser utilizada. Porém, como já foi mencionado, dependendo da excitação e do comportamento do robô, a identificação pode ser prejudicada e a predição comprometida. Para evitar que estes problemas ocorram, a atualização dos parâmetros pode ser interrompida no momento de ocorrência de falhas na estimação. Através da observação do algoritmo recursivo de mínimos quadrados mostrado no Apêndice B, vê-se que o "chaveamento" entre atualização ou não-atualização dos parâmetros

pode ser feito com a simples inibição de uma equação. A função de chaveamento pode ser escolhida como a verificação da matriz de covariância dos parâmetros estimada, $\mathbf{P}$. Assim, se $\mathbf{P}$ (ou um de seus elementos) é menor que um certo limite, a ser escolhido experimentalmente, é feita a atualização, caso contrário não. Uma explicação para este procedimento, é que as variâncias dos parâmetros, representadas pela diagonal de $\mathbf{P}$, são grandes, quando estes estão se alterando ou não são muito "confiáveis". Para o bom funcionamento do método, o modelo inicial deve ser bom, permitindo que $\mathbf{P}$ seja inicializada com valores pequenos. A determinação do limite não é muito fácil, já que se o valor for muito pequeno, mudanças nas características do processo não serão identificadas. A grande vantagem deste procedimento, é que quando ocorrem problemas, a matriz $\mathbf{P}$ cresce rapidamente (dependendo do fator de esquecimento) saindo do limite escolhido.

Existem duas falhas distintas que podem ocorrer durante a estimação: o sistema não está corretamente excitado; o robô é impedido de se movimentar, por colisão, falta de bateria ou defeito mecânico. A Figura 4.17 mostra o comportamento dos quatro primeiros valores da diagonal de $\mathbf{P}$ e de seus respectivos parâmetros, a_1^Θ, a_2^Θ, b_1^Θ e b_2^Θ, no decorrer do tempo. Os parâmetros foram inicializados com os valores encontrados através da identificação em batelada e a diagonal da matriz $\mathbf{P}$ com uma variância igual a $0,01$. O fator de esquecimento foi de $0,995$. Aproximadamente no tempo $t = 20\,\mathrm{s}$ o robô foi desligado, simulando uma falha mecânica ou uma colisão, sendo religado em $t = 53\,\mathrm{s}$. Devido a esta falha os ganhos do modelo, representados pelos dois últimos parâmetros, tendem a zero, como já era esperado. Na Figura 4.18 tem-se uma situação semelhante. Neste caso, porém, o robô foi desligado por *software*, ou seja, foi provocado um sinal não-excitante que mantinha o robô parado. Observe que em ambas as figuras a soma dos dois primeiros parâmetros é sempre igual a unidade devido a integração.

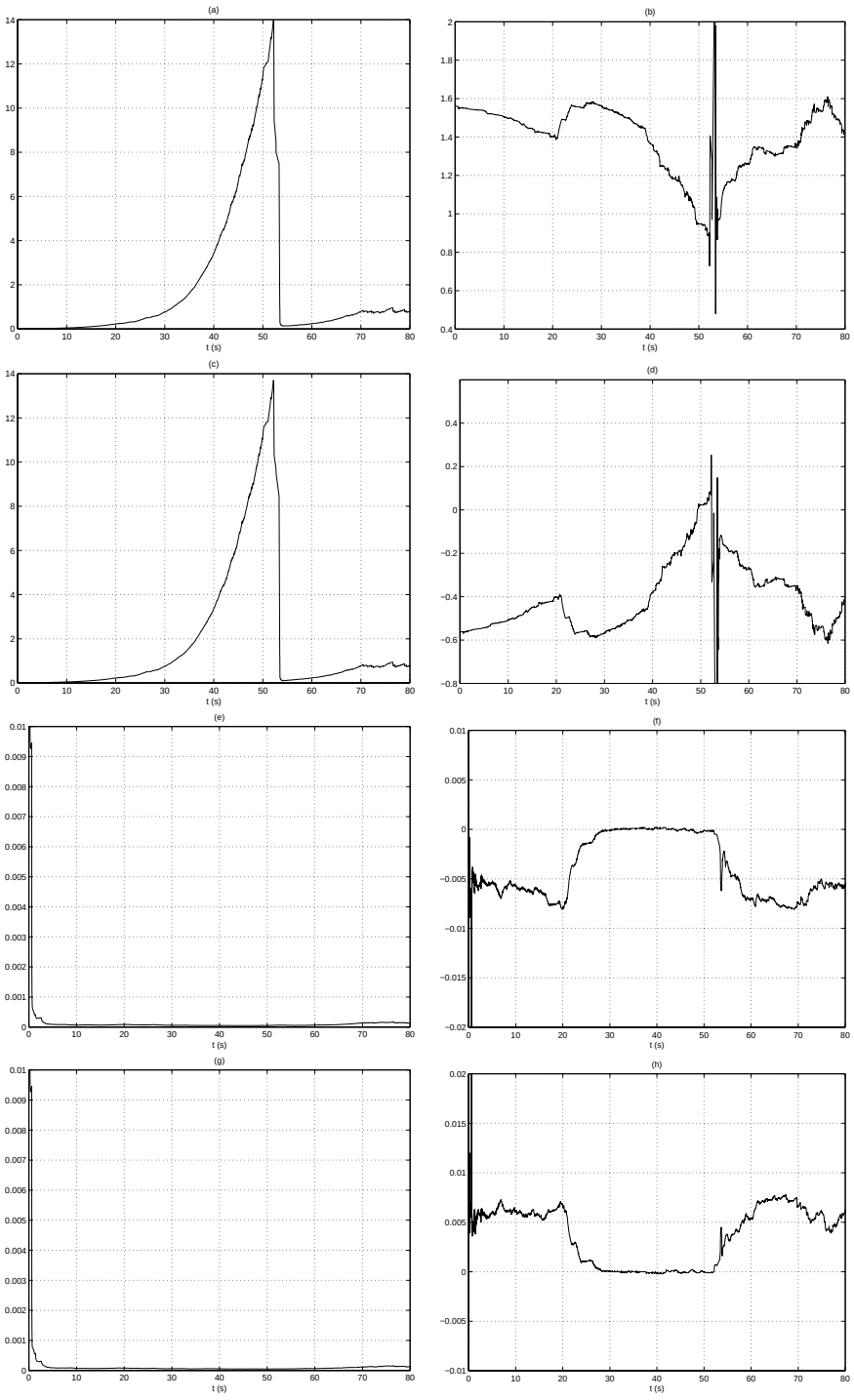

Figura 4.17: Comportamento dos parâmetros e de suas variâncias na presença de uma falha de movimentação do robô: (a) - σ^2(variância) de a_1^Θ, (b) - a_1^Θ, (c) - σ^2 de a_2^Θ, (d) - a_2^Θ, (e) - σ^2 de b_1^Θ, (f) - b_1^Θ, (g) - σ^2 de b_2^Θ, (h) - b_2^Θ,

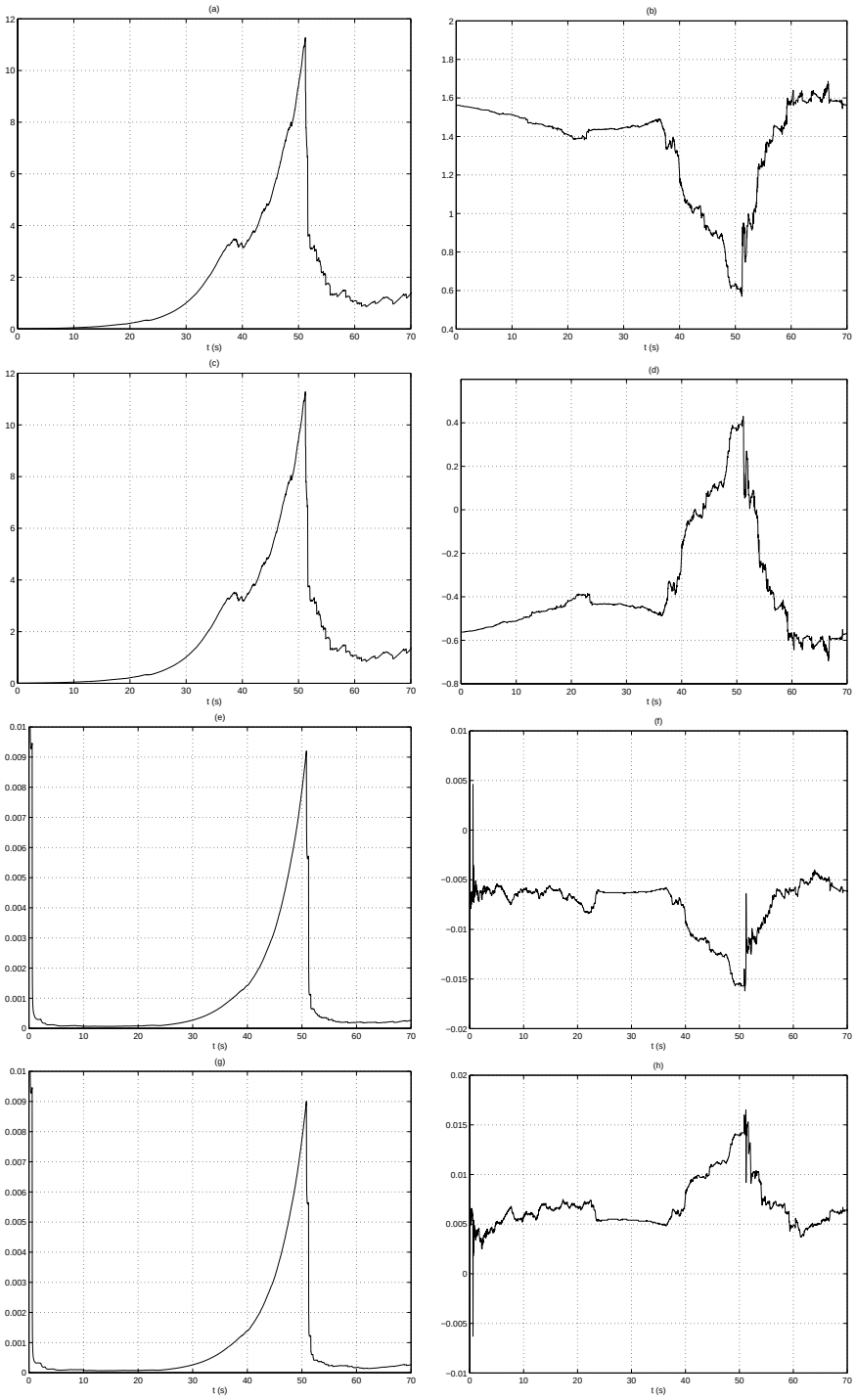

Figura 4.18: Comportamento dos parâmetros e de suas variâncias na presença de uma falha de excitação do robô: (a) - σ^2(variância) de a_1^Θ, (b) - a_1^Θ, (c) - σ^2 de a_2^Θ, (d) - a_2^Θ, (e) - σ^2 de b_1^Θ, (f) - b_1^Θ, (g) - σ^2 de b_2^Θ, (h) - b_2^Θ,

Pelas figuras anteriores, vê-se que, realmente, a simples observação da variância estimada dos parâmetros é suficiente para detectar as duas falhas possíveis. Como não é necessário isolar as falhas, ou seja, para as duas falhas a atualização dos parâmetros deve ser inibida, este procedimento é ainda mais simples. Assim, o algoritmo estendido de mínimos quadrados recursivos mostrado no Apêndice B, pode ser alterado e resumido como:

$flag \Leftarrow 1$

$\lambda \Leftarrow \lambda_1$

loop

$y_k \Leftarrow leitura$

$\boldsymbol{\psi}_k \Leftarrow atualiza(y_{k-1}, \xi_{k-1})$

$\mathbf{K}_k \Leftarrow \mathbf{P}_{k-1}\boldsymbol{\psi}_k \left[\boldsymbol{\psi}_k{}'\mathbf{P}_{k-1}\boldsymbol{\psi}_k + \lambda\right]^{-1}$

if $flag = 1$ **then**

$\hat{\boldsymbol{\theta}}_k \Leftarrow \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{K}_k \left[y_k - \boldsymbol{\psi}_k{}'\hat{\boldsymbol{\theta}}_{k-1}\right]$

end if

$\mathbf{P}_k \Leftarrow \frac{\mathbf{P}_{k-1} - \mathbf{K}_{k-1}\boldsymbol{\psi}_k{}'\mathbf{P}_{k-1}}{\lambda}$

if $(\mathbf{P}_k(1,1) > \delta_1$ and $\mathbf{P}_{k-1}(1,1) \leq \delta_1)$ **then**

$flag \Leftarrow 0$

$\lambda \Leftarrow \lambda_2$

else if $(\mathbf{P}_k(1,1) \leq \delta_2$ and $\mathbf{P}_{k-1}(1,1) > \delta_2)$ **then**

$flag \Leftarrow 1$

$\lambda \Leftarrow \lambda_1$

end if

$\xi_k \Leftarrow y_k - \psi_k'\hat{\boldsymbol{\theta}}_k$

end loop

Onde as constantes δ_1, δ_2, λ_1 e λ_2 são determinados de forma empírica. Este algoritmo, permite que os parâmetros iniciais sejam corrigidos pois é

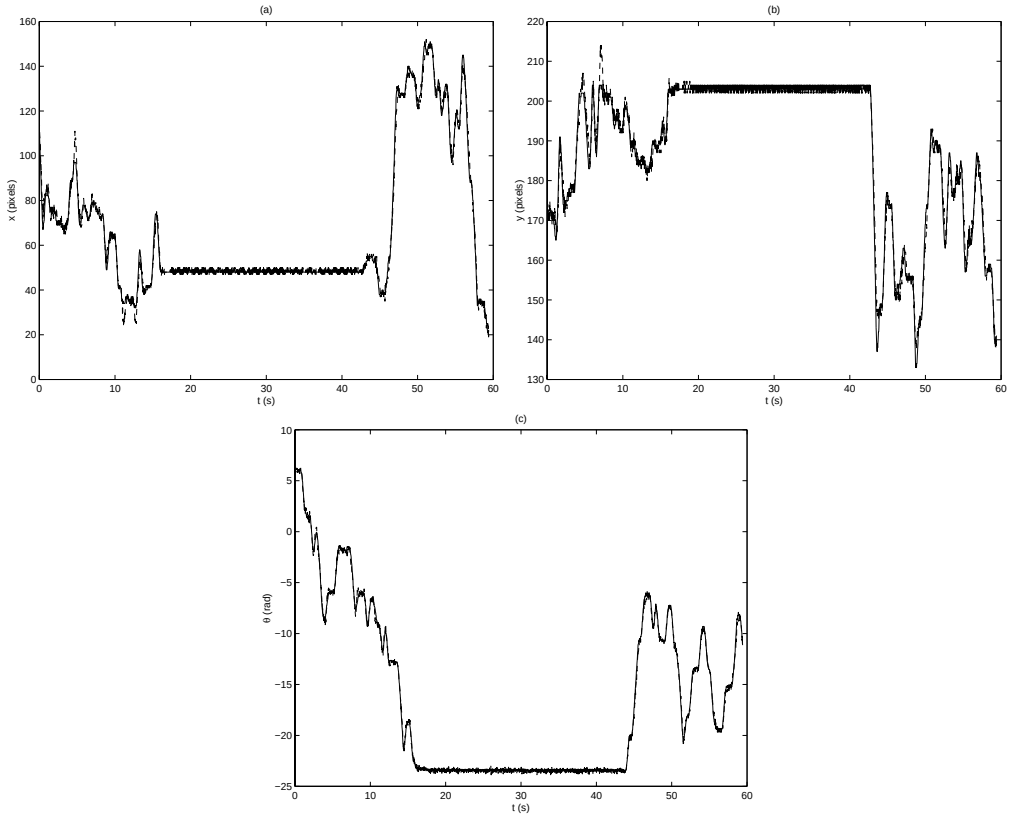

Figura 4.19: Validação do modelo com estimação em tempo real e predição de 8 passos à frente. (a) - submodelo de x, (b) - submodelo de y, (c) - submodelo de θ. A linha contínua mostra os valores medidos pelo sistema de visão e a linha pontilhada representa a predição 8 passos à frente.

garantido que a atualização somente é desligada quando há um aumento das variâncias dos parâmetros. A mudança no valor de λ foi feita para impedir problemas de estimação no momento que esta é reinicializada Para evitar que os dados "errados" obtidos durante a falha interfiram nos valores das novas medições, pode-se diminuir o fator de esquecimento durante a falha, voltando a aumentá-lo após a ocorrência da mesma. Desta forma, é como se houvesse um "esquecimento" do sistema, em relação ao intervalo de falha. Outra vantagem, da diminuição do fator de esquecimento, é que o reinicio da identificação se torna facilmente detectável. Os resultados da Figura 4.19

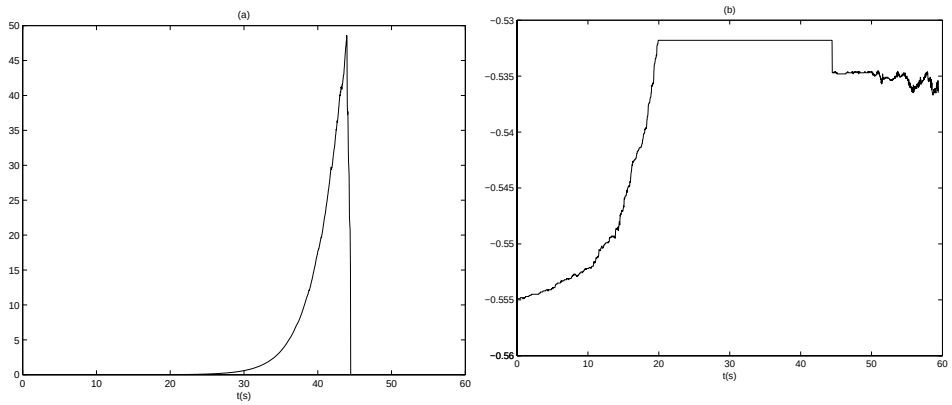

Figura 4.20: Comportamento do parâmetro a_2^{Θ} e de sua variância estimada com o uso do algoritmo de prevenção de falhas de estimação. (a) - variância, (b) - parâmetro.

mostram a predição de 8 passos à frente da posição, (x, y), e orientação, θ, do robô obtidos como o uso do algoritmo anterior. Para tanto, o modelo (4.2) foi atualizado de forma recursiva e o fator de esquecimento utilizado foi de $0,995$ durante a estimação e $0,99$ no momento da falha. A falha foi simulada com o envio de um sinal pouco excitante que mantinha o robô parado. Os limites δ_1 e δ_2 utilizados foram respectivamente $0,05$ e $0,5$. Note que o modelo sofreu pouca deterioração durante o período de desligamento, já que a predição se manteve com o mesmo erro durante todo o tempo. Isto pode ser verificado através da Figura 4.20, que mostra a variância do parâmetro a_2^{Θ} e seu próprio valor para os resultados da Figura 4.19. Note que o parâmetro permanece constante logo que a falha de excitação é detectada. Os desvios padrões para os erros de predição das variáveis x, y e θ nesta figura foram respectivamente, $\sigma_x = 2,49$ pixels, $\sigma_y = 2,14$ pixels, $\sigma_\theta = 0,28$ radianos.

Afim de mostrar a eficácia do algoritmo, a Figura 4.21 mostra a predição de 8 passos à frente sem o uso do mesmo na presença de uma falha de excitação. Note que há uma deterioração da predição de todas as variáveis durante e após a ocorrência da falha. A Figura 4.22, que está relacionada

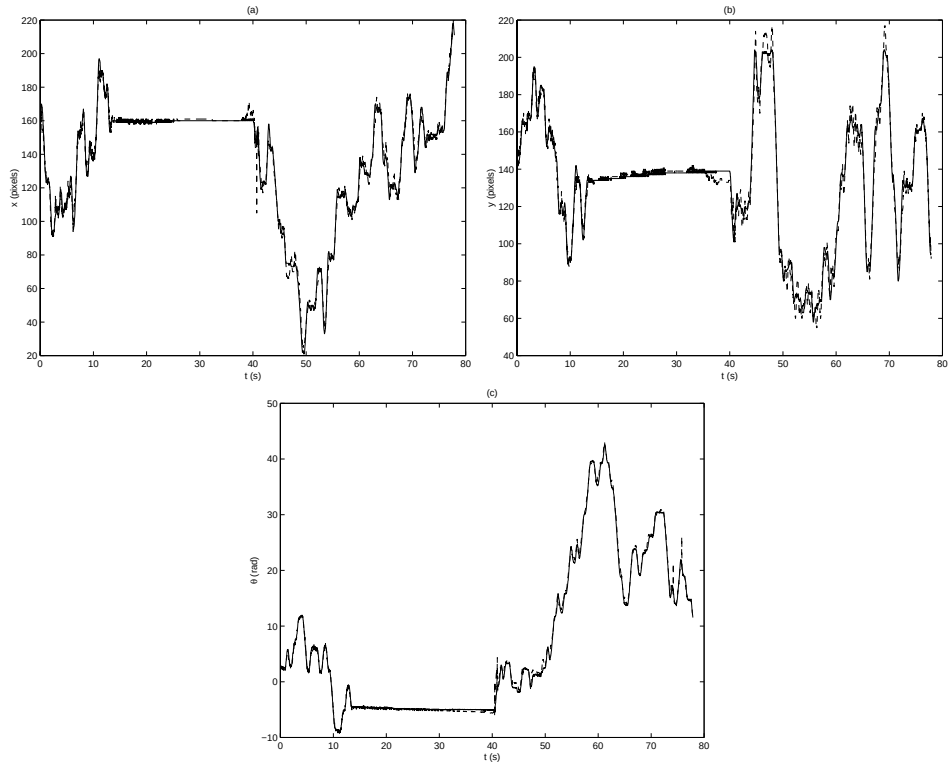

Figura 4.21: Predição em tempo real de 8 passos à frente sem o algoritmo de detecção de falhas. (a) - submodelo de x, (b) - submodelo de y, (c) - submodelo de θ. A linha contínua representa os valores reais e a linha pontilhada a predição.

com a anterior, mostra que mesmo após o fim da falha, o parâmetro leva algum tempo para se recuperar, devido ao alto valor do fator de esquecimento (0,995). A Figura 4.23 mostra uma comparação entre os erros de predição de 8 passos à frente para o modelo de θ com e sem a detecção de falhas.

A metodologia apresentada nesta seção é baseada em um algoritmo heurístico que, apesar de ter se mostrado eficaz para os casos testados, não tem sua eficiência garantida. Para atingir esta garantia, pode-se usar outros métodos baseados, por exemplo, em técnicas de inteligência computacional. Como este não é o objetivo principal do trabalho, o uso e verificação destes métodos é deixado como proposta de continuação.

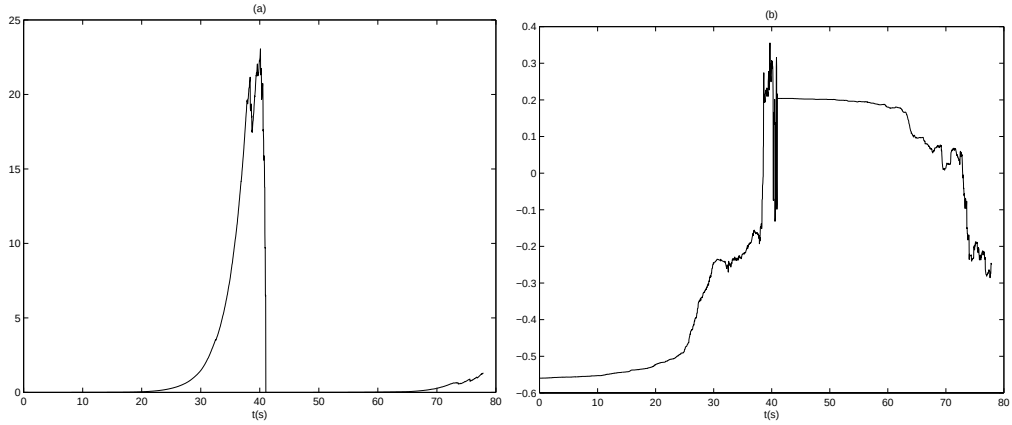

Figura 4.22: Comportamento do parâmetro a_2^{Θ} e de sua variância estimada sem o uso do algoritmo de prevenção de falhas de estimação. (a) - variância, (b) - parâmetro.

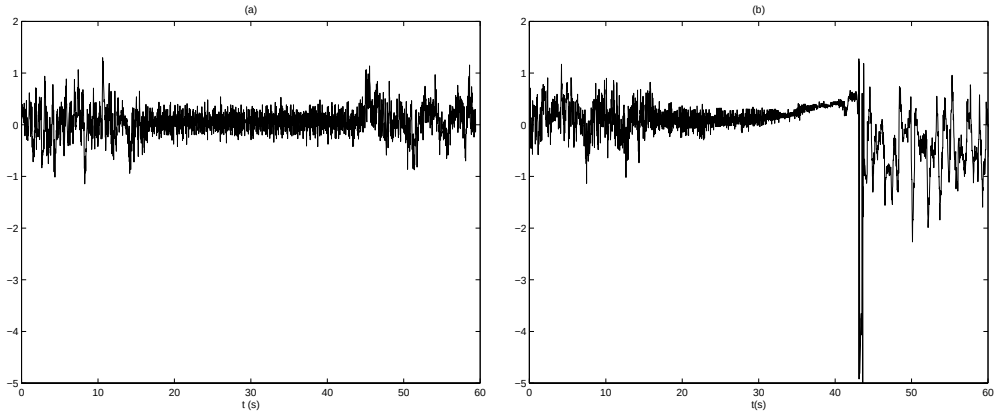

Figura 4.23: Erros de predição em tempo real de 8 passos à frente para o modelo de θ. (a) - com o algoritmo de detecção de falhas, (b) - sem o algoritmo.

Capítulo 5

Resultados Experimentais: Controlador

> Experiência não é o que acontece a você, mas sim o que você faz com o que acontece.
>
> *Aldoux Huxley (1894–1955)*

NESTE capítulo é mostrado o projeto do controlador para o robô do time MIneiROSOT. Para tanto, os modelos encontrados no capítulo anterior serão usados para análise e simulação dos sistemas de controle projetados. O Apêndice A descreve as características de um simulador, usado nos testes do controlador, que considera inclusive as perturbações existentes no robô real. Após a etapa de simulação os controladores são implementados e validados no robô real.

5.1 Controle de ângulo

Como proposto no Capítulo 3, o controlador de direção do robô pode ser um controlador linear do tipo PI, projetado com o auxílio do gráfico de lugar das raízes. O modelo utilizado no projeto, será aquele encontrado no capítulo anterior através de identificação em batelada. Desta forma, a função de transferência de ângulo, considerando as condições de desacoplamento

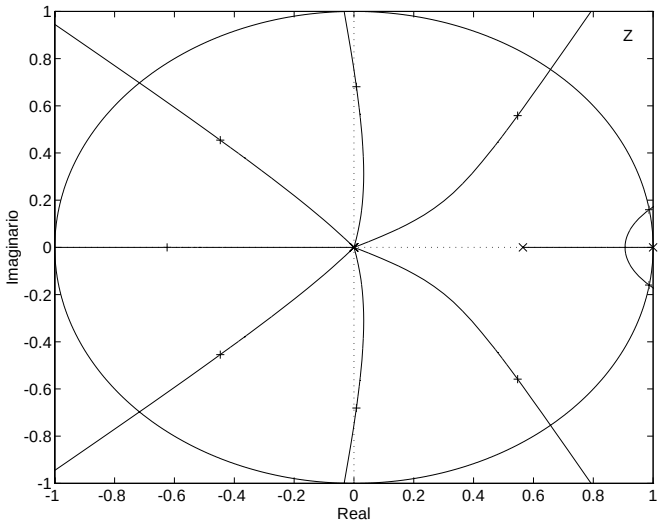

Figura 5.1: Diagrama de lugar das raízes do modelo de ângulo.

determinadas no Capítulo 3, é:

$$G_{\Theta}(z) = \frac{0,0130\ z^{-7}}{(z-1)(z-0,5640)} = \frac{K\ z^{-7}}{(z-1)(z-0,5640)}\ . \qquad (5.1)$$

O lugar das raízes para o modelo é mostrado na Figura 5.1. Por esta figura, vê-se que o ganho que leva o sistema ao seu limite de estabilidade (indicado por '+') é $0,0758$. A visualização deste ganho e posterior escolha no novo valor de K pode ser feita muito facilmente com o auxilio do software MATLAB [MATLAB, 1992] e das funções da *toolbox* de controle [Cavallo et al., 1996] que entre outras coisas, permite a obtenção do valor dos ganhos e seus respectivos pólos no gráfico de lugar das raízes.

Antes de iniciar o projeto do controlador devem-se escolher as características desejáveis para o sistema controlado. Assim, como deseja-se um sistema rápido e com pouco sobresinal e sabendo-se, a partir da Equação (3.32), que a constante de tempo mais rápida do sistema é 47 ms, um sobresinal

de no máximo 20% e um tempo de acomodação de 500 ms podem ser consideradas características satisfatórias para a malha fechada[1]. A princípio, satisfazer estas duas condições simultaneamente é uma tarefa muito difícil, principalmente devido ao grande atraso de 215 ms. Apesar disso, utilizando as Equações (3.30) e (3.31) vê-se que para estas escolhas $\zeta = 0,456$ e $\omega_n = 0,790$.

O segundo passo do projeto é incluir a função de transferência do controlador PI na malha de forma que a função de transferência de ramo direto se pareça com aquela mostrada na Equação (3.33). Assim, pelo método do dipolo onde $z_o = 0,99$, pode-se escrever a função de ramo direto como:

$$D(z)G_\Theta(z) = \frac{K_D\,(z - 0,99)}{(z-1)^2(z - 0,5640)} \quad . \tag{5.2}$$

O diagrama de lugar das raízes da função (5.2) é mostrada, juntamente com as curvas de ζ e ω_n desejados, na Figura 5.2. Note que este gráfico é muito similar ao da Figura 5.1 devido ao conceito de dipolo adotado. Como esperado, pode-se observar também, que não é possível, com esta estrutura de controlador, satisfazer as características desejáveis para o sistema. Desta forma, somente o sobresinal máximo pode ser satisfeito pois este é o único gráfico interceptado pelo lugar das raízes do sistema. O ganho K_D para este sobresinal é $0,02$. Assim, pelas Equações (3.27) e (3.28) e considerando que o processo já possui um ganho de $0,013$, as constantes do controlador são: $K_p = 1,53$ e $K_i = 0,57$. A resposta temporal ao degrau e à rampa para este sistema, simulado através da Equação (5.2), é mostrado na Figura 5.3. Por esta figura, nota-se que o sistema é muito lento para a resposta ao degrau,

[1]Na verdade, os critérios de projeto são também um desafio no projeto dos sistemas de controle. No caso de robôs móveis, a especificação destes critérios pode ser feita, por exemplo, pelos níveis superiores que definem as referências para os controladores.

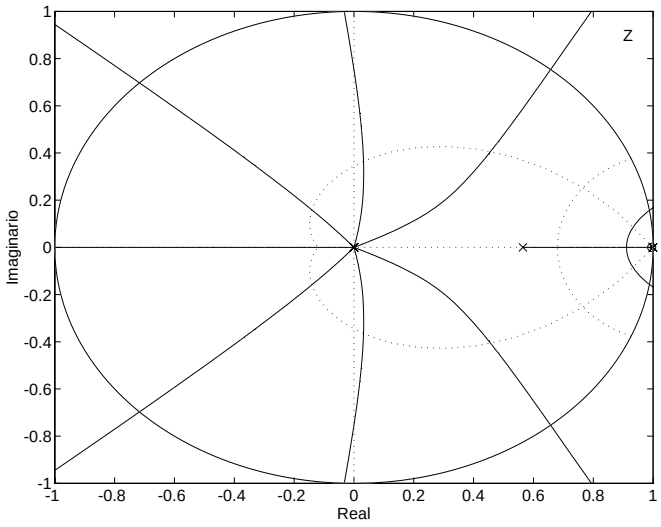

Figura 5.2: Diagrama de lugar das raízes do sistema controlado com PI.

demorando cerca de 6 s para se estabilizar, apesar de apresentar erro nulo em regime permanente para os dois tipos de entrada, como já era esperado. Para diminuir o tempo de subida, o ganho do controlador pode ser aumentado apesar da Figura 5.2 sugerir que este aumento provocaria um sobresinal acima do desejado. Para superar este problema, pode-se então, aumentar o zero do controlador, melhorando o cancelamento do pólo integrador adicionado, permitindo assim pequenos aumentos do ganho. Desta forma, as Figuras 5.4 e 5.5 mostram respectivamente, as respostas temporais do sistema para $z_o = 0,995$ e $K_D = 0,026$ e, no limite, $z_o = 1$ e $K_D = 0,031$. Estas figuras mostram que realmente é possível melhorar o tempo de subida, apesar de haver uma deterioração da resposta à rampa, com o aumento do ganho e diminuição do zero do controlador. Apesar disso, o tempo de resposta não foi satisfatório, devido principalmente ao atraso, que inviabiliza o aumento do ganho do controlador. O uso de um componente derivativo, poderia melhorar a resposta do sistema, mas como a presença de ruído pode dificultar o

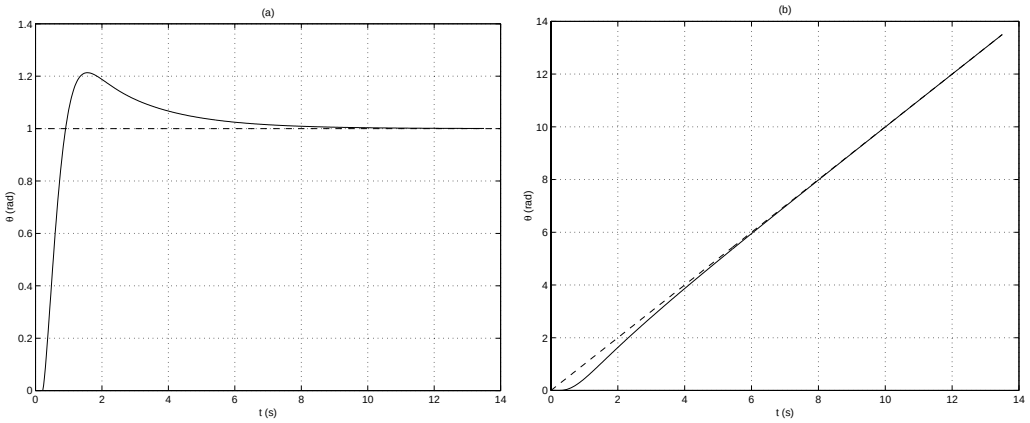

Figura 5.3: Resposta temporal do sistema de ângulo controlado com PI: $K_p = 1,53$ e $K_i = 0,57$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência.

uso de tal componente optou-se pela não adoção desta técnica. Para tentar melhorar o controlador serão utilizadas outras metodologias, como será visto nas próximas seções.

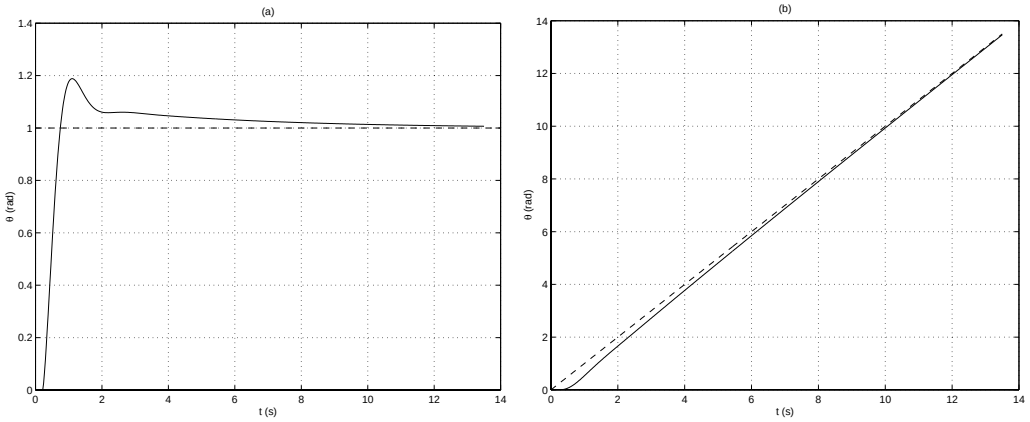

Figura 5.4: Resposta temporal do sistema de ângulo controlado com PI: $K_p = 1,995$ e $K_i = 0,37$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência.

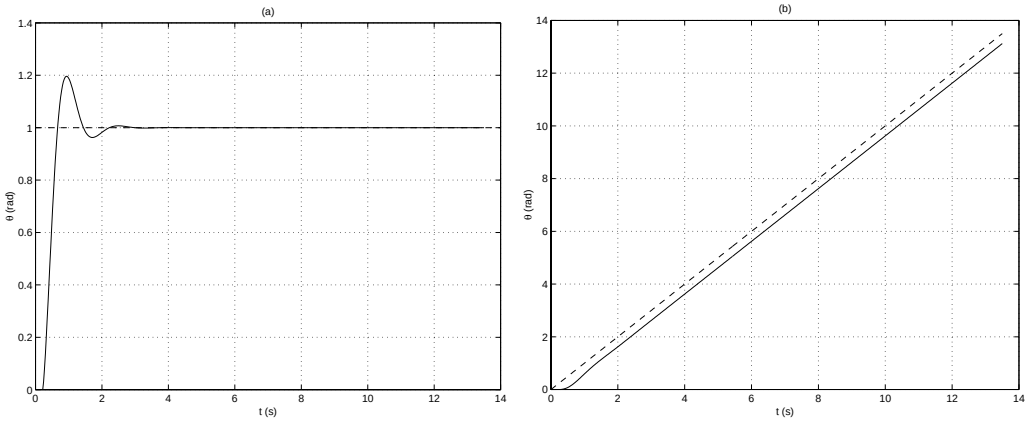

Figura 5.5: Resposta temporal do sistema de ângulo controlado com PI: $K_p = 2,385$ e $K_i = 0$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência.

Até agora, todos os resultados mostrados foram obtidos através da simulação da função de transferência mostrada na Equação (5.2). Não foram, portanto, considerados os ruídos e não-linearidades do sistema. A Figura 5.6 mostra a resposta do sistema obtida através do simulador mostrado no Apêndice A. Nesta figura usou-se $K_p = 1,53$ e $K_i = 0,57$ e uma constante de desacoplamento que multiplica a entrada u_1 igual a $0,97$. Observando a Figura 5.6(a) vê-se que esta difere um pouco de sua correspondente obtida através da função de transferência (5.2). A principal diferença é que aparecem degraus na resposta transitória do sistema. A explicação para este fato está relacionada com a quantização ou arredondamento das variáveis manipuladas, que ocorre antes de sua transmissão ao robô. Os degraus aparecem porque em certos momentos o arredondamento da saída do controlador permanece constante, ou seja, não varia mais que um intervalo de quantização que no caso é 1 bit. Na Figura 5.6(b) este fato não acontece, pois a referência está variando impedindo que a saída do controlador permaneça constante. A Figura 5.7 mostra que o problema é ainda pior se a ação integral for desligada. Neste caso, a saída permanece constante e com a tendência de nunca se aproximar da referência resultando num erro constante em regime permanente. A diferença do primeiro caso, é que a ação integral, com o passar do tempo, aumenta a saída do controlador fazendo com que seja "rompida a barreira" de 1 bit.

As Figuras 5.8 e 5.9 mostram a resposta do robô real para dois dos controladores de ângulo projetados. A estimação em tempo real da função de desacoplamento não mostrou bons resultados pois foi observada grande oscilação no seu valor. Por este motivo optou-se por manter este valor fixo em $0,97$, que foi o valor médio observado durante a identificação em tempo real. A princípio, o uso de um valor constante é razoável, já que, se há variação

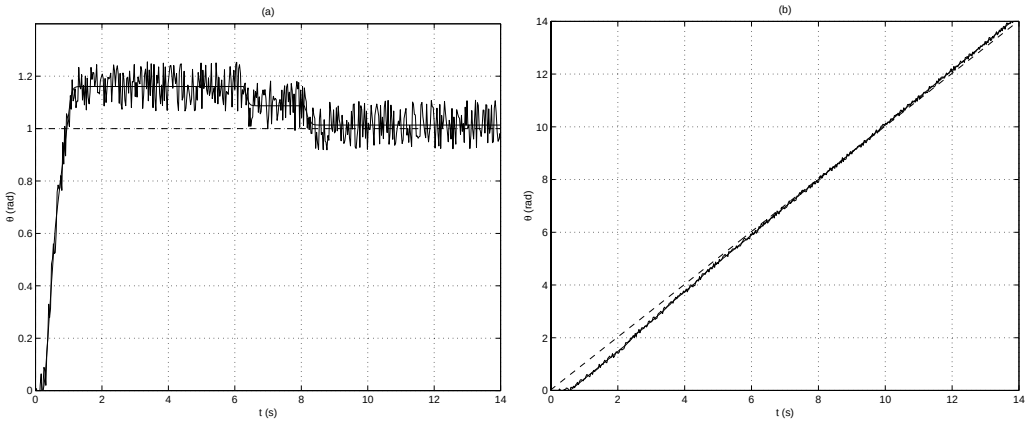

Figura 5.6: Resposta temporal do simulador controlado com PI: $K_p = 1,53$ e $K_i = 0,57$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e as linhas contínuas os valores reais do ângulo com e sem ruído de medição.

na relação entre os ganhos dos motores, ela é feita de forma lenta.

A comparação entre as Figuras 5.8 e 5.9 e as anteriores mostra que o comportamento do simulador é bastante similar ao do sistema real, inclusive em relação ao erros provocados pelo arredondamento, que foram bem aproximados na simulação. A principal diferença é que o sistema real parece ser um pouco mais lento que o simulado. Esta diferença deve estar no fato de que o modelo e os dados do controle foram obtidos em momentos diferentes onde as condições, como tensão alimentação do robô, nível de sujeira do campo e outras variáveis se alteraram. De qualquer forma, os sinais são muito parecidos e a diferença de velocidade entre a simulação e a realidade dá um indício de que os ganhos calculados com o simulador podem ser ligeiramente aumentados antes de serem aplicados ao sistema real.

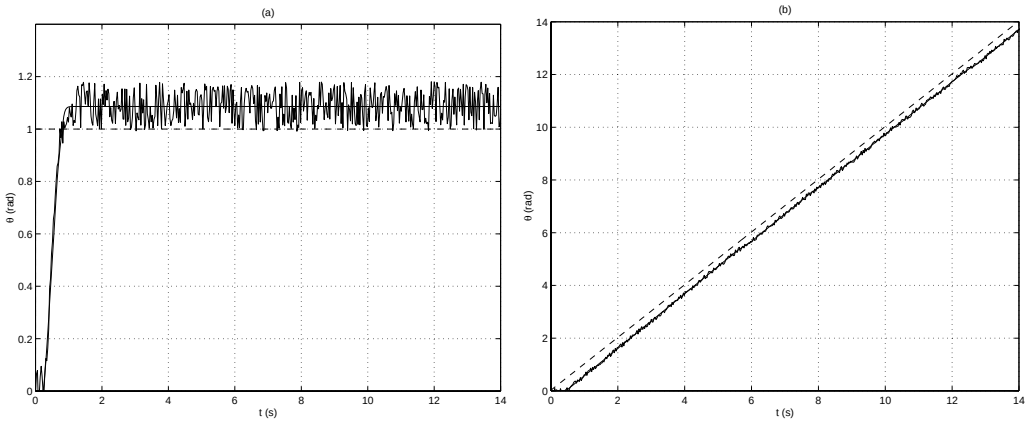

Figura 5.7: Resposta temporal do simulador controlado com PI: $K_p = 2,385$ e $K_i = 0$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e as linhas contínuas os valores reais do ângulo com e sem ruído de medição.

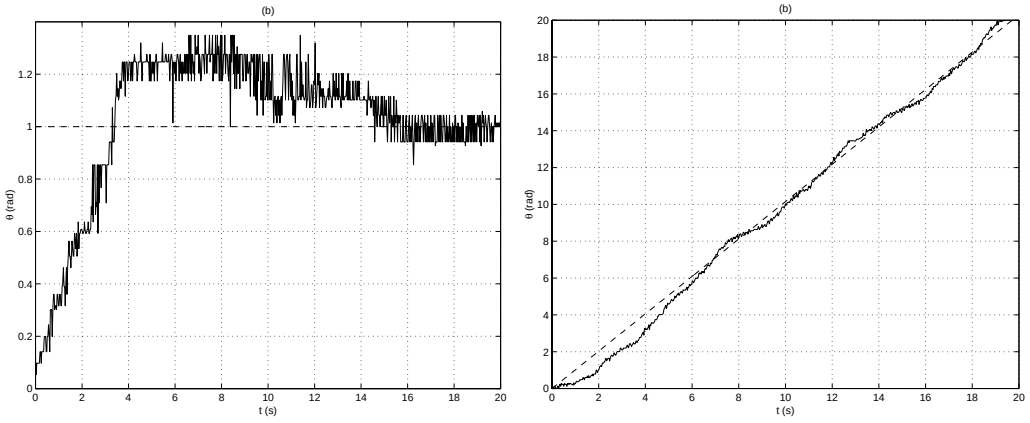

Figura 5.8: Resposta temporal do sistema real controlado com PI: $K_p = 1,53$ e $K_i = 0,57$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

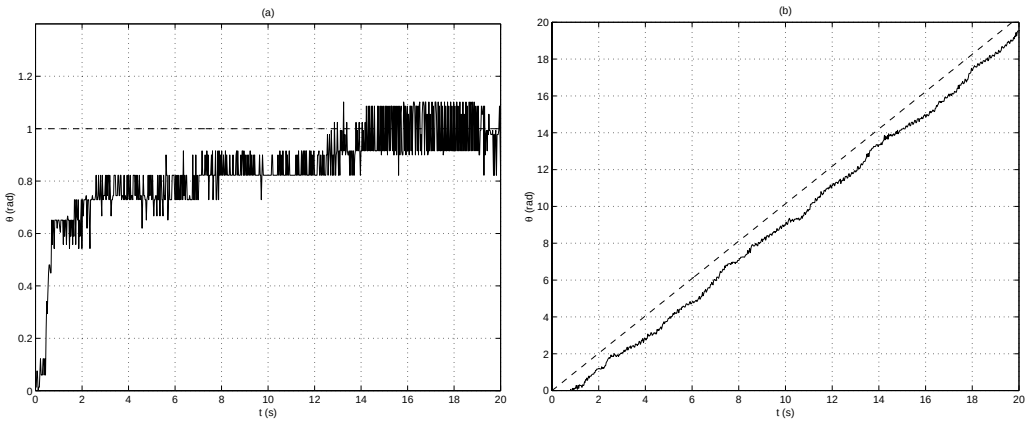

Figura 5.9: Resposta temporal do sistema real controlado com PI: $K_p = 2,385$ e $K_i = 0$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

Observando os gráficos de resposta temporal do sistema de ângulo controlado vê-se que existem dois problemas principais: o arredondamento dos sinais de controle e a lentidão do sistema. O problema do arredondamento do sinal de controle que pode ser notado nos resultados simulados (Figuras 5.6 e 5.7) e nos reais (Figuras 5.8 e 5.9), pode ser corrigido com a alteração do *hardware* do sistema através da mudança do sistema que faz a interface entre o computador e o transmissor. Como esta alteração não é muito simples, o uso de um estrutura de controle onde o sinal de controle é diretamente proporcional ao erro, também pode ajudar, apesar do problema não poder ser totalmente corrigido. Para o controlador atual, onde o sinal de controle é proporcional à diferença temporal do erro, um aumento do ganho do controlador pode permitir que pequenas variações do erro consigam excitar o sistema. Esta última alternativa, além do arredondamento, seria útil também na correção do segundo problema, ou seja, aumentaria a velocidade de resposta do robô. Entretanto, como pôde se visto na Figura 5.2, o aumento de ganho é limitado, principalmente devido ao tempo morto, que diminui

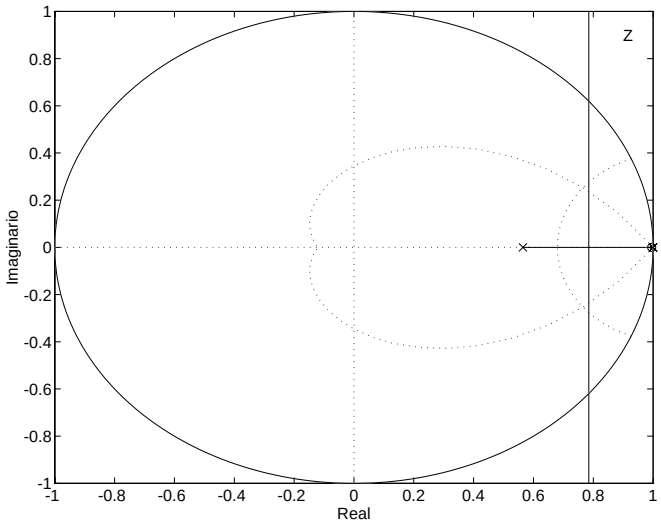

Figura 5.10: Lugar das raízes do sistema sem atraso controlado com PI.

a margem de ganho do sistema. Então, para permitir ganhos maiores, uma compensação do tempo morto faz-se necessária, como será mostrado a seguir.

5.1.1 Compensação de tempo morto

Como foi mencionado, uma compensação do atraso e um posterior ajuste de ganhos poderia melhorar o desempenho do sistema que, por enquanto, está insatisfatório. Para provar esta afirmação, considera-se o gráfico de lugar das raízes, mostrado na Figura 5.10, onde o atraso do sistema controlado foi removido. Por este gráfico, supõe-se que a margem de ganho foi aumentada em relação à Figura 5.1, permitindo que os ganhos também sejam maiores. Isto é comprovado quando verifica-se que o ganho no limite de estabilidade passou de $0,07$ para $0,44$. Observa-se também, que está mais fácil obter os parâmetros de projeto desejáveis, já que o lugar das raízes praticamente intercepta os gráficos de ζ e ω_n no mesmo ponto. Utilizando este gráfico, onde $z_0 = 0,99$, um valor de K_D que satisfaz os critérios do projeto é $0,115$.

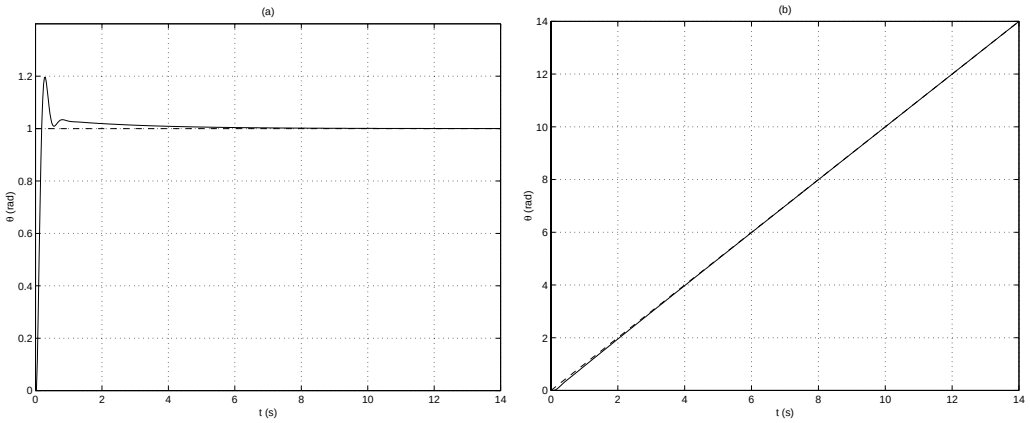

Figura 5.11: Resposta temporal do sistema de ângulo compensado e controlado com PI: $K_p = 8,802$ e $K_i = 3,276$.(a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência.

Com este valor, tem-se que $K_p = 8,802$ e $K_i = 3,276$. A Figura 5.11 mostra a resposta temporal da função de transferência sem atraso e controlada, para estes ganhos. Note que o tempo de resposta diminuiu significativamente em comparação com o sistema original com atraso e praticamente foram atingidos os requisitos de projeto.

A Figura 5.12 mostra a resposta temporal do simulador sem o atraso. Observe que, em oposição à situação original, além de mais rápido, o sistema não apresenta os "degraus" causados pelo arredondamento ou quantização do sinal de controle. Porém, pode ser notada uma oscilação adicional causada pela maior amplificação do ruído de $(e_k - e_{k-1})$. Fisicamente, esta oscilação aparece como um pequena variação do robô em torno da referência de ângulo. Como este ruído está no nível de grandeza do ruído original do sistema de medição, o problema pode ser desconsiderado, já que os benefícios obtidos com a compensação do atraso e aumento do ganho foram muito maiores. Apesar disso, uma conseqüência ruim da oscilação poderia ser um aumento do desgaste dos atuadores (servos) causada pela variação rápida dos sinais de

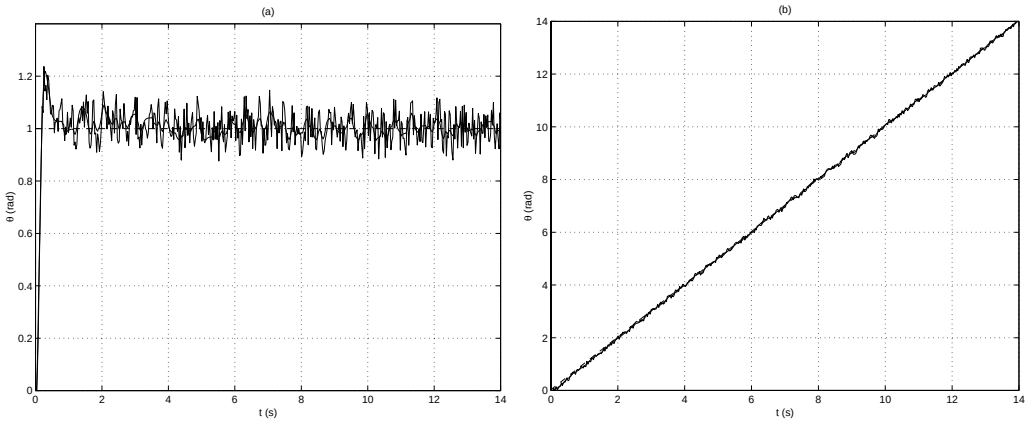

Figura 5.12: Resposta temporal do simulador sem o atraso controlado com PI: $K_p = 8,802$ e $K_i = 3,276$.(a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

controle. Entretanto, como na prática os robôs não "trabalham" por muito tempo, o desgaste é minimizado.

Até aqui, todos os resultados foram mostrados através de simulações. Nestes casos, a eliminação do tempo morto é trivial, já que se tem o controle sobre todos os códigos de programa, inclusive a equação que simula o robô. O problema é, então, eliminar o atraso na prática. Como foi proposto no final do Capítulo 3, um preditor baseado no modelo identificado anteriormente pode ser obtido. O modelo pode, então, ser utilizado para predizer 7 intervalos de amostragem à frente, e a saída deste sistema pode ser utilizada como sinal de realimentação para o controlador, conforme é mostrado na Figura 3.13. A Figura 5.13 mostra a resposta do sistema real com esta configuração. Note que o sistema tornou-se mais rápido e como no caso simulado, houve um aumento da oscilação envolvida. Além disso, o sistema parece não se estabilizar completamente na referência apesar de estar sempre em torno deste valor. Na resposta à rampa, apesar de ser esperado um erro nulo, isto não ocorreu. Estes problemas foram causados pelos erros do estimador que, apesar da pre-

cisão mostrada na Seção 4.2.3, não conseguiu estimar o ângulo corretamente. Apesar disso, os resultados obtidos com a compensação do tempo morto, se mostraram melhores que os anteriores pois os níveis de erro envolvidos são pequenos e, a princípio, incapazes de comprometer o objetivo do controlador de ângulo. Para exemplificar estas observações, na Figura 5.14 é mostrada a resposta a um degrau igual a π para o mesmo sistema. Nesta figura, percebe-se que o erro é praticamente insignificante em relação à amplitude do degrau.

Em todos os resultados práticos, tentou-se utilizar a identificação em tempo real para atualizar o modelo de ângulo e assim executar a predição de forma mais precisa. Entretanto, observou-se que desde o início os sinais de controle não eram suficientemente excitantes para que a estimação de parâmetros fosse correta. Com isso, o algoritmo proposto na Seção 4.2.3 sempre atuava no início do processo desligando a estimação. A princípio isto seria um bom resultado, já que foi mostrado que o modelo inicial, encontrado a partir da identificação em batelada, representa bem o sistema. O problema é que o tempo que o algoritmo demora para identificar a falha é suficiente para distorcer um pouco os parâmetros do modelo. Uma possível solução para este problema seria portanto, melhorar o tempo de resposta do sistema de detecção de falhas. Como este não é o objetivo do trabalho, antes de todos os testes práticos foi realizado uma identificação em tempo real com sinais do tipo PRBS e os parâmetros encontrados foram mantidos fixos durante a predição. É importante ressaltar que os parâmetros praticamente não se alteraram de um teste para outro, indicando que o robô não sofre variações significativas.

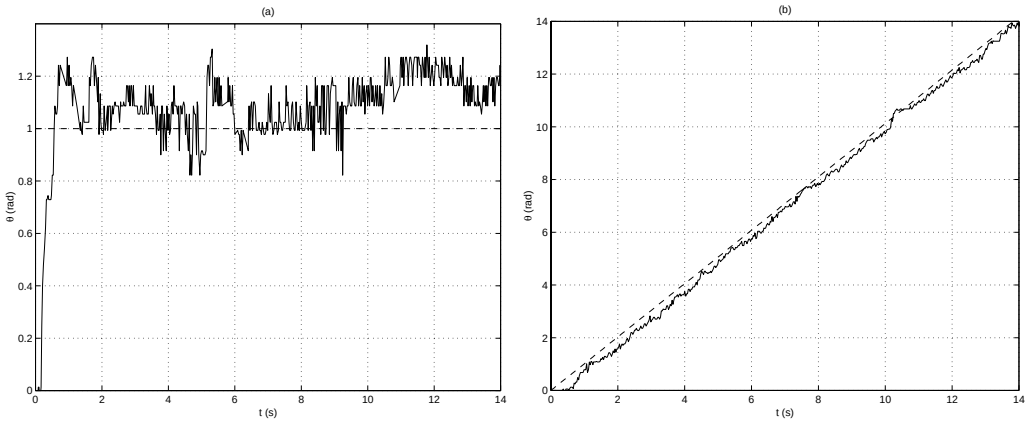

Figura 5.13: Resposta temporal do sistema real com preditor e controlado com PI: $K_p = 8,802$ e $K_i = 3,276$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

5.1.2 Controlador preditivo

Um preditor foi usado no item anterior para compensar o tempo morto e mostrou bons resultados. Se este mesmo preditor, for utilizado para predizer um passo além, anulando também o atraso mínimo de 1 período de amostragem, a resposta do sistema pode melhorar ainda mais. Este processo eqüivale a adicionar um zero em $z = 0$, sem a necessidade de adicionar um pólo. Assim, o lugar das raízes do novo sistema e com $z_0 = 0,99$ é mostrado na Figura 5.15. A Figura 5.16 mostra a resposta do simulador com $K_D = 0,25$ ($K_p = 19,135$ e $K_i = 7,123$) para esta situação. Da mesma forma, a resposta do sistema real é mostrada na Figura 5.17. Vê-se que neste caso, há uma oscilação ainda maior que no caso anterior, provocada pelo aumento do ganho e pelos erros de predição que tendem a ser maiores. Além disso, não houve uma melhoria significativa no tempo de resposta do robô, não justificando, aparentemente, o uso da predição de mais um intervalo de amostragem à frente. Para contrariar um pouco esta observação,

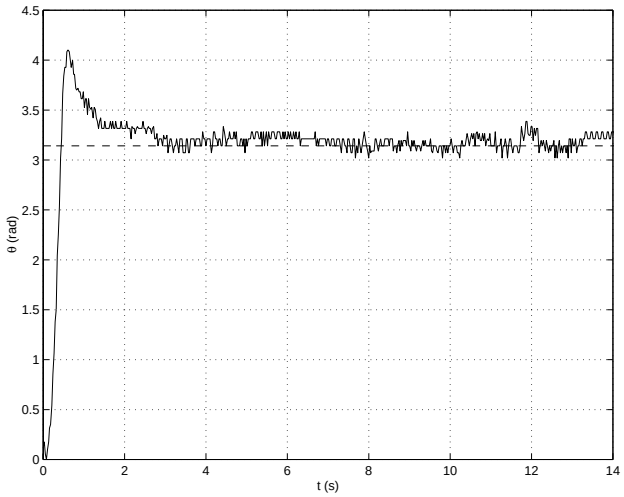

Figura 5.14: Resposta temporal do sistema real com preditor e controlado com PI: $K_p = 8,802$ e $K_i = 3,276$. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

a Figura 5.18 mostra a resposta ao um degrau igual a π. Percebe-se neste caso, uma pequena melhoria em relação à figura equivalente (Figura 5.14) mostrada anteriormente.

Nesta seção, foram mostrados diversas alternativas para o controle de orientação do robô baseadas no uso de controladores lineares do tipo PI. Até agora, entretanto, estes controladores foram utilizados apenas para orientar o robô parado, ou seja, o projeto dos controladores de ângulo é apenas parte da tarefa de se projetar um controlador que leve o robô de uma posição à outra ou que o faça seguir trajetórias. Na próxima seção, os controladores projetados serão utilizados juntamente com um controlador de posição para fazer com que o robô execute estas tarefas, como atingir um alvo ou seguir trajetórias.

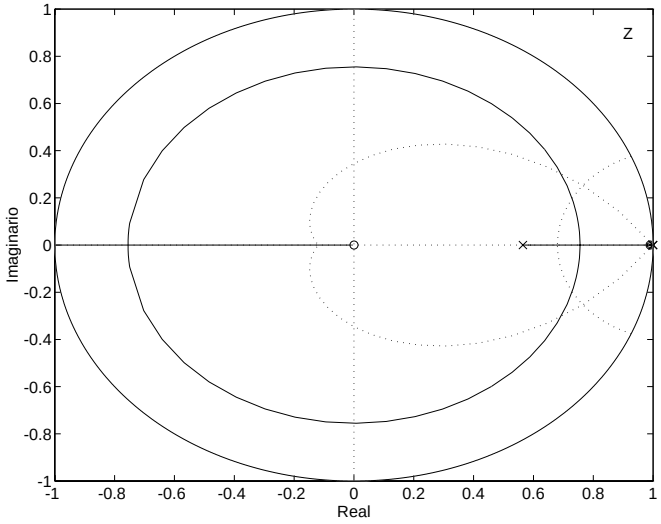

Figura 5.15: Diagrama de lugar das raízes do sistema com preditor controlado com PI.

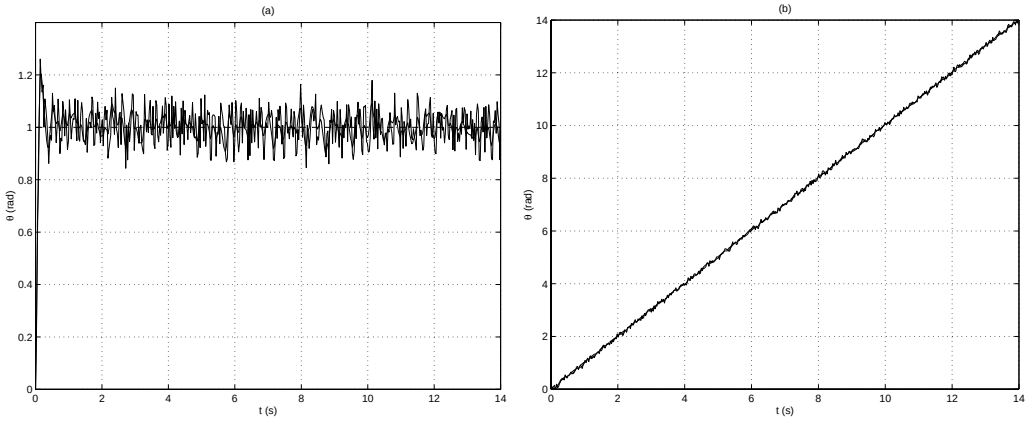

Figura 5.16: Resposta temporal do simulador com preditor e controlado com PI: $K_p = 19,135$ e $K_i = 7,123$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

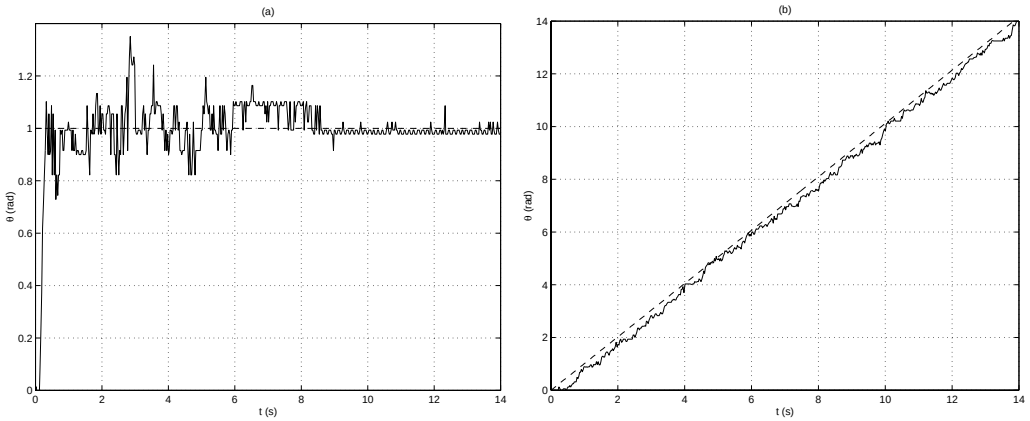

Figura 5.17: Resposta temporal do sistema real com preditor e controlado com PI: $K_p = 19,135$ e $K_i = 7,123$. (a) - Resposta ao degrau, (b) - Resposta à rampa. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

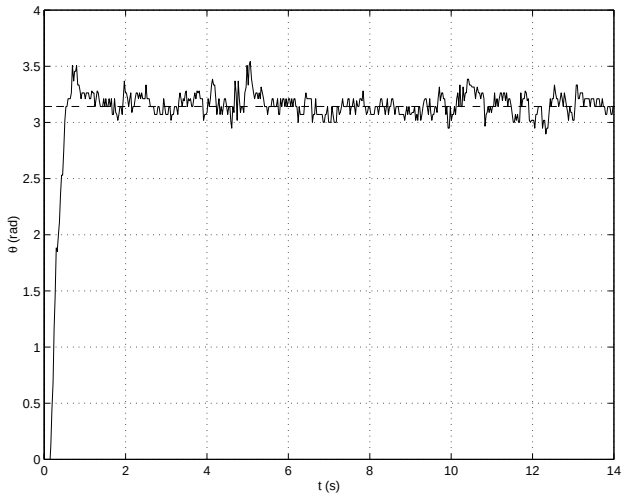

Figura 5.18: Resposta ao degrau do sistema real com preditor e controlado com PI: $K_p = 19,135$ e $K_i = 7,123$. A linha tracejada representa a referência e a linha contínua o valor real do ângulo.

5.2 Rastreamento de trajetórias

Como proposto no Capítulo 3, o projeto do controlador de distância é mais fácil que aquele para o ângulo, já que deve-se determinar um único ganho proporcional para o controlador. Assim a saída do controlador é simplesmente um valor proporcional à distância do robô ao alvo. Como as funções de transferência envolvidas são de certa forma complicadas, nesta seção optou-se por encontrar um ganho de forma empírica, ou seja, por tentativa e erro. Assim, para ajustar e validar este controlador será usado basicamente o simulador mostrado no Apêndice A e o próprio robô real. Além disso, o controlador de distância não tem sentido se o ângulo não estiver sendo ajustado. Desta forma, os controladores projetados na seção anterior serão utilizados durante os testes.

Como foi mencionado na Seção 3.3.3, constantes de desacoplamento podem ser utilizadas para tornar independentes as saídas lineares e angulares do controlador. Afim de testar o uso de tais constantes, a Figura 5.19(a) mostra o comportamento do robô, simulado em duas situações distintas. Na primeira, representada pela linha tracejada, a constante de desacoplamento não foi utilizada e na segunda, representada pela linha contínua, multiplicou-se o sinal enviado ao servo da direta (sinal u_1) por $1,03$ ou seja b_2^Θ/b_1^Θ. Em ambos os casos, o controlador de ângulo foi desligado e foi aplicado um degrau constante de 30 *bits* ao sinal de controle de distância. O robô foi inicializado com um ângulo de $\pi/4$ radianos. Note que realmente houve um desacoplamento com o uso da constante $1,03$ pois neste caso, a saída do sistema não apresentou praticamente nenhuma componente angular permitindo que o robô se movesse em linha reta. A Figura 5.20(b) mostra a mesma situação mas com a aplicação um degrau de 10 *bits*, ou seja o robô terá uma menor velocidade. Vê-se neste caso que não há diferença alguma entre as situações com ou sem

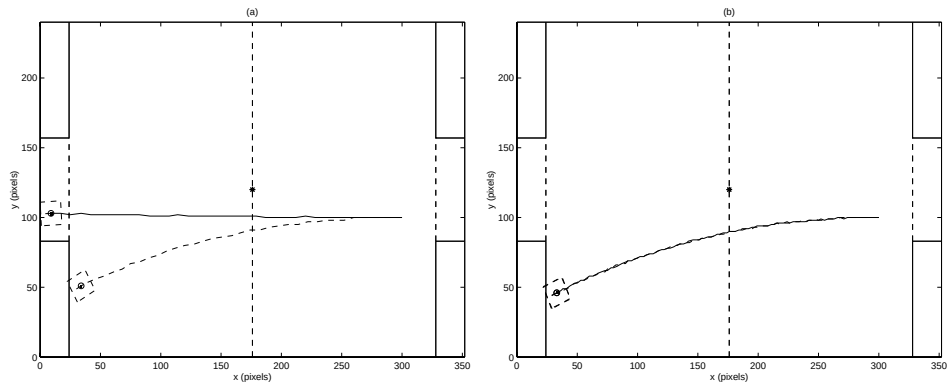

Figura 5.19: Efeito do desacoplamento entre as componentes angulares e lineares. A linha tracejada mostra o comportamento do simulador sem o desacoplamento e a linha contínua com o desacoplamento. (a) - altas velocidades, (b) - baixas velocidades.

desacoplamento. Isto ocorre, devido ao problema do arredondamento, identificado na seção anterior. Como a constante de desacoplamento é pequena, após o arredondamento não existe diferença entre os sinais, com ou sem esta constante. Na seção anterior este problema foi superado com o aumento do ganho que consequentemente aumentaria o sinal de controle. Aqui, isto, a princípio não fará muita diferença, pois com o controlador, o sinal de controle próximo ao alvo será pequeno não importando o valor de seu ganho. Isto entretanto, não é um problema grave já que para pequenas velocidades lineares, o controlador de ângulo pode atuar e manter a trajetória correta. Não há portanto, um completo desacoplamento entre as malhas de controle. Apesar disso, em todos os testes as constantes de desacoplamento serão utilizadas na tentativa de melhorar o desempenho do sistema, quando isso for possível. A Figura 5.20 mostra o uso constante de desacoplamento para o sistema real. Note que o valor deve ser diferente daquela calculada para o simulador, já que o robô real sem desacoplamento tende para o lado contrário do robô simulado. Como no caso do ângulo, o valor utilizado foi obtido *a*

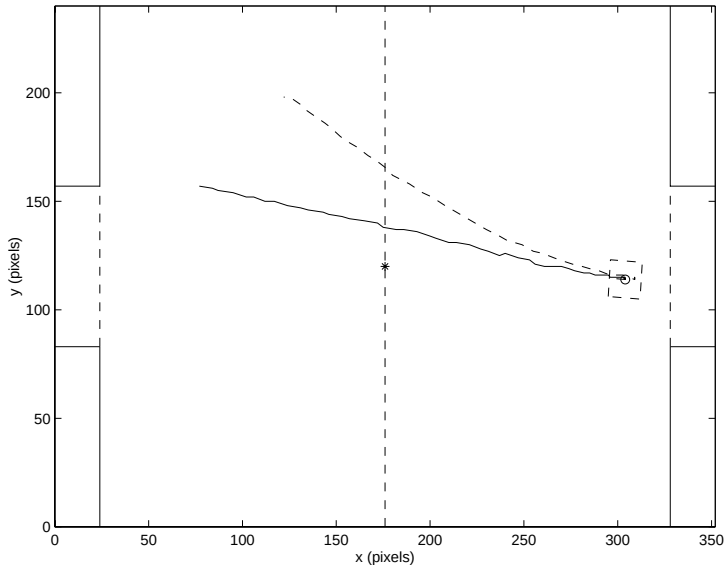

Figura 5.20: Efeito do desacoplamento entre as componentes angulares e lineares para o robô real. A linha tracejada mostra o comportamento do robô sem o desacoplamento e a linha contínua com o desacoplamento.

priori e mantido constante durante os testes.

O controlador do robô deve ser projetado de forma que o mesmo atinja um alvo ou siga uma trajetória. Desta forma, para testar e projetar os controladores, serão usados quatro tipos de referências: um alvo fixo, uma trajetória retilínea, uma trajetória circular e uma trajetória senoidal. Então, o controlador será considerado bom, se o robô atingir o alvo e percorrer todas as trajetórias com erros pequenos.

A Figura 5.21 mostra o comportamento do simulador quando a referência é um alvo fixo, representado por '□' na Figura 5.21(d). Neste caso, o robô é inicializado a uma certa distância do alvo contendo um erro de orientação. O ganho de distância, K, utilizado foi de $0,05$. Por ter apresentado bons resultados, o controlador PI de ângulo foi ajustado com $K_D = 0,115$ e $z_0 = 0,99$ ($K_p = 8,802$ e $K_i = 3,276$) e preditor para compensação do atraso no ângulo. A não ser que seja dito o contrário, esta configuração para o controlador de

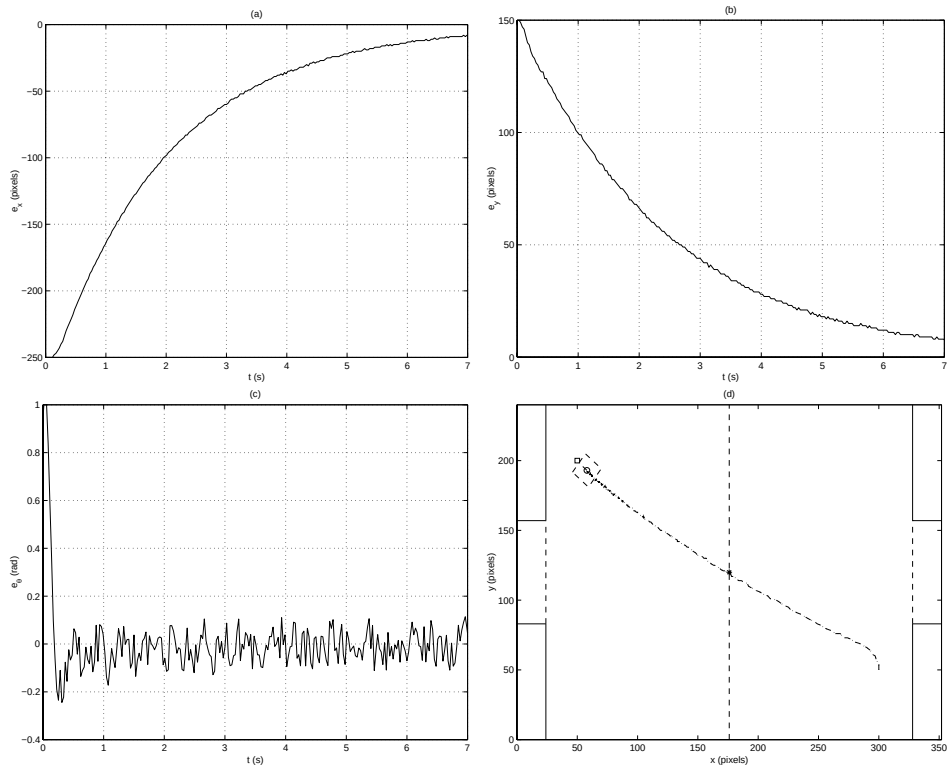

Figura 5.21: Comportamento do simulador quando o alvo é um ponto fixo para $K = 0,05$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. O símbolo '□' representa o alvo.

ângulo foi usada em todos os resultados experimentais desta seção. Não é utilizado preditor para a malha de distância pois, a princípio, este não traria melhorias ao sistema. Note que o ângulo é corrigido de forma mais rápida que a distância e por isso, o robô executa uma trajetória praticamente retilínea. Comportamento similar pode ser verificado na Figura 5.22 onde os mesmos ganhos dos controladores foram utilizados no sistema real. Pelas figuras de erros, vê-se que o erro de ângulo é o que apresenta maior ruído. Além do maior nível de ruído desta medida, do maior ganho de seu controlador e dos erros de predição, este grande ruído é, no caso real, causado também pelas próprias características do alvo utilizado. Para facilitar as mudanças de

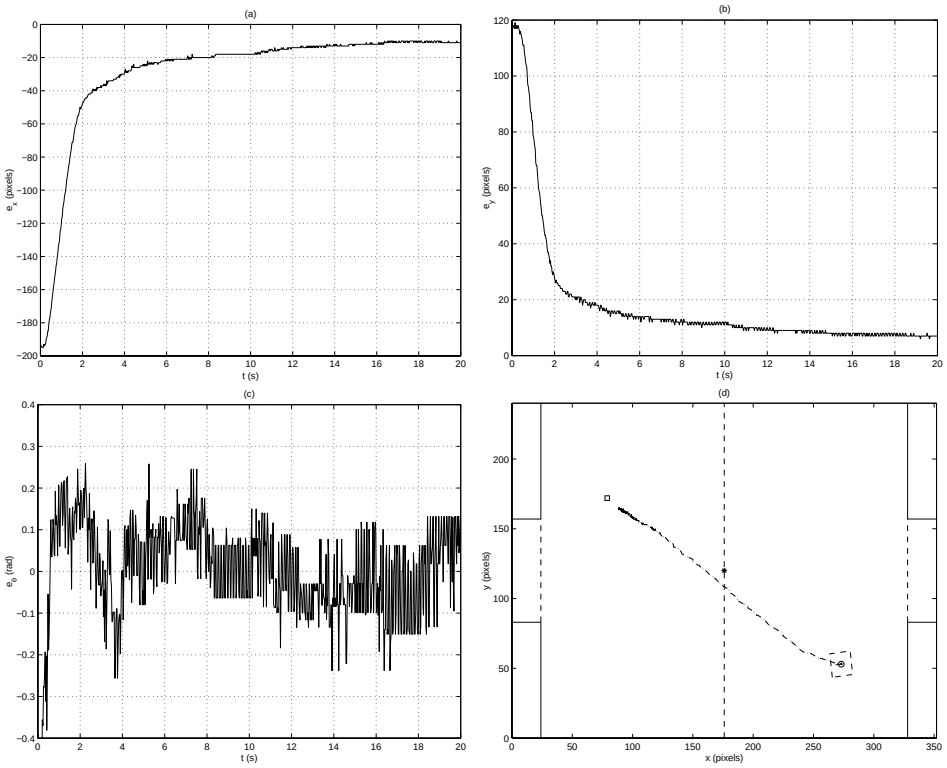

Figura 5.22: Comportamento do robô real quando o alvo é um ponto fixo para $K = 0,05$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. O símbolo '□' representa o alvo (bola).

referência, a medida de posição da bola usada nas partidas de futebol, foi utilizada como alvo. Como o sistema de visão já fornece esta informação sem prejudicar o tempo de processamento, esta alternativa na prática mostrou-se muito útil. Note também, que como o ganho de distância é baixo, quando o robô está muito perto da bola, a grandeza $K \cdot d$, não é suficiente para fazer o robô andar, devido ao problema da quantização, fazendo com que este nunca atinja o alvo (bola). Para corrigir este problema o ganho deve ser aumentado.

Como o comportamento do simulador se mostrou muito próximo da reação do robô real, a partir de agora, a etapa de simulação, apesar de realizada, será omitida durante o ajuste e validação dos ganhos. A Figura 5.23 mostra

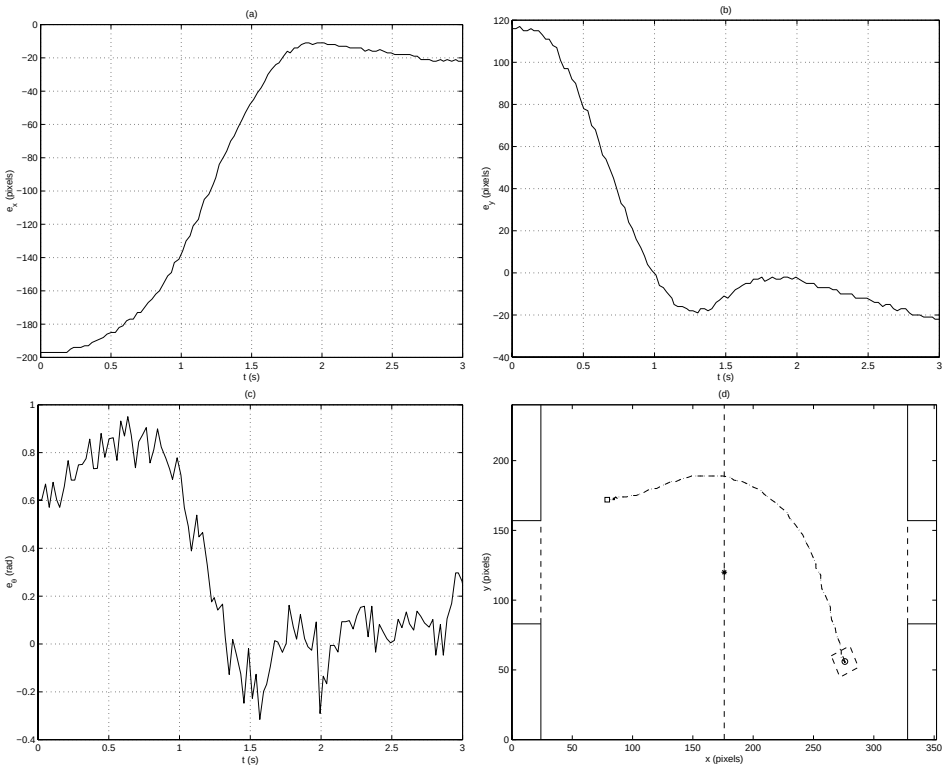

Figura 5.23: Comportamento do robô real quando o alvo é um ponto fixo para $K = 0,1$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. O símbolo '□' representa o alvo (bola).

o comportamento do robô real quando o ganho K foi dobrado, passando para $0,1$. Observa-se que neste caso a correção da distância é executada quase na mesma velocidade do ajuste de ângulo e por isso, o caminho do robô tende a ser menos linear. É claro, que a forma do caminho é determinada não só pelo ganho mas principalmente, pelo ângulo inicial do robô. Se o robô estiver apontado para a bola a trajetória é mais retilínea. No caso anterior, isto entretanto, não era tão importante pois o ângulo era corrigido logo no início do movimento que tendia a ser sempre reto. Então, percebe-se que o uso de um controlador de ângulo diferente, levaria a outros resultados, indicando que existe uma dependência entre os dois controladores. Observando

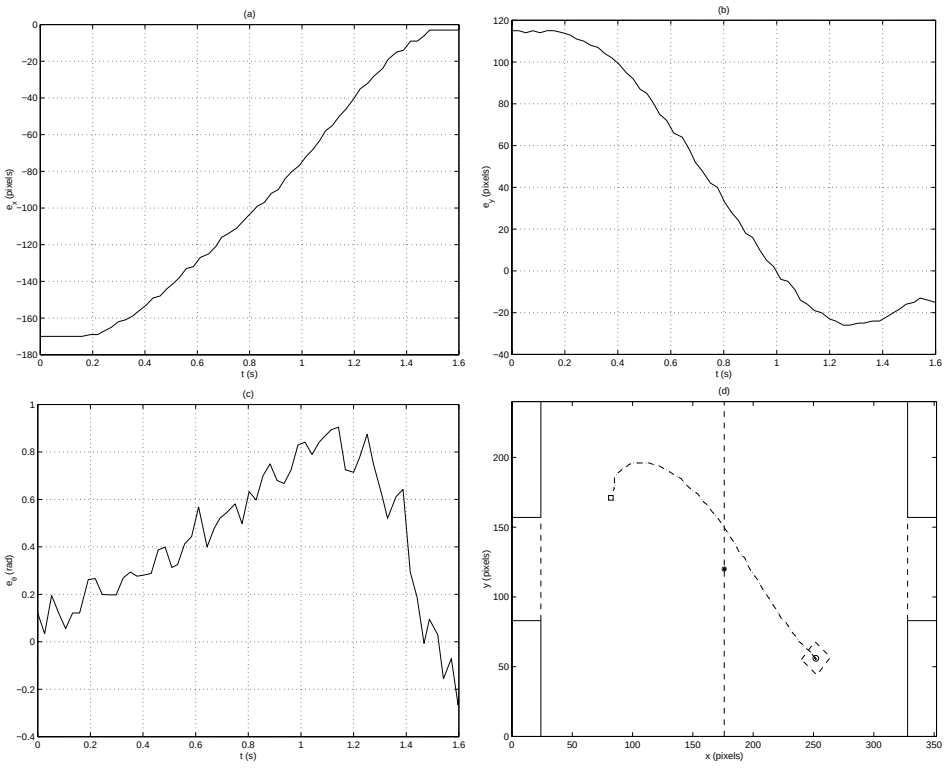

Figura 5.24: Comportamento do robô real quando o alvo é um ponto fixo para $K = 0,15$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. O símbolo '□' representa o alvo (bola).

os comportamentos dos erros, vê-se que a grande vantagem do aumento de ganho foi em relação ao tempo de resposta. Apesar de ter executado uma trajetória mais longa, o robô atingiu o alvo na metade do tempo, o que, em certos casos é desejável. Além disso, houve colisão com a bola o que não aconteceu no caso anterior. A Figura 5.24 mostra o comportamento do robô real para $K = 0,15$. Neste caso o tempo para atingir o alvo foi ainda menor, mesmo com o robô tendo ultrapassado o seu objetivo. O problema disso, é que se houver obstáculos, estes poderiam ser atingidos. Então, a escolha do ganho quando se deseja atingir alvos fixos, pode depender da aplicação. Se o alvo estiver em situações favoráveis, onde não haja obstáculos próximos,

grandes valores de ângulos podem ser utilizados. Para o presente campo de trabalho, por exemplo, constatou-se que valores de ganhos maiores que $0,15$ provocavam muitas colisões com as paredes no caso de alvos muito longe da posição do robô, o que não é desejado.

Os resultados anteriores, mostraram que o robô atingia o alvo com caminhos e tempos variáveis, dependendo da posição e orientação iniciais e do valor dos ganhos. Para que este comportamento seja mais controlado, o robô pode ser forçado a executar trajetórias pré-definidas. Estas trajetórias, nada mais são do que alvos, ordenados em seqüência e com menores distâncias entre eles. Então, a cada instante, a referência do robô varia de um alvo para outro, e se o controlador estiver bem ajustado e todos os alvos forem atingidos, no final do processo o robô terá percorrido um caminho pré-determinado. A velocidade em que o caminho é percorrido pode ser controlada pela distância entre os alvos, ou pontos que formam o caminho, que na verdade é a sua derivada. Por isso, se os pontos estiverem mais longe uns dos outros o robô andará mais rápido e *vice-versa*. O ideal é que a trajetória se inicie na posição do robô, levando em consideração inclusive o seu ângulo atual, impedindo assim, que este execute movimentos descontrolados. Como a geração de trajetórias não é o objetivo deste trabalho, este procedimento será executado em tempo real mas com o formato e posição dos caminhos sendo definido *a priori*. Na medida do possível o robô será inicializado no começo dos mesmos. Quando isto não for possível, o robô interceptará a trajetória de forma livre, como nos resultados anteriores.

A Figura 5.25 mostra o robô real rastreando um trajetória retilínea. O robô foi inicializado de forma que houvesse um grande erro em relação ao início de caminho e o seu ganho de distância K, foi ajustado para $0,15$. Pela Figura 5.25(d) vê-se que o robô intercepta a trajetória e se mantém junto

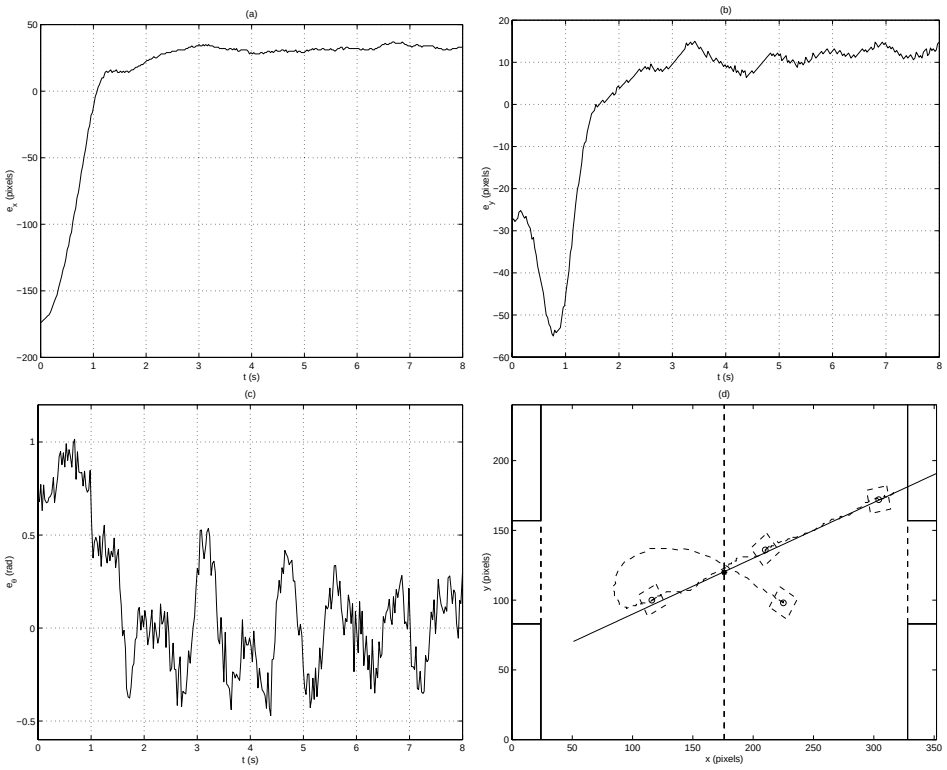

Figura 5.25: Comportamento do robô real seguindo uma trajetória retilínea para $K = 0,15$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

a esta com pequenos erros. Nas Figuras 5.25(a) e (b) entretanto, percebe-se que existe um erro significativo nas componentes x e y. Isto ocorre por dois motivos principais: o sistema só possui uma integração e por isto era esperado um erro constante para entradas em rampa; existe um grande atraso no sistema. Este segundo motivo, não permite que o sistema atinja um alvo móvel, a não ser que haja um comportamento sobreamortecido causado por altos valores de ganhos, o que não é desejado. Assim, o comportamento do robô será sempre de perseguição aos alvos móveis, nunca chegando a seu objetivo. Apesar deste comportamento aparentemente parecer ruim, ele faz com que o robô percorra a trajetória com velocidades contínuas e não pare

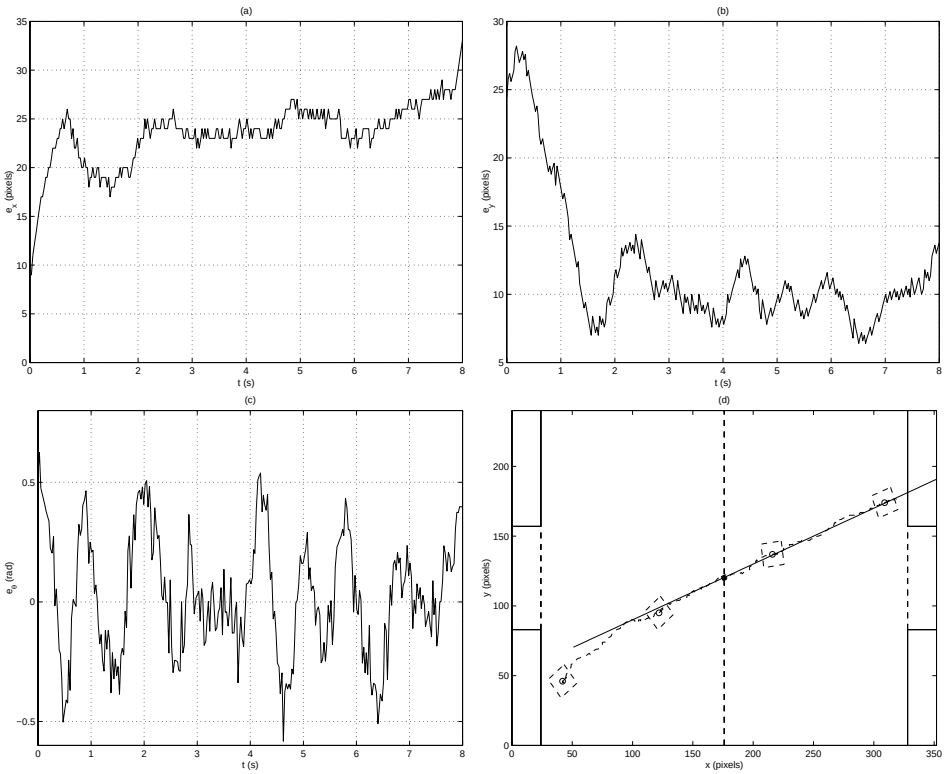

Figura 5.26: Comportamento do robô real seguindo uma trajetória retilínea para $K = 0,2$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

a cada ponto. Pela Figura 5.25(c) pode-se notar que novamente, o erro de orientação é a variável mais ruidosa, como já era esperado. Apesar disso, vê-se que este "ruído" não atrapalhou o comportamento do robô, a não ser pela pequenas oscilações que foram previstas na seção anterior. Note que para esta variável não existe atraso nem erro em regime permanente pois foi adicionado uma segunda integração e compensado o atraso.

A Figura 5.26 mostra a mesma situação com o ganho passando para $K = 0,2$. Nota-se que houve uma diminuição de todos os erros com exceção de e_θ que permaneceu inalterado. Isto indica que o robô aumentou a sua velocidade linear e conseguiu chegar mais perto de seus alvos (pontos da

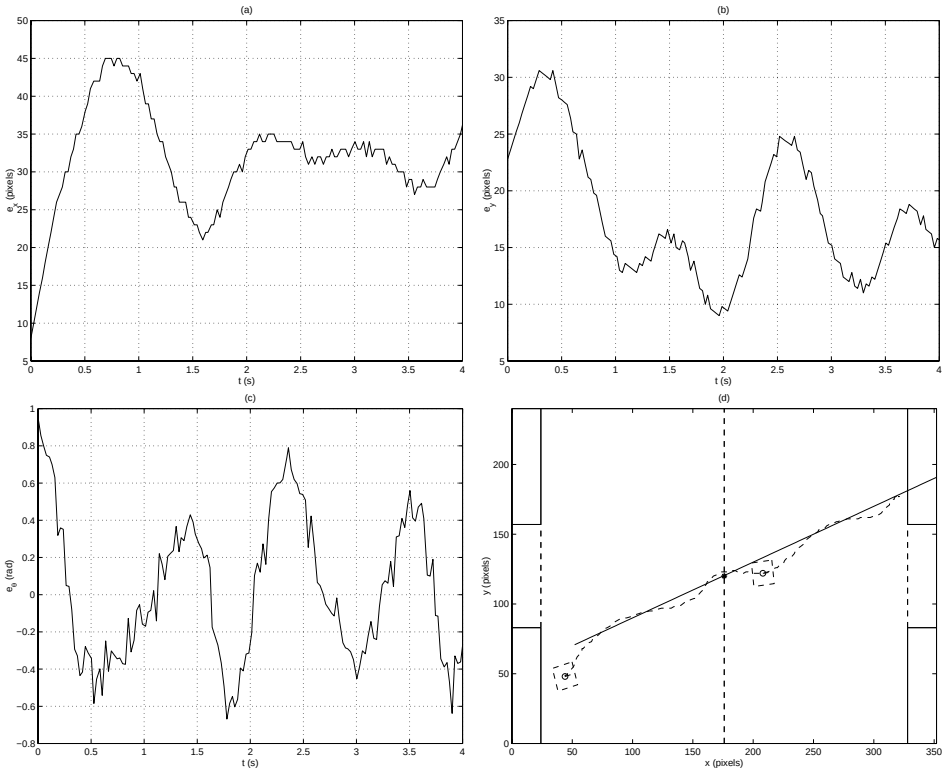

Figura 5.27: Comportamento do robô real seguindo uma trajetória retilínea para $K = 0,2$, $K_p = 8,802$ e $K_i = 3,276$ com maior velocidade. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

reta). Apesar disso, este aumento não foi escolhido, tendo sido causado pelo aumento do ganho. Na Figura 5.27 é mostrado um caso onde tentou-se dobrar a velocidade de movimentação do robô. Para tanto, uma nova trajetória onde a distância entre os pontos é duas vezes maior foi usada como referência. Note que os erros de x e y aumentaram um pouco em relação ao caso anterior, mas o robô percorreu a mesma distância na metade do tempo. Este comportamento pode ser melhor observado na Figura 5.28 que mostra as velocidades estimadas para os dois casos. Na Figura 5.28(a) observa-se que a velocidade é de aproximadamente 40 *pixels*/s, aumentando para 80 *pixels*/s na

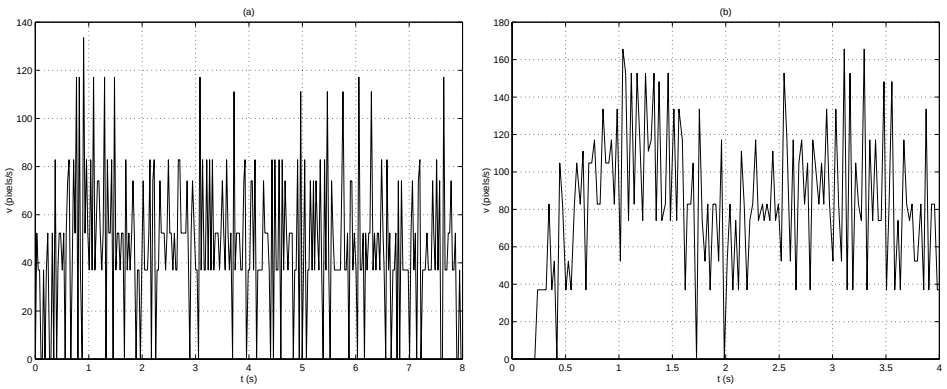

Figura 5.28: Comparação entre as velocidades estimadas do robô real para duas trajetórias retilíneas. (a) - valores de x estão espaçados de 1 *pixel*, (b) - valores de x estão espaçados de 2 *pixels*.

Figura 5.28(b) que representa a velocidade da situação onde o espaçamento, ou discretização da trajetória dobrou. A velocidade estimada foi calculada como a diferença entre as distâncias consecutivas, calculadas a cada ponto, dividida pelo período de amostragem.

Na Figura 5.29 é mostrado o comportamento do robô perante uma trajetória circular e com $K = 0,1$. A trajetória deveria ser percorrida no sentido horário. Pela Figura 5.29(d) nota-se que visualmente o robô apresenta grandes erros. As Figuras 5.25(a) e (b) entretanto, não têm muito significado pois ha uma grande variação do erro. Isto ocorre, porque em trajetória circulares as variáveis x e y variam de forma senoidal e como o robô está sempre atrasado em relação à trajetória a forma do erro também será senoidal ou alguma coisa parecida. O erro de ângulo está novamente com o mesmo comportamento, como já era esperado, já que não houve ajuste no controlador desta variável.

A Figura 5.30 mostra a resposta do robô para a mesma trajetória circular mas com o ganho dobrado. Note que neste caso o robô executa uma trajetória bem próxima da original. Observe também que o robô foi inicializado pra-

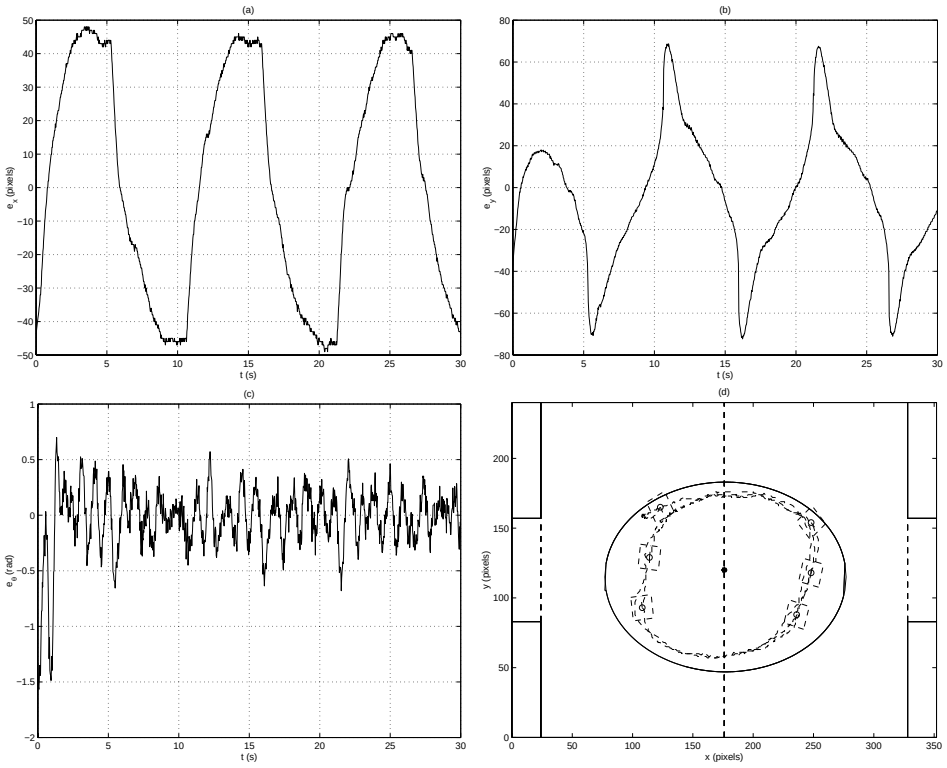

Figura 5.29: Comportamento do robô real seguindo uma trajetória circular para $K = 0,1$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

ticamente do centro da trajetória e conseguiu interceptá-la rapidamente. As regiões onde estão os maiores erros, ou seja, próximo aos gols, são também onde, por construção da curva, existem menos pontos. Desta forma, o robô imprime maior velocidade e comete maiores erros. Apesar de ser possível gerar uma circunferência com os pontos igualmente distribuídos esta situação foi mantida para efeito de ilustração. O efeito desta "deformação", pode ser observado também no caso anterior. Na Figura 5.31 onde o ganho foi aumentado para $0,3$, um novo comportamento é causado por este mesmo motivo. Como o ajuste do ganho não consegue acompanhar o aumento de velocidade, os erros são maiores. O interessante, é que como os erros de orientação são

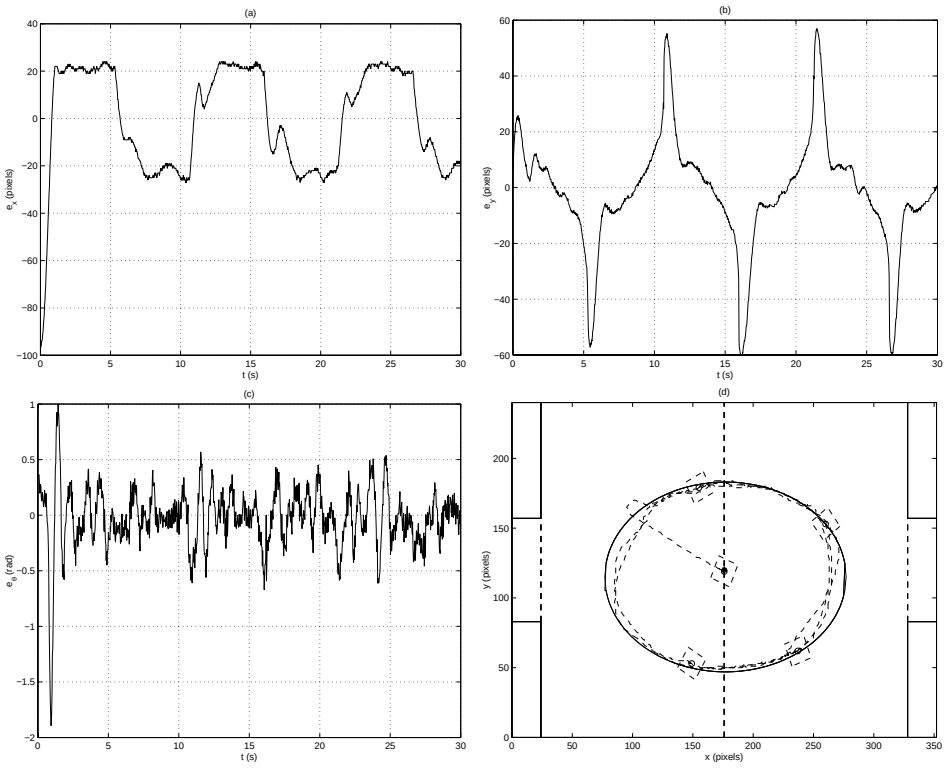

Figura 5.30: Comportamento do robô real seguindo uma trajetória circular para $K = 0,2$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

muito grandes, o controlador inverte a frente do robô permitindo com que o ajuste seja mais rápido. Para mostrar que realmente a distância entre os pontos é importante no desempenho do sistema, a Figura 5.32 mostra o robô com o ganho $K = 0,2$ percorrendo uma trajetória circular onde foi garantido que os pontos são igualmente espaçados. Para isso, a trajetória foi gerada com o uso de coordenadas polares, em oposição à anterior que originou-se de funções cartesianas de x e y. Note que para esta nova situação, o desempenho do robô se mantém inalterado durante todo o percurso.

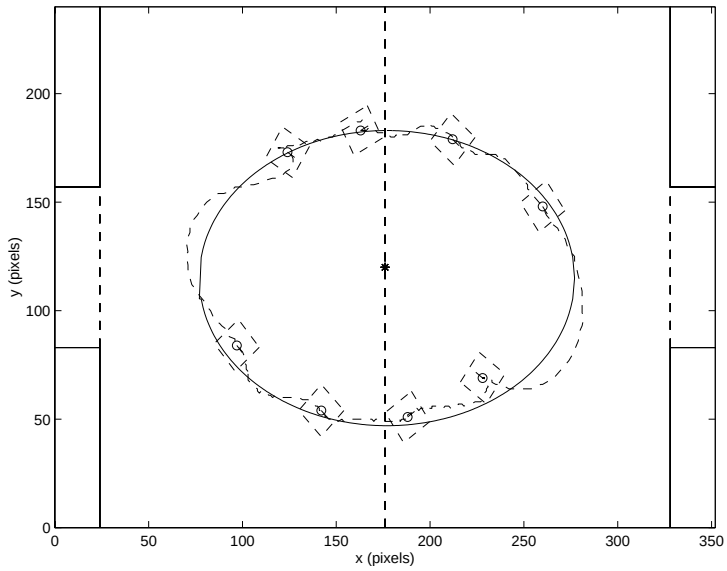

Figura 5.31: Comportamento do robô real seguindo uma trajetória circular para $K = 0,3$, $K_p = 8,802$ e $K_i = 3,276$. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

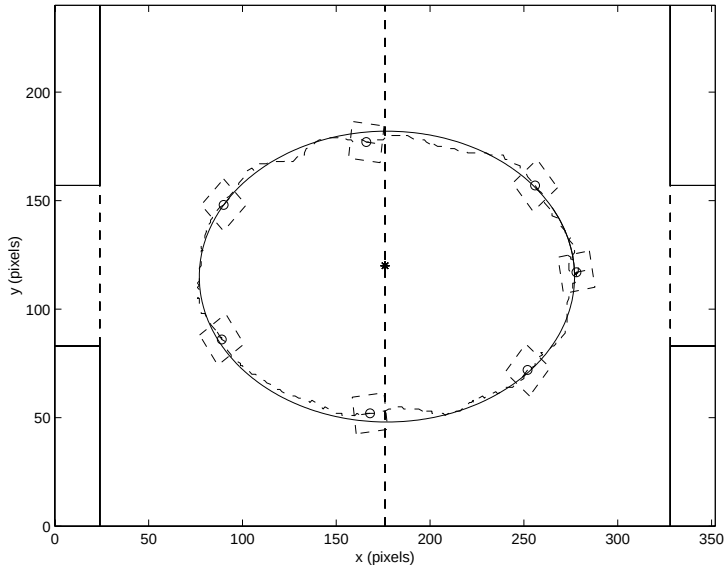

Figura 5.32: Comportamento do robô real seguindo uma trajetória circular onde os pontos estão igualmente espaçados, para $K = 0,2$, $K_p = 8,802$ e $K_i = 3,276$. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

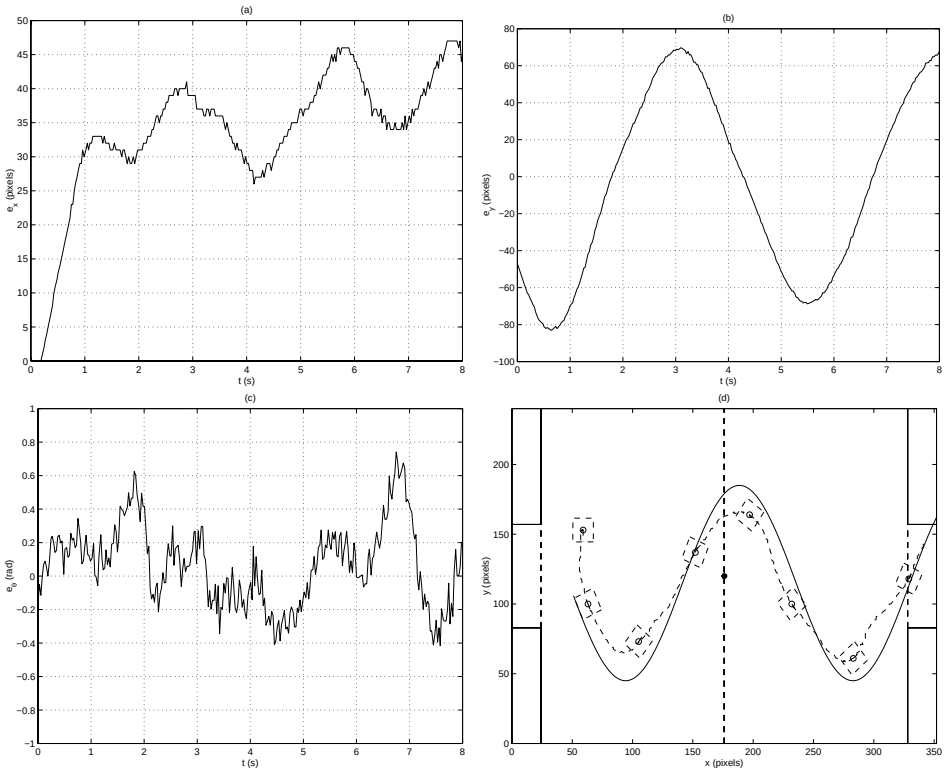

Figura 5.33: Comportamento do robô real seguindo uma trajetória senoidal para $K = 0,1$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

Apesar de ser mostrado na literatura que as trajetórias ideais para robôs móveis autônomos são as retas e os círculos, uma trajetória interessante e fácil de ser gerada é a trajetória senoidal. Assim, este tipo de trajetória também será utilizado como validação e ajuste do controlador projetado. Uma particularidade desta função é que existe um grande número de pontos nos picos e vales e um pequeno número de pontos entre eles. Assim, o robô tenderá a se deslocar mais rapidamente durante as "retas" e diminuir a velocidade nas "curvas". A Figura 5.33 mostra o comportamento do robô seguindo um caminho senoidal. Nesta figura o ganho foi ajustado como $K = 0,1$. Note que a trajetória percorrida pelo robô, apesar de ser parecida com a original,

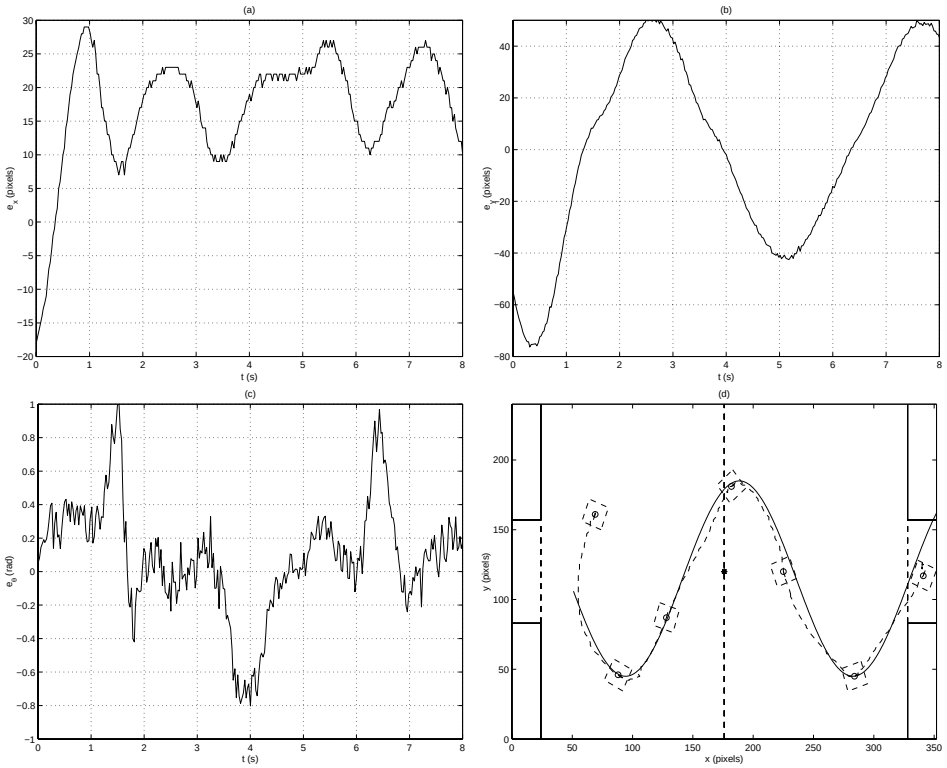

Figura 5.34: Comportamento do robô real seguindo uma trajetória senoidal para $K = 0,2$, $K_p = 8,802$ e $K_i = 3,276$. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

possuí erros visuais consideráveis. Pelo mesmo motivo apresentado para o caminho circular o erro em y apresenta a forma senoidal. Em oposição, o erro em x se assemelha daquele para a trajetória retilínea, já que esta coordenada varia da mesma forma nos dois casos. Para melhorar o desempenho do sistema, o ganho foi aumentado para $K = 0,2$ e os resultados são mostrados na Figura 5.34. Note que neste caso, a trajetória executada pelo robô é muito próxima da real. Por este motivo e pelos resultados anteriores, observa-se que uma boa escolha de ganho é $K = 0,2$. Este valor somente não apresentou bons resultados quando o alvo é um ponto fixo muito distante da posição inicial do robô. Entretanto, quando o alvo está próximo este ganho pode

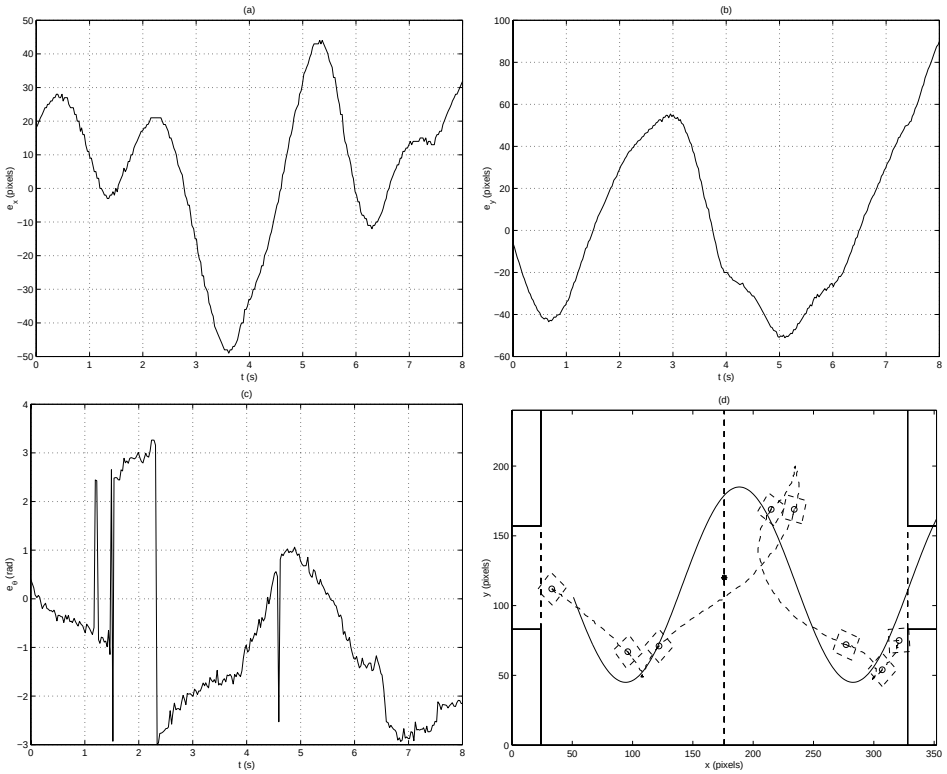

Figura 5.35: Comportamento do robô real seguindo uma trajetória senoidal para $K = 0,2$, $K_p = 1,995$ e $K_i = 0,37$ e sem predição de ângulo para compensação de tempo morto. (a) - erro de x, (b) - erro de y, (c) - erro de θ, (d) - caminho executado pelo robô. A linha contínua representa a referência e a tracejada o caminho percorrido pelo robô.

ser utilizado como mostrou os resultados onde o robô deveria interceptar a trajetória.

Como foi mencionado, todos os resultados mostrados até aqui, utilizam $K_D = 0,115$ e $z_0 = 0,99$ como ajustes do controlador de ângulo. Na seção anterior mostrou-se que o uso de tais parâmetros somente era possível se houvesse uma compensação de tempo morto. Como prova de que esta compensação através de predição do ângulo é importante para o controle do robô, a Figura 5.35 mostra o comportamento do robô quando a predição é eliminada e os parâmetros do controlador são reajustados para $K_D = 0,026$

e $z_0 = 0,995$, um dos pares utilizados na Seção 5.1. A trajetória utilizada é a mesma senóide usada anteriormente e $K = 0,2$. Note que, como esperado, o erro de ângulo é menos ruidoso. Entretanto, a trajetória executada pelo robô está muito distante da referência pois controlador de ângulo não consegue corrigir a orientação a tempo deixando o robô descontrolado.

5.3 Cooperação entre robôs

Um dos objetivos do futebol de robôs é estudar a cooperação entre os agentes. Por isso, o sistema de controle projetado deve ser capaz de controlar diversos robôs diferentes de forma simultânea. Até este ponto, todos os resultados foram mostrados para um único robô. Entretanto, toda a implementação do projeto foi executada pensando-se na possibilidade de se estar usando mais de um robô. Assim, será mostrado nesta seção, através de um resultado prático, que o uso de mais um robô não influencia o comportamento total do sistema. Os recursos implementados, tais como a identificação e predição em tempo real são úteis na adaptação do sistema a cada robô, mesmo que eles possuam pequenas diferenças construtivas.

A Figura 5.36 mostra o comportamento de dois robôs seguindo trajetórias circulares idênticas mas com os centros deslocados. Como simplificação, os robôs somente foram desenhados em suas posições iniciais e finais. Note que os dois robôs foram iniciados em posições aleatórias mas conseguiram interceptar a trajetória rapidamente. Os ganhos dos controladores de ambos os robôs são $K = 0,2$, $K_p = 1,995$ e $K_i = 0,37$. O robô que percorre a trajetória que esta mais à direita, foi aquele usado em todos os testes anteriores. O segundo robô, é muito similar ao primeiro, possuindo pequenas diferenças mecânicas nos rolamentos de apoio e na ligação entre as placas de

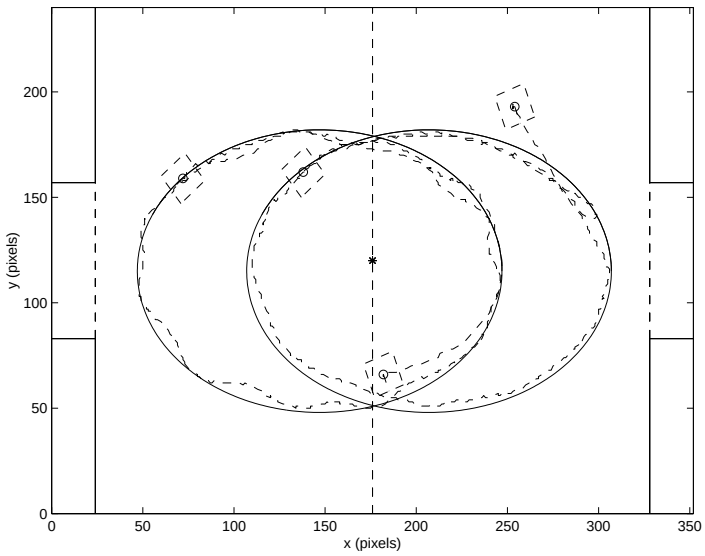

Figura 5.36: Dois robôs seguindo trajetórias circulares idênticas com os centros deslocados. A linha contínua representa a referência e a linha tracejada o caminho percorrido pelos robôs.

alumínio que formam o corpo. Além disso, este robô possui um potenciômetro que permite que o valor do sinal de controle necessário para manter o robô parado, seja ajustado. Através da identificação em tempo real deste robô, obteve-se o seguinte submodelo para a orientação:

$$\theta_k = 1,5179\,\theta_{k-1} - 0,5179\,\theta_{k-2} - 0,0070\,u_{1\,k-8} + 0,0059\,u_{2\,k-8} \ ,$$

indicando que realmente existe uma diferença entre os robôs (ver Equação (4.2)), principalmente em relação aos ganhos que multiplicam os sinais de controle. Apesar das diferenças, pela Figura 5.36 nota-se que o comportamento de ambos os robôs é muito similar, mostrando que o sistema é robusto a estas pequenas variações. Além disso, é fácil observar em comparação com os resultados anteriores, que houve modificações insignificantes no desempenho do sistema após a inclusão de um novo robô, comprovando o fato de que os controladores projetados estão preparados para operar com mais de um agente.

Capítulo 6

Conclusões e Perspectivas Futuras

> Nunca conseguimos o que queremos, mas se nos esforçarmos um pouco conseguimos o que precisamos.
>
> *Mick Jagger (1943–)*

NESTE trabalho foi proposta e desenvolvida uma forma de controlar a posição e a orientação de robôs móveis não-holonômicos. A metodologia proposta é uma alternativa simples às técnicas atualmente utilizadas que se baseiam em controladores não lineares e multivariáveis ou em técnicas de inteligência computacional como lógica nebulosa ou redes neurais. Em oposição a estas abordagens o controlador proposto é baseado em sistemas lineares e monovariáveis clássicos. Além disso, uma das grandes diferenças deste trabalho em relação aos anteriores, é que aqui supõe-se que os robôs não possuem malhas de controle de velocidade independentes em cada roda, o que é muito importante para o bom desempenho dos controladores de posição. Todas as propostas apresentadas, foram desenvolvidas para robôs móveis com duas entradas de controle e movimentação bidimensional, mas podem ser estendidas a outros tipos de veículos onde as variáveis angulares podem ser matematicamente desacopladas das variáveis lineares. O projeto do controlador é baseado nos modelos dinâmicos do sistema, e é justamente

na determinação destes modelos que está uma das grandes contribuições do trabalho. Os modelos dinâmicos utilizados na bibliografia são sempre baseados na física do processo o que dificulta a determinação de alguns parâmetros que não podem ser diretamente medidos. A modelagem proposta no trabalho se utiliza de estruturas determinadas a partir de considerações físicas do robô mas a determinação dos parâmetros do modelo é feita a partir de um conjunto de dados práticos, o que facilita o processo de identificação. Uma vez determinado o modelo, a proposta é usa-lo, não só para projetar e simular o controlador, como também para predizer o comportamento futuro do robô e permitir uma compensação de tempo morto.

Para validar as metodologias propostas, foi utilizada uma plataforma de futebol de robôs que se constitui, basicamente, de um computador que controla, por rádio, três micro-robôs observados por um sistema de visão computacional. Uma análise desta configuração, mostrou que a ela estão associadas algumas características que de certa forma dificultam o controle:

1. os robôs não possuem um processamento ou sensoriamento local e por isso todo o controle deve ser feito baseado em informações visuais a uma taxa de amostragem insuficiente para medir grandezas rápidas, como por exemplo a velocidade das rodas;

2. existe um grande atraso no sistema que indiretamente, torna o sistema de controle lento e com uma pequena margem de estabilidade;

3. o sistema de comunicação do computador com o robô é quantizado em poucos valores, impedindo que pequenas variações do sinal de controle sejam percebidas pelos motores responsáveis pela movimentação.

Com exceção do item 2, cujas causas não foram completamente isoladas, estas características, estão diretamente associadas ao *hardware* do sistema e

por isso, mudanças em algumas partes deste *hardware* poderiam melhorar consideravelmente o desempenho dos robôs e facilitar o projeto do sistema de controle. Durante o trabalho, entretanto, optou-se por manter a parte física do processo inalterada, buscando alternativas no *software* para corrigir ou minimizar os efeitos causado pelas deficiências do *hardware*.

O trabalho mostrou que o item 1 é um problema que dificulta o controle pois algumas perturbações rápidas não podem ser compensadas sem o uso de controladores dedicados com taxas de amostragem maiores. Além disso, pôde ser visto que os controladores locais de velocidade seriam muito importantes para o desacoplamento entre as variáveis lineares e angulares. Foi proposta um forma de minimizar este problema através do uso de funções de desacoplamento na saída dos controladores. Entretanto, esta alternativa não se mostrou viável primeiro porque não se conseguiu determinar em tempo real o valor desta função e principalmente, pela presença da quantização mencionada no item 3. O atraso do sistema, mencionado no item 2, foi parcialmente resolvido com o uso de um preditor para a orientação do robô, baseado no modelo identificado, apesar de não se ter conseguido que atualização em tempo real deste modelo funcionasse de forma eficiente. O item 3 não pode ser resolvido por *software*, apesar de algumas de suas conseqüências terem sido minimizadas com simples ajustes nos ganhos dos controladores.

Pelos resultados de validação do modelo, viu-se que a modelagem estocástica de robôs móveis é uma alternativa simples para uma questão complicada que é a de se encontrar equações dinâmicas que representem um sistema físico. Como na literatura este tipo de modelagem não foi muito discutido, a conclusão a que se pode chegar é que o método empregado é muito mais fácil de ser implementado que os métodos tradicionais de identificação determinística, apesar de ambos poderem ser aplicados em situações

semelhantes. A técnica aplicada neste trabalho é baseada na identificação estocástica de parâmetros de uma estrutura determinada através da análise física do sistema. O trabalho que mais se assemelha a esta técnica é mostrado em [Elnagar and Gupta, 1998]. Por comparação pode ser visto que os resultados apresentados no Capítulo 4, são semelhantes e até melhores aos mostrados neste artigo para predição da posição dos robôs apesar das diferenças nos objetivos dos dois trabalhos. Foi mostrado que os resultados de predição poderiam melhorar ainda mais, se as técnicas de identificação em tempo real fossem apuradas com o uso de algoritmos de detecção de falhas.

O uso da predição para compensação do tempo morto em robôs móveis, também não foi um tema encontrado na literatura. Os resultados práticos, entretanto, mostraram que esta é uma solução prática que apresentou bons resultados. O desempenho do controlador proposto se mostrou bastante satisfatório, considerando a sua aplicação. As pequenas oscilações e erros presentes na saída do sistema certamente não comprometerão o comportamento dos robôs em partidas de futebol. Os resultados do Capítulo 5 são comparáveis aos mostrados em [Yang and Kim, 1999] onde um controlador de estrutura variável foi utilizado para controlar robôs de futebol. A principal diferença é que os robôs utilizados neste artigo são muito precisos possuindo controladores de locais de velocidade e não é mencionada a presença de atrasos.

Com tudo isso, os resultados práticos em geral foram satisfatórios e comprovaram a metodologia sugerida no Capítulo 3. Entretanto, os resultados mostraram que à medida que a velocidade de deslocamento dos robôs aumenta, os erros do controle também aumentam. Este problema é causado tanto pela falta de controle local, que não permite um grande aumento de velocidade linear sem provocar velocidades angulares indesejadas, quanto pela

presença do atraso, que não permite um aumento da velocidade de resposta do controlador de ângulo e cuja compensação provoca oscilações nesta variável. O atraso também provoca erros no controle de posição dos robôs, pois os alvos, que geralmente são móveis, correm o risco de não serem atingidos. Assim, algumas propostas de trabalhos futuros, que algumas vezes são excludentes, podem ser feitas afim de melhorar ainda mais o desempenho dos robôs:

- alteração da interface entre o computador e o transmissor para melhorar a quantização;
- detecção de falhas na estimação de parâmetros em tempo real para melhorar a predição e permitir a adaptação do controlador;
- implementação de controladores locais de velocidade nos robôs;
- identificação e eliminação das causas do atraso;
- uso de preditores também para o alvo, permitindo que os erros sejam minimizados.

Além disso, o controlador deve ser utilizado para que os robôs "joguem futebol" e participem de competições nacionais e internacionais. Por isso, devem ser desenvolvidos dois níveis superiores ao nível de controle: um para proporcionar a cooperação e executar a estratégia de jogo, escolhendo as jogadas e quais robôs devem executá-las e outro para planejar os caminhos de forma que as estratégias sejam cumpridas. O controlador projetado no trabalho, garantirá então que estes caminhos possam ser executados de forma correta.

Referências Bibliográficas

[Åström and Wittenmark, 1995] Åström, K. J. and Wittenmark, B. (1995). *Adaptive Control*. Addinson-Wesley Publishing Company, Inc., 2nd edition.

[Abidi and Gonzales, 1992] Abidi, M. A. and Gonzales, R. C. (1992). *Data Fusion in Robotics and Machine Intelligence*. Academic Press.

[Achim et al., 1996] Achim, S., Stone, P., and Veloso, M. (1996). Building a dedicated robotic soccer system. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Osaka, Japan.

[Aguirre, 1995] Aguirre, L. A. (1995). A nonlinear correlation function for selecting the delay time in dynamical reconstructions. *Physics Letters A*, 203:88–94.

[Aguirre, 1999] Aguirre, L. A. (1999). *Identificação de Sistemas e Estimação de Parâmetros*. (in portuguese) to be published in 2000 by Editora UFMG.

[Aguirre and Jácome, 1998] Aguirre, L. A. and Jácome, C. R. F. (1998). Cluster analysis of NARMAX models for signal-dependent systems. *IEE Proceedings on Control Theory and Applications*, 145(4):409–414.

[Aguirre et al., 1998] Aguirre, L. A., Rodrigues, G. G., and Jácome, C. R. F. (1998). Identificação de sistemas não-lineares utilizando modelos NARMAX polinomiais - uma revisão e novos resultados. *Controle & Automação*, 9(2):90–106.

[Aicardi et al., 1995] Aicardi, M., Casalino, G., Bicchi, A., and Balestrino, A. (1995). Loop steering of unicycle-lik vehicles via lyapunov techniques. *IEEE Robotics & Automation Magazine*, 2(1):27–35.

[Arimoto et al., 1994] Arimoto, S., Parra-Vega, V., and Naniwa, T. (1994). A class of linear velocity observes for nonlinear mechanical systems. In *Proceedings of the 1st Asian Control Conference*, Tokyo, Japan.

[Asada and Slotine, 1985] Asada, H. and Slotine, J.-J. E. (1985). *Robot Analysis and Control*. Wiley-Interscience.

[Beer and Johnston Jr., 1991] Beer, F. P. and Johnston Jr., E. R. (1991). *Mecânica Vetorial para Engenheiros - Cinemática e Dinâmica*. McGraw Hill, 5th edition.

[Bloch et al., 1992] Bloch, A. M., Reyhanoglu, M., and McClamroch, N. H. (1992). Control and stabilization of nonholonomic dynamic systems. *IEEE Transactions on Automatic Control*, 37(11):1746–1757.

[Caccavale et al., 1999] Caccavale, F., Natale, C., and Villani, L. (1999). Task-space tracking control without velocity measurements. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 512–517, Detroit, Michigan.

[Caminhas, 1997] Caminhas, W. M. (1997). *Estratégias de Detecção e Diagnóstico de Falhas em Sistemas Dinâmicos*. PhD thesis, Universidade Estadual de Campinas, Campinas, São Paulo, Brasil.

[Campion et al., 1996] Campion, G., Bastion, G., and D'Andréa-Novel, B. (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62.

[Campos et al., 1998] Campos, M., Anício, M., Carvalho, M., Dias, R., Hartmann, A., Nagem, D., Oliveira, V., Oliveira, E., Pereira, G., Ribeiro, A., Sanches, F., and Silveira, M. (1998). MIneiROSOT – the development of a centralized control set of soccer-playing micro-robots. In *Proccedings of FIRA Robot World Cup*, pages 57–62, Paris.

[Campos and de Souza Coelho, 1999] Campos, M. F. M. and de Souza Coelho, L. (1999). Autonomous dirigible navigation using visual tracking and pose estimation. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 2584–2589, Detroit, Michigan.

[Caracciolo et al., 1999] Caracciolo, L., Luca, A. D., and Iannitti, S. (1999). Trajectory tracking control of a four-wheel differentially driven mobile robot. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 2632–2638, Detroit, Michigan.

[Cavallo et al., 1996] Cavallo, A., Setola, R., and Vasca, F. (1996). *Using MATLAB – SIMULINK and Control System Toolbox: A Practical Approach*. Prentice Hall.

[Cortesao et al., 1998] Cortesao, R., Millela, F., and Nunes, U. (1998). Joint robust position control using linear Kalman filters. In *Proceedings of the 5th International Workshop on Advanced Motion Control*, pages 417 – 422, Coimbra.

[de Wit and Sordalen, 1992] de Wit, C. C. and Sordalen, O. J. (1992). Exponential stabilization of mobile robots with nonholonomic contraints. *IEEE Transactions on Automatic Control*, 37(11):1791–1797.

[Elnagar and Gupta, 1998] Elnagar, A. and Gupta, K. (1998). Motion prediction of moving objects based on autoregressive model. *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, 28(6):803–810.

[FIRA, 1999] FIRA (1999). Federation of International Robot-soccer Association. http://www.fira.net.

[Frank and Seliger, 1991] Frank, P. M. and Seliger, R. (1991). Fault detection and isolation in automatic processes. *Control and Dynamics Systems*, 49:241–287.

[Fu et al., 1987] Fu, K. S., Gonzalez, R. C., and Lee, C. S. G. (1987). *Robotics: Control, sensing, vision and intelligence*. McGraw Hill.

[Fukuda and Kubota, 1997] Fukuda, T. and Kubota, N. (1997). Evolutionary robotic system. In *1997 Micro-Robot World Cup Soccer Tournament Proceedings*, pages 19–26.

[Futaba, 1999] Futaba (1999). Futaba. http://www.futaba-na.com.

[Gomes and Ramos, 1998] Gomes, S. B. V. and Ramos, J. J. G. (1998). Airship dynamic modeling for autonomous operation. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 3462–3476.

[Gorinevsky et al., 1996] Gorinevsky, D., Kapitanovsky, A., and Goldenberg, A. (1996). Neural network architeture for trajectory generation and control of automated car paking. *IEEE Transactions on Control Systems Technology*, 4(1).

[Harris and Billings, 1995] Harris, C. J. and Billings, S. A. (1995). *Self-Tuning and Adaptative Control: Theory and Aplications*. Peter Peregrinus.

[Hemerly, 1998] Hemerly, E. M. (1998). Controle em coordenadas polares de robôs móveis com rodas. In *Proceedings of XII Brazilian Automatic Control Conference*, pages 605–610, Uberlândia, MG, Brazil.

[Hong et al., 1997] Hong, S. G., Eom, T. D., Lee, C. Y., Kim, M.-S., Sugisaka, M., and Lee, J. J. (1997). Designing soccer-playing robot team (f.b.i.) based on the centralized approach. In *Proceedings of the Micro-Robot World Cup Soccer Tournament*, pages 119–123, Taejon, Korea.

[Horn, 1986] Horn, B. K. P. (1986). *Robot Vision*. The MIT Press.

[Jagannathan et al., 1994] Jagannathan, S., Zhu, S. Q., and Lewis, F. L. (1994). Path planning and control of a mobile base with nonholonomic constraints. *Robotica*, 12:529–539.

[Jiang and enk Nijmeijer, 1997] Jiang, Z.-P. and enk Nijmeijer (1997). Tracking control of mobile robots: A case study in backstepping. *Automática*, 33:1393–1399.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, pages 35–45.

[Kang et al., 1999] Kang, W., Xi, N., and Tan, J. (1999). Analysis and design of non-time based motion controller for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 2964–2969, Detroit, Michigan.

[Kerningan and Ritchie, 1978] Kerningan, B. W. and Ritchie, D. M. (1978). *The C Programming Language*. Prentice-Hall.

[Kim et al., 1996] Kim, D.-Y., Yoo, J.-R., Park, H.-K., Lee, Y.-J., and Chung, M. J. (1996). Development of multiple mobile robots playing soccer. In *1996 Micro-Robot World Cup Soccer Tournament Proceedings*, pages 83–86.

[Kim et al., 1998] Kim, J.-H., Kim, K.-C., Kim, D.-H., Kim, Y.-J., and Vadakkepat, P. (1998). Path planning and role selection mechanism for soccer robots. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 3216–3221, Leuven, Belgium.

[Kim et al., 1997] Kim, J.-H., Shim, H.-S., Kim, H.-S., Jung, M.-J., Choi, I.-H., and Kim, J.-O. (1997). A cooperative multi-agent system and its real time application to robot soccer. In *Proceedings of the 1997 IEEE*

International Conference on Robotics & Automation, pages 638–643, Albuquerque.

[Kim et al., 1999] Kim, S. H., Choi, J. S., and Byung (1999). Development of BEST nano-robot. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 2680–2685, Detroit, Michigan.

[Kitano et al., 1997] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., and Matsubara, H. (1997). RoboCup – a challenge problem for AI. *AI Magazine*, 18(1):73–85.

[Kobayashi et al., 1994] Kobayashi, K., Watanabe, K., and Munekata, F. (1994). Accurate navigation via differential gps and local sensors. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 9–16.

[Kozlowski and Dutkiewicz, 1996] Kozlowski, K. R. and Dutkiewicz, P. (1996). Experimental identification of parameters for a class of geared robots. *Robotica*, 14(5):561–574.

[Kuo, 1995] Kuo, B. C. (1995). *Automatic Control Systems*. Prentice Hall, 7th edition.

[la Rosa et al., 1999] la Rosa, J. L. D., Garcia, R., Innocenti, B., Munoz, I., Figueras, A., and Ramon, J. A. (1999). Rogi team description. In *RoboCup-99 Team Descriptions*, pages 64–71.

[Lafferriere and Sussmann, 1991] Lafferriere, G. and Sussmann, H. J. (1991). Motion planning for controlable systems without drift. In *Proceedings of the 1991 IEEE International Conference on Robotics & Automation*, pages 1148–1153, Sacramento, California.

[Lee and Bautista, 1998] Lee, S. and Bautista, J. (1998). Motion control for micro-robots playing soccer games. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 2599–2604, Leuven, Belgium.

[Lee et al., 1999] Lee, T.-C., Song, K.-T., Ching-Hung, and Teng, L. C.-C. (1999). Tracking control of mobile robots using saturation feedback controller. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 2639–2644, Detroit, Michigan.

[Leontaritis and Billings, 1985] Leontaritis, I. J. and Billings, S. A. (1985). Input-output parametric models for nonlinear systems part I: deterministic nonlinear systems. *International Journal of Control*, 41(2):303–328.

[Lizarralde and Wen, 1996] Lizarralde, F. and Wen, J. T. (1996). Attitude control without angular velocity measurement: A passivity approach. *IEEE Transactions on Automatic Control*, 41:468–472.

[Ljung, 1987] Ljung, L. (1987). *System Identification – Theory for the User*. Prentice Hall, New Jersey.

[Luenberger, 1971] Luenberger, D. G. (1971). An introduction to observers. *IEEE Transactions on Automatic Control*, 10(6):596–602.

[Ma et al., 1999] Ma, Y., Kosecká, J., and Sastry, S. S. (1999). Vision guided navigation for a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*, 15(3):521–536.

[MATLAB, 1992] MATLAB (1992). *The student version of MATLAB*. Englewood Cliffs NJ: Prentice-Hall, Inc.

[McKerow, 1998] McKerow, P. (1998). *Introduction to Robotics*. Addison Wesley.

[M'Closkey and Murray, 1997] M'Closkey, R. T. and Murray, R. M. (1997). Exponential stabilization of driftless nolinear control systems using homogeneous feedback. *IEEE Transactions on Automatic Control*, 42(5):614–628.

[McMillan, 1994] McMillan, G. K. (1994). *Tuning and Control Loop Performance: A Practitioner's Guide*. Instrument Society of America, 3th edition.

[Microchip, 1996] Microchip (1996). PIC16/17 microcontroller data book.

[Murray and Sastry, 1993] Murray, R. M. and Sastry, S. S. (1993). Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700–716.

[Normey-Rico et al., 1998] Normey-Rico, J. E., Ortega, J. G., and Alcalá, I. (1998). Control predictivo para sguimiento de cominos en un robot del tipo "synchro-drive". In *Proceedings of XII Brazilian Automatic Control Conference*, pages 611–616, Uberlândia, MG, Brazil.

[Ogata, 1993] Ogata, K. (1993). *Engenharia de Controle Moderno*. Prentice Hall do Brasil, 2th edition.

[Olsen and Bekey, 1986] Olsen, H. B. and Bekey, G. A. (1986). Identification of robot dynamics. In *Proceedings of the 1986 IEEE International Conference on Robotics & Automation*, pages 1004–1010, San Francisco.

[Ozaki et al., 1991] Ozaki, T., Suzuki, T., Furuhashi, T., Okuma, S., and Uchikawa, Y. (1991). Trajectory control of robotic manipulators using neural networks. *IEEE Transactions on Industrial Electronics*, 38(3):195–202.

[Park et al., 1999] Park, K., Chung, H., and Leel, J. G. (1999). Point stabilization of mobile robots via state space exact feedback linearization. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 2626–2631, Detroit, Michigan.

[Philips and Nagle, 1995] Philips, C. L. and Nagle, H. T. (1995). *Digital Control System Analysis and Design*. Prentice Hall, 3th edition.

[Samson, 1995] Samson, C. (1995). Control of chained systems application to path following and time-varying point-stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40(1):64–77.

[Sargent et al., 1997] Sargent, R., Bailey, B., Witty, C., and Wright, A. (1997). Fast visual tracking and coordinated control for soccer-playing robots. In *Proceedings of the Micro-Robot World Cup Soccer Tournament*, pages 59–65, Taejon, Korea.

[Stallings, 1996] Stallings, W. (1996). *Data and Computer Communications*. Prentice Hall, 5th edition.

[Stemmer et al., 1997] Stemmer, M. R., Pieri, E. R. D., and Borges, F. A. P. (1997). Comparação de performance entre controladores clássicos e um controlador torque computado neural aplicados ao robô PUMA 560. In *Anais do III Congresso Brasileiro de Redes Neurais*, pages 390–395, Florianópolis.

[Stroustrup, 1986] Stroustrup, B. (1986). *The C++ Programming Language*. Addison-Wesley.

[Tavares Filho, 1997a] Tavares Filho, R. F. (1997a). Controlando um transmissor RC através de um computador. Technical report, Instituto de Automação - Centro Tecnológico para Informática.

[Tavares Filho, 1997b] Tavares Filho, R. F. (1997b). Convertendo um servo motor RC em um dispositivo de tração para micro-robôs. Technical report, Instituto de Automação - Centro Tecnológico para Informática.

[Teixeira et al., 1998] Teixeira, R. A., Braga, A. P., and Menezes, B. R. (1998). Controle de um manipulador robótico Puma 560 utilizando redes neurais artificiais com adaptação em tempo real. In *V Simpósio Brasileiro de Redes Neurais*, pages 146–151.

[van den Bosch and van der Klauw, 1994] van den Bosch, P. P. J. and van der Klauw, A. C. (1994). *Modeling, Identification and Simulation of Dynamical Systems*. CRC Press, 1th edition.

[Veloso et al., 1999] Veloso, M., Bowlibg, M., and Achin, S. (1999). The CMUnited-99 small robot team. In *RoboCup-99 Team Descriptions*, pages 24–32.

[Walsh et al., 1994] Walsh, G., Tilbury, D., Sastry, S., Murray, R., and Laumond, J. P. (1994). Stabilization of trajectories for systems. *IEEE Transactions on Automatic Control*, 39(1):216–222.

[Wang et al., 1994] Wang, Y., Linnett, J. A., and Roberts, J. (1994). Kinematics, kinematics contraints and path planning for wheeled mobile robots. *Robotica*, 12:391–400.

[Wen, 1995] Wen, J. T. (1995). Control of nonholonomic systems. In Levine, W., editor, *Control Handbook*. CRC Press.

[Werger, 1998] Werger, B. (1998). Principles of minimal control for comprehensive team behavior. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 3504–3509, Leuven, Belgium.

[Yang et al., 1998] Yang, J.-M., Choi, I.-H., and Kim, J.-H. (1998). Sliding mode control of a nonholonomic wheeled mobile robot for trajectory tracking. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 2983–2988, Leuven, Belgium.

[Yang and Kim, 1999] Yang, J.-M. and Kim, J.-H. (1999). Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 15(3):578–587.

[Zhang and Ostrowski, 1999] Zhang, H. and Ostrowski, J. P. (1999). Visual servoing with dynamics: Control of an unmanned blimp. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 618 – 623, Detroit, Michigan.

Apêndice A

Simulador para o MATLAB

> Até onde as leis da matemática se referem à realidade, elas não estão certas; e até onde elas estão certas, não se referem à realidade.
>
> *Albert Einstein (1879–1955)*

BASEADO nas características físicas do sistema e no modelo encontrado no Capítulo 4 foi desenvolvido um simulador para o MATLAB [MATLAB, 1992]. Este simulador tem objetivo de reproduzir o comportamento do sistema real no computador para evitar que todos os testes sejam feitos no sistema real. Apesar do sistema ser um plataforma para futebol de robôs, houve uma preocupação de se reproduzir principalmente o comportamento dinâmico e cinemático dos agentes, sendo que outras variáveis como choques com a bola e com a parede, foram deixados para uma segunda etapa.

A.1 Robô

Para reproduzir o comportamento do robô foi usado o modelo encontrado com a identificação em batelada no Capítulo 4. Assim, o modelo dinâmico

do robô foi representado pelo seguinte conjunto de equações:

$$\begin{aligned} x_k &= 1,6824\, x_{k-1} - 0,6824\, x_{k-2} + [0,0441\, u_{1\,k-8} + 0,0403\, u_{2\,k-8}] \cos(\theta_{k-1}) \\ y_k &= 1,7150\, y_{k-1} - 0,7152\, y_{k-2} + [0,0336\, u_{1\,k-8} + 0,0318\, u_{2\,k-8}] \sin(\theta_{k-1}) \\ \theta_k &= 1,5640\, \theta_{k-1} - 0,5640\, \theta_{k-2} - 0,0063\, u_{1\,k-8} + 0,0065\, u_{2\,k-8} \ . \end{aligned} \tag{A.1}$$

É importante ressaltar que os sinais de controle u_1 e u_2 têm as mesmas características do sinal real ou seja são números inteiros de 8 bits.

Além do modelo dinâmico, o simulador permite que o robô possa ser desenhado a cada passo de cálculo do modelo e é representado por um quadrado de 15 pixels de lado cujo centro e a frente aparecem em destaque.

A.2 Visão

Como foi mostrado na Seção 4.1.2 o ruído fornecido pelo sistema de visão tem características especiais devido principalmente à resolução do sinal de vídeo. Para reproduzir os sinais de saída do sistema de visão foram usados as seguintes linhas do MATLAB:

```
xm=ceil(x+rand-1);
ym=ceil(y+rand-1);
tetam=teta+0.095*(rand*2-1);
```

onde `xm,ym` e `tetam` são as variáveis medidas pelo sistema de visão simulado e `x,y` e `teta` são as variáveis de saída do modelo dinâmico do robô.

O ruído em x e y são calculados da mesma forma. A função `ceil()` arredonda para um número inteiro acima e a função `rand` retorna um número aleatório entre 0 e 1. Para θ, o número $0,095$ é usado para ajustar a variância do ruído. A Figura A.1 mostra a saída do sistema de visão simulado para

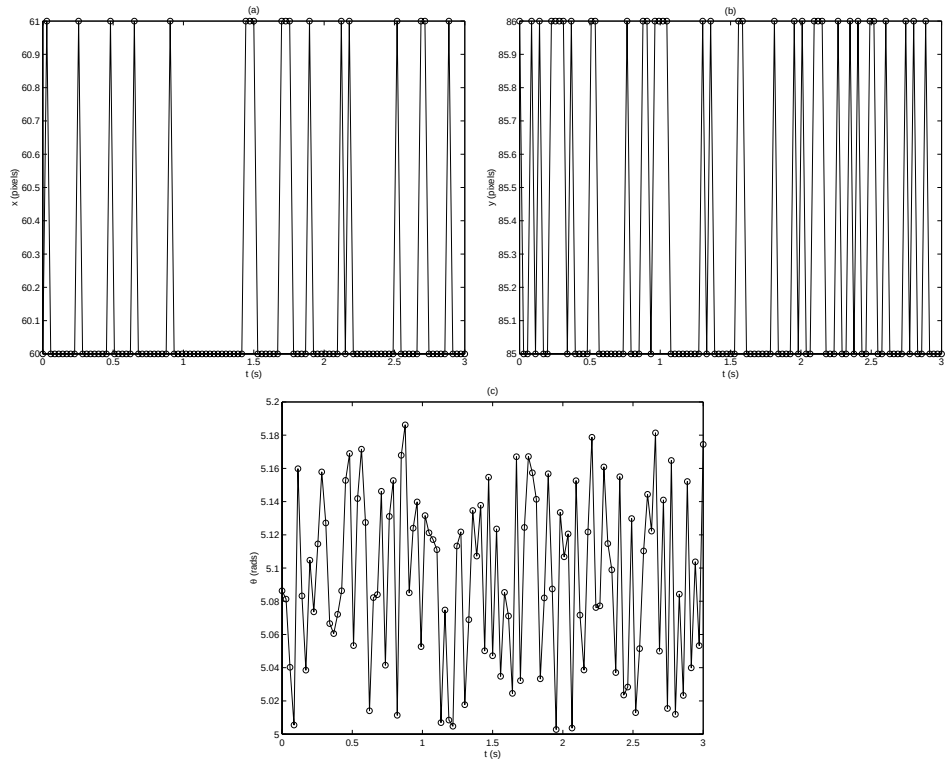

Figura A.1: Simulador do sistema de visão: (a) – coordenada x; (b) – coordenada y; (c) – orientação.

o robô parado. Comparando com a Figura 4.5 vê-se que com exceção da variável θ existe muita semelhança entre o sinais do simulador e os sinais reais. Apesar do ruído de θ não ter a mesma forma do ruído real preocupou-se em garantir que os dois tivessem variâncias parecidas.

O Campo foi desenhado nas dimensões máximas possíveis para o sistema visão, ou seja 352×240 pixels. As paredes só têm a função de parar o robô caso haja uma colisão, não tendo sido considerada nenhuma função de reflexão ou absorção de impacto. A Figura A.2 mostra a visão do campo com um robô percorrendo uma trajetória qualquer.

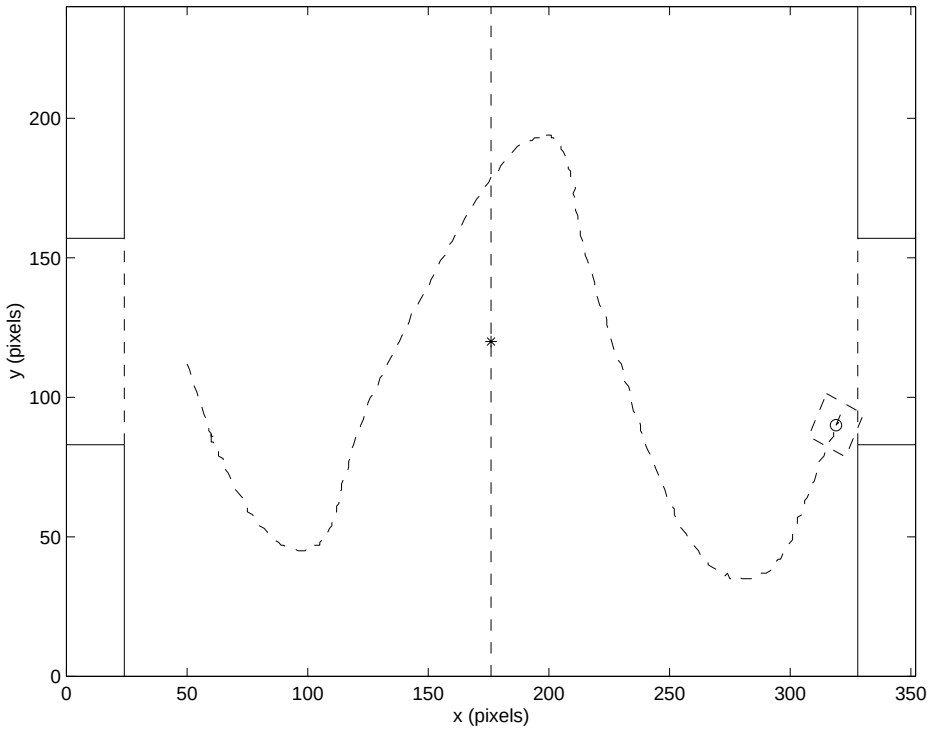

Figura A.2: Desenho do campo no simulador.

Apêndice B

Estimador estendido de mínimos quadrados

> Se o erro é corrigido assim que ele for reconhecido, o caminho do erro é o caminho da verdade.
>
> *Hans Reichenbach (1891–1953)*

O OBJETIVO deste apêndice é descrever os algoritmos estendidos de mínimos quadrados (EMQ) tradicional e recursivo, sem se preocupar entretanto em provar o seu funcionamento. A principal diferença entre os métodos, é que o primeiro se utiliza de um conjunto de dados coletados *a priori* e no segundo a estimação é feita a medida que os dados são coletados. A dedução completa do método e dos algoritmos pode ser encontrada em [Ljung, 1987, van den Bosch and van der Klauw, 1994, Aguirre, 1999].

A principal característica do EMQ é a tentativa de se modelar o ruído como um processo média móvel (MA) [Ljung, 1987] para evitar que haja polarização dos parâmetros. Por este motivo, este estimador utiliza um modelo do tipo ARMAX (*Auto-Regressive Moving Average with eXogenous inputs*) que pode ser obtido com a adição do termo MA a um modelo ARX. Como não é possível medir o ruído do processo, o método usa o resíduo de predição do modelo como estimativa deste ruído.

B.1 Estimação em batelada

Quando um conjunto de dados é coletados antes do inicio do processo de estimação de parâmetros, este é conhecido como estimação em batelada. Desta forma, apesar do algoritmo EMQ ser um algoritmo executado em batelada ele é também iterativo, significando que a estimação de parâmetros é feita em mais de um passo de cálculo. Este algoritmo pode ser descrito da seguinte forma:

1. Monte a equação matricial $\mathbf{y} = \Psi\boldsymbol{\theta} + \mathbf{e}$, onde $\mathbf{y}$ é o vetor contendo as saídas do sistema a cada instante de amostragem, Ψ é a matriz de regressores (ver [Aguirre, 1999]), $\boldsymbol{\theta}$ é o vetor de parâmetros e $\mathbf{e}$ é o vetor de erros;

2. Determine os parâmetros como no método de mínimos quadrados (MQ): $\hat{\theta}_{MQ} = [\Psi'\Psi]^{-1}\Psi'\mathbf{y}$, onde a aspa significa transposição;

3. Calcule o vetor de resíduos $\boldsymbol{\xi}_1 = \mathbf{y} - \Psi\hat{\boldsymbol{\theta}}_{MQ}$;

4. Faça $i = 2$ (i indica o número da iteração);

5. Inclua a coluna ξ_{i-1} a ψ, e monte a matriz estendida de regressores, Ψ_i^*, e estime $\hat{\boldsymbol{\theta}}_{EMQ} = [\Psi_i^{*\prime}\Psi_i^*]^{-1}\Psi_i^{*\prime}\mathbf{y}$. Se estiver na segunda iteração, Ψ_i^* é montada a partir de Ψ_{i-1}^*;

6. Calcule o vetor de resíduos $\boldsymbol{\xi}_i = \mathbf{y} - \Psi_i^*\hat{\boldsymbol{\theta}}_{EMQ}$;

7. Faça $i = i + 1$ e volte ao passo 5. Repita até convergir.

Para verificar a convergência, pode-se monitorar a variância dos resíduos ou o vetor de parâmetros.

B.2 Estimação recursiva

Ao contrário da estimação em batelada, a estimação recursiva é feita em geral em tempo real, a medida que os dados vão sendo coletados. Uma forma de se implementar o EMQ de maneira recursiva é:

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_{k-1}\boldsymbol{\psi}_k \left[\boldsymbol{\psi}_k{}'\mathbf{P}_{k-1}\boldsymbol{\psi}_k + \lambda\right]^{-1} \\
\hat{\boldsymbol{\theta}}_k &= \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{K}_k \left[y_k - \boldsymbol{\psi}_k{}'\hat{\boldsymbol{\theta}}_{k-1}\right] \\
\mathbf{P}_k &= \frac{\mathbf{P}_{k-1} - \mathbf{K}_{k-1}\boldsymbol{\psi}_k{}'\mathbf{P}_{k-1}}{\lambda} \\
\boldsymbol{\xi}_k &= y_k - \psi_k'\hat{\boldsymbol{\theta}}_k
\end{aligned}$$

onde $\mathbf{P}$ é a matriz de covariância estimada do vetor de parâmetros, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{\psi}$ é o vetor de regressores e λ é conhecido como fator de esquecimento. Este fator que na prática recebe valores na faixa $0,950 \leq \lambda \leq 0,999$ é usado para dar pesos diferentes aos valores medidos durante a estimação em relação ao momento em que estes foram adquiridos. Se $\lambda = 1$ todos os valores passados têm o mesmo peso. Note que na primeira iteração, $\boldsymbol{\psi}_k$ não contém resíduos. A última equação do estimador é utilizada para calcular o resíduo na iteração k e com ele atualizar o vetor de regressores.