Campus de São Carlos

CONTROLE H_∞ NÃO LINEAR DE ROBÔS MÓVEIS
COM RODAS

Autor: Gilson Antonio dos Reis

Orientador: Prof. Dr. Marco Henrique Terra

UNIVERSIDADE DE SÃO PAULO

**ESCOLA DE ENGENHARIA
DE SÃO CARLOS**

Gilson Antonio dos Reis

CONTROLE $\mathcal{H}_\infty$ NÃO LINEAR DE ROBÔS MÓVEIS COM RODAS

Dissertação apresentada à Es-
cola de Engenharia de São Carlos
da Universidade de São Paulo,
como parte dos requisitos para
obtenção do título de Mestre em
Engenharia Elétrica.

Orientador: Prof. Dr. Marco Henrique Terra

Serviço de Pós-Graduação EESC/USP

EXEMPLAR REVISADO

Data de entrada no Serviço... 23 / 08 / 2006

Ass.: *Denise*

São Carlos
2005

DEDALUS - Acervo - EESC

31100104314

Class.	TESE ✓
Cutt.	1199
Tombo	T185/06
Sysno	154 3517

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

R375c Reis, Gilson Antonio dos
Controle H_∞ não linear de robôs móveis com rodas /
Gilson Antonio dos Reis. -- São Carlos, 2005.

Dissertação (Mestrado) -- Escola de Engenharia de São
Carlos-Universidade de São Paulo, 2005.
Área: Engenharia Elétrica.
Orientador: Prof. Dr. Marco Henrique Terra.

1. Controle H_∞ não linear. 2. Robôs móveis com rodas.
I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **GILSON ANTONIO DOS REIS**

Dissertação defendida e julgada em 19-08-2005 perante a Comissão Julgadora:

Prof. Associado **MARCO HENRIQUE TERRA (Orientador)**
(Escola de Engenharia de São Carlos/USP)

Aprovado.

Dr. **ADRIANO ALMEIDA GONÇALVES SIQUEIRA**
(Pesquisador Autônomo)

Aprovado

Prof. Dr. **RAMON ROMANKEVICIUS COSTA**
(Universidade Federal do Rio de Janeiro/UFRJ)

Aprovado.

Prof. Associado **HOMERO SCHIABEL**
Coordenador do Programa de Pós-Graduação em Engenharia
Elétrica

Profa. Titular **MARIA DO CARMO CALIJURI**
Presidente da Comissão de Pós-Graduação

Dedicatória

Em memória de minha mãe, Florisbela,
a pessoa que mais me incentivou em meus estudos.
Dura perda ocorrida durante a minha graduação.

Agradecimentos

Ao poder superior, Deus, que nunca me desamparou.

Ao meu pai, Antonio, que sempre me ajudou em todos os sentidos para alcançar meus objetivos.

Aos meus irmãos, Milton, Nivaldo e Rosirene que me apoiaram nas oportunidades de estudo e deram-me forças nos momentos mais difíceis.

Ao Prof. Dr. Marco Henrique Terra pela orientação e pelas contribuições para a realização deste trabalho.

Ao Prof. Dr. Adriano Almeida Gonçalves Siqueira pelas contribuições e por estar sempre disposto a tirar minhas dúvidas.

Aos Profs. Drs. Adilson, Luís Fernando, Murilo e Vilma pelas disciplinas ministradas, importantes à minha formação.

Aos colegas de laboratório, Aline, Antonio, Arthur, Cleber, Elmer, Everaldo, Gildson, José Carlos, Karla, Márcio, Natanael, Rafael, Tatiana, Thiago e Victor pela paciência, descontrações e convivência tranqüila.

Às secretárias Denise e Marisa pela disposição em ajudar no que fosse possível.

Aos funcionários do departamento de Engenharia Elétrica, em especial, ao técnico Rui Bertho por ter construído os suportes e protótipos necessários para serem usados nos experimentos práticos.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro durante a realização deste trabalho.

Resumo

REIS, G. A. (2005). *Controle $\mathcal{H}_\infty$ não linear de robôs móveis com rodas*. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2005.

Este trabalho apresenta o projeto de dois controladores robustos, baseados no critério $\mathcal{H}_\infty$ não linear, para o acompanhamento de trajetória de Robôs Móveis com Rodas (RMRs). Estes controladores estabilizam o sistema em malha fechada e garantem que a norma $\mathcal{L}_2$ induzida entre os sinais de entrada (distúrbios) e saída seja limitada por um nível de atenuação $\gamma > 0$. Para o projeto, as equações dinâmicas não lineares do robô são descritas na forma quase Linear a Parâmetros Variantes (quase-LPV), sendo os parâmetros parte do estado. Os controladores são resolvidos via Desigualdades Matriciais Lineares (DMLs) e Equações Algébricas de Riccati (EAR). Resultados em simulação com um estudo comparativo entre essas duas estratégias de controle e um controlador Proporcional Derivativo (PD) em conjunto com um controlador do tipo torque calculado são apresentados. Além disso, a implementação de dois métodos de localização de RMRs através de imagens é realizada.

Palavras-chave: controle $\mathcal{H}_\infty$ não linear; robôs móveis com rodas.

Abstract

REIS, G. A. (2005). Nonlinear $\mathcal{H}_\infty$ control of Wheeled Mobile Robots. M.Sc. Dissertation - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2005.

This work presents the design of two robust controllers, based on nonlinear $\mathcal{H}_\infty$ approach, for tracking trajectory of Wheeled Mobile Robots (WMRs). These controllers stabilize the close-loop system and guarantee that induced $\mathcal{L}_2$ norm between input (disturbances) and output signals be bounded by an attenuation level $\gamma > 0$. For the design, the nonlinear dynamic equations of the robot are described in quasi-Linear Parameter Varying (quasi-LPV) form being the parameters part of the states. The controllers are solved via Linear Matrix Inequalities (LMIs) and Algebraic Riccati Equation (ARE). Simulation results with a comparison study among these two control strategies and a Proportional-Derivative (PD) controller plus calculated torque are presented. Moreover, implementation of two methods of localization of WMRs based on images is accomplished.

Keywords: nonlinear $\mathcal{H}_\infty$ control; wheeled mobile robots.

Publicação

1- REIS, G. A.; SIQUEIRA A. A. G.; TERRA M. H.(2005). Nonlinear $\mathcal{H}_\infty$ Control via quasi-LPV Representation and Game Theory for Wheeled Mobile Robots. In: *Proc. IEEE Mediterranean Conference on Control and Automation*, 13. June 2005, Limassol, Cyprus.

Lista de Figuras

FIGURA 2.1	Robô móvel.	10
FIGURA 2.2	Velocidades do robô móvel.	11
FIGURA 3.1	Representação do erro de postura do robô móvel.	20
FIGURA 5.1	Convenção: (a) imagem contínua, (b) imagem digital. . . .	33
FIGURA 5.2	Transformada de Hough Circular: (a) imagem, (b) matriz acumuladora.	36
FIGURA 6.1	Distúrbios de torque aplicados às rodas.	39
FIGURA 6.2	Sistema de controle de acompanhamento de trajetória para Robôs Móveis.	39
FIGURA 6.3	Acompanhamento de trajetória de referência usando o con- trolador quase-LPV: (a) sem distúrbio (b) com distúrbio.	41
FIGURA 6.4	Erros de posição usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.	41
FIGURA 6.5	Erro de direção usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.	42
FIGURA 6.6	Derivada dos erros de posição usando o controlador quase- LPV: (a) sem distúrbio (b) com distúrbio.	42
FIGURA 6.7	Derivada do erro de direção usando o controlador quase- LPV: (a) sem distúrbio (b) com distúrbio.	42

FIGURA 6.8 Velocidade angular da roda direita usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.	43
FIGURA 6.9 Velocidade angular da roda esquerda usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.	43
FIGURA 6.10 Torques τ_d e τ_e aplicados às rodas direita e esquerda, respectivamente, usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.	43
FIGURA 6.11 Acompanhamento de trajetória de referência usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	44
FIGURA 6.12 Erros de posição usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	44
FIGURA 6.13 Erro de direção usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	45
FIGURA 6.14 Derivada dos erros de posição usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	45
FIGURA 6.15 Derivada do erro de direção usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	45
FIGURA 6.16 Velocidade angular da roda direita usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	46
FIGURA 6.17 Velocidade angular da roda esquerda usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	46
FIGURA 6.18 Torques τ_d e τ_e aplicados às rodas direita e esquerda, respectivamente, usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.	46
FIGURA 6.19 Acompanhamento de trajetória de referência usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.	47
FIGURA 6.20 Erros de posição usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.	48

FIGURA 6.21 Erro de direção usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.	48
FIGURA 6.22 Derivada dos erros de posição usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.	48
FIGURA 6.23 Derivada do erro de direção usando o controlador PD: (a) sem distúrbio (b) com distúrbio.	49
FIGURA 6.24 Velocidades angulares da roda direita usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.	49
FIGURA 6.25 Velocidades angulares da roda esquerda usando o controla- dor PD + TC: (a) sem distúrbio (b) com distúrbio.	49
FIGURA 6.26 Torques τ_d e τ_e aplicados às rodas direita e esquerda, res- pectivamente, usando o controlador PD +TC: (a) sem distúrbio (b) com distúrbio.	50
FIGURA 6.27 Protótipo do campo de atuação.	52
FIGURA 6.28 Protótipo do robô para o método dos momentos.	53
FIGURA 6.29 Imagem do protótipo do robô na plataforma para o método dos Momentos. Os ruídos foram editados para efeito de ilustração. . .	54
FIGURA 6.30 Protótipo do robô para a THC.	55
FIGURA 6.31 Imagem do protótipo do robô na plataforma para o método da THC. Os ruídos foram editados para efeito de ilustração.	56
FIGURA A.1 Ambiente gráfico. O robô é o triângulo em verde, o ponto azul é a posição do robô e em vermelho a trajetória de referência. . .	73
FIGURA A.2 Formação da máscara. (a) imagem real; (b) círculo de raio R; (c) máscara.	80

Lista de Tabelas

TABELA 6.1	Índices de desempenho.	51
TABELA 6.2	Tempo de processamento e postura do protótipo do robô usando o método dos momentos.	54
TABELA 6.3	Tempo de processamento e postura do protótipo do robô usando a THC.	55

Lista de Algoritmos

ALGORITMO 4.1	Seqüência de passos para resolver (4.20), sendo que a sobrescrito significa iteração atual	32
ALGORITMO 5.1	Seqüência de passos para encontrar centros de círculos em imagem usando o método dos Momentos e determinar a postura de um RMR	35
ALGORITMO 5.2	Seqüência de passos para encontrar centros de circunferências em imagem usando a THC e determinar a postura de um RMR	37

Lista de Abreviaturas e Siglas

DMLs	Desigualdades Matriciais Lineares
LPV	Linear a Parâmetros Variantes
quase-LPV	quase Linear a Parâmetros Variantes
PD	Controlador Proporcional Derivativo
TC	Torque Calculado
TJ	Teoria dos Jogos
RMR	Robô Móvel com Rodas
FLDP	Função de Lyapunov Dependente de Parâmetros
ANSI	American National Standardization Institute
OpenGL	Open Graphics Library

Lista de Símbolos

a	Comprimento do robô
b	Distância entre uma roda atuada e o eixo de simetria
r	Raio das rodas atuadas
θ_d, θ_e	Deslocamentos angulares das rodas direita e esquerda
P_c	Centro de massa da plataforma do robô
P_o	Ponto central entre as rodas atuadas, no eixo de simetria
P_r	Ponto de referência
d	Distância entre P_o e P_c
v_o	Velocidade linear do robô em P_o
v_c	Velocidade linear do robô em P_c
(X, Y)	Sistema de coordenadas inercial
(X_o, Y_o)	Sistema de coordenadas local
α	Ângulo (direção do robô) entre o eixo X e o eixo de simetria do robô no sentido anti-horário
α_r	Direção de referência
α_e	Erro direção
x_c, y_c	Posição de P_c em relação ao sistema (X, Y)
x_o, y_o	Posição de P_o em relação ao sistema (X, Y)
x_r, y_r	Posição de P_r em relação ao sistema (X, Y)
x_d, y_d	Posição do centro de massa da roda direita relacionada ao sistema (X, Y)
x_e, y_e	Erros de posição em relação ao sistema (X_o, Y_o)
$\dot{\alpha} = \omega$	Velocidade angular do robô

$q_1 = [x_c \ y_c \ \alpha \ \theta_d \ \theta_e]^T$	Vetor de coordenadas generalizadas
$q_2 = [\theta_d \ \theta_e]^T$	Vetor de coordenadas generalizadas
q_e	Vetor de erro de postura
q_a	Vetor de postura atual
q_r	Vetor de postura de referência
L	Lagrangiano
λ	Multiplicador de Lagrange
λ_{max}	Máximo autovalor
T	Energia cinética
V	Energia potencial
$M(q_1)$	Matriz de Inércia
m	Massa total do robô
m_P	Massa da plataforma do robô
m_r	Massa de cada roda atuada incluindo a massa do rotor do motor
I_c	Momento de inércia da plataforma em relação ao eixo vertical em P_c
I_r	Momento de inércia de cada roda com o rotor do motor em relação ao eixo da roda
I_m	Momento de inércia em relação ao eixo definido no plano da roda (perpendicular ao eixo das rodas)
$C(q_1, \dot{q}_1)$	Matriz de forças de coriolis e centrípetas
τ_d, τ_e	Torque da roda direita e esquerda
J	Função custo
v^d, ω^d	Velocidades linear e angular desejadas para o robô
v_r, ω_r	Velocidades linear e angular de referência
K_x, K_y, K_α	Ganhos para o controlador cinemático
V	Função de Lyapunov
$\dot{\theta}_d^d, \dot{\theta}_e^d$	Velocidades angulares desejadas para as rodas
z	Saídas controladas
w, δ	Distúrbios
x	Estado

$\ T_{zw}\ _\infty$	Norma $\mathcal{H}_\infty$ da função de transferência entre o distúrbio w e a saída z
ρ	Parâmetros variantes
P	Conjunto de parâmetros ρ
F_P^ν	Conjunto de variação dos parâmetros
$F(\rho)$	Ganho de realimentação de estado dependente do parâmetro ρ
N	Número de pontos do conjunto de parâmetros
γ	Nível de atenuação
$\tilde{x}$	Vetor do erro de estado
u	Entrada de controle
T_o	Matrix de transformação de estado
R, Q	Matrizes de ponderação
$K_{p1}, K_{p2}, K_{d1}, K_{d2}$	Ganhos para o controlador PD
$e = [q_e \dot{q}_e]^T$	Vetor dos erros de posição e velocidade
$\mathcal{L}_2[e]$	Norma $\mathcal{L}_2$ dos erros
$E[\tau]$	Somatório das áreas dos torques
C^1	Funções continuamente diferenciáveis
C^0	Funções contínuas

Sumário

Resumo	iii
Abstract	iv
Publicação	v
Lista de Figuras	vi
Lista de Tabelas	ix
Lista de Algoritmos	x
Lista de Abreviaturas e Siglas	xi
Lista de Símbolos	xii
1 Introdução	1
1.1 Motivação	1
1.2 Revisão Bibliográfica	1
1.2.1 Robôs móveis com rodas	1
1.2.2 Controle $\mathcal{H}_\infty$ para robôs	5
1.2.3 Localização usando visão computacional	7
1.3 Objetivos	8
1.4 Originalidades do trabalho	8
1.5 Disposição dos capítulos	9
2 Modelagem do Robô Móvel	10
2.1 Modelo Cinemático	11
2.2 Modelo Dinâmico	16

3	Controlador Baseado na Cinemática	20
4	Controladores Baseados no Modelo Dinâmico	22
4.1	Controle $\mathcal{H}_\infty$ não linear via representação quase-LPV	22
4.1.1	Ganho $\mathcal{L}_2$ para sistemas não lineares variantes no tempo . . .	23
4.1.2	Síntese do controle $\mathcal{H}_\infty$ para sistemas LPV por realimentação do estado	23
4.1.3	Considerações computacionais	25
4.1.4	Representação quase-LPV do modelo dinâmico do robô . . .	26
4.2	Controle $\mathcal{H}_\infty$ não linear via Teoria dos Jogos	28
5	Determinação da Postura de RMRs	33
5.1	Método usando Momentos	34
5.2	Método usando Transformada de Hough Circular	35
6	Resultados	38
6.1	Simulação computacional dos controladores	38
6.1.1	Controle via representação quase-LPV	40
6.1.2	Controle via Teoria dos Jogos (TJ)	44
6.1.3	Controle Proporcional Derivativo e Torque Calculado (PD+TC)	47
6.1.4	Estudo comparativo	50
6.2	Implementação de localização por visão computacional	51
6.2.1	Método dos Momentos	53
6.2.2	Método da Transformada de Hough Circular	55
7	Conclusão e Trabalho Futuro	57
	Referências Bibliográficas	59
A	Programas Computacionais	65
A.1	Controladores	65
A.2	Ambiente gráfico	70
A.3	Captura de imagens	74
A.4	Localização utilizando visão computacional	76
A.4.1	Método dos Momentos	76

A.4.2	Transformada de Hough Circular	78
-------	--	----

Capítulo 1

Introdução

1.1 Motivação

Considerando que robôs móveis podem estar sujeitos a perturbações externas, como desníveis da superfície ou colisão com obstáculos ou outros robôs, o controle utilizado tem que ser robusto o suficiente para a realização de uma tarefa pré-estabelecida. Na busca de tal controle optou-se por trabalhar com o modelo dinâmico do robô na representação quase Linear a Parâmetros Variantes (quase-LPV), pois não foi encontrada nenhuma aplicação desta técnica nos trabalhos relacionados a Robôs Móveis com Rodas (RMRs) e ela proporciona uma melhor representação que preserva a natureza dinâmica do sistema. Para a atenuação dos distúrbios externos e incertezas paramétricas o critério $\mathcal{H}_\infty$ tem sido utilizado para outros tipos de robôs e será aplicado em robôs móveis neste trabalho.

1.2 Revisão Bibliográfica

1.2.1 Robôs móveis com rodas

Controladores para RMRs têm sido alvo de pesquisas para muitos pesquisadores que atuam em robótica a partir dos anos 80. A expressão Robôs Móveis com Rodas será utilizada para diferenciar a categoria de robôs considerada neste trabalho de

outros tipos de robôs (aquáticos, aéreos, etc) e em boa parte do texto será utilizado apenas o termo Robôs Móveis.

Em [CAMPION et al. (1996)], os autores definiram dois tipos de rodas para RMRs: *rodas convencionais*, cuja velocidade no ponto de contato da roda com o solo é zero e são divididas em rodas fixas, centradas orientáveis e centradas não orientáveis (conhecida também por castor); e *rodas suecas* nas quais somente a componente da velocidade ao longo do movimento no ponto de contato da roda com o solo é admitida ser nula.

Os RMRs mais comuns estudados na literatura são: unicycle [AICARDI et al. (1995); MORIN E SAMSON (2000); LEE et al. (2001)] (este nome é devido a equação cinemática do robô ser equivalente ao de uma roda que não gira em falso e nem desliza no sentido do eixo), carro convencional [ALMEIDA et al. (1997)], carro convencional com *trailers* [VENDITTELLI E ORIOLO (2000); SAMSON (1995); JIANG E NIJMEIJER (1999)] e unicycle com *trailers* [M'CLOSKEY E MURRAY (1997)].

O robô móvel utilizado neste trabalho é um unicycle com duas rodas convencionais fixas atuadas independentemente e uma roda convencional tipo castor. Considera-se também que o centro de massa (P_c) é diferente do ponto no centro do eixo das rodas atuadas (P_o).

Equações matemáticas (modelos) para RMRs têm sido encontradas para cada tipo de robô. Modelos cinemáticos para cinco tipos de robôs são apresentados em [CAMPION et al. (1996)]. Em [COELHO (2001)], o autor apresenta uma modelagem completa (cinemática e dinâmica) para um robô unicycle, considerando três casos: no primeiro é considerado $P_c = P_o$, no segundo $P_c \neq P_o$ e no terceiro $P_c \neq P_o$ e incluindo as forças de restrições.

Robôs Móveis com Rodas constituem uma classe de sistemas mecânicos caracterizados por restrições não holonômicas. Para entender o que são restrições não holonômicas, considere que o sistema mecânico pode ser descrito por um vetor de coordenadas generalizadas de dimensão n

$$q = [q_1 \ q_2 \ \dots \ q_n]^T,$$

e a velocidade generalizada em um ponto genérico de uma trajetória suave $q(t)$ é o vetor tangente

$$\dot{q} = [\dot{q}_1 \quad \dot{q}_2 \quad \dots \quad \dot{q}_n]^T.$$

Um sistema mecânico pode estar sujeito a um conjunto de restrições cinemáticas, envolvendo as coordenadas generalizadas e suas derivadas, ou seja, k restrições de primeira ordem

$$\phi_i^T(q, \dot{q}) = 0,$$

sendo $i = \{1, 2, \dots, k\}$.

Na maioria dos casos as restrições são lineares nas velocidades e podem se denotadas por

$$A(q)\dot{q} = 0$$

sendo $A(q)$ a matriz jacobiana.

Se as restrições cinemáticas podem ser integráveis, isto é, existem k funções h_i tais que

$$\frac{dh_i(q(t))}{dt} = \frac{\partial h_i(q(t))}{\partial q} \dot{q} = 0, \quad i = \{1, 2, \dots, k\},$$

então, as restrições cinemáticas são de fato restrições geométricas. Portanto um conjunto de restrições cinemáticas é chamado holonômico se é integrável (limitação geométrica) e em caso contrário, é chamada de não holonômica (limitação cinemática).

Usando Álgebra de Lie, [COELHO E NUNES (2003)] mostraram como pode-se determinar quantas restrições cinemáticas não holonômicas estão presentes em um sistema de restrições.

Uma limitação na estabilidade de um ponto de equilíbrio de um sistema com restrições não holonômicas é que a estabilidade de Lyapunov não pode ser alcançada por uma lei de realimentação invariante no tempo e suave [BLOCH E MCCLAMROCH (1989); ORIOLO et al. (2002)]. Este resultado é baseado nos resultados apresentados em [BROCKETT (1983)].

Estudos de controlabilidade e estabilizabilidade para sistemas não holonômicos

podem ser encontrados em [BLOCH E MCCLAMROCH (1990)], onde se utiliza como exemplo o movimento de uma faca, e em [ORIOLO et al. (2002)], que avalia um robô do tipo unicycle.

Na literatura são apresentados três tipos básicos de objetivos a serem alcançados por um RMR: *estabilização de postura* (posição e direção do robô), alcançar uma postura de referência iniciando em uma dada postura; *acompanhamento de trajetória*, o robô deve seguir uma trajetória de referência em função do tempo; *seguindo um caminho*, o robô deve seguir uma trajetória de referência em função de parâmetros independentes do tempo, podendo ser geométrico [COELHO E NUNES (2003)] ou em função do trajeto e das velocidades ao longo do caminho [SARKAR et al. (1994)]. Neste trabalho, o controle será realizado para acompanhamento de trajetória de referência, sendo a trajetória de referência o objetivo a ser alcançado pelo ponto P_o do robô, diferentemente da trajetória desejada, que se refere às velocidades desejadas para as rodas do robô tais que o robô alcance a referência, conforme apresentado no Capítulo 3.

Na literatura foram encontradas inúmeras publicações de trabalhos relacionados a controle de RMRs e alguns estão resumidos a seguir.

[ORIOLO et al. (2002)] apresentaram um método unificado para resolver os problemas de acompanhamento de trajetória e estabilização de postura, que utiliza o modelo cinemático de um unicycle e a linearização por realimentação dinâmica. O método consiste em encontrar um compensador dinâmico dependente da direção do robô e das variáveis de controle geradas por um controlador proporcional derivativo (PD).

[SARKAR et al. (1994)] propuseram um método de controle por realimentação saída-estado que alcança estabilidade assintótica usando o modelo cinemático e dinâmico. A escolha das saídas está ligada diretamente ao objetivo de controle. Para o acompanhamento de trajetória, o vetor de saída consiste de parte das coordenadas generalizadas em função do tempo, e para seguir um caminho, o vetor de saída consiste em um caminho geométrico, por exemplo uma reta ou uma circunferência.

[COELHO E NUNES (2003)] mostraram o uso da álgebra de Lie para sistemas

de controle não lineares com restrições não holonômicas. O controle de um RMR é realizado por realimentação saída-estado, sendo as saídas o erro de posição do robô com relação a uma circunferência e a velocidade linear do robô para o problema de seguir um caminho.

[FUKAO et al. (2000)] apresentaram um método para projetar um controlador adaptativo para o modelo dinâmico de um robô móvel com restrições não holonômicas com parâmetros desconhecidos, utilizando *Backstepping* adaptativo. A metodologia de projeto recursiva chamada *Backstepping*, [KRSTIC et al. (1995)], é uma construção sistemática de ambas as leis de controle de realimentação e função de Lyapunov associada. Propriedades fortes de acompanhamento e estabilidade local ou global são construídas dentro de sistemas não lineares em um número de passos que nunca é maior do que a ordem do sistema.

[DO et al. (2004)] apresentaram um controle adaptativo global variante no tempo que resolve simultaneamente ambas estabilização de postura e acompanhamento de trajetória para robôs móveis com parâmetros da dinâmica e cinemática desconhecidos. A síntese de controle é baseada em técnicas de *Backstepping* e método direto de Lyapunov.

[JI et al. (2003)] desenvolveram um sistema de controle adaptativo tolerante a falha. O robô é modelado como um sistema contínuo com um controlador supervisor. Um ajuste em um controle hierárquico é desenvolvido para o controlador supervisor que determina uma estratégia de controle adequada para eliminar a falha.

Através de uma função de Lyapunov, [KANAYAMA et al. (1990)] propuseram um controle de acompanhamento de trajetória estável, que envolve o erro de postura (erro de direção e posição com relação ao sistema de coordenadas local do robô) e a trajetória de referência, e gera velocidades desejadas para o robô móvel seguir a trajetória de referência, ou seja, zerar o erro de postura.

1.2.2 Controle $\mathcal{H}_\infty$ para robôs

Poucos trabalhos têm sido apresentados com técnicas de controle $\mathcal{H}_\infty$ direcionados a RMRs. [HWANG et al. (2004)] propuseram uma combinação de um controla-

dor baseado na cinemática, encontrado via teoria de estabilidade de Lyapunov, e um controlador $\mathcal{H}_\infty$ robusto baseado na dinâmica para acompanhamento de trajetória. Este controle robusto é baseado no mesmo critério de desempenho de um controlador usado neste trabalho, porém, a solução proposta em [HWANG et al. (2004)] é dada para matrizes dinâmicas constantes. Neste trabalho, o modelo dinâmico é gerado usando matrizes dependentes do estado, ou seja, na forma quase-LPV que caracteriza melhor este tipo de sistema. A representação quase-LPV de um sistema não linear é uma equação no espaço de estado, na qual as matrizes do sistema estão em função dos parâmetros dependentes do estado.

Controladores para sistemas LPVs foram desenvolvidos em [WU (1995); WU et al. (1996)] com o objetivo de estabilizar o sistema em malha fechada com atenuação de distúrbios. [HUANG E JADBABAIE (1998)] resolveram o problema de controle $\mathcal{H}_\infty$ usando desigualdades matriciais lineares de dimensão infinita.

Diversas técnicas de controle $\mathcal{H}_\infty$ têm sido aplicadas a robôs manipuladores. [CHEN et al. (1994)] utilizaram a Teoria dos Jogos para resolver o problema de controle $\mathcal{H}_\infty$ para manipuladores totalmente atuados. Resultados experimentais obtidos da implementação de um controle $\mathcal{H}_\infty$ não linear via Teoria dos Jogos e um controle via representação quase-LPV em um manipulador subatuado são apresentados em [SIQUEIRA E TERRA (2004)].

Neste trabalho, o controle do robô móvel utiliza dois controladores. Um controlador baseado no modelo cinemático, proposto em [KANAYAMA et al. (1990)], gera as velocidades desejadas para as rodas e um controlador robusto baseado no modelo dinâmico encontra os torques a serem aplicados nas rodas. Dois controladores baseados no modelo dinâmico são encontrados resolvendo-se Desigualdades Matriciais Lineares (DMLs) para a equação no espaço de estado com representação quase-LPV e Equações Algébricas de Riccati (EAR) geradas pela Teoria dos Jogos (TJ). Também projetou-se um controlador Proporcional Derivativo em conjunto com um controlador do tipo torque calculado que serviu como referência de análise para o estudo comparativo dos controladores robustos.

1.2.3 Localização usando visão computacional

A determinação da postura de RMRs é uma área de pesquisa de destaque para a implementação dos controladores, pois uma boa precisão da posição e direção é de fundamental importância para o funcionamento do sistema como um todo.

Os sensores mais utilizados são os *encoders* (por exemplo [ORIOLO et al. (2002); SARKAR et al. (1994)]), sonares [BOZMA E KUC (1994)] ou a união destes dois [CORRADINI et al. (2003)]. A utilização de câmeras está sendo largamente pesquisada, sendo que elas podem ser embarcadas [LEE et al. (2004)], como em visão omnidirecional que produzem imagens de 360° do ambiente [YAGI et al. (1994)], ou externas como em [GUPTA et al. (2005); LIU et al. (2004)], na qual a câmera fica em cima de um campo de atuação (visão global).

Posições em imagens 2D pode ser obtida com extração de características como pontos, linhas, áreas ou formas geométricas. Um procedimento para detecção aproximada de círculos e arcos em imagens monocromática (escala em níveis de cinza) foi mostrado em [KIMME E SKLANSKY (1975)], fazendo primeiro uma extração de bordas e depois aplicando a Transformada de Hough com matrizes tridimensionais de acumuladores. [HO E CHEN (1995)] utilizaram simetria geométrica para encontrar círculos/elipses em imagens após aplicar um processo de extração de bordas. Um método para detecção de características geométricas (linhas, círculos e elipses) foi proposto por [MCLAUGHLIN E ALDER (1998)] e comparado com a transformação de Hough.

Para um RMR seguir um caminho desenhado na superfície, [BIANCHI et al. (2001)] utilizaram uma câmera de *video* colorida sobre a área de movimentação do robô (visão global). A identificação da posição, direção e caminho a seguir são encontrados particionando a imagem em regiões de cores de *pixels* usando a técnica de imposições de limiares.

[LUCA et al. (2002)] utilizaram a informação visual de uma câmera no teto para identificar a postura de um RMR. Em cima do robô foi criada uma superfície de cor preta com três *leds* localizados nos vértices de um triângulo isósceles. O método consiste primeiro em tornar binária a imagem monocromática capturada, depois

aplica-se um operador dilatação (veja [SONKA et al. (1998)]) e extração das regiões de luminosidades dos *leds* e por último as posições dos vértices do triângulo são encontradas por um algoritmo baseado na relação das distâncias entre as posições candidatas a vértices. Portanto, a posição será um dos vértices e a direção dada por relações trigonométricas.

Este trabalho implementa e analisa dois métodos de localização de um RMR (protótipo) em imagens monocromáticas. Eles são baseados na determinação de centros de círculos, através do método dos Momentos e da Transformada de Hough Circular (THC), mostrados em [SONKA et al. (1998)]. O robô tem a cor de fundo da imagem (preta). Para o primeiro método foram definidos dois círculos de cores claras e distintas e para o segundo método foram definidas três circunferências de cores claras. A localização destes círculos/circunferências no robô foi estrategicamente definido e o tratamento da imagem e lógicas para a determinação da postura estão apresentados no Capítulo 5.

1.3 Objetivos

Desenvolver e implementar controladores robustos para um Robô Móvel com Rodas (RMR), baseados no modelo dinâmico e no critério $\mathcal{H}_\infty$, visando atenuar os efeitos de distúrbios externos e incertezas paramétricas.

1.4 Originalidades do trabalho

O modelo dinâmico não linear de um RMR (uníciclo) com $P_c \neq P_o$ é descrito em equação de espaço de estado na representação quase-LPV, sendo os parâmetros as velocidades angulares das rodas.

Projeto de dois controladores baseados no modelo dinâmico usando técnicas de controle $\mathcal{H}_\infty$, o primeiro obtido através de Desigualdades Matriciais Lineares (DMLs) via representação quase-LPV e o segundo obtido resolvendo uma equação de Riccati gerada pela Teoria dos Jogos (TJ) para o problema de acompanhamento de trajetória

de referência para RMRs.

Estudo comparativo entre os dois controladores robustos desenvolvidos e um controlador proporcional derivativo em conjunto com um controlador do tipo torque calculado é apresentado.

Implementação e análise dos métodos dos Momentos e da Transformada de Hough Circular para a localização de RMRs usando visão computacional em imagens monocromáticas são realizados.

Resultados de simulação dos controladores serão apresentados nesta dissertação. As implementações estão sendo realizadas.

1.5 Disposição dos capítulos

No **Capítulo 2** é descrita a modelagem completa, cinemática e dinâmica, do robô móvel.

No **Capítulo 3** é apresentado o controle cinemático proposto por [KANAYAMA et al. (1990)] para obter as velocidades angulares desejadas, dadas a postura atual e a trajetória de referência.

No **Capítulo 4** é abordado o desenvolvimento de dois controladores robustos baseados no modelo dinâmico usando técnicas de controle $\mathcal{H}_\infty$ para RMRs.

No **Capítulo 5** são aplicados dois métodos para detecção de centro de círculos/circunferências em imagens que serão utilizados para a determinação de postura do RMR.

No **Capítulo 6** são mostrados os resultados simulados do controle de um RMR e a implementação do módulo de visão.

No **Capítulo 7** são relatadas as conclusões e as propostas de trabalhos futuros.

No **Apêndice A** são mostradas as principais lógicas dos programas desenvolvidos.

Capítulo 2

Modelagem do Robô Móvel

O Robô Móvel com Rodas (RMR) considerado nesta dissertação, com o centro de massa P_c localizado a uma distância d de P_o (ponto central entre as rodas atuadas, no eixo de simetria do robô), pode ser apresentado como na Fig. 2.1, sendo (X,Y)

Figura 2.1: Robô móvel.

o sistema de coordenadas inercial; (X_o, Y_o) o sistema de coordenadas local; a o comprimento do robô; d a distância entre P_o e P_c ; b a distância entre uma roda atuada e o eixo de simetria do robô; r o raio das rodas atuadas; α o ângulo (direção do robô) entre o eixo X e o eixo de simetria do robô no sentido anti-horário; θ_d e θ_e os deslocamentos angulares das rodas direita e esquerda, respectivamente.

A modelagem do robô apresentada a seguir está baseada em [COELHO (2001)].

2.1 Modelo Cinemático

Esta seção apresenta as equações de restrições cinemáticas de um robô móvel. Considerando as velocidades angulares das rodas, tais que $\dot{\theta}_d > \dot{\theta}_e > 0$ em um determinado instante, então as velocidades no robô podem ser apresentadas como na Fig. 2.2.

Figura 2.2: Velocidades do robô móvel.

Sabendo que o robô não pode deslizar, ou seja, movimentar-se na direção do eixo das rodas atuadas, obtém-se a **primeira restrição**. A velocidade do ponto P_o , v_o , deve estar na direção do eixo de simetria do robô, isto é,

$$\tan(\alpha) = \frac{\dot{y}_o}{\dot{x}_o} = \frac{\sin(\alpha)}{\cos(\alpha)} \Rightarrow \dot{y}_o \cos(\alpha) - \dot{x}_o \sin(\alpha) = 0. \quad (2.1)$$

Da Fig. 2.2, tem-se as seguintes equações

$$r\dot{\theta}_d = v_o + b\dot{\alpha}, \quad (2.2)$$

$$r\dot{\theta}_e = v_o - b\dot{\alpha}, \quad (2.3)$$

e

$$v_o = \dot{x}_o \cos(\alpha) + \dot{y}_o \sin(\alpha). \quad (2.4)$$

Somando (2.2) e (2.3), tem-se

$$v_o = (r\dot{\theta}_d + r\dot{\theta}_e)/2, \quad (2.5)$$

Subtraindo (2.2) e (2.3), obtém-se

$$2b\dot{\alpha} = r\dot{\theta}_d - r\dot{\theta}_e, \quad (2.6)$$

e substituindo (2.5) em (2.4), resulta em

$$r\dot{\theta}_d = 2\dot{x}_o \cos(\alpha) + 2\dot{y}_o \sin(\alpha) - r\dot{\theta}_e \quad (2.7)$$

ou

$$r\dot{\theta}_e = 2\dot{x}_o \cos(\alpha) + 2\dot{y}_o \sin(\alpha) - r\dot{\theta}_d. \quad (2.8)$$

As rodas devem rolar sem girar em falso, ou seja, a velocidade do ponto da roda atuada em contato com o solo deve ser zero. Assim, a **segunda restrição** referente à roda direita é obtida substituindo-se (2.8) em (2.6)

$$\dot{x}_o \cos(\alpha) + \dot{y}_o \sin(\alpha) + b\dot{\alpha} - r\dot{\theta}_d = 0. \quad (2.9)$$

A **terceira restrição** referente à roda esquerda, semelhante à anterior, é obtida substituindo (2.7) em (2.6)

$$\dot{x}_o \cos(\alpha) + \dot{y}_o \sin(\alpha) - b\dot{\alpha} - r\dot{\theta}_e = 0. \quad (2.10)$$

Definindo a coordenada generalizada $q = [x_o \ y_o \ \alpha \ \theta_d \ \theta_e]^T$ e $q_2 = [\theta_d \ \theta_e]^T$, as

restrições podem ser representadas em espaço de estado na forma

$$A_o(q)\dot{q} = \begin{bmatrix} -\sin(\alpha) & \cos(\alpha) & 0 & 0 & 0 \\ -\cos(\alpha) & -\sin(\alpha) & -b & r & 0 \\ -\cos(\alpha) & -\sin(\alpha) & b & 0 & r \end{bmatrix} \dot{q} = 0. \quad (2.10)$$

A matriz $A_o(q)$ tem posto pleno e pode ser particionada em duas matrizes da forma $A_o(q) = [A_1(q)_{(3 \times 3)} \quad A_2(q)_{(3 \times 2)}]$. Como $A_1(q)$ é não singular, então é possível encontrar uma matriz $S_o(q) = [-A_1^{-1}(q)A_2 \quad I_{(2 \times 2)}]^T$ (veja [BLOCH et al. (1992)] para detalhes) que está no espaço nulo de $A_o(q)$, ou seja, $A_o(q)S_o(q) = 0$. Então encontra-se

$$S_o(q) = \begin{bmatrix} \frac{r}{2} \cos(\alpha) & \frac{r}{2} \cos(\alpha) \\ \frac{r}{2} \sin(\alpha) & \frac{r}{2} \sin(\alpha) \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Esta matriz $S_o(q)$ permite transformar as velocidades angulares das rodas atuais nas velocidades do ponto P_o no sistema de coordenadas inercial e, portanto, a equação cinemática do robô móvel será dada por

$$\dot{q}(t) = S_o(q)\dot{q}_2(t) \quad (2.11)$$

ou

$$\dot{x}_o = \frac{r}{2} \cos(\alpha) \dot{\theta}_d + \frac{r}{2} \cos(\alpha) \dot{\theta}_e, \quad (2.12)$$

$$\dot{y}_o = \frac{r}{2} \sin(\alpha) \dot{\theta}_d + \frac{r}{2} \sin(\alpha) \dot{\theta}_e, \quad (2.13)$$

$$\dot{\alpha} = \frac{r}{2b} (\dot{\theta}_d - \dot{\theta}_e). \quad (2.14)$$

O controle do RMR será realizado para que o ponto P_o acompanhe uma trajetória de referência. Entretanto, para a obtenção do modelo dinâmico na Seção 2.2 é interessante que o modelo cinemático esteja relacionado ao ponto P_c . Assim, pode-

se relacionar as velocidades em P_o , $(\dot{x}_o$ e $\dot{y}_o)$, com as velocidades em P_c , $(\dot{x}_c$ e $\dot{y}_c)$, derivando as relações de posições que podem ser obtidas analisando a Fig. 2.2 que são dadas por

$$\begin{cases} x_c - x_o = d\cos(\alpha) \\ y_c - y_o = d\sin(\alpha) \end{cases} \Rightarrow \begin{cases} \dot{x}_o = \dot{x}_c + d\sin(\alpha)\dot{\alpha} \\ \dot{y}_o = \dot{y}_c - d\cos(\alpha)\dot{\alpha} \end{cases} \quad (2.15)$$

Substituindo (2.15) em (2.1), (2.9) e (2.10), obtém-se as restrições em função das velocidades $(\dot{x}_c, \dot{y}_c$ e $\dot{\alpha})$ em P_c

$$\dot{y}_c \cos(\alpha) - \dot{x}_c \sin(\alpha) - d\dot{\alpha} = 0, \quad (2.16)$$

$$\dot{x}_c \cos(\alpha) + \dot{y}_c \sin(\alpha) + b\dot{\alpha} - r\dot{\theta}_d = 0, \quad (2.17)$$

$$\dot{x}_c \cos(\alpha) + \dot{y}_c \sin(\alpha) - b\dot{\alpha} - r\dot{\theta}_e = 0. \quad (2.18)$$

Este sistema de três equações de restrições cinemáticas possui uma equação holonômica e duas equações não holonômicas (veja [COELHO E NUNES (2003)]). A equação de restrição holonômica pode ser obtida subtraindo (2.17) de (2.18), ou seja,

$$\dot{\alpha} = \frac{r}{2b}(\dot{\theta}_d - \dot{\theta}_e). \quad (2.19)$$

Integrando (2.19), tem-se

$$\alpha = \frac{r}{2b}(\theta_d - \theta_e) + c_1,$$

sendo c_1 uma constante de integração. Portanto, verifica-se que (2.19) é uma equação de restrição holonômica como apresentado na Seção 1.2.1.

E duas restrições não holonômicas são dadas por (2.16)

$$\dot{y}_c \cos(\alpha) - \dot{x}_c \sin(\alpha) - d\dot{\alpha} = 0 \quad (2.20)$$

e, somando (2.17) e (2.18),

$$\dot{x}_c \cos(\alpha) + \dot{y}_c \sin(\alpha) - \frac{r}{2}(\dot{\theta}_d + \dot{\theta}_e) = 0. \quad (2.21)$$

Definindo a coordenada generalizada $q_1 = [x_c \ y_c \ \alpha \ \theta_d \ \theta_e]^T$, então as restrições podem ser representadas em espaço de estado na forma

$$A_c(q_1)\dot{q}_1 = \begin{bmatrix} -\sin(\alpha) & \cos(\alpha) & -d & 0 & 0 \\ -\cos(\alpha) & -\sin(\alpha) & -b & r & 0 \\ -\cos(\alpha) & -\sin(\alpha) & b & 0 & r \end{bmatrix} \dot{q}_1 = 0.$$

Novamente tem-se que $A_c(q_1)$ tem posto pleno e pode ser particionada em duas matrizes da forma $A_c(q_1) = [A_3(q_1)_{(3 \times 3)} \ A_2(q_1)_{(3 \times 2)}]$. Como $A_3(q_1)$ é não singular então é possível encontrar uma matriz $S_c(q_1) = [-A_3^{-1}(q_1)A_2 \ I_{(2 \times 2)}]^T$ que está no espaço nulo de $A_c(q_1)$, ou seja, $A_c(q_1)S_c(q_1) = 0$. Então encontra-se

$$S_c(q_1) = \begin{bmatrix} \frac{r}{2b}(b \cos(\alpha) - d \sin(\alpha)) & \frac{r}{2b}(b \cos(\alpha) + d \sin(\alpha)) \\ \frac{r}{2b}(b \sin(\alpha) + d \cos(\alpha)) & \frac{r}{2b}(b \sin(\alpha) - d \cos(\alpha)) \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Esta matriz $S_c(q_1)$ permite transformar as velocidades angulares das rodas atuadas nas velocidades do ponto P_c no sistema de coordenadas inercial, formando uma outra equação cinemática do robô móvel, dada por

$$\dot{q}_1(t) = S_c(q_1)\dot{q}_2(t) \quad (2.22)$$

ou

$$\dot{x}_c = \frac{r}{2b}(b \cos(\alpha) - d \sin(\alpha))\dot{\theta}_d + \frac{r}{2b}(b \cos(\alpha) + d \sin(\alpha))\dot{\theta}_e, \quad (2.23)$$

$$\dot{y}_c = \frac{r}{2b}(b \sin(\alpha) + d \cos(\alpha))\dot{\theta}_d + \frac{r}{2b}(b \sin(\alpha) - d \cos(\alpha))\dot{\theta}_e, \quad (2.24)$$

$$\dot{\alpha} = \frac{r}{2b}(\dot{\theta}_d - \dot{\theta}_e). \quad (2.25)$$

2.2 Modelo Dinâmico

Um método bem conhecido para a formulação de modelos dinâmicos é através da abordagem definida por Lagrange [GOLDSTEIN (1981)] que descreve o comportamento dinâmico de um robô em termos de trabalho realizado pelo sistema e pela energia armazenada no sistema. Considerando as restrições em que o robô está sujeito, a equação de Lagrange é da forma

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_1} \right) - \frac{\partial L}{\partial q_1} = \tau - A_c(q_1)^T \lambda, \quad (2.26)$$

sendo τ o torque, λ o multiplicador de Lagrange e L o Lagrangiano dado pela diferença entre a energia cinética $\mathcal{T}$ e potencial $\mathcal{V}$.

O robô móvel considerado movimenta-se em um campo de atuação plano e sem inclinação e, portanto, a energia potencial $\mathcal{V}$ é zero. Assim o Lagrangiano será encontrado calculando a energia cinética total do robô dado pelo somatório da energia cinética de suas partes constituintes, ou seja,

$$\mathcal{T}(q_1, \dot{q}_1) = \sum_{i=1}^n \mathcal{T}_i(q_1, \dot{q}_1) = \frac{1}{2} \dot{q}_1^T M(q_1) \dot{q}_1. \quad (2.27)$$

A energia cinética de um corpo movendo-se linearmente ($\mathcal{T}_{lin}$) e rotacionalmente ($\mathcal{T}_{rot}$) é dada por $\mathcal{T} = \mathcal{T}_{lin} + \mathcal{T}_{rot} = \frac{1}{2}mv^2 + \frac{1}{2}I_{cm}\omega^2$, sendo m a massa do corpo; v a velocidade linear no centro de massa; ω a velocidade angular do corpo; e I_{cm} o momento de inércia do corpo relativo ao centro de massa.

A energia cinética da plataforma relativa ao movimento linear em P_c é dada por

$$\mathcal{T}_{linP} = \frac{1}{2}m_p v_c^2 = \frac{1}{2}m_p (\dot{x}_c^2 + \dot{y}_c^2), \quad (2.28)$$

sendo m_p a massa da plataforma do robô.

Devido à rotação em torno de P_o há uma energia cinética de rotação da plataforma em P_c , dada por

$$\mathcal{T}_{rotP} = \frac{1}{2}I_c \dot{\alpha}^2, \quad (2.29)$$

sendo I_c o momento de inércia da plataforma do robô em relação ao eixo vertical em P_c . A energia total da plataforma em P_c é escrita da seguinte maneira

$$T_P = T_{linP} + T_{rotP} = \frac{1}{2}m_p(\dot{x}_c^2 + \dot{y}_c^2) + \frac{1}{2}I_c\dot{\alpha}^2. \quad (2.30)$$

As velocidades lineares na roda direita, $(\dot{x}_d$ e $\dot{y}_d)$, com relação às velocidades em P_o , $(\dot{x}_o$ e $\dot{y}_o)$, podem ser obtidas de acordo com a Fig. 2.2, a partir das relações

$$\begin{cases} x_d - x_o = b\sin(\alpha) \\ y_o - y_d = b\cos(\alpha) \end{cases} \Rightarrow \begin{cases} \dot{x}_d = \dot{x}_o + b\cos(\alpha)\dot{\alpha} \\ \dot{y}_d = \dot{y}_o + b\sin(\alpha)\dot{\alpha}. \end{cases} \quad (2.31)$$

Elevando ao quadrado cada equação de (2.31) e somando-as, obtém-se

$$\dot{x}_d^2 + \dot{y}_d^2 = \dot{x}_o^2 + \dot{y}_o^2 + b^2\dot{\alpha}^2 + 2b\dot{\alpha}(\dot{x}_o\cos(\alpha) + \dot{y}_o\sin(\alpha)). \quad (2.32)$$

A energia cinética da roda direita relativa ao movimento linear é dada por

$$\begin{aligned} T_{linRd} = \frac{1}{2}m_r(r\dot{\theta}_d)^2 = \frac{1}{2}m_r(\dot{x}_d^2 + \dot{y}_d^2) = \frac{1}{2}m_r(\dot{x}_o^2 + \dot{y}_o^2) + \frac{1}{2}m_r(b^2\dot{\alpha}^2) \\ + m_rb\dot{\alpha}(\dot{x}_o\cos(\alpha) + \dot{y}_o\sin(\alpha)), \end{aligned} \quad (2.33)$$

sendo m_r a massa de cada roda atuada mais o rotor do motor. Devido ao movimento rotacional da plataforma e da roda direita, vale a seguinte relação

$$T_{rotRd} = \frac{1}{2}I_m\dot{\alpha}^2 + \frac{1}{2}I_r\dot{\theta}_d^2, \quad (2.34)$$

sendo I_m o momento de inércia em relação ao eixo definido no plano da roda (perpendicular ao eixo da roda) e I_r o momento de inércia da roda com o rotor do motor em relação ao eixo da roda. Então a energia cinética total da roda direita é dada por

$$\begin{aligned} T_{Rd} = T_{linRd} + T_{rotRd} = \frac{1}{2}m_r(\dot{x}_o^2 + \dot{y}_o^2) + \frac{1}{2}m_r(b^2\dot{\alpha}^2) \\ + m_rb\dot{\alpha}(\dot{x}_o\cos(\alpha) + \dot{y}_o\sin(\alpha)) + \frac{1}{2}I_m\dot{\alpha}^2 + \frac{1}{2}I_r\dot{\theta}_d^2. \end{aligned} \quad (2.35)$$

Seguindo o mesmo raciocínio, a energia cinética da roda esquerda é dada por

$$\begin{aligned} \mathcal{T}_{Re} = \frac{1}{2}m_r(\dot{x}_o^2 + \dot{y}_o^2) + \frac{1}{2}m_r(b^2\dot{\alpha}^2) - m_rb\dot{\alpha}(\dot{x}_o\cos(\alpha) + \dot{y}_o\sin(\alpha)) \\ + \frac{1}{2}I_m\dot{\alpha}^2 + \frac{1}{2}I_r\dot{\theta}_e^2. \end{aligned} \quad (2.36)$$

A energia cinética das rodas é dada por

$$\mathcal{T}_R = \mathcal{T}_{Rd} + \mathcal{T}_{Re} = 2m_r(\dot{x}_o^2 + \dot{y}_o^2) + m_r(b^2\dot{\alpha}^2) + \frac{1}{2}I_r(\dot{\theta}_d^2 + \dot{\theta}_e^2) + I_m\dot{\alpha}^2. \quad (2.37)$$

Elevando ao quadrado cada equação de (2.15) e somando-as, obtém-se

$$\dot{x}_o^2 + \dot{y}_o^2 = \dot{x}_c^2 + \dot{y}_c^2 + d^2\dot{\alpha}^2 + 2d\dot{\alpha}(\dot{x}_c\sin(\alpha) - \dot{y}_c\cos(\alpha)). \quad (2.38)$$

Substituindo (2.38) em (2.37), obtém-se a energia cinética das rodas em função das velocidades $\dot{x}_c$, $\dot{y}_c$ e $\dot{\alpha}$

$$\begin{aligned} \mathcal{T}_R = m_r(\dot{x}_c^2 + \dot{y}_c^2) + m_r(d^2 + b^2)\dot{\alpha}^2 \\ + (2m_r)d\dot{\alpha}(\dot{x}_c\sin(\alpha) - \dot{y}_c\cos(\alpha)) + \dot{\alpha}^2 I_m + \frac{1}{2}I_r(\dot{\theta}_d^2 + \dot{\theta}_e^2). \end{aligned} \quad (2.39)$$

A energia cinética total do robô é dada por

$$\mathcal{T} = \mathcal{T}_P + \mathcal{T}_R = \frac{1}{2}m(\dot{x}_c^2 + \dot{y}_c^2) + \frac{1}{2}I\dot{\alpha}^2 + \frac{1}{2}I_r(\dot{\theta}_d^2 + \dot{\theta}_e^2) + 2m_rd\dot{\alpha}(\dot{x}_c\sin(\alpha) - \dot{y}_c\cos(\alpha)), \quad (2.40)$$

sendo

$$m = m_p + 2m_r \text{ e } I = I_c + 2m_r(d^2 + b^2) + 2I_m.$$

Substituindo $L = \mathcal{T}$ em (2.26), obtém-se a seguinte equação dinâmica para o robô

$$M(q_1)\ddot{q}_1 + C(q_1, \dot{q}_1)\dot{q}_1 = E\tau - A_c(q_1)^T\lambda, \quad (2.41)$$

sendo $\lambda = [\lambda_1 \ \lambda_2 \ \lambda_3]^T$ o vetor de forças de restrições; $E = [0_{2 \times 4} \ I_{2 \times 2}]^T$ a matriz de transformação de entrada; $\tau = [\tau_d \ \tau_e]^T$ o vetor de entrada de torque nas rodas;

$$C(q_1, \dot{q}_1) = \begin{bmatrix} 0 & 0 & 2m_r d \dot{\alpha} \cos(\alpha) & 0 & 0 \\ 0 & 0 & 2m_r d \dot{\alpha} \sin(\alpha) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

a matriz de forças de Coriolis e centrípeta e

$$M(q_1) = \begin{bmatrix} m & 0 & 2m_r d \sin(\alpha) & 0 & 0 \\ 0 & m & -2m_r d \cos(\alpha) & 0 & 0 \\ 2m_r d \sin(\alpha) & -2m_r d \cos(\alpha) & I & 0 & 0 \\ 0 & 0 & 0 & I_r & 0 \\ 0 & 0 & 0 & 0 & I_r \end{bmatrix}$$

a matriz de inércia.

Capítulo 3

Controlador Baseado na Cinemática

O controle do robô será realizado primeiro por uma lei de controle baseada no modelo cinemático e depois por uma lei de controle baseada no modelo dinâmico.

O ponto considerado no robô para acompanhar a trajetória de referência é o ponto P_o . Assim considere o erro de postura no sistema de coordenadas local $q_e = [x_e \ y_e \ \alpha_e]^T$, entre a postura de referência $q_r = [x_r \ y_r \ \alpha_r]^T$ e a postura atual $q_a = [x_o \ y_o \ \alpha]^T$ no sistema de coordenadas inercial, como ilustrada na Fig. 3.1.

Figura 3.1: Representação do erro de postura do robô móvel.

As equações dos erros (com relação às coordenadas locais) são obtidas pela análise

trigonométrica da Fig. 3.1 e são dadas por

$$\begin{aligned}x_e &= (x_r - x_o) \cos(\alpha) + (y_r - y_o) \sin(\alpha), \\y_e &= -(x_r - x_o) \sin(\alpha) + (y_r - y_o) \cos(\alpha), \\ \alpha_e &= \alpha_r - \alpha,\end{aligned}\tag{3.1}$$

sendo $(x_r, y_r) = P_{or}$ a trajetória de referência escolhida e $\alpha_r = \tan^{-1}(\frac{\dot{y}_r}{\dot{x}_r})$.

As velocidades linear (v^d) e angular (ω^d) desejadas do robô móvel foram propostas por [KANAYAMA et al. (1990)] e usadas por [FIERRO E LEWIS (1995); FIERRO E LEWIS (1998); FUKAO et al. (2000)] conforme mostradas abaixo

$$\begin{aligned}v^d &= v_{or} \cos(\alpha_e) + k_x x_e, \\ \omega^d &= \omega_r + v_{or}(k_y y_e + k_\alpha \sin(\alpha_e)),\end{aligned}\tag{3.2}$$

sendo k_x, k_y, k_α constantes,

$$v_{or} = \sqrt{(\dot{x}_r)^2 + (\dot{y}_r)^2} \text{ e } \omega_r = \dot{\alpha}_r.\tag{3.3}$$

A lei de controle (3.2) tem estabilidade assintoticamente uniforme ao redor de $q_e = 0$ quando satisfaz as seguintes condições: $v_{or} > 0$; v_{or} e ω_r são contínuas; v_{or} , ω_r , k_x e k_α são limitados; $\dot{v}_{or}$ e $\dot{\omega}_r$ são suficientemente pequenas [KANAYAMA et al. (1990)].

No Capítulo 4, o controle baseado no modelo dinâmico do robô é resolvido considerando as velocidades angulares desejadas das rodas, $\dot{q}_2^d$. Então é necessária uma relação de velocidades que pode ser obtida através de (2.2) e (2.3), resultando em

$$\dot{q}_2^d = \begin{bmatrix} \dot{\theta}_d^d \\ \dot{\theta}_e^d \end{bmatrix} = \begin{bmatrix} 1/r & b/r \\ 1/r & -b/r \end{bmatrix} \begin{bmatrix} v^d \\ \omega^d \end{bmatrix},\tag{3.4}$$

sendo $\dot{\theta}_d^d$ e $\dot{\theta}_e^d$ as velocidades angulares desejadas das rodas direita e esquerda, respectivamente.

Capítulo 4

Controladores Baseados no Modelo Dinâmico

Uma vez encontradas as velocidades desejadas do robô (v^d e ω^d) utilizando o controlador cinemático e posteriormente as velocidades angulares desejadas das rodas ($\dot{\theta}_d^d$ e $\dot{\theta}_e^d$), torna-se possível aplicar técnicas de controle $\mathcal{H}_\infty$ ao modelo dinâmico para obter os torques a serem aplicados às rodas para o acompanhamento da trajetória de referência.

4.1 Controle $\mathcal{H}_\infty$ não linear via representação quase-LPV

O sistema dinâmico não linear do robô móvel, será formulado nesta seção como um sistema quase Linear a Parâmetros Variantes (quase-LPV). A lei de controle não linear apresentada a seguir é baseada em Desigualdades Matriciais Lineares (DMLs), que fornecem uma lei de controle por realimentação de estado $u = F(\rho)x$ de maneira a estabilizar o sistema em malha fechada garantindo que um ganho $\mathcal{L}_2$ entre o distúrbio e a saída seja limitado por um nível de atenuação $\gamma > 0$.

4.1.1 Ganho $\mathcal{L}_2$ para sistemas não lineares variantes no tempo

Considere um sistema não linear variante no tempo com entrada de distúrbio afim $w \in \mathbb{R}^p$ e saída controlada $z \in \mathbb{R}^q$

$$\begin{aligned}\dot{x} &= f(x, t) + g(x, t)w, \\ z &= h(x, t) + k(x, t)w,\end{aligned}\tag{4.1}$$

sendo $f(0, t) = 0$ e $h(0, t) = 0$ para todo $t \in [0, T]$, e $x \in \mathbb{R}^n$ o estado. Assume-se que $f(x, t)$, $g(x, t)$, $h(x, t)$ e $k(x, t)$ são funções continuamente diferenciáveis em relação a x e contínuas em t . O sistema (4.1) possui ganho $\mathcal{L}_2 \leq \gamma$ no intervalo $[0, T]$ se

$$\int_0^T \|z(t)\|^2 dt \leq \gamma^2 \int_0^T \|w(t)\|^2 dt,\tag{4.2}$$

para todo $T \geq 0$ e todo $w \in \mathcal{L}_2(0, T)$ com o sistema iniciando em $x(0) = 0$. Para sistemas lineares invariantes no tempo, a condição de ganho $\mathcal{L}_2 \leq \gamma$ corresponde à condição da norma $\mathcal{H}_\infty$ da função de transferência entre a entrada de distúrbio e a saída controlada ser limitada por γ , ou seja, $\|T_{zw}(s)\|_\infty \leq \gamma$.

4.1.2 Síntese do controle $\mathcal{H}_\infty$ para sistemas LPV por realimentação do estado

O problema consiste no controle por realimentação de estado dependente de parâmetro para estabilizar o sistema em malha fechada e fazer a norma $\mathcal{L}_2$ menor que um nível de desempenho especificado γ . Considere o problema de síntese do controle por realimentação do estado

$$\begin{aligned}\dot{x} &= A(\rho(t))x + B_1(\rho(t))w + B_2(\rho(t))u \\ z_1 &= C_1(\rho(t))x \\ z_2 &= C_2(\rho(t))x + u\end{aligned}\tag{4.3}$$

sendo $x \in \mathbb{R}^n$ o estado, $u \in \mathbb{R}^{q_4}$ a entrada de controle, $w \in \mathbb{R}^p$ o distúrbio de entrada, $z_1 \in \mathbb{R}^{q_3}$ e $z_2 \in \mathbb{R}^{q_4}$ as saídas controladas, $A(\cdot)$, $B_1(\cdot)$, $B_2(\cdot)$, $C_1(\cdot)$ e $C_2(\cdot)$ matrizes contínuas de dimensões apropriadas e $\rho(t) \in F_P^\nu$ definido por

$$F_P^\nu = \{ \rho \in C^1(\mathbb{R}^+, \mathbb{R}^m) : \rho(t) \in P, \|\dot{\rho}_i\| \leq \nu_i, i = 1, \dots, m \},$$

sendo $P \subset \mathbb{R}^m$ um conjunto compacto e $\nu = [\nu_1 \dots \nu_m]^T$ com $\nu_i \geq 0$.

Definição 4.1 (WU (1995)) *O problema de realimentação de estado dependente de parâmetro, para o sistema LPV (4.3), é resolvido se existir uma função $Z \in C^1(\mathbb{R}^s, S^{n \times n})$ e uma $F \in C^0(\mathbb{R}^s \times \mathbb{R}^s, \mathbb{R}^{n_u \times n})$ tais que para todo $\rho(t) \in P$ e $\|\dot{\rho}_i\| \leq \nu_i$, $i = 1, 2, \dots, s$, $Z(\rho) > 0$ e*

$$\begin{bmatrix} A_F^T(\rho, \dot{\rho})Z(\rho) + Z(\rho)A_F(\rho, \dot{\rho}) + \sum_{i=1}^s \left(\nu_i \frac{\partial Z}{\partial \rho_i} \right) + C^T(\rho, \dot{\rho})C(\rho, \dot{\rho}) & Z(\rho)B_1(\rho) \\ B_1^T(\rho)Z(\rho) & -\gamma^2 I \end{bmatrix} < 0, \quad (4.4)$$

sendo $A_F(\rho, \dot{\rho}) := A(\rho) + B_2(\rho)F(\rho, \dot{\rho})$, $C(\rho, \dot{\rho}) := C_1(\rho) + D_{12}F(\rho, \dot{\rho})$ e $D_{12} = [0 \ I]^T$.

Se o problema de realimentação de estado dependente de parâmetro (4.4) tem solução, então a lei de controle $u = F(\rho(t))x$ estabilizará exponencialmente o sistema em malha fechada e garantirá que a norma $\mathcal{L}_2$ induzida seja menor que γ . O seguinte teorema fornece uma condição de existência para o controlador de realimentação de estado expresso em DMLs para o sistema LPV em malha aberta (4.3).

Teorema 4.1 (WU (1995)) *Dado um conjunto compacto $P \in \mathbb{R}^s$, um nível de desempenho $\gamma > 0$ e o sistema (4.3), o problema de realimentação de estado tem solução se e somente se existir uma função $X \in C^1(\mathbb{R}^s, S^{n \times n})$ tal que para todo $\rho \in P$, $X(\rho) > 0$ e*

$$\begin{bmatrix} E(\rho) & X(\rho)C_1^T(\rho) & B_1(\rho) \\ C_1(\rho)X(\rho) & -I & 0 \\ B_1^T(\rho) & 0 & -\gamma^2 I \end{bmatrix} < 0, \quad (4.5)$$

sendo

$$E(\rho) = - \sum_{i=1}^m \pm \left(\nu_i \frac{\partial X}{\partial \rho_i} \right) + \hat{A}(\rho)X(\rho) + X(\rho)\hat{A}(\rho)^T - B_2(\rho)B_2^T(\rho)$$

$$\text{e } \hat{A}(\rho) = A(\rho) - B_2(\rho)C_2(\rho).$$

Este teorema fornece um ganho $F(\rho)$ de realimentação de estado tal que

$$u = -(B_2(\rho)^T X^{-1}(\rho) + C_2(\rho))x, \quad (4.6)$$

garante que o sistema em malha tenha ganho $\mathcal{L}_2 \leq \gamma$ para toda variação paramétrica $\rho(t) \in F_P^\nu$.

O resultado acima é uma generalização natural da teoria de controle $\mathcal{H}_\infty$ para sistemas lineares. Uma função de Lyapunov dependente de parâmetros na forma $V(x, t) = x^T(t)X^{-1}(\rho(t))x(t)$ é assumida. Como resultado, deve-se resolver as DMLs paramétricas (4.5) que é um problema de otimização convexo com dimensão infinita.

4.1.3 Considerações computacionais

Um esquema computacional prático [WU (1995); WU et al. (1996); HUANG E JADBABAIE (1998); SIQUEIRA E TERRA (2004)] pode ser utilizado para resolver as desigualdades matriciais lineares presentes na análise e síntese dos problemas LPV. Por simplicidade, considere o problema de encontrar $X(\rho(t))$ na Equação (4.5). Primeiro, escolha um conjunto de funções $\mathcal{C}^1$, $\{f_i(\rho(t))\}_{i=1}^M$, como base para $X(\rho)$, ou seja,

$$X(\rho(t)) = \sum_{i=1}^M f_i(\rho(t))X_i, \quad (4.7)$$

sendo $X_i \in S^{n \times n}$ a matriz coeficiente para $f_i(\rho(t))$. Se $X(\rho(t))$ em (4.5) é substituída por (4.7), o problema de realimentação do estado transforma-se no seguinte problema de otimização

$$\min_{\{X_i\}_{i=1}^M} \gamma^2,$$

sujeito a

$$\begin{bmatrix} E^*(\rho) & \sum_{j=1}^M f_j(\rho) X_j C_1^T(\rho) & B_1(\rho) \\ C_1(\rho) \sum_{j=1}^M f_j(\rho) X_j & -I & 0 \\ B_1^T(\rho) & 0 & -\gamma^2 I \end{bmatrix} < 0,$$

$$\sum_{j=1}^M f_j(\rho) X_j > 0, \quad (4.8)$$

sendo

$$E^*(\rho) = - \sum_{i=1}^m \pm \left(\nu_i \sum_{j=1}^M \frac{\partial f_j}{\partial \rho_i} X_j \right) + \sum_{j=1}^M f_j(\rho) (\hat{A}(\rho) X_j + X_j \hat{A}(\rho)^T) - B_2(\rho) B_2^T(\rho).$$

Note que (4.8) são DMLs em termos das variáveis matriciais $\{X_i\}_{i=1}^M$ que devem ser satisfeitas para todo parâmetro $\rho(t)$ em P . Para resolver este problema de otimização de dimensão infinita, divide-se o conjunto de parâmetros P em N pontos $\{\rho_k\}_{k=1}^N$ em cada dimensão. Então calcula-se as DMLs acima para estes pontos. Desde que (4.5) consiste em 2^m vínculos, um total de $(2^m + 1)N^m$ desigualdades matriciais afins em termos das M variáveis matriciais $\{X_i\}$ devem ser resolvidas. Uma aproximação da densidade de pontos particionados, N , que garante uma solução global das DMLs é dada em [WU (1995); WU et al. (1996)].

Este esquema computacional possui algumas limitações. O número de parâmetros considerados e o número de divisões N devem ser escolhidos tais que a solução seja alcançada em um número de iterações realizáveis. Outro problema é a falta de justificativa teórica na escolha das funções base para $X(\rho)$. Geralmente, escolhem-se funções similares às encontradas nas matrizes de estado $A(\rho(t))$ [APKARIAN E ADAMS (1998)].

4.1.4 Representação quase-LPV do modelo dinâmico do robô

Com a equação em espaço de estado formulada na representação quase-LPV (o parâmetro ρ em função do estado, ou seja, $\rho = \rho(x)$) resultante do modelo dinâmico (2.41), o controle $\mathcal{H}_\infty$ para sistemas LPV pode ser aplicado a RMRs, gerando um

controlador não linear baseado na dinâmica. Diferenciando (2.22) com respeito ao tempo, tem-se

$$\ddot{q}_1 = \dot{S}_c(q_1)\dot{q}_2 + S_c(q_1)\ddot{q}_2. \quad (4.9)$$

Substituindo (4.9) e (2.22) na equação (2.41) e multiplicando por S_c^T à esquerda, obtém-se

$$M_2\ddot{q}_2 + C_2(\dot{q}_1)\dot{q}_2 = S_c^T E\tau = \tau \quad (4.10)$$

sendo M_2 uma matriz simétrica constante, não singular, dada por $S_c(q_1)^T M(q_1) S_c(q_1)$ e $C_2(\dot{q}_1) = C_2(\dot{\alpha}) = S_c(q_1)^T C(q_1, \dot{q}_1) S_c(q_1) + S_c(q_1)^T M(q_1) \dot{S}_c(q_1)$.

Note que nesta passagem desaparece a matriz de restrição que estava presente no termo $A_c^T \lambda$ da equação dinâmica, pois $S_c^T A_c^T = 0$ (A_c está no espaço nulo de S_c). Acrescentando um distúrbio de torque $w = [w_d \ w_e]^T$ e substituindo (2.25) na equação (4.10) segue que

$$\ddot{q}_2 = A(\dot{q}_2)\dot{q}_2 + B\tau + Bw, \quad (4.11)$$

sendo $A(\dot{q}_2) = -M_2^{-1}C_2(\dot{q}_2)$ e $B = M_2^{-1}$. Somando e subtraindo $\ddot{q}_2^d$ e $A(\dot{q}_2)\dot{q}_2^d$ em (4.11), (sendo que ‘d’ sobrescrito significa valor desejado) e definindo o estado como

$$\tilde{x} = \begin{bmatrix} \dot{\tilde{q}}_2 \\ \tilde{q}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_d - \dot{\theta}_e^d \\ \theta_d - \theta_e^d \\ \theta_e - \theta_e^d \end{bmatrix}$$

a representação quase-LPV, em espaço de estado, para o controle de acompanhamento de trajetória do RMR é dada por

$$\dot{\tilde{x}} = \begin{bmatrix} A(\dot{q}_2) & 0 \\ I & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} I \\ 0 \end{bmatrix} u + \begin{bmatrix} B \\ 0 \end{bmatrix} w \quad (4.12)$$

sendo

$$u = -\ddot{q}_2^d + A(\dot{q}_2)\dot{q}_2^d + B\tau,$$

ou

$$\tau = B^{-1}(\ddot{q}_2^d - A(\dot{q}_2)\dot{q}_2^d + u). \quad (4.13)$$

4.2 Controle $\mathcal{H}_\infty$ não linear via Teoria dos Jogos

Nesta seção, o controle $\mathcal{H}_\infty$ não linear proposto em [CHEN et al. (1994)] é aplicado aos RMRs onde se leva em consideração a equação em espaço de estado da dinâmica do erro de ordem reduzida (4.12). Considere a seguinte transformação de estado

$$\tilde{z} = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \end{bmatrix} = T_0 \tilde{x} = \begin{bmatrix} T_{11} & T_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\tilde{q}}_2 \\ \tilde{q}_2 \end{bmatrix}, \quad (4.14)$$

sendo $T_{11}, T_{12} \in \mathbb{R}^{2 \times 2}$ matrizes constantes a serem determinadas. Assume-se que a matriz T_{11} é diagonal, ou seja, $T_{11} = t_{11}I$. Aplicando a transformação de estado (4.14) em (4.12), obtém-se

$$\dot{\tilde{x}} = A_T(\dot{q}_2)\tilde{x} + B_{T_1}u + B_{T_2}w, \quad (4.15)$$

sendo

$$\begin{aligned} A_T(\dot{q}_2) &= T_0^{-1} \begin{bmatrix} A(\dot{q}_2) & 0 \\ T_{11}^{-1} & -T_{11}^{-1}T_{12} \end{bmatrix} T_0, \\ B_{T_1} &= T_0^{-1} \begin{bmatrix} M_2^{-1} \\ 0 \end{bmatrix}, \\ B_{T_2} &= T_0^{-1} \begin{bmatrix} M_2^{-1} \\ 0 \end{bmatrix} \quad T_{11} = B_{T_1}T_{11}, \end{aligned}$$

$$u = T_{11}(-M_2(\ddot{q}_2^d - T_{11}^{-1}T_{12}\dot{\tilde{q}}_2) - C_2(\dot{q}_2)(\dot{q}_2^d - T_{11}^{-1}T_{12}\tilde{q}_2) + \tau)$$

ou

$$\tau = T_{11}^{-1}u + M_2(\ddot{q}_2^d - T_{11}^{-1}T_{12}\dot{\tilde{q}}_2) + C_2(\dot{q}_2)(\dot{q}_2^d - T_{11}^{-1}T_{12}\tilde{q}_2). \quad (4.16)$$

Considerando a dinâmica do erro de acompanhamento, (4.15), o critério de de-

sempenho que inclui um nível de atenuação de distúrbio desejado γ é definido como

$$\min_{u(\cdot) \in \mathcal{L}_2} \max_{0 \neq w(\cdot) \in \mathcal{L}_2} \frac{\int_0^\infty \left(\frac{1}{2} \tilde{x}^T(t) Q \tilde{x}(t) + \frac{1}{2} u^T(t) R u(t) \right) dt}{\int_0^\infty \left(\frac{1}{2} w^T(t) w(t) \right) dt} \leq \gamma^2, \quad (4.17)$$

sendo Q e R matrizes de ponderação simétricas definidas positivas e $\tilde{x}(0) = 0$. Este critério de desempenho é semelhante ao apresentado em (4.2), sendo que neste caso matrizes de ponderação são acrescentadas no estado e na entrada de controle.

A solução do problema de controle $\mathcal{H}_\infty$ (4.17) relacionado à equação de estado (4.12), pode ser explicitamente encontrada pela teoria dos jogos diferenciais [BASAR E BERNHARD (1990); BASAR E OLSDER (1982)] e uma escolha apropriada da função de Lyapunov $V(\tilde{x}, t)$ [CHEN et al. (1994)].

A metodologia apresentada por [CHEN et al. (1994)] para resolver este problema é resumida a seguir. O critério de desempenho (4.17) pode ser modificado para formar o seguinte problema *minimax*

$$\min_{u(\cdot) \in \mathcal{L}_2} \max_{0 \neq w(\cdot) \in \mathcal{L}_2} \int_0^\infty \left(\frac{1}{2} \tilde{x}^T(t) Q \tilde{x}(t) + \frac{1}{2} u^T(t) R u(t) - \frac{1}{2} \gamma^2 w^T(t) w(t) \right) dt \leq 0,$$

com $\tilde{x}(0) = 0$. Definindo a função custo

$$J(\tilde{x}(t), u, w, t) = \int_t^\infty L(\tilde{x}(s), u(s), w(s)) ds,$$

sendo $L(\tilde{x}, u, w)$ o Lagrangiano dado por

$$L(\tilde{x}, u, w) = \frac{1}{2} \tilde{x}^T(t) Q \tilde{x}(t) + \frac{1}{2} u^T(t) R u(t) - \frac{1}{2} \gamma^2 w^T(t) w(t).$$

Definindo-se a função de Lyapunov

$$V(\tilde{x}(t), t) = \min_{u(\cdot)} \max_{w(\cdot)} J(\tilde{x}(t), u, w, t),$$

o critério de desempenho (4.17) fica

$$V(\tilde{x}(0), 0) = \min_{u(\cdot)} \max_{w(\cdot)} J(\tilde{x}(0), u, w, 0) \leq 0,$$

com $\tilde{x}(0) = 0$. De acordo com a teoria dos jogos diferenciais, a solução deste problema *minimax* é encontrada se existir uma função de Lyapunov continuamente diferenciável $V(\tilde{x}, t)$ que satisfaz a seguinte equação *minimax* de Bellman-Isaacs

$$-\frac{\partial V(\tilde{x}, t)}{\partial t} = \min_{u(\cdot)} \max_{w(\cdot)} \left\{ L(\tilde{x}, u, w) + \left(\frac{\partial V(\tilde{x}, t)}{\partial \tilde{x}} \right)^T \tilde{x} \right\},$$

com condição terminal $V(\tilde{x}(\infty), \infty) = 0$. Escolhendo a função de Lyapunov da forma

$$V(\tilde{x}, t) = \frac{1}{2} \tilde{x}^T P(\tilde{x}, t) \tilde{x}, \quad (4.18)$$

sendo $P(\tilde{x}, t)$ uma matriz simétrica definida positiva para todo t , a equação de Bellman-Isaacs fornece a seguinte equação de Riccati

$$\begin{aligned} \dot{P}(\tilde{x}, t) + P(\tilde{x}, t) A_T(\tilde{x}, t) + A_T^T(\tilde{x}, t) P(\tilde{x}, t) \\ - P(\tilde{x}, t) B_{T_1}(\tilde{x}, t) \left(R^{-1} - \frac{1}{\gamma^2} T_{11}^2 \right) B_{T_1}^T(\tilde{x}, t) P(\tilde{x}, t) + Q = 0. \end{aligned}$$

O controle ótimo correspondente e o pior caso de distúrbio são dados, respectivamente, por

$$u^* = -R^{-1} B_{T_1}^T(\tilde{x}, t) P(\tilde{x}, t) \tilde{x}$$

e

$$w^* = \frac{1}{\gamma^2} B_{T_2}^T(\tilde{x}, t) P(\tilde{x}, t) \tilde{x}.$$

Com uma escolha apropriada da matriz $P(\tilde{x}, t)$ e sendo a matriz $(C_2(\dot{q}_1) - \frac{1}{2} \dot{M}_2)$ anti-simétrica [CHEN et al. (1994)], a equação de Riccati pode ser simplificada para uma equação matricial algébrica. De acordo com [CHEN et al. (1994); JOHANSSON (1990)] uma escolha apropriada para $P(\tilde{x}, t)$ é feita em função de M_2 , T_0 e uma matriz K simétrica definida positiva a ser determinada, ou seja,

$$P = T_0^T \begin{bmatrix} M_2 & 0 \\ 0 & K \end{bmatrix} T_0. \quad (4.19)$$

A equação algébrica simplificada é dada por

$$\begin{bmatrix} 0 & K \\ K & 0 \end{bmatrix} - T_0^T [I \ 0]^T \left(R^{-1} - \frac{1}{\gamma^2} T_{11}^2 \right) [I \ 0] T_0 + Q = 0. \quad (4.20)$$

O controle ótimo e o pior distúrbio podem ser reescritos, respectivamente como

$$u^* = -R^{-1} [I \ 0] T_0 \tilde{x} \quad (4.21)$$

e

$$w^* = \frac{T_{11}}{\gamma^2} [I \ 0] T_0 \tilde{x}.$$

A condição terminal é satisfeita para esta escolha de $P(\tilde{x}, t)$, [CHEN et al. (1994)]. Então, para solucionar o problema de controle $\mathcal{H}_\infty$, deve-se encontrar matrizes K e T_0 que resolvam a equação algébrica (4.20). Considere a matriz Q fatorada da seguinte forma

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix}, \quad (4.22)$$

sendo $Q_{11} =: q_{11}I$, $Q_{12} =: q_{12}I$, $Q_{22} =: q_{22}I$ e também $K =: kI$, $R =: r_u I$ e $T_{12} = t_{12}I$ com q_{11} , q_{12} , q_{22} , k , r_u , t_{11} e t_{12} positivos. Então, (4.20) é reduzida em três equações algébricas

$$q_{11} - t_{11}^2 r_u^{-1} + \gamma^{-2} t_{11}^4 = 0, \quad (4.23)$$

$$q_{22} - t_{12}^2 r_u^{-1} + \gamma^{-2} t_{11}^2 t_{12}^2 = 0, \quad (4.24)$$

$$k + q_{12} - t_{11} t_{12} r_u^{-1} + \gamma^{-2} t_{11}^3 t_{12} = 0. \quad (4.25)$$

O Algoritmo 4.1 mostra como encontrar t_{11} , t_{12} e $k > 0$ para o menor $\gamma > 0$.

Algoritmo 4.1 Seqüência de passos para resolver (4.20), sendo que a sobrescrito significa iteração atual

- 1 - Escolha um $\gamma > 0$ de valor elevado;
 - 2 - Escolha também $t_{11}^a \gg \sqrt{q_{11}r_u}$;
 - 3 - Utilizando o valor de t_{11}^a calcule, através de (4.23), o nível de atenuação γ^a ;
 - 4 - Calcule t_{12}^a em (4.24) utilizando t_{11}^a e γ^a ;
 - 5 - Calcule k^a em (4.25) usando t_{11}^a , t_{12}^a e γ^a ;
 - 6 - Se $k^a > 0$ e $\gamma^a < \gamma$ então

$\gamma \leftarrow \gamma^a$;
 $t_{11} \leftarrow t_{11}^a$;
 $t_{12} \leftarrow t_{12}^a$;
 $k \leftarrow k^a$;
 - 7 - Se $\gamma^a > \gamma^{a-1}$ então pare. Caso contrário continue;
 - 8 - Diminua t_{11}^a ;
 - 9 - Se $t_{11}^a > \sqrt{q_{11}r_u}$ então retorne ao passo 3. Caso contrário pare.
-

Capítulo 5

Determinação da Postura de RMRs

A localização do robô será realizada encontrando a posição de centros de círculos em etiquetas fixadas sobre o robô em imagens bi-dimensionais monocromáticas. Para os métodos de detecção de círculos nas Seções 5.1 e 5.2, define-se a convenção utilizada para processamento de imagens, como mostrado na Fig. 5.1, sendo L o nú-

Figura 5.1: Convenção: (a) imagem contínua, (b) imagem digital.

mero de linhas e C o número de colunas da imagem digital, $f(i, j)$ a intensidade do nível de cinza da imagem na coordenada (i, j) com $f(i, j) \in \{0, 1, 2, 3, 4, \dots, 254, 255\}$, sendo o valor 0 representando a cor preta e 255 a branca.

5.1 Método usando Momentos

Este método possibilita encontrar o centro de círculos em imagens, calculando as coordenadas dos seus centróides. O *Momento*, apresentado em [SONKA et al. (1998)], de ordem $(p + q)$ de uma função contínua bi-dimensional é definido como

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} i^p j^q f(i, j) di dj$$

com $p, q \in \{0, 1, 2, \dots\}$.

Em imagens digitalizadas, tem-se

$$m_{pq} = \sum_i \sum_j i^p j^q f(i, j),$$

sendo que o somatório $(\sum_i \sum_j)$ só ocorre quando a intensidade do nível de cinza, $f(i, j)$, corresponde à intensidade da região de interesse (círculo). A área da região é dada por

$$m_{00} = \sum_i \sum_j f(i, j),$$

sendo $f(i, j) \neq 0$. As coordenadas do centróide, normalizada pela área da região é dada por

$$\bar{i} = \frac{m_{10}}{m_{00}} = \frac{\sum_i \sum_j i f(i, j)}{m_{00}} \quad (5.1)$$

e

$$\bar{j} = \frac{m_{01}}{m_{00}} = \frac{\sum_i \sum_j j f(i, j)}{m_{00}}. \quad (5.2)$$

O algoritmo 5.1 mostra como encontrar a postura de um robô usando o método dos Momentos.

Algoritmo 5.1 Seqüência de passos para encontrar centros de círculos em imagem usando o método dos Momentos e determinar a postura de um RMR

- 1 - Coloque sobre o robô duas etiquetas de círculos de intensidade de nível de cinza diferentes, por exemplo, para um campo de atuação do robô de cor escura coloque um círculo branco com centro no eixo de simetria representando a frente e um círculo cinza claro com centro no ponto P_o do robô;
 - 2 - Calibre o sistema capturando uma imagem e verificando a faixa de valores dos *pixels* de cada círculo;
 - 3 - Em cada quadro de imagem capturada encontre a posição do centro do círculo branco, denotado como $(\bar{i}_b, \bar{j}_b)$, e do círculo cinza, denotado como $(\bar{i}_c, \bar{j}_c)$, usando (5.1) e (5.2) para cada faixa de valores correspondentes aos respectivos círculos;
 - 4 - A posição do robô será $(x_o, y_o) = (\bar{i}_c, \bar{j}_c)$ e a direção será dada pela inclinação da reta que passa por $(\bar{i}_b, \bar{j}_b)$ e $(\bar{i}_c, \bar{j}_c)$.
-

5.2 Método usando Transformada de Hough Circular

Esta seção mostra um método para encontrar o centro de circunferências de raio fixo, R . Para isto utiliza-se uma máscara que também é uma circunferência de raio R , cuja função é atualizar os valores de uma matriz de dimensão $(L+2R) \times (C+2R)$ definida como Matriz Acumuladora.

O método Transformada de Hough Circular, mostrado em [SONKA et al. (1998)], consiste em posicionar o centro da máscara nos *pixels* da circunferência da imagem e os valores dos *pixels* nas posições correspondentes à circunferência da máscara na matriz acumuladora (inicialmente zerada) são incrementadas de uma unidade. A Fig. 5.2b ilustra bem o método, sendo as circunferências de linhas contínuas a contribuição da máscara, os losangos em negrito os *pixels* da circunferência da imagem considerada e o círculo em negrito a região de maior contribuição.

Após percorrer toda a imagem a matriz acumuladora estará totalmente atualizada, e, portanto, a posição do *pixel* de maior valor, denotada por $f_{max}(i, j)$, será considerada o centro da circunferência na matriz acumuladora e conseqüentemente o centro da circunferência na imagem será dado por

$$\bar{i} = i - R$$

Figura 5.2: Transformada de Hough Circular: (a) imagem, (b) matriz acumuladora.

e

$$\bar{j} = j - R.$$

O algoritmo 5.2 mostra como encontrar a postura de um robô usando a THC.

Algoritmo 5.2 Seqüência de passos para encontrar centros de circunferências em imagem usando a THC e determinar a postura de um RMR

- 1 - Coloque sobre o robô três etiquetas de circunferências de raio R (cor branca, por exemplo, para um campo de atuação de cor escura), tais que a média das duas posições dos centros das duas circunferências, localizadas no fundo do robô, seja o ponto P_o e a outra circunferência localizada no eixo de simetria, na frente do robô, bem afastada das outras duas;
 - 2 - Calibre o sistema capturando uma imagem e verificando o menor valor dos *pixels* pertencentes às circunferências;
 - 3 - A cada quadro de imagem capturada, verificar quais os *pixels* são supostamente pertencentes às circunferências (Valor do nível de cinza maior ou igual ao menor valor encontrado na calibração) e atualizar a matriz acumuladora, conforme apresentada anteriormente;
 - 4 - Encontre a posição na matriz acumuladora de maior valor de nível de cinza $f_{max}(i, j)$. E, portanto, a posição da primeira circunferência na imagem será $(\bar{i}_1, \bar{j}_1) = (i - R, j - R)$;
 - 5 - Zerar uma região na matriz acumuladora nas proximidades de $(i - R, j - R)$;
 - 6 - Encontre a posição na matriz acumuladora de maior valor de nível de cinza $f_{max}(i, j)$. E, portanto, a posição da segunda circunferência na imagem será $(\bar{i}_2, \bar{j}_2) = (i - R, j - R)$;
 - 7 - Zerar uma região na matriz acumuladora nas proximidades de $(i - R, j - R)$;
 - 8 - Encontre a posição na matriz acumuladora de maior valor de nível de cinza $f_{max}(i, j)$. E, portanto, a posição da terceira circunferência na imagem será $(\bar{i}_3, \bar{j}_3) = (i - R, j - R)$;
 - 9 - A posição do robô, $(x_o, y_o) = (\bar{i}_o, \bar{j}_o)$, será a média entre as posições dos dois centros das circunferências mais próximas;
 - 10 - A direção do robô será dada pela inclinação da reta que passa por $(\bar{i}_o, \bar{j}_o)$ e a posição do centro da circunferência mais distante.
-

Capítulo 6

Resultados

6.1 Simulação computacional dos controladores

Os resultados em simulação foram obtidos utilizando-se os parâmetros nominais da plataforma LABMATE® [YAMAMOTO E YUN (1992)], ou seja, $m_p = 30(Kg)$, $m_r = 1(Kg)$, $a = 2(m)$, $b = 0.75(m)$, $d = 0.3(m)$, $r = 0.15(m)$ e os momentos de inércia $I_c = 15.625(Kg.m^2)$, $I_r = 0.005(Kg.m^2)$ e $I_m = 0.0025(Kg.m^2)$. Uma incerteza paramétrica representando 10% dos parâmetros nominais foi considerada no modelo dinâmico e cinemático do robô.

A trajetória de referência é $x_r = 0.52t$ e $y_r = 2\sin(0.3t)$, com a condição inicial $(x_{o_o}, y_{o_o}, \alpha_o) = (0, 0, 0)$ e tempo de amostragem $t_s = 0.05$. Para o controlador baseado na cinemática, os ganhos foram selecionados como $k_x = 7$, $k_y = 52$ e $k_\alpha = 27$. Para permitir uma análise de robustez dos controladores, distúrbios externos (veja Fig.6.1) foram introduzidos nos torques das rodas da forma

$$w_d = 80e^{-(t-6)^4} \sin(1.3\pi t) \text{ e } w_e = -80e^{-(t-6)^4} \sin(1.3\pi t).$$

Figura 6.1: Distúrbios de torque aplicados às rodas.

A Fig. 6.2 mostra o diagrama de blocos completo da estratégia de controle do robô móvel. O controle usando um controlador baseado na dinâmica do tipo Proporcional Derivativo em conjunto com Torque Calculado (PD + TC) será apresentado na seção 6.1.3.

Figura 6.2: Sistema de controle de acompanhamento de trajetória para Robôs Móveis.

6.1.1 Controle via representação quase-LPV

A matriz do sistema $A(\dot{q}_2)$ está em função apenas das velocidades das rodas $\dot{q}_2$. Portanto, no projeto do controlador quase-LPV pode-se considerar como função dos erros de velocidades $\dot{\tilde{q}}_2$. Assim o parâmetro ρ é definido como $\rho(\tilde{x}) = \dot{\tilde{q}}_2$. O conjunto P é dado por

$$-\pi \leq \dot{\tilde{\theta}}_d \leq \pi(\text{rad/s}) \quad \text{e} \quad -\pi \leq \dot{\tilde{\theta}}_e \leq \pi(\text{rad/s}),$$

sendo $\nu = [\ddot{\tilde{\theta}}_{d_{\max}} \quad \ddot{\tilde{\theta}}_{e_{\max}}] = [2.5\pi \quad 2.5\pi](\text{rad/s}^2)$, e P dividido em $N = 3$ para cada parâmetro $\dot{\tilde{\theta}}_d$ e $\dot{\tilde{\theta}}_e$. As funções bases dependentes dos parâmetros selecionadas são

$$f_1 = 10 + \cos \dot{\tilde{\theta}}_d + \sin \dot{\tilde{\theta}}_d + \cos \dot{\tilde{\theta}}_e + \sin \dot{\tilde{\theta}}_e, \quad f_2 = \cos \dot{\tilde{\theta}}_d + \sin \dot{\tilde{\theta}}_e \quad \text{e} \quad f_3 = -\sin \dot{\tilde{\theta}}_d + \cos \dot{\tilde{\theta}}_e.$$

A desigualdade (4.5) é resolvida usando o software MATLAB® [GAHINET et al. (1995)], sendo que em (4.3)

$$x = \tilde{x}, \quad A(\rho(t)) = \begin{bmatrix} A(\dot{\tilde{q}}_2) & 0 \\ I & 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} I_{2 \times 2} \\ 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} B \\ 0 \end{bmatrix},$$

$$C_1 = I_{4 \times 4} \quad \text{e} \quad C_2 = 0.$$

As matrizes X_i são encontradas para o melhor nível de atenuação $\gamma = 4.3$ (testes realizados com as perturbações e incertezas paramétricas) e são dadas por

$$X_1 = 10^{-2} \times \begin{bmatrix} 2.732 & -0.058 & -2.037 & -0.433 \\ -0.058 & 2.905 & 0.481 & -2.181 \\ -2.037 & 0.481 & 3.254 & 0.003 \\ -0.433 & -2.181 & 0.003 & 3.379 \end{bmatrix},$$

$$X_2 = 10^{-2} \times \begin{bmatrix} -2.730 & 0.058 & 2.035 & 0.433 \\ 0.058 & -2.904 & -0.481 & 2.179 \\ 2.035 & -0.481 & -3.252 & -0.003 \\ 0.433 & 2.179 & -0.003 & -3.377 \end{bmatrix},$$

e

$$X_3 = 10^{-2} \times \begin{bmatrix} 2.723 & -0.058 & -2.030 & -0.432 \\ -0.058 & 2.896 & 0.479 & -2.173 \\ -2.030 & 0.479 & 3.244 & 0.003 \\ -0.432 & -2.173 & 0.003 & 3.368 \end{bmatrix}.$$

Os resultados são mostrados nas Figs. 6.3 a 6.10.

Figura 6.3: Acompanhamento de trajetória de referência usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.4: Erros de posição usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.5: Erro de direção usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.6: Derivada dos erros de posição usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.7: Derivada do erro de direção usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.8: Velocidade angular da roda direita usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.9: Velocidade angular da roda esquerda usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

Figura 6.10: Torques τ_d e τ_e aplicados às rodas direita e esquerda, respectivamente, usando o controlador quase-LPV: (a) sem distúrbio (b) com distúrbio.

6.1.2 Controle via Teoria dos Jogos (TJ)

O controlador baseado na TJ foi obtido seguindo a sequência abaixo:

1. Valores escolhidos: $q_{11} = 0.3$, $q_{12} = 0$, $q_{22} = 1.2$ e $r_u = 0.7$;
2. Usando o Algoritmo 4.1 da Seção 4.2, obteve-se: $t_{11} = 0.65$, $\gamma = 0.77$, $t_{12} = 1.3$ e $k = 0.6$.

Os resultados são mostrados nas Figs. 6.11 e 6.18.

Figura 6.11: Acompanhamento de trajetória de referência usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.12: Erros de posição usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.13: Erro de direção usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.14: Derivada dos erros de posição usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.15: Derivada do erro de direção usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.16: Velocidade angular da roda direita usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.17: Velocidade angular da roda esquerda usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

Figura 6.18: Torques τ_d e τ_e aplicados às rodas direita e esquerda, respectivamente, usando o controlador TJ: (a) sem distúrbio (b) com distúrbio.

6.1.3 Controle Proporcional Derivativo e Torque Calculado (PD+TC)

O controle PD consiste de um controle independente para cada roda, dado por

$$\begin{aligned} u_d &= -[\theta_d - \theta_d^d]k_{p1} - [\dot{\theta}_d - \dot{\theta}_d^d]k_{d1}, \\ u_e &= -[\theta_e - \theta_e^d]k_{p2} - [\dot{\theta}_e - \dot{\theta}_e^d]k_{d2}, \end{aligned} \quad (6.1)$$

sendo os ganhos proporcionais e derivativos ajustados em $K_{p1} = K_{p2} = 10.9$ e $K_{d1} = K_{d2} = 5$, tais que os erros de postura sem distúrbios e incertezas fiquem próximos aos dos erros dos controladores via controle $\mathcal{H}_\infty$. Os torques aplicados ao robô móvel são obtido considerando (4.13), que de fato é a equação conhecida do torque calculado.

Os resultados são mostrados nas Figs. 6.19 a 6.26.

Figura 6.19: Acompanhamento de trajetória de referência usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

Figura 6.20: Erros de posição usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

Figura 6.21: Erro de direção usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

Figura 6.22: Derivada dos erros de posição usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

Figura 6.23: Derivada do erro de direção usando o controlador PD: (a) sem distúrbio (b) com distúrbio.

Figura 6.24: Velocidades angulares da roda direita usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

Figura 6.25: Velocidades angulares da roda esquerda usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

Figura 6.26: Torques τ_d e τ_e aplicados às rodas direita e esquerda, respectivamente, usando o controlador PD + TC: (a) sem distúrbio (b) com distúrbio.

6.1.4 Estudo comparativo

Os controladores foram comparados baseados nos seguintes índices de desempenho

- Norma $\mathcal{L}_2$ dos erros de posição e velocidade

$$\mathcal{L}_2[e] = \left(\frac{1}{(t_r - t_0)} \int_{t_0}^{t_r} \|e\|_2^2 dt \right)^{\frac{1}{2}},$$

$$\text{sendo } e = \begin{bmatrix} q_e & \dot{q}_e \end{bmatrix}^T,$$

- Energia total

$$E[\tau] = \sum_{i=1}^2 \left(\int_{t_0}^{t_r} \|\tau_i\| dt \right)$$

sendo $t_0 = 0$ até $t_r = 20(s)$ o tempo de simulação e $\tau_i(t)$ o torque da roda i (direita ou esquerda).

Os resultados estão apresentados na Tabela 6.1.

Fazendo uma análise da Tabela 6.1 e observando as figuras provenientes da simulação dos controladores, pode-se afirmar que sem distúrbio e sem incertezas paramétricas os erros de acompanhamento de trajetória são praticamente os mesmos com menor fornecimento de energia realizado pelo controlador PD + TC. Entretanto, quando os distúrbios e incertezas estão presentes o controlador quase-LPV apresen-

Tabela 6.1: Índices de desempenho.

Controladores	γ	Distúrbios externo e incertezas paramétricas	$\mathcal{L}_2[e]$	$E[\tau]$
quase-LPV	3.7	presente	0.50348	285.1
		ausente	0.41728	81.4
TJ	0.77	presente	0.54702	286.5
		ausente	0.42011	69.4
PD+TC	—	presente	0.57647	288.8
		ausente	0.42285	67.5

apresentou melhor desempenho no erro de acompanhamento de trajetória de referência e na energia fornecida ao robô.

6.2 Implementação de localização por visão computacional

Protótipo do campo de atuação: Foi construída uma armação de ferro (veja Fig. 6.27), cuja parte superior possui um suporte para câmera e na base é posicionada uma plataforma (campo de atuação). Um protótipo do robô é feito de papel cartão preto (cor da plataforma) e colado sobre uma peça metálica. Este protótipo é colocado sobre a plataforma e movimentado através de um ímã posicionado na parte de baixo da plataforma.

Câmera: Utilizou-se um câmera da empresa COHU, Inc. Electronics Division série 2200 e acoplada a esta uma lente da empresa Rainbow modelo H10×8M-II que possui um controlador de ajuste de zoom, íris e foco.

Computador: Utilizou-se um computador Pentium 4 de CPU 3.2GHz e 2GB de RAM e o sistema operacional usado é o Linux Debian Kernel versão 2.2.20. Os programas foram feitos em linguagem C e os movimentos do protótipo do robô foram visualizados em uma ambiente gráfico baseado em OpenGL que utiliza as bibliotecas GL.h, GLU.h e GLUT.h.

Placa de aquisição: Utilizou-se uma placa de aquisição de imagem chamada

Figura 6.27: Protótipo do campo de atuação.

PXC200F da empresa ImaginationCorporation. Na implementação obteve-se 30 quadros capturados por segundo, ou seja, 33 *ms* por quadro.

Na verificação das posições dos centros dos círculos levou-se em consideração apenas a imagem, pois, na implementação dos controladores também será referenciado apenas a imagem, já que a trajetória de referência nas coordenadas inerciais devem ser transformada adequadamente para o espaço da imagem.

As imagens capturadas têm resolução de 320×240 *pixels* referente a área da plataforma de 24×32 *cm* e a dimensão do robô é de 7×4 *cm*. Os valores em níveis de cinza das imagens consideradas nos programas serão os valores das imagens capturadas pela câmera.

A lente acoplada a câmera é ajustada (foco e íris) para evitar os ruídos provenientes da variação da iluminação ambiente.

6.2.1 Método dos Momentos

O protótipo do robô utilizado para a análise do desempenho do método dos Momentos está ilustrado na Fig. 6.28, sendo que o centro do círculo cinza representa

Figura 6.28: Protótipo do robô para o método dos momentos.

a posição do robô, P_o , e a direção, α , será o ângulo entre a reta que passa pelos centros dos círculos e a linha da imagem que passa pelo centro do círculo cinza (maiores detalhes no Programa A.7). Os círculos utilizados têm 6 mm de raio e as faixas de valores de níveis de cinza encontradas na calibração são $110 < f(i_b, j_b) < 118$ e $83 < f(i_c, j_c) < 89$. Para verificar a precisão e a robustez do método, os seguintes procedimentos foram seguidos e os resultados estão descritos na Tabela 6.2

- 1 - Deixou-se o protótipo do robô parado sobre a plataforma;
- 2 - Executou-se o Algoritmo 5.1 e foram medidos a média aritmética do tempo de processamento, as posições e a direção do protótipo do robô para 1000 iterações consecutivas, sem adição de ruídos;
- 3 - Executou-se o Algoritmo 5.1 e movimentou-se, com a utilização do imã, um papel contendo ruídos, ou seja, pontos, retas e círculos da mesma cor dos círculos do protótipo do robô na plataforma;
- 4 - Atribuiu-se a uma linha da imagem valores de cor branca (Ruído 1) no Algoritmo 5.1, conforme ilustrado na Fig. 6.29;
- 5 - Executou-se o Algoritmo 5.1 e foram medidos a média aritmética do tempo de processamento, as posições e a direção do protótipo do robô para 1000 iterações consecutivas;

- 6 - Atribuiu-se a uma linha da imagem valores de cor cinza (Ruído 2) no Algoritmo 5.1, conforme ilustrado na Fig. 6.29;
- 7 - Executou-se o Algoritmo 5.1 e foram medidos novamente a média aritmética do tempo de processamento, as posições e a direção do protótipo do robô para 1000 iterações consecutivas;
- 8 - Capturou-se uma imagem e foram medidas as posições dos centros dos círculos através da função *impixel* no Matlab;
- 9 - Calculou-se a direção através dos valores dos centros dos círculos obtidos no Matlab.

Figura 6.29: Imagem do protótipo do robô na plataforma para o método dos Momentos. Os ruídos foram editados para efeito de ilustração.

Tabela 6.2: Tempo de processamento e postura do protótipo do robô usando o método dos momentos.

	Sem Ruído				Ruído 1				Ruído 2			
	Tempo (ms)	Postura			Tempo (ms)	Postura			Tempo (ms)	Postura		
		i_o	j_o	$\alpha(^{\circ})$		i_o	j_o	$\alpha(^{\circ})$		i_o	j_o	$\alpha(^{\circ})$
Momentos	0.40	166	170	312.52	0.41	166	170	66.64	0.42	106	165	290.61
Imagem	—	166	170	312.65	—	166	170	312.65	—	166	170	312.65

6.2.2 Método da Transformada de Hough Circular

O mesmo experimento da Seção 6.2.1 foi utilizado para a análise do desempenho do método via Transformada de Hough Circular. Neste caso utilizou-se o protótipo mostrado na Fig 6.30, sendo a posição do robô a média aritmética entre as posições

Figura 6.30: Protótipo do robô para a THC.

dos dois centros de circunferências mais próximos e a direção será o ângulo, no sentido anti-horário, entre a reta que passa pela posição do robô e a posição do centro da circunferência que representa a frente e a linha da imagem que passa pela posição do robô.

Cada circunferência tem raio externo de 6 mm e espessura da borda de 1 mm. Na calibração encontrou-se o menor valor de níveis de cinza (limiar) dado por $f_{min}(i, j) = 110$. Então valores acima de $f_{min}(i, j)$ serão considerados pertencentes a uma circunferência. Para verificar a precisão e a robustez foram executados os mesmos procedimentos da Seção 6.2.1, sendo os resultados apresentados na Tabela 6.3 de acordo com o Algoritmo 5.2 e os ruídos ilustrados na Fig. 6.31.

Tabela 6.3: Tempo de processamento e postura do protótipo do robô usando a THC.

	Sem Ruído				Ruído 1				Ruído 2			
	Tempo (ms)	Postura			Tempo (ms)	Postura			Tempo (ms)	Postura		
		i_o	j_o	$\alpha(^{\circ})$		i_o	j_o	$\alpha(^{\circ})$		i_o	j_o	$\alpha(^{\circ})$
THC	1.35	165	168	312.77	1.64	165	168	312.77	1.71	165	168	312.77
Imagem	—	165	168	312.77	—	165	168	312.77	—	165	168	312.77

Com a análise dos resultados das Tabelas 6.2 e 6.3 verifica-se que o método dos momentos possui um tempo de processamento menor e exige condições de iluminação adequada para bom desempenho, ou seja, é muito susceptível a ruídos. Já a

Figura 6.31: Imagem do protótipo do robô na plataforma para o método da THC. Os ruídos foram editados para efeito de ilustração.

THC possui uma grande robustez, ou seja, a localização da postura do protótipo só é prejudicada se o ruído possuir dimensão maior ou igual a das circunferências e o tempo de processamento é relativamente alto comparado com o método dos Momentos, porém é suficiente para implementação em um sistema de controle de RMRs.

Capítulo 7

Conclusão e Trabalho Futuro

Dois controladores robustos baseados no modelo dinâmico do robô, usando técnicas de controle $\mathcal{H}_\infty$, foram propostos para o problema de acompanhamento de trajetória de RMRs. Um estudo comparativo entre os controladores considerando incertezas paramétricas e distúrbios externos, foi apresentado. Um controlador proporcional derivativo em conjunto com um controlador do tipo torque calculado foi utilizado e serviu como referência de análise para o estudo comparativo dos controladores robustos. Simulações mostraram que os resultados com relação ao erro de acompanhamento da trajetória de referência para os três controladores são praticamente os mesmos quando os distúrbios e incertezas estão ausentes, sendo que o menor valor da energia é fornecido pelo controlador proporcional derivativo. Entretanto, quando os distúrbios e incertezas estão presentes o controlador quase-LPV apresentou melhor desempenho no erro de acompanhamento da trajetória de referência e na energia fornecida ao robô.

Dois métodos para encontrar a postura de RMRs também foram analisados. O método dos momentos possui um tempo de processamento menor, mas é muito susceptível a ruídos. Já o método usando a transformada de Hough circular apresentou grande robustez a ruídos, ou seja, apenas ruídos de dimensão maior ou igual a da circunferência prejudica a determinação da postura do RMR.

Pretende-se implementar os controladores propostos utilizando um robô chamado SmartROB, desenvolvido pelo Instituto de Robótica (IFR) do Instituto Federal de

Tecnologia da Suíça. A posição e direção do robô serão encontradas com o método usando a THC com a imagem obtida por uma câmera localizada no teto do laboratório, o controle será realizado remotamente por um computador que receberá a leitura dos *encoders* e enviará o valor dos torques via porta serial.

Referências Bibliográficas

AICARDI, M.; CASALINO, G.; BICCHI, A.; BALESTRINO, A. (1995). Closed loop steering of unicycle like vehicles via Lyapunov techniques. *IEEE Robotics & Automation Magazine*, v.2, n.1, p.27–35.

ALMEIDA, J.; PEREIRA, F. L.; SOUSA, J. B. (1997). A hybrid feedback control system for a nonholonomic car-like vehicle. In: *IEEE International Conference on Robotics and Automation*. 1997, volume 3, p. 2614 – 2619.

APKARIAN, P.; ADAMS, R. J. (1998). Advanced Gain-Scheduling Techniques for Uncertain Systems. *IEEE Transactions on Control Systems Technology*, v.6, n.1, p.21–32.

BASAR, T.; BERNHARD, P. (1990). *H_∞ -Optimal Control and Related Minimax Problems*. Berlin: Birkhauser.

BASAR, T.; OLSDER, J. (1982). *Dynamic Noncooperative Game Theory*. New York: Academic Press.

BIANCHI, R. A. C.; SIMÕES, A. S.; COSTA, A. H. R. (2001). Comportamentos Relativos para Seguir Pistas em um Robô Móvel Guiado por Visão. *Simpósio Brasileiro de Automação Inteligente*, , n.5, Canela, .

BLOCH, A. M.; MCCLAMROCH, N. H. (1989). Control of Mechanical Systems with Classical Nonholonomic Constraints. In: *IEEE Conference on Decision and Control*. 1989, volume 1, p. 201–205.

BLOCH, A. M.; MCCLAMROCH, N. H. (1990). Controllability and stabilizability

properties of a nonholonomic control system. In: *IEEE Conference on Decision and Control*. 1990, volume 3, p. 1312 – 1314.

BLOCH, A. M.; REYHANOGLU, M.; MCCLAMROCH, N. H. (1992). Control and Stabilization of Nonholonomic Dynamic Systems. *IEEE Transactions on Automatic Control*, v.37, n.11, p.1746–1757. OK

BOZMA, O.; KUC, R. (1994). A physical model-based analysis of heterogeneous environments using sonar-ENDURA method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.16, n.5, p.497 – 506. OK

BROCKETT, R. W. (1983). Asymptotic stability and feedback stabilization. In: *Differential Geometric Control Theory*, R. W. Brockett, R. S. Millman and H. J. Sussmann. 1983, Boston, MA: Birkhäuser, p. 181–191.

CAMPION, G.; BASTIN, G.; D'ANDRÉA-NOVEL, B. (1996). Strutural Properties and Classification of Kinematic and Dynamic Model of Wheeled Mobile Robots. *IEEE Transactions on Robotics and Automation*, v.12, n.1, p.47–62. OK

CHEN, B. S.; LEE, T. S.; FENG, J. H. (1994). A nonlinear $\mathcal{H}_\infty$ Control design in robotic systems under parameter perturbation and external disturbance. *International Journal of Control*, v.59, n.2, p.439–461.

COELHO, P. (2001). Aplicação de Lagrangianos na Dinâmica de Robôs Móveis com Rodas. *Technical Report, No. ISR-LCIR-2001/01 (Instituto de sistemas e robótica), Universidade de Coimbra, Portugal*.

COELHO, P.; NUNES, U. (2003). Lie Algebra Application to Mobile Robot Control: A Tutorial. *Robotica*, v.21p.483 – 493.

CORRADINI, M. L.; IPPOLITI, G.; LONGHI, S. (2003). Neural Networks Based Control of Mobile Robots: Development and Experimental Validation. *journal of Robotic Systems*, v.20, n.10, p.587–600.

DO, K. D.; JIANG, Z. P.; PAN, J. (2004). Simultaneous tracking and stabilization of mobile robots: an adaptive approach. *IEEE Transactions on Automatic Control*, v.49, n.7, p.1147 – 1151. OK

- FIERRO, R.; LEWIS, F. L. (1995). Control of a nonholonomic Mobile Robot: Backstepping Kinematics into Dynamics. In: *IEEE Conference on Decision and Control*. 1995, p. 3805 – 3810.
- FIERRO, R.; LEWIS, F. L. (1998). Control of a nonholonomic mobile robot using neural networks. *IEEE Transactions on Neural Networks*, v.9, n.4, p.589 – 600.
- FUKAO, T.; NAKAGAWA, H.; ADACHI, N. (2000). Adaptive tracking control of a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*, v.16, n.5, p.609 – 615.
- GAHINET, P.; NEMIROVSKI, A.; LAUB, A. J.; CHILALI, M. (1995). *LMI Control Toolbox*. The MathWorks Inc.
- GOLDSTEIN, H. (1981). *Classical Mechanics*. Addison-Wesley Publishing Company.
- GUPTA, G. S.; MESSOM, C. H.; DEMIDENKO, S. (2005). Real-time identification and predictive control of fast mobile robots using global vision sensing. *IEEE Transactions on Instrumentation and Measurement*, v.54, n.1, p.200 – 214.
- HO, C. T.; CHEN, L. H. (1995). A Fast Ellipse/Circle Detector Using Geometric Symmetry. *Pattern Recognition*, v.28, n.1, p.117–124.
- HUANG, Y.; JADBABAIE, A. (1998). Nonlinear $\mathcal{H}_\infty$ Control: An Enhanced Quasi-LPV Approach. In: *IEEE International Conference on Decision and Control*, 37. 1998, Tampa, Florida, USA. Workshop in $\mathcal{H}_\infty$ nonlinear control by J. C. Doyle, Caltech.
- HWANG, C. K.; CHEN, B. S.; CHANG, Y. T. (2004). Combination of Kinematical and Robust Dynamical Controllers for Mobile Robotics Tracking Control: (I) Optimal $\mathcal{H}_\infty$ Control. In: *Proc. IEEE International Conference on Control Applications*. 2004, p. 1205–1210.
- JI, M.; ZHANG, Z.; BISWAS, G.; SARKAR, N. (2003). Hybrid fault adaptive control of a wheeled mobile robot. *IEEE/ASME Transactions on Mechatronics*, v.8, n.2, p.226 – 233.

- JIANG, Z.-P.; NIJMEIJER, H. (1999). A recursive technique for tracking control of nonholonomic systems in chained form. *IEEE Transactions on Automatic Control*, v.44, n.2, p.265 – 279. OK
- JOHANSSON, R. (1990). Quadratic Optimization of Motion Coordination and Control. *IEEE Transactions on Automatic Control*, v.35, n.11, p.1197–1208. OK
- KANAYAMA, Y.; KIMURA, Y.; MIYAZAKI, F.; NOGUCHI, T. (1990). A Stable Tracking Control Method for an Autonomous Mobile Robot. In: *Proc. IEEE International Conference on Robotics and Automation*. 1990, p. 384 – 389.
- KIMME, C.; SKLANSKY, J. (1975). Finding circles by an array of accumulators. *Communication of the ACM* 18(2), p. 120–122.
- KRSTIC, M.; KANELAKOPOULOS, I.; KOKOTOVIC, P. (1995). *Nonlinear and Adaptive Control Design*. John Wiley and Sons, Inc., 605 Third Avenue, New York, NY.
- LEE, P.; DEAN, T.; YAP, A.; WALTER, D.; KITCHEN, L.; BARNES, N. (2004). On-board vision using visual-servoing for RoboCup F-180 league mobile robots. *ROBOCUP 2003: ROBOT SOCCER WORLD CUP VII LECTURE NOTES IN COMPUTER SCIENCE* 3020, p. 422–433.
- LEE, T.-C.; SONG, K.-T.; LEE, C.-H.; TENG, C.-C. (2001). Tracking control of unicycle-modeled mobile robots using a saturation feedback controller. *IEEE Transactions on Control Systems Technology*, v.9, n.2, p.305 – 318.
- LIU, H.; ZHA, H.; LIN, F. (2004). Competition analysis system for soccer robots based on global vision and trajectory restrictions. *IEEE International Conference on Systems, Man and Cybernetics*, v.6p.5296 – 5300.
- LUCA, A. D.; ORIOLO, G.; PAONE, L.; GIORDANO, P. R. (2002). Experiments in Visual Feedback Control of a Wheeled Mobile Robot. In: *IEEE Conference on Robotics and Automation*. 2002, p. 2073–2078.
- MCLAUGHLIN, R. A.; ALDER, M. D. (1998). The Hough Transform Versus the

UpWrite. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.20, n.4, p.396–400.

M'CLOSKEY, R. T.; MURRAY, R. M. (1997). Exponential stabilization of driftless nonlinear control systems using homogeneous feedback. *Automatic Control, IEEE Transactions on*, v.42, n.5, p.614 – 628.

MORIN, P.; SAMSON, C. (2000). Practical stabilization of a class of nonlinear systems. Application to chain systems and mobile robots. In: *IEEE Conference on Decision and Control*. 2000, volume 3, p. 2989 – 2994.

ORIOLO, G.; LUCA, A. D.; VENDITTELLI, M. (2002). WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation. *IEEE Trans. Contr. Syst. Technol.*, v.10, n.6, p.835–852.

SAMSON, C. (1995). Control of Chained Systems Application to Path Following and Time-Varying Point-Stabilization of Mobile Robots. *IEEE Transactions on Automatic Control*, v.40, n.1, p.64–77.

SARKAR, N.; YUN, X.; KUMAR, V. (1994). Control of Mechanical Systems with Rolling Constraints: Application to Dynamic Control of Mobile Robots. *Int. J. Rob. Res.*, v.13, n.1, p.55–69.

SIQUEIRA, A. A. G.; TERRA, M. H. (2004). Nonlinear and Markovian $\mathcal{H}_\infty$ Controls of Underactuated Manipulators. *IEEE Transaction on Control Systems Technology*, v.12, n.6, p.811–826.

SONKA, M.; HLAVAC, V.; BOYLE, R. (1998). *Image Processing, Analysis, and Machine Vision*. Wiley Inter Science.

VENDITTELLI, M.; ORIOLO, G. (2000). Stabilization of the general two-trailer system. In: *IEEE International Conference on Robotics and Automation*. 2000, volume 2, p. 1817 – 1823.

WU, F. (1995). *Control of linear parameter-varying systems*. 150p. PhD Thesis - Department of Mechanical Engineering, University of California, Berkeley. 1995.

WU, F.; YANG, X. H.; PACKARD, A.; BECKER, G. (1996). Induced $\mathcal{L}_2$ -Norm Control for LPV Systems With Bounded Parameter Variation Rates. *International Journal of Robust and Nonlinear Control*, v.6, n.9-10, p.983–998.

YAGI, Y.; KAWATO, S.; TSUJI, S. (1994). Real-time omnidirectional image sensor (COPIS) for vision-guided navigation. *IEEE Transactions on Robotics and Automation*, v.10, n.1, p.11–22. Ok

YAMAMOTO, Y.; YUN, X. (1992). Coordinating locomotion and manipulation of a mobile manipulator. In: *Proc. IEEE International Conference on Decision and Control*. 1992, p. 1205–1210.

Apêndice A

Programas Computacionais

Este apêndice apresenta partes dos programas para a implementação dos controladores e do módulo de visão. Omitirá, portanto, as bibliotecas ANSI, as declarações de variáveis, diretivas, condições iniciais e passagens não tão relevantes para a compreensão das lógicas utilizadas. As variáveis nos pseudo-códigos serão apresentadas com a mesma simbologia definidas nos capítulos desta dissertação.

A.1 Controladores

As matrizes do modelo em espaço de estado foram calculadas algebricamente para serem utilizadas nos programas em linguagem C. Estas matrizes estão apresentadas a seguir. A matriz M_2 é definida como

$$M_2 = S^T M S =: \begin{bmatrix} M_2^{11} & M_2^{12} \\ M_2^{12} & M_2^{11} \end{bmatrix},$$

sendo $M_2^{11} = c^2(m_p(b^2 + d^2) + 2m_r(b^2 - d^2) + I) + I_r$ e $M_2^{12} = c^2(m_p(b^2 - d^2) + 2m_r(b^2 + d^2) - I)$. A matriz B é definida como

$$B = (M_2)^{-1} =: \begin{bmatrix} B^{11} & B^{12} \\ B^{12} & B^{11} \end{bmatrix},$$

sendo $B^{11} = M_2^{11}/\text{Det}M_2$, $B^{12} = -M_2^{12}/\text{Det}M_2$ e $\text{Det}M_2 = (M_2^{11})^2 - (M_2^{12})^2$. A matriz $C_2(\dot{q}_2)$ é definida como

$$C_2(\dot{q}_2) = S^T CS + S^T M \dot{S} =: \begin{bmatrix} C_2^{11} & C_2^{12} \\ -C_2^{12} & -C_2^{11} \end{bmatrix},$$

sendo $C_2^{11} = -2bc^2 dm_r(1 - c(\dot{\theta}_d - \dot{\theta}_e))$ e $C_2^{12} = -C_2^{11} + 2bc^2 dm_p$. A matriz $A(\dot{q}_2)$ é definida como

$$A(\dot{q}_2) = -BC_2(\dot{q}_2) =: \begin{bmatrix} A^{11} & A^{12} \\ -A^{12} & -A^{11} \end{bmatrix},$$

sendo $A^{11} = B^{12}C_2^{12} - B^{11}C_2^{11}$ e $A^{12} = B^{12}C_2^{11} - B^{11}C_2^{12}$

O Programa A.1 foi desenvolvido para encontrar a solução X de (4.5), utilizando o *software* Matlab.

Programa A.1 *Pseudo-código em Matlab para encontrar a solução das DMLs em (4.5).*

```

:
NvezesN = 0;
for PosVetor $\dot{\theta}_d$  = 1 : N
    for PosVetor $\dot{\theta}_e$  = 1 : N
        NvezesN = NvezesN + 1;
        :
        A(:, :, NvezesN) = -inv(M2) * C2;
        f1(NvezesN) = 1;
        f2(NvezesN) = cos( $\dot{\theta}_d$ (PosVetor $\dot{\theta}_d$ ));
        f3(NvezesN) = sin( $\dot{\theta}_e$ (PosVetor $\dot{\theta}_e$ ));
        f(:, :, NvezesN) = [0 0; -sin( $\dot{\theta}_d$ (PosVetor $\dot{\theta}_d$ )) 0; 0 cos( $\dot{\theta}_e$ (PosVetor $\dot{\theta}_e$ ))];
    end
end
:
setlmis([]); % Inicializa a descrição de um sistema de DMLs
X1 = lmivar(1, [4 1]); % Define a variável X1
X2 = lmivar(1, [4 1]);

```

```

X3 = lmivar(1, [4 1]);
for cont = 1 : NvezesN
    auxf = f(:, :, cont);
    for aux1 = -1 : 2 : 1
        for aux2 = -1 : 2 : 1
            A2 = [A(:, :, cont) zeros(2); eye(2) zeros(2)];
            BRL = newlmi; % DMLs de (4.5)

            % Definindo os termos
            lmiterm([BRL 1 1 X1], -(aux1*νmax*auxf(1,1)+aux2*νmax*auxf(1,2)), 1);
            lmiterm([BRL 1 1 X2], -(aux1*νmax*auxf(2,1)+aux2*νmax*auxf(2,2)), 1);
            lmiterm([BRL 1 1 X3], -(aux1*νmax*auxf(3,1)+aux2*νmax*auxf(3,2)), 1);
            lmiterm([BRL 1 1 X1], f1(cont) * A2, 1, 's');
            lmiterm([BRL 1 1 X2], f2(cont) * A2, 1, 's');
            lmiterm([BRL 1 1 X3], f3(cont) * A2, 1, 's');
            lmiterm([BRL 1 1 0], -B2 * B2T);
            lmiterm([BRL 1 2 X1], f1(cont), C1T);
            lmiterm([BRL 1 2 X2], f2(cont), C1T);
            lmiterm([BRL 1 2 X3], f3(cont), C1T);
            lmiterm([BRL 1 3 0], B1);
            lmiterm([BRL 2 2 0], -1);
            lmiterm([BRL 2 3 0], zeros(2));
            lmiterm([BRL 3 3 0], -γ2);

            Xpos = newlmi; % X > 0
            lmiterm([-Xpos 1 1 X1], f1(cont), 1);
            lmiterm([-Xpos 1 1 X2], f2(cont), 1);
            lmiterm([-Xpos 1 1 X3], f3(cont), 1);
        end
    end
end

DMLs = getlmis; % Obtenção das DMLs
[tmin, xfeas] = feasp(DMLs); % Encontra a solução das DMLs
X1 = dec2mat(DMLs, xfeas, X1) % Soluções caso for factível

```

$$X_2 = \text{dec2mat}(DMLs, xfeas, X_2)$$

$$X_3 = \text{dec2mat}(DMLs, xfeas, X_3)$$

Sendo

$$X = \sum_{i=1}^3 f_i(\dot{q}_2) X_i =: \begin{bmatrix} X^{11} & X^{12} & X^{13} & X^{14} \\ X^{21} & X^{22} & X^{23} & X^{24} \\ X^{31} & X^{32} & X^{33} & X^{34} \\ X^{41} & X^{42} & X^{43} & X^{44} \end{bmatrix}$$

com

$$X_i =: \begin{bmatrix} X_i^{11} & X_i^{12} & X_i^{13} & X_i^{14} \\ X_i^{21} & X_i^{22} & X_i^{23} & X_i^{24} \\ X_i^{31} & X_i^{32} & X_i^{33} & X_i^{34} \\ X_i^{41} & X_i^{42} & X_i^{43} & X_i^{44} \end{bmatrix},$$

Então o controle do RMR via representação quase-LPV, pode ser realizado utilizando um programa em linguagem C como no Programa A.2 e para o controle via TJ como no Programa A.3

Programa A.2 *Pseudo-código em linguagem C para o controle via representação quase-LPV.*

⋮

for(*cont* = 1; *cont* <= *iteracoes*; *cont*++){

$$x_e[cont] = \cos(\alpha[cont]) * (x_r[cont] - x_o[cont]) + \sin(\alpha[cont]) * (y_r[cont] - y_o[cont]);$$

⋮

$$v^d[cont] = v_{or}[cont] * \cos(\alpha_e[cont]) + k_x * x_e[cont];$$

$$\omega^d[cont] = \omega_r[cont] + k_y * v_{or}[cont] * y_e[cont] + k_\alpha * v_{or}[cont] * \sin(\alpha_e[cont]);$$

⋮

$$X^{11} = X_1^{11} + f_2[cont] * X_2^{11} + f_3[cont] * X_3^{11};$$

⋮

$$X^{44} = X_1^{44} + f_2[cont] * X_2^{44} + f_3[cont] * X_3^{44};$$

$$\begin{aligned} \text{Cof } X^{11} = & X^{22} * X^{33} * X^{44} - X^{22} * X^{34} * X^{43} - X^{23} * X^{32} * X^{44} + \\ & X^{23} * X^{34} * X^{42} + X^{24} * X^{32} * X^{43} - X^{24} * X^{33} * X^{42}, \end{aligned}$$

$$\begin{aligned}
& \vdots \\
CofX^{42} &= X^{11} * X^{23} * X^{34} - X^{11} * X^{24} * X^{33} - X^{13} * X^{21} * X^{34} + \\
& \quad X^{13} * X^{24} * X^{31} + X^{14} * X^{21} * X^{33} - X^{14} * X^{23} * X^{31}; \\
DetX &= X^{11} * CofX^{11} + X^{21} * CofX^{21} + X^{31} * CofX^{31} + X^{41} * CofX^{41}; \\
InvX^{11} &= CofX^{11} / DetX; \\
& \vdots \\
InvX^{24} &= CofX^{42} / DetX; \\
u_d[cont] &= -InvX^{11} * (\dot{\theta}_d[cont] - \dot{\theta}_d^d[cont]) - InvX^{12} * (\dot{\theta}_e[cont] - \dot{\theta}_e^d[cont]) - \\
& \quad InvX^{13} * (\theta_d[cont] - \theta_d^d[cont]) - InvX^{14} * (\theta_e[cont] - \theta_e^d[cont]); \\
u_e[cont] &= -InvX^{21} * (\dot{\theta}_d[cont] - \dot{\theta}_d^d[cont]) - InvX^{22} * (\dot{\theta}_e[cont] - \dot{\theta}_e^d[cont]) - \\
& \quad InvX^{23} * (\theta_d[cont] - \theta_d^d[cont]) - InvX^{24} * (\theta_e[cont] - \theta_e^d[cont]); \\
C_2^{11} &= -2 * b * c^2 * d * m_r * (1 - c * (\dot{\theta}_d[cont] - \dot{\theta}_e[cont])); \\
C_2^{12} &= -C_2^{11} + 2 * b * c^2 * d * m_p; \\
A^{11} &= B^{12} * C_2^{12} - B^{11} * C_2^{11}; \\
A^{12} &= B^{12} * C_2^{11} - B^{11} * C_2^{12}; \\
\tau_d[cont] &= M_2^{11} * \ddot{\theta}_d^d[cont] + M_2^{12} * \ddot{\theta}_e^d[cont] - (M_2^{11} * A^{11} - M_2^{12} * A^{12}) * \dot{\theta}_d^d[cont] - \\
& \quad (M_2^{11} * A^{12} - M_2^{12} * A^{11}) * \dot{\theta}_e^d[cont] + M_2^{11} * u_d[cont] + M_2^{12} * u_e[cont]; \\
\tau_e[cont] &= M_2^{12} * \ddot{\theta}_d^d[cont] + M_2^{11} * \ddot{\theta}_e^d[cont] - (M_2^{12} * A^{11} - M_2^{11} * A^{12}) * \dot{\theta}_d^d[cont] - \\
& \quad (M_2^{12} * A^{12} - M_2^{11} * A^{11}) * \dot{\theta}_e^d[cont] + M_2^{12} * u_d[cont] + M_2^{11} * u_e[cont]; \\
& \vdots \\
\ddot{\theta}_d[cont + 1] &= A^{11} * \dot{\theta}_d[cont] + A^{12} * \dot{\theta}_e[cont] + B^{11} * (\tau_d[cont] + w_d[cont]) + \\
& \quad B^{12} * (\tau_e[cont] + w_e[cont]) \\
\ddot{\theta}_e[cont + 1] &= -A^{12} * \dot{\theta}_d[cont] - A^{11} * \dot{\theta}_e[cont] + B^{12} * (\tau_d[cont] + w_d[cont]) + \\
& \quad B^{11} * (\tau_e[cont] + w_e[cont]); \\
& \vdots \\
\dot{\theta}_e[cont + 1] &= \ddot{\theta}_e[cont + 1] * t_s + \dot{\theta}_e[cont]; \\
& \vdots \\
\dot{x}_o &= (r/2) * \cos(\alpha[cont]) * (\dot{\theta}_d[cont + 1] + \dot{\theta}_e[cont + 1]); \\
& \vdots \\
\alpha[cont + 1] &= \dot{\alpha}[cont + 1] * t_s + \alpha[cont]; \\
& \}
\end{aligned}$$

Programa A.3 *Pseudo-código em linguagem C para o controle via TJ.*

```

:
:
for(cont = 1; cont <= iteracoes; cont++){
    xe[cont] = cos(α[cont])*(xr[cont]-xo[cont])+ sen (α[cont])*(yr[cont]-yo[cont]);
    :
    ud[cont] = -(t11/ru) * (θ̇d[cont] - θ̇dd[cont]) - (t12/ru) * (θd[cont] - θdd[cont]);
    ue[cont] = -(t11/ru) * (θ̇e[cont] - θ̇ed[cont]) - (t12/ru) * (θe[cont] - θed[cont]);
    :
    τd[cont] = ud[cont]/t11 + M211*θ̈dd[cont] + M212*θ̈ed[cont] - M211*(t12/t11)*(θ̇d[cont] -
        θ̇dd[cont]) - M212*(t12/t11)*(θ̇e[cont] - θ̇ed[cont]) + C211*θ̇dd[cont] + C212*
        θ̇ed[cont] - C211*(t12/t11)*(θd[cont] - θdd[cont]) - C212*(t12/t11)*
        (θe[cont] - θed[cont]);
    τe[cont] = ue[cont]/t11 + M212*θ̈dd[cont] + M211*θ̈ed[cont] - M212*(t12/t11)*(θ̇d[cont] -
        θ̇dd[cont]) - M211*(t12/t11)*(θ̇e[cont] - θ̇ed[cont]) - C212*θ̇dd[cont] - C211*
        θ̇ed[cont] + C212*(t12/t11)*(θd[cont] - θdd[cont]) + C211*(t12/t11)*
        (θe[cont] - θed[cont]);
    :
}

```

A.2 Ambiente gráfico

O ambiente gráfico foi construído em OpenGL em linguagem C e o Programa A.4 apresenta partes do código fonte com as principais funções.

Programa A.4 *Parte do código fonte, em Linguagem C, do ambiente gráfico .*

```

//Bibliotecas ANSI
#include < GL/gl.h > //gl.h é a biblioteca de ambiente gráfico OpenGL
#include < GL/glu.h > /*glu.h é a biblioteca que complementa o

```

```

OpenGL (manipulação de matrizes, polígonos e outros)*/
#include < GL/glut.h > /*glut.h é a biblioteca para OpenGL de ferra-
    mentas para usuários independente da plataforma*/
//Protótipos
:
GLvoid initGL();
GLvoid window_display();
GLvoid window_reshape(GLsizeiwidth, GLsizeiheight);
void render_scene();
void Robo();
int main(int argc, char ** argv){/*Rotina principal do programa
    responsável por:
        1 - Inicializar OpenGL, GLU e GLUT;
        2 - Cria uma janela usando GLUT;
        3 - Inicializa toda a estrutura de dados necessário para desenhar
            a cena;
        4 - Declara ao GLUT quais funções vai manobrar os eventos;
        5 - Declara ao GLUT para iniciar e esperar por eventos.*/
    glutInit(&argc, argv); //Inicializa a bibliotecas Glut.h
    glutInitDisplayMode(GLUT_RGBA); //Especificação de cores
    :
    initGL(); /*initGL é uma função definida pelo usuário que inicializa
        a descrição tipicamente do OpenGL*/
    glutDisplayFunc(&window_display); /*window_display é uma função
        que redesenha a janela*/
    glutReshapeFunc(&window_reshape); /*window_reshape é uma função
        que ajusta ao tamanho da janela*/
    glutMainLoop(); /*Inicia o laço de eventos da janela, esperando por te-
       clas a serem precionadas, mudança no tamanho ou na cena da janela*/
    return 1;
}
GLvoid initGL(){

```

```

    glClearColor(RED, GREEN, BLUE, ALPHA); //Inicializa a cor de
        fundo da tela com os valores especificados pelo usuário
}
GLvoid window_display() {
    :
    render_scene(); //Redesenha a cena
    glFlush(); //Declara ao OpenGL que apenas redesenha a cena
    glutPostRedisplay(); //Declara ao OpenGL que a janela necessita ser
        redesenhada
}
GLvoid window_reshape(GLsizeiwidth, GLsizeiheight) {
    glClear(GL_COLOR_BUFFER_BIT); //Apaga o que foi desenhado na
        tela, limpando o "buffer" de cores
    :
    glMatrixMode(GL_MODELVIEW); //Aplica ao programa todas as
        transformações
}
void render_scene() {
    : //Processo que encontra a postura do robô
    //Transformação de coordenadas (imagem para a tela do monitor)
     $x_o = (6.0 * posY - 963.0) / 319.0$ ; //posY é a coluna da posição do robô
     $y_o = (-4.0 * posX + 482.0) / 239.0$ ; //posX é a linha da posição do robô
    glColor3f(0.0, 0.0, 1.0); //Cor atual em azul (posição do robô)
    glPointSize(5); //Tamanho do ponto (posição do robô)
    glBegin(GL_POINTS); //Desenha o ponto (posição do robô)
    glVertex3f( $x_o$ ,  $y_o$ , 0);
    glEnd();
    glColor3f(0.0, 1.0, 0.0); //Cor atual em verde (robô)
    glTranslatef( $x_o$ ,  $y_o$ , 0); //Translada o robô de (0,0) para ( $x_o$ ,  $y_o$ )
    glRotatef( $\alpha$ , 0, 0, 1); //Rotaciona o robô de 0° para  $\alpha$  (em graus)
    Robo();
}

```

```
void Robo(){  
    glEnable(GL_TEXTURE_2D);  
    glBegin(GL_POLYGON); //Constroi um triângulo isóceles (Robô)  
    glVertex3f(0, -0.05, 0);  
    glVertex3f(0.3, 0, 0);  
    glVertex3f(0, 0.05, 0);  
    glEnd();  
    glDisable(GL_TEXTURE_2D);  
}
```

A Fig. A.1 mostra o acompanhamento da trajetória de referência do robô, no ambiente gráfico do controle via PD em conjunto com Torque Calculado.

Figura A.1: Ambiente gráfico. O robô é o triângulo em verde, o ponto azul é a posição do robô e em vermelho a trajetória de referência.

A.3 Captura de imagens

A captura da imagem, em níveis de cinza, ocorre com as seguintes sintaxes mostadas no Programa A.5.

Programa A.5 *Parte do código fonte, em Linguagem C, que captura imagens monocromáticas.*

```

:
//Declaração de variáveis
char *devname = " /dev/pxc0pgm";
char *prgname;
FILE *f;
unsigned char *buf; char s[80];
int main(int argc, char **argv){
    prgname = argv[0];
    if(argc > 1 && argv[1][0] == '/') {
        devname = argv[1];
        argc --; argv ++;
    } else if (argc > 1 && argv[argc - 1][0] == '/') {
        devname = argv[argc - 1];
        argc --;
    } else if (getenv("PXCDEVICE")) {
        devname = getenv("PXCDEVICE");
    }
    f = fopen(devname, "r");
    if(!f) {
        fprintf(stderr, "%s : %s : %s\n", prgname, devname, strerror(errno));
        exit(1);
    }
    fgets(s, 80, f);
    if(sscanf(s, " P5%d%d", &xsize, &ysize) != 2) {

```

```

        fprintf(stderr, "%s : %s : notapgmfile\n", prgname, devname);
        exit(1);
    }
    buf = malloc(xsize * ysize);
    if(!buf){
        fprintf(stderr, "%s : malloc() : %s\n", prgname, strerror(errno));
        exit(1);
    }
}

void render_scene(){//Rotina da ambiente gráfico
:
    fgets(s, 80, f); /* skippgmheader */
    if(fread(buf, 1, xsize * ysize, f) != xsize * ysize){
        fprintf(stderr, "%s : %s : read() : %s\n", prgname, devname, strerror(errno));
        exit(1);
    }
:
}

```

O Programa A.6 foi utilizado para verificar o tempo de processamento

Programa A.6 *Sintaxe em Linguagem C para verificação de tempo de processamento.*

```

#include < time.h >
#include < sys/time.h >
void print_time(char *message){
    static char date[80], date1[80], date2[80], date3[80];
    struct timeval tv;
    struct tm *tptr;
    gettimeofday(&tv, NULL);
    tptr = localtime(&tv.tv_sec);

```

```

    strftime(date, 80, "%H : %M : %S", tptr);
    strftime(date1, 80, "%H", tptr);
    strftime(date2, 80, "%M", tptr);
    strftime(date3, 80, "%S", tptr);
    if(message){
        printf("%s.%03li : %s\n", date, tv.tv_usec/1000, message);
    }else{
        printf("\n\n%s", date1);
        printf("\n\n%s", date2);
        printf("\n\n%s", date3);
        printf("\n%03li", tv.tv_usec/1000);
    }
}

int main(){
    :
    print_time(NULL); //Início da contagem de tempo
    printf("\n");
    ://Processo a ser determinado o tempo
    print_time(NULL); //Final da contagem de tempo
    :
}

```

A.4 Localização utilizando visão computacional

A.4.1 Método dos Momentos

O protótipo do robô a ser utilizado no método dos Momentos está ilustrado na Fig. 6.28. O Programa A.7 apresenta as sintaxes para a determinação de postura do robô usando o método dos Momentos.

Programa A.7 *Pseudo-código em Linguagem C para localização usando o método dos Momentos.*

```
//Inicialização dos momentos referentes ao círculo branco
 $m_{00}b = 0$ ;  $m_{10}b = 0$ ;  $m_{01}b = 0$ ;

//Inicialização dos momentos referentes ao círculo cinza
 $m_{00}c = 0$ ;  $m_{10}c = 0$ ;  $m_{01}c = 0$ ;
 $cont = 0$ ;

for( $L = 1$ ;  $L \leq 240$ ;  $L++$ ){//Varre as linhas da imagem
    for( $C = 1$ ;  $C \leq 320$ ;  $C++$ ){//Varre as colunas da imagem
        if(( $buf[cont] \geq 110$ )&&( $buf[cont] \leq 118$ )){/*Verifica se o pixel
(armazenado em  $buf[cont]$ ) pertence ao círculo branco*/
             $m_{00}b = m_{00}b + 1$ 
             $m_{10}b = m_{10}b + L$ ;
             $m_{01}b = m_{01}b + C$ ;
        }
        if(( $buf[cont] \geq 83$ )&&( $buf[cont] \leq 89$ )){
             $m_{00}c = m_{00}c + 1$ ;
             $m_{10}c = m_{10}c + L$ ;
             $m_{01}c = m_{01}c + C$ ;
        }
         $cont++$ ;
    }
}

//Cálculo da posição do Robô( $x_o = \bar{i}$ ,  $y_o = \bar{j}$ )
 $\bar{i} = m_{10}c/m_{00}c$ ;
 $\bar{j} = m_{01}c/m_{00}c$ ;

//Cálculo da posição do centro do círculo branco
 $\bar{i}_b = m_{10}b/m_{00}b$ ;
 $\bar{j}_b = m_{01}b/m_{00}b$ ;

//Encontra a direção do robô em graus
```



```

if((( $\bar{i} - \bar{i}_b$ ) == 0) && ( $\bar{j}_b > \bar{j}$ )){
     $\alpha = 0$ ;
}
if((( $\bar{i} - \bar{i}_b$ ) == 0) && ( $\bar{j}_b < \bar{j}$ )){
     $\alpha = 180$ ;
}
if((( $\bar{j} - \bar{j}_b$ ) == 0) && ( $\bar{i} > \bar{i}_b$ )){
     $\alpha = 90$ ;
}
if((( $\bar{j} - \bar{j}_b$ ) == 0) && ( $\bar{i} < \bar{i}_b$ )){
     $\alpha = 270$ ;
}
if((( $\bar{i} - \bar{i}_b > 0$ ) && ( $\bar{j} - \bar{j}_b < 0$ )){
     $\alpha = \text{atan}((\bar{i} - \bar{i}_b) / -(\bar{j} - \bar{j}_b)) * 180/\pi$ ;
}
if((( $\bar{i} - \bar{i}_b > 0$ ) && ( $\bar{j} - \bar{j}_b > 0$ )){
     $\alpha = 180 - \text{atan}((\bar{i} - \bar{i}_b) / (\bar{j} - \bar{j}_b)) * 180/\pi$ ;
}
if((( $\bar{i} - \bar{i}_b < 0$ ) && ( $\bar{j} - \bar{j}_b > 0$ )){
     $\alpha = 180 + \text{atan}(-(\bar{i} - \bar{i}_b) / (\bar{j} - \bar{j}_b)) * 180/\pi$ ;
}
if((( $\bar{i} - \bar{i}_b < 0$ ) && ( $\bar{j} - \bar{j}_b < 0$ )){
     $\alpha = 360 - \text{atan}((\bar{i} - \bar{i}_b) / (\bar{j} - \bar{j}_b)) * 180/\pi$ ;
}

```

A.4.2 Transformada de Hough Circular

Esta seção mostra como o programa para a determinação da postura do robô usando a THC em imagens é elaborado. A matriz acumuladora, ilustrada na Fig. 5.2b, é transformada em um vetor acumulador, formado pelas linhas da matriz

acumuladora da forma

$$VetAcum = [f(1, 1) f(1, 2) \dots f(1, C+2R) f(2, 1) \dots f(2, C+2R) \dots f(L+2R, C+2R)].$$

Sabendo-se que as circunferências possuem raios fixos e é necessário que o tempo de processamento fosse o menor possível, criou-se um programa no Matlab chamado *THC.m* que gera um arquivo de saída em formato texto salvando-o como *SaidaTHC.c*. Depois é só compilar usando o comando `cc -o SaidaTHC SaidaTHC.c -lm -lglut` e executar usando o comando `./SaidaTHC` em um terminal. Desta forma, foi possível realizar uma grande quantidade de testes para diferentes tamanhos de circunferências e optar pelo melhor. As principais lógicas utilizadas no programa *THC.m* são apresentadas nos passos seguintes.

1. Captura-se uma imagem contendo o protótipo do robô;
2. Uma parte desta imagem é salva em outro arquivo chamado *circunferencia.bmp* e editado pelo usuário (conforme mostrado na Fig. A.2a). Esta imagem, a posição do centro da circunferência (indicada com o mouse pelo usuário) e o valor do linear (obtido pelo histograma da imagem) são as únicas entradas no programa *THC.m* e automaticamente é gerado o arquivo *SaidaTHC.c*;
3. As seqüências lógicas que formam o programa *THC.m* atuam da seguinte forma
 - (i) Identifica a maior distância, R , em pixels do centro a um *pixel* da borda da circunferência;
 - (ii) Constrói um círculo de raio R usando a distância euclidiana (veja a Fig A.2b);
 - (iii) Extrai a borda do círculo, conforme ilustrado na Fig. A.2c;
 - (iv) Cria sintaxes para o arquivo *SaidaTHC.c* que atualiza o vetor acumulador, ou seja, acrescenta uma unidade ao elemento que está a uma distância R do pixel considerado borda da circunferência na imagem (veja o Programa A.8);

- (v) Cria sintaxes para o arquivo *SaidaTHC.c* que zera a região em volta dos dois primeiros centros de circunferência encontrados (veja o Programa A.9);
- (vi) Identifica quais são os centros de circunferências encontrados, ou seja, o centro mais distante é a circunferência que representa a frente do protótipo;
- (vii) Calcula a posição do robô P_o fazendo a média entre os dois centros de circunferências mais próximas;
- (viii) Encontra a direção utilizando as mesmas sintaxes apresentadas no Programa A.7.

A figura abaixo apresenta a evolução da imagem até a formação da máscara.

Figura A.2: Formação da máscara. (a) imagem real; (b) círculo de raio R ; (c) máscara.

Programa A.8 Parte do código fonte de *THC.m* que salva no arquivo *SaidaTHC.c* as sintaxes que atualizará o vetor acumulador.

```

:
saida = fopen('SaidaTHC.c','w');
:
cont2 = 0;
forlinha = 1 : L
    forcoluna = 1 : C
        cont2 = cont2 + 1;
    
```

```

    if(mascara(linha,coluna) == 255)
        Diferenca = PosCentro - cont2; %PosCentro é a posição
do centro da máscara no vetor acumulador
        ModDiferenca = abs(Diferenca);
        if(Diferenca > 0)
            fprintf(saida,'\t\t\tVetorAcum[PosCentro-');
            fprintf(saida,num2str(Diferenca));
            fprintf(saida,'] = VetorAcum[PosCentro-');
            fprintf(saida,num2str(Diferenca));
            fprintf(saida,'] + 1; \n');
        end
        if(Diferenca < 0)
            fprintf(saida,'\t\t\tVetorAcum[PosCentro+');
            fprintf(saida,num2str(ModDiferenca));
            fprintf(saida,'] = VetorAcum[PosCentro+');
            fprintf(saida,num2str(ModDiferenca));
            fprintf(saida,'] + 1; \n');
        end
    end
end
end
end
:

```

Programa A.9 Parte do código fonte de *THC.m* que salva no arquivo *SaidaTHC.c* as sintaxes para zerar regiões do vetor acumulador.

```

:
fprintf(saida,'\t\t\tVetorAcum[pos1 + cont5] = 0; ');
n = 3; % escolhido pelo usuário
for cont = -n : n % n é o numero de píxels, a serem zerados, nas
proximidades do centro da circunferência encontrada
    cont_abs = abs(cont);

```



```

    if(cont == 0)
    Else
        if(cont < 0)
            for cont2 = -n : n
                cont2_abs = abs(cont2);
                if(cont2 == (-n))
                    fprintf(saida, '\n\tVetor Acum(pos-'); % pos é a posição do
máximo valor no vetor acumulador
                    Linha_tirar = cont_abs * (C + 2 * R);
                    Valor_inf = Linha_tirar + cont2_abs;
                    fprintf(a, num2str(Valor_inf));
                    fprintf(saida, ': pos-');
                    Valor_sup = Linha_tirar - cont2_abs;
                    fprintf(a, num2str(Valor_sup));
                    fprintf(saida, ') = 0; ');
                end
            end
        end
        if(cont > 0)
            for cont2 = -n : n
                cont2_abs = abs(cont2);
                if(cont2 == (-n))
                    fprintf(saida, '\n\tVetor Acum(pos+');
                    Linha_colocar = cont_abs * (C + 2 * R);
                    Valor_inf2 = Linha_colocar - cont2_abs;
                    fprintf(a, num2str(Valor_inf2));
                    fprintf(saida, ': pos+');
                    Valor_sup2 = Linha_colocar + cont2_abs;
                    fprintf(a, num2str(Valor_sup2));
                    fprintf(saida, ') = 0; ');
                end
            end
        end
    end

```

```

    end
  end
end
:

```

O Programa A.10 apresenta parte do programa *SaidaTHC.c* em linguagem C para determinação da postura do robô.

Programa A.10 *Parte do código fonte do arquivo saidaTHC.c que mostra a lógica para encontrar o centro de circunferências em imagens.*

```

:
void render_scene(){//Rotina da ambiente gráfico
:
  for(cont = 1; cont <= 84836; cont++){// Zera todo o vetor acumulador
    VetorAcum[cont] = 0;
  }
  PosCentro = 2331;
  cont2 = 0;
  for(L = 1; L <= 240; L++){
    PosCentro = PosCentro + 14;    for(C = 1; C <= 320; C++){
      PosCentro = PosCentro + 1;
      if(buf[cont2] >= 110){/*110 é o menor valor em nível de cinza
        considerado pertencente a circunferência (limiar)*/
/* As sintaxes abaixo são gerados pelo
    Programa A.8*/
      VetorAcum[PosCentro - 2338] = VetorAcum[PosCentro - 2338] + 1;
      VetorAcum[PosCentro - 2007] = VetorAcum[PosCentro - 2007] + 1;
      :
      VetorAcum[PosCentro + 2338] = VetorAcum[PosCentro + 2338] + 1;
    }

```

```

        cont2++;
    }
}
for(cont = 1; cont <= 3; cont++){
    cont3 = 0;
    for(cont2 = 2346; cont2 <= 82490; cont2++){
        if(VetorAcum[cont2] > cont3){
            cont3 = VetorAcum[cont2];
            pos = cont2;
        }
    }
    quoc = pos/334;
    //Recuperação da posição da circunferência na imagem
    L[cont4] = quoc - 7;
    C[cont4] = (pos - quoc * 334) - 7;
    for(cont5 = -3; cont5 <= 3; cont5++){
        VetorAcum[pos + cont5] = 0; //Zera região (Programa A.9)
        VetorAcum[pos - 1002 + cont5] = 0;
        VetorAcum[pos - 668 + cont5] = 0;
        :
        VetorAcum[pos + 1002 + cont5] = 0;
    }
}

D1 = sqrt(pow((L[1] - L[2]), 2) + pow((C[1] - C[2]), 2)); /*Calcula a dis-
tância entre os centros das três circunferências*/
D2 = sqrt(pow((L[1] - L[3]), 2) + pow((C[1] - C[3]), 2));
D3 = sqrt(pow((L[3] - L[2]), 2) + pow((C[3] - C[2]), 2));
if(D1 < D2){
    if(D1 < D3){
        Dmin = D1;
    }else{
        Dmin = D3;
    }
}

```

```

    }
    }else{
    if( $D2 < D3$ ){
         $Dmin = D2$ ;
    }else{
         $Dmin = D3$ ;
    }
    }

    if( $Dmin == D1$ ){/* $Dmin=D1$  significa que ( $L[3],C[3]$ ) é a posição do-
        centro de circunferência na frente do robô.  $L[3]$  é mesmo que  $\bar{i}_b$ 
        e  $C[3]$  é o mesmo que  $\bar{j}_b$ .*/
         $PosX = floor((L[1] + L[2])/2)$ ; //PosX é o mesmo que  $\bar{i}$ 
         $PosY = floor((C[1] + C[2])/2)$ ; //PosY é o mesmo que  $\bar{j}$ 
        :///Encontra a direção como no Programa A.7
    }

    if( $Dmin == D2$ ){/* $Dmin=D2$  significa que ( $L[2],C[2]$ ) é a posição do
        centro de circunferência na frente do robô.  $L[2]$  é mesmo que  $\bar{i}_b$ 
        e  $C[2]$  é o mesmo que  $\bar{j}_b$ .*/
         $PosX = floor((L[1] + L[3])/2)$ ; //PosX é o mesmo que  $\bar{i}$ 
         $PosY = floor((C[1] + C[3])/2)$ ; //PosY é o mesmo que  $\bar{j}$ 
        ://Encontra a direção como no Programa A.7
    }

    if( $Dmin == D3$ ){/* $Dmin=D3$  significa que ( $L[1],C[1]$ ) é a posição do
        centro de circunferência na frente do robô.  $L[1]$  é mesmo que  $\bar{i}_b$ 
        e  $C[1]$  é o mesmo que  $\bar{j}_b$ .*/
         $PosX = floor((L[2] + L[3])/2)$ ; //PosX é o mesmo que  $\bar{i}$ 
         $PosY = floor((C[2] + C[3])/2)$ ; //PosY é o mesmo que  $\bar{j}$ 
        ://Encontra a direção como no Programa A.7
    }
    }
    :
     $xo = (6.0 * posY - 963.0)/319.0$ ;
    :

```



```
}//Final da rotina render_scene()
```

```
:
```