ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR Silas Franco dos Reis Alves E APROVADA PELA COMISSÃO JULGADORA EM 16 / 12 / 2011

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Silas Franco dos Reis Alves

Plataforma de Software para Técnicas de Navegação e Colaboração de Robôs Móveis Autônomos

Campinas, 2011

07/2012

Silas Franco dos Reis Alves

Plataforma de Software para Técnicas de Navegação e Colaboração de Robôs Móveis Autônomos

Dissertação apresentada ao Curso de Mestrado da Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Orientador: João Maurício Rosário
Co-orientador: Humberto Ferasoli Filho

Campinas
2011

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Alves, Silas Franco dos Reis
AL87p Plataforma de software para técnicas de navegação e colaboração de robôs móveis autônomos / Silas Franco dos Reis Alves. --Campinas, SP: [s.n.], 2011.

Orientadores: João Maurício Rosário, Humberto Ferasoli Filho.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Robôs móveis. 2. Navegação de robôs móveis. 3. Sistemas colaboração. 4. Robôs - Programação. 5. Software - Reutilização. I. Rosário, João Maurício. II. Ferasoli Filho, Humberto. III. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. IV. Título.

Título em Inglês: Software platform for autonomous mobile robot navigation and collaboration
Palavras-chave em Inglês: Mobile robot, Mobile robot navigation, Collaboration systems, Robots - Programming, Software - Reuse
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora: Ely Carneiro de Paiva, Leonimer Flávio de Melo
Data da defesa: 16-12-2011
Programa de Pós Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

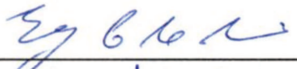
Dissertação de Mestrado

Plataforma de Software para Técnicas de Navegação e Colaboração de Robôs Móveis Autônomos

Autor: Silas Franco dos Reis Alves
Orientador: João Maurício Rosário
Co-orientador: Humberto Ferasoli Filho

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. João Maurício Rosário, Presidente
Universidade Estadual de Campinas



Prof. Dr. Ely Carneiro de Paiva
Universidade Estadual de Campinas

Prof. Dr. Leonimer Flávio de Melo
Universidade Estadual de Londrina

Campinas, 16 de dezembro de 2011

Dedico este trabalho à minha família e aos amigos, que me incentivaram e contribuíram para minha formação.

Agradecimentos

No desenvolvimento deste trabalho, contei com o apoio de diversas pessoas, sem as quais não o teria completado. Assim, agradeço:

À minha família pelo apoio e incentivo em todos os momentos.

Aos professores João Maurício Rosário e Humberto Ferasoli Filho, por terem me orientado e indicado o caminho a trilhar.

Aos todos os colegas do Laboratório de Automação Integrada e Robótica que me ajudaram no desenvolvimento deste trabalho, em especial ao Alvaro Uribe, ao Luciano Frezzato e ao Fabian Lara.

Aos funcionários da UNICAMP, em especial ao Almiro Franco, à Regina Camilo, ao André Santos, à Denise Villela, que me deram apoio, mesmo quando distantes.

Aos professores e funcionários do DCo da UNESP de Bauru, que me acolheram e cederam seu tempo para me ajudar, em especial aos professores René Pegoraro, Marco Caldeira, Wilson Yonezawa, e aos funcionários Sergio Komori e Isméria Rosa.

A todos os meus amigos, que participaram direta ou indiretamente desta jornada.

À FAPESP, pelo financiamento do desenvolvimento deste projeto.

E a todas as pessoas que, apesar de não explicitamente incluídas, contribuíram para o desenvolvimento deste trabalho.

Você não deve mudar uma coisa, uma rocha, um grão de areia, até que você saiba que bem e mal seguirão tal ato. (...) Ele deve seguir o conhecimento, e servir a necessidade. Acender uma vela é criar uma sombra.
(Ursula K. Le Guin; Um Mago de Terramar, 1968)

Resumo

A navegação e a colaboração são aspectos importantes da robótica móvel. A navegação confere aos robôs móveis as habilidades básicas de interação com o ambiente, os obstáculos e agentes nele situado. Já a colaboração permite que os robôs coordenem sua navegação e interação com o ambiente de forma que os permita realizar tarefas complexas de forma rápida e eficiente. Neste trabalho de pesquisa foi desenvolvida uma plataforma de software que oferece suporte a algumas técnicas tradicionais de navegação e colaboração de robôs móveis. Com esta plataforma, é possível programar diferentes robôs com os mesmos componentes de software, o que reduz o tempo de desenvolvimento do aplicativo ao incentivar o reuso de software. Além disso, as técnicas de navegação e colaboração fornecidas pela plataforma amenizam o esforço em desenvolver o software de controle para robôs móveis colaborativos, pois a plataforma permite que o usuário concentre seus esforços na solução dos problemas pertinentes a aplicação do robô, uma vez que as técnicas de navegação e colaboração são fornecidas pela plataforma.

Palavras-Chave: Robôs Móveis, Navegação de Robôs Móveis, Sistemas Colaborativos, Robôs – Programação, Software – Reutilização

Abstract

The navigation and collaboration are important aspects of mobile robotics. The navigation provides to mobile robots the basic skills of interaction with the environment, and the obstacles and agents located therein. The collaboration allows the robots to coordinate their navigation and interaction with the environment in a way that enables them to perform complex tasks quickly and efficiently. This research project developed a software platform that supports some traditional navigation techniques and collaboration of mobile robots. With this platform, different robots can be programmed with the same software components, reducing the application's development time by encourage software reuse. Furthermore, the techniques of navigation and collaboration provided by the platform alleviate the effort to develop the control software for collaborative mobile robots, because the platform allows the user to focus their efforts on solving the problems relevant to the robot's application, since the navigation techniques and collaboration are provided by the platform.

Keywords: *Mobile Robot, Mobile Robot Navigation, Collaboration Systems, Robot – Programming, Software – Reuse*

Lista de Ilustrações

Lista de Abreviaturas e Siglas

Letras Latinas

a	Aceleração do robô móvel
$A(q)$	Matriz associada às restrições não-holonômicas
$B(q)$	Matriz de transformação das entradas de controle
b_0	Coeficiente de atrito viscoso da combinação entre o motor, a redução e as rodas
C	Centro geométrico do robô
C_0	Centro geométrico do primeiro círculo da etiqueta utilizada pela Visão Global
C_1	Centro geométrico do segundo círculo da etiqueta utilizada pela Visão Global
$C(q,\dot{q})$	Matriz de torques de *Coriolis* e centrípetos
CX_lY_l	Sistema de coordenadas local do robô
$C_rX_rY_r$	Sistema de coordenadas de referência
d_a	Distância com a qual o robô móvel deve começar a desacelerar
d_e	Erro de distância
$e_{\dot{\varphi}}$	Erro da velocidade angular
e_p	Erro de posição
e_q	Erro de postura em função do tempo
$f(n)$	Função de custo
$\mathcal{F}_g$	O mesmo que OX_gY_g
$\mathcal{F}_l$	O mesmo que CX_lY_l
$\mathcal{F}_r$	O mesmo que $C_rX_rY_r$
$g(n)$	Custo estimado do nó inicial para o nó n
$h(n)$	Custo de n para o nó objetivo
$H(q)$	Matriz de inércia simétrica definida positiva
i	Número da roda do robô móvel

i_a	Corrente de armadura
I_c	Momento de inércia do robô sem rodas e motores em respeito ao eixo vertical através do ponto P
J_0	Momentos de inércia do motor, redução e roda em referência ao eixo do motor
k_1	Ângulo de erro mínimo aceito para que o robô possa se mover linearmente
k_1	Ganho do controlador de trajetórias
k_2	Ganho da velocidade angular
k_2	Ganho do controlador de trajetórias
k_3	Ganho do controlador de trajetórias
k_t	Fator de ganho de tração
k_d	Ganho derivativo
k_e	Constante de força eletromotriz
k_i	Ganho integrativo
k_n	Taxa de redução
k_p	Ganho proporcional
k_v	Constante de velocidade linear arbitrária
k_τ	Constante de torque do motor
k_ω	Constante de velocidade angular arbitrária
l	Largura da grade
L_a	Impedância da armadura
M	Ponto de meta (objetivo) do robô móvel no plano $\mathcal{F}_g$
m_c	Massa do robô sem rodas e motores
OX_gY_g	Sistema de coordenadas global ou inercial
p	Posição atual do robô
P	Ponto de interseção entre o eixo de simetria e o eixo das rodas
p_r	Postura de referência
q	Vetor de coordenadas generalizadas
$\ddot{q}$	Vetor das acelerações em coordenadas generalizadas
$\dot{q}$	Vetor das velocidades em coordenadas generalizadas

q_0	Posição inicial da trajetória $\mathcal{F}_g$
q_1	Posição final da trajetória $\mathcal{F}_g$
q_f	Ponto localizado no centro da grade final
q_i	Ponto localizado no centro da grade inicial
r	Raio das rodas
R	Distância entre as rodas e o eixo de simetria
R_a	Resistência da armadura
s	Variável da transformada de Laplace
$S(q)$	Matriz jacobiana ou o modelo cinemático
t	Tempo
$\mathcal{T}$	Trajetória no espaço livre do plano
t_{curva}	Tempo necessário para descrever uma curva
t_{reta}	Tempo necessário para descrever uma reta
u	Tensão de armadura aplicada
v	Velocidade linear
$\mathcal{V}$	Vetor de velocidade do robô móvel com tração diferencial $(v, \omega)^{\text{T}}$
v_1	Velocidade linear da primeira roda de um robô de tração omnidirecional
v_2	Velocidade linear da segunda roda de um robô de tração omnidirecional
v_3	Velocidade linear da terceira roda de um robô de tração omnidirecional
v_{max}	Velocidade linear máxima
v_c	Velocidade linear de controle
v_i	Velocidade linear para a i-ésima roda
$v_r(t)$	Velocidade linear de referência em função do tempo
$\mathcal{V}_r(t)$	Vetor de velocidade de referência em função do tempo
w_1	Variável auxiliar
w_2	Variável auxiliar
x	Posição no eixo das abscissas
$\dot{x}$	Velocidade linear no eixo das abscissas
x_M	Posição do ponto M no eixo das abcissas do plano $\mathcal{F}_g$
x_c	Posição do ponto C no eixo das abcissas do plano $\mathcal{F}_g$

$\dot{x}_c$	Velocidade do robô móvel no eixo das abcissas do plano $\mathcal{F}_g$
$\dot{x}_e$	Erro de velocidade linear no eixo das abcissas
$x_g(t)$	Posição x real do robô em função do tempo
$x_g = x_c$	
$\dot{x}_r$	Velocidade linear de referência do robô em relação ao eixo das abcissas do plano $\mathcal{F}_g$
$x_r(t)$	Posição x de referência do robô em função do tempo
y	Posição no eixo das ordenadas
$\dot{y}$	Velocidade linear no eixo das ordenadas
y_c	Posição do ponto C no eixo das ordenadas do plano $\mathcal{F}_g$
$\dot{y}_c$	Velocidade do robô móvel no eixo das ordenadas do plano $\mathcal{F}_g$
$\dot{y}_e$	Erro de velocidade linear no eixo das ordenadas
$y_g(t)$	Posição y real do robô em função do tempo
$y_g = y_c$	
y_M	Posição do ponto M no eixo das ordenadas do plano $\mathcal{F}_g$
$\dot{y}_r$	Velocidade linear de referência do robô em relação ao eixo das ordenadas do plano $\mathcal{F}_g$
$y_r(t)$	Posição y de referência do robô em função do tempo
z_1	Variável auxiliar
$\dot{z}_1$	Variável auxiliar
z_2	Variável auxiliar
$\dot{z}_2$	Variável auxiliar
z_3	Variável auxiliar
$\dot{z}_3$	Variável auxiliar

.....

Letras Gregas

α	Ângulo entre o segmento $\overline{C_0C_1}$ e a abscissa do plano $\mathcal{F}_g$
β_1	Variável auxiliar

β_2	Variável auxiliar
β_3	Variável auxiliar
β_4	Variável auxiliar
Δ_x	Diferença entre p e p_r em respeito ao eixo das abcissas
Δ_y	Diferença entre p e p_r em respeito ao eixo das ordenadas
θ	Ângulo do robô em relação ao eixo das abscissas
$\dot{\theta} = \omega$	
θ_e	Erro angular
$\dot{\theta}_e$	Erro de velocidade angular
$\theta_g(t)$	Ângulo θ real do robô em função do tempo
$\dot{\theta}_r$	Velocidade angular de referência do robô
$\theta_r(t)$	Ângulo θ de referência do robô em função do tempo
ρ	Vetor de forças de restrição
τ	Vetor de torqu
φ_1	Ângulo da primeira roda de um robô de tração omnidirecional
$\dot{\varphi}_1$	Velocidade angular da primeira roda de um robô de tração omnidirecional
φ_2	Ângulo da segunda roda de um robô de tração omnidirecional
$\dot{\varphi}_2$	Velocidade angular da segunda roda de um robô de tração omnidirecional
φ_3	Ângulo da terceira roda de um robô de tração omnidirecional
$\dot{\varphi}_3$	Velocidade angular da terceira roda de um robô de tração omnidirecional
$\dot{\varphi}_d$	Velocidade angular da roda direita de um robô de tração diferencial
$\dot{\varphi}_e$	Velocidade angular da roda esquerda de um robô de tração diferencial
φ_i	Ângulo da i-ésima roda do robô
$\dot{\varphi}_i$	Velocidade angular da i-ésima roda do robô
$\dot{\varphi}_{i,c}$	Velocidade angular de controle da i-ésima roda do robô
$\dot{\varphi}_{\mathrm{m}}$	Velocidade angular do eixo do motor
$\ddot{\varphi}_{\mathrm{m}}$	Aceleração angular do eixo do motor
ω	Velocidade angular
ω_c	Velocidade angular de controle
$\omega_r(t)$	Velocidade angular de referência em função do tempo

Superescritos

®	Marca Registrada

.....

Abreviações

fem	Força Eletromotriz
fps	*Frames per second* (Quadros por segundo)

.....

Siglas

AEDROMO	Ambiente Experimental Didático com Robôs Móveis
API	*Application Programming Interface* (Interface de Programação de Aplicativos)
BER	Base Experimental Robótica
CORBA	*Commom Object Request Broker* (Intermediário Comum para Requisições de Objetos)
DARPA	*Defense Advanced Research Projects Agency* (Agência de Projetos de Pesquisa Avançada de Defesa)
GISDI	Grupo de Integração de Sistemas e Dispositivos Inteligentes
GNU	*GNU is Not Unix* (GNU não é Unix)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
ICE	*Internet Communication Engine* (Motor de Comunicação para Internet)
IEEE	*Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos)
IP	*Internet Protocol* (Protocolo de Internet)
IPC	*Inter-Process Communication* (Comunicação entre Processos)
ISO	*International Organization for Standardization* (Organização Internacional de Padronização)
LAI	Laboratório de Automação Inteligente

LISDI	Laboratório de Integração de Sistemas e Dispositivos Inteligentes
MIRO	*Middleware for Mobile Robots* (Middleware para Robôs Móveis)
NASA	*National Aeronautics and Space Administration* (Administração Nacional do Espaço e da Aeronáutica)
NASREM	*NASA/NBS Standard Reference Model* (O Modelo de Referência da NASA/NBS)
NBS	*National Bureau of Standards* (Bureau Nacional de Padrões)
OO	Orientação a Objetos
OpenCV	*Open Computer Vision* (Visão Computacional Aberta)
PC	*Personal Computer* (Computador Pessoal)
POSIX	*Portable Operating System Interface* (Interface Portável ente Sistemas Operacionais)
ROS	*Robot Operating System* (Sistema Operacional para Robôs)
SLAM	*Simultaneous Localization and Mapping* (Localização e Mapeamento Simultâneos)
SO	Sistema Operacional
SWIG	*Simplified Wrapper and Interface Generator* (Gerador de Interface e Embalador Simplificado)
TCA	*Task Control Architecture* (Arquitetura de Controle de Tarefas)
TCP	*Transmission Control Protocol* (Protocolo com Controle de Transmissão)
UDP	*User Datagram Protocol* (Protocolo de Datagrama de Usuário)
UPnP	*Universal Plug-and-Play* (Plug-and-Play Universal)
WS4D	*Web Services for Devices* (Serviços de Rede para Dispositivos)
XMPP	*Extensible Messaging and Presence Protocol* (Protocolo Extensível de Mensagem e Presença)

Sumário

1 INTRODUÇÃO

A navegação e a colaboração são aspectos importantes da robótica móvel. A navegação confere aos robôs móveis as habilidades básicas de interação com o ambiente, os obstáculos e agentes nele situado. Já a colaboração permite que os robôs coordenem sua navegação e interação com o ambiente de forma que os permita realizar tarefas complexas de forma rápida e eficiente. Durante as últimas décadas, os pesquisadores desenvolveram diferentes técnicas de navegação e colaboração para robôs móveis. Entretanto, experimentar estas técnicas não é uma tarefa corriqueira, pois pode envolver adequações mecânicas, de hardware ou de software. Adequar a estrutura mecânica ou o hardware de um robô móvel é uma tarefa complexa, cara e demorada. Por outro lado, a adequação de software é menos trabalhosa graças às ferramentas e linguagens de programação, muito embora ainda exista uma solução de desenvolvimento ótima.

Ainda que as arquiteturas de software dos robôs comerciais ofereçam componentes de software que abstraem a aquisição de dados sensoriais e o controle de atuadores, há pouco suporte nativo para técnicas de navegação e colaboração. Comumente, as técnicas de navegação e colaboração são implementadas pelo usuário, visto que ambas as áreas são extensas. Contudo, se por um lado a obrigatoriedade de escrever métodos de navegação e colaboração resulte na escolha do método que melhor se adapte a aplicação do robô móvel, por outro lado esta obrigação dificulta o desenvolvimento da aplicação do robô. Neste caso, o usuário deve escrever tanto os algoritmos referentes à aplicação quanto os referentes à navegação e colaboração do robô. Além disso, no que pauta a pesquisa nestas duas áreas, o usuário deve implementar um ou mais métodos tradicionais de navegação e colaboração a fim de compará-los a novos métodos em desenvolvimento.

Neste trabalho de pesquisa foi desenvolvida uma plataforma de software que oferece suporte a algumas técnicas tradicionais de navegação e colaboração de robôs móveis. Com esta plataforma, é possível programar diferentes robôs com os mesmos componentes de software, o que reduz o tempo de desenvolvimento do aplicativo ao incentivar o reuso de software. Além disso, as técnicas de navegação e colaboração fornecidas pela plataforma

amenizam o esforço em desenvolver o software de controle para robôs móveis colaborativos. Isso é possível pois a plataforma permite que o usuário concentre seus esforços na solução dos problemas pertinentes a aplicação do robô, uma vez que as técnicas de navegação e colaboração são fornecidas pela plataforma.

Para que a plataforma possa ser utilizada por diferentes robôs, seja de forma remota ou embarcada, foi utilizado o Sistema Operacional GNU/Linux e bibliotecas portáveis. Desta forma, a plataforma pode ser executada em qualquer hardware que suporte um sistema GNU/Linux.

Para o desenvolvimento desta plataforma, foi realizado um estudo abrangente que abordou: a definição de robôs móveis autônomos, que é o objeto de estudo do trabalho; das técnicas de navegação e colaboração, bem como das arquiteturas de software; que são o foco de estudo do trabalho; e das arquiteturas de controle para robôs móveis, necessárias para orquestrar os diferentes módulos funcionais de um robô móvel.

Finalmente, a plataforma desenvolvida contempla apenas um pequeno conjunto das técnicas de navegação e colaboração, pois ambas as áreas são extensas. Contudo, embora esteja longe de ser uma solução completa, a plataforma aqui proposta resolve algumas dificuldades encontradas no desenvolvimento de robôs móveis através da padronização dos componentes de software que controlam os robôs móveis e da implementação de técnicas de navegação e colaboração também padronizadas.

1.1 Objetivos Gerais

O objetivo geral deste trabalho é desenvolver e avaliar uma plataforma de software que auxilie na pesquisa e no ensino através da criação de componentes de software reutilizáveis voltada para navegação e colaboração de robôs móveis.

1.2 Objetivos específicos

Os objetivos específicos deste trabalho são os seguintes:

- Implementar algumas técnicas de navegação e colaboração os robôs móveis disponíveis nos laboratórios da UNESP e UNICAMP;
- Desenvolver uma plataforma de software que facilite o desenvolvimento do software de controle de robôs móveis através da padronização dos componentes de software para controlar robôs diferentes e da abstração do processo de aquisição de dados, do acionamento de atuadores, e de técnicas de navegação e colaboração.

1.3 Organização da Dissertação

Esta dissertação foi organizada em oito capítulos que descrevem de forma linear o desenvolvimento do trabalho. Os capítulos 1 e 2 tratam da apresentação do trabalho, enquanto os capítulos 3 e 4 discutem o arcabouço teórico do trabalho. Em sequência, os capítulos 5 e 6 apresentam o desenvolvimento do trabalho e seus aspectos práticos. Finalmente, o derradeiro capítulo 7 encerra a dissertação. Neste sentido, o conteúdo de cada um dos capítulos é apresentado a seguir.

O capítulo 1 introduz o tema do trabalho e apresenta seus objetivos gerais e específicos, além de apresentar a organização da dissertação.

O capítulo 2 apresenta o método utilizado para o desenvolvimento deste trabalho e discute em quais capítulos cada etapa do método é descrito.

O capítulo 3 realiza uma síntese da revisão bibliográfica que analisa, mesmo que *en passant* devido à concisão do texto, os vários aspectos de um robô móvel. Nesta síntese, apresenta-se: a definição de robôs móveis autônomos; uma discussão sobre os elementos

que compõe um robô móvel; uma visão geral sobre navegação e colaboração de robôs móveis; a definição e discussão das arquiteturas de controle; e a definição e estudo das arquiteturas de software.

O capítulo 4 aprofunda a discussão da navegação e apresenta as técnicas e métodos escolhidos para comporem este trabalho, além de apresentar e discutir as técnicas de comunicação aqui adotada.

O capítulo 5 introduz e discute a plataforma de software implementada neste trabalho. Nele, são evidenciados como a plataforma dá suporte aos objetivos almejados por este trabalho e como os componentes de software foram desenvolvidos.

Em sequência, o capítulo 6 apresenta os robôs utilizados nos testes práticos e discute o desenvolvimento e resultados dos experimentos realizados com a plataforma desenvolvida. Tais experimentos demonstram como a plataforma foi capaz de alcançar os objetivos propostos para este trabalho, bem como discute o desempenho da arquitetura e das técnicas e métodos adotados.

Finalmente, o capítulo 7 encerra o trabalho com a conclusão e uma discussão dos possíveis trabalhos futuros.

2 MÉTODO

O desenvolvimento deste trabalho foi dividido em seis etapas: estudo da fundamentação teórica; estudo dos robôs móveis; planejamento da plataforma; desenvolvimento do software; testes e validação; documentação final.

Durante o estudo da fundamentação teórica, foi realizada uma revisão bibliográfica sobre os diferentes aspectos dos robôs móveis autônomos – tema deste trabalho. Neste sentido, foi estudada a definição de robótica móvel e autonomia, a navegação de robôs móveis, a colaboração entre robôs móveis, as arquiteturas de controle para robótica e as arquiteturas de software que dão suporte ao desenvolvimento de tais arquiteturas. Concordando com o foco principal do trabalho, foi dada maior atenção aos aspectos de navegação e colaboração de robôs móveis, bem como para as arquiteturas de software, que são discutidos mais profundamente nos capítulos 4 e 5.

Em paralelo ao levantamento do estado da arte, foram estudados os robôs móveis disponíveis para a composição do ambiente, conforme é apresentado na seção 6.1. Esta etapa foi importante pois a maior parte dos robôs móveis disponíveis foi concebida para um uso específico, de modo que seus sensores e atuadores não foram escolhidos para permitir o teste de diversas formas de navegação, mas sim para ser capaz de cumprir sua aplicação. Assim, conforme a pesquisa das técnicas de navegação e colaboração avançava, eram verificados quais destas técnicas poderiam ser implantadas nos robôs existentes e quais arquiteturas de software eram suportadas por eles. Consequentemente, esta etapa foi responsável por determinar as técnicas e métodos de navegação e colaboração discutidos no capítulo4.

No final da primeira e segunda etapa, foram determinadas quais técnicas de navegação e colaboração seriam implementadas e quais as adequações necessárias nos robôs e no ambiente. Com estas informações, é iniciada a terceira etapa: planejamento da plataforma de software. Durante o planejamento da plataforma de software, foi escolhida qual a arquitetura de software seria usada, bem como qual a linguagem de programação e paradigmas de software adotados. A escolha foi feita de tal modo que permitiu tanto a implementação

das técnicas de navegação e colaboração, quanto o reuso dos componentes de software que compõe cada técnica. Logo, os aspectos tecnológicos foram avaliados e definidos durante esta etapa. São várias as ferramentas computacionais e linguagens de programação que podem ser adotadas para o desenvolvimento de sistemas robóticos, portanto foi importante avaliar as soluções de computação disponíveis, desde o sistema operacional até a linguagem e ferramentas.

A quarta etapa segue o planejamento da plataforma de software para desenvolver os componentes de software que compõe a plataforma. O desenvolvimento do software utilizará a metodologia de desenvolvimento Scrum Solo. O Scrum Solo é um método ágil de desenvolvimento de software incremental baseado no Scrum, porém adaptado para equipes pequenas ou para um único programador. No caso deste trabalho, cada iteração do trabalho é definida e discutida junto aos orientadores. Como consequência do método, o software será feito de forma incremental. Esta abordagem apresenta vantagens sobre o método tradicional de desenvolvimento em cascata, uma vez que os métodos tradicionais são sensíveis aos erros e modificações do escopo do trabalho. A evolução em espiral adotada pelos métodos ágeis auxilia na mitigação de falhas, muito embora também sofra com erros descobertos tardiamente.

Tanto o planejamento da plataforma quanto sua implementação são apresentados no capítulo 5.

A penúltima etapa do trabalho consistiu no teste da plataforma para validar seu funcionamento. Para isso, foram elaborados experimentos que demonstram o funcionamento dos diferentes componentes de software desenvolvidos. Tais experimentos são apresentados no capítulo 6.

Finalmente, a última etapa consistiu na redação dos relatórios e da presente dissertação.

3 SÍNTESE DA REVISÃO BIBLIOGRÁFICA

A robótica móvel é por natureza multidisciplinar e, por consequência, absorve os fundamentos teóricos de diferentes áreas do conhecimento. Portanto, a robótica móvel pode ser observada sob diferentes perspectivas que em alguns casos são conflitantes entre si. Este capítulo de revisão bibliográfica apresenta um estudo sistemático dos robôs móveis segundo o ponto de vista da Engenharia e discute, quando oportuno, a divergência deste ponto de vista com aqueles de outras disciplinas, como a Biologia e as Ciências Cognitivas.

3.1 Robôs Móveis Autônomos

Bekey (2005) define um robô como "uma máquina que sente, pensa e age". Isto determina que o robô deve possuir sensores, habilidade de processamento que emule algum aspecto de cognição, e atuadores. Esta é uma definição ampla que engloba diferentes tipos de sistemas automatizados, inclusive aqueles que normalmente não são considerados robôs. Nesta mesma linha de raciocínio, Matarić (2007) apresenta uma definição de robô mais completa: "um robô é um sistema autônomo que existe no mundo físico, pode sentir e agir sobre ele para encontrar algum objetivo". Esta definição é mais concreta, pois introduz o conceito de autonomia e a necessidade de um corpo físico e um objetivo. Ou seja, o robô deve possuir um corpo físico, ser capaz de extrair informações sobre o ambiente, processar estas informações e acionar seus atuadores de forma que ele cumpra sua missão ou objetivo.

Entretanto, essa definição continua ampla e cobre diversos tipos de dispositivos robóticos, desde os robôs manipuladores até os robôs móveis. Os robôs móveis, como o nome implica, são aqueles capazes de locomoverem-se livremente pelo mundo. Neste sentido, a robótica móvel se difere de outras áreas de pesquisa como os manipuladores robóticos convencionais, inteligência artificial e visão computacional, pois enfatiza os problemas relacionados ao entendimento de um ambiente de grande escala, isto é, um ambiente substancialmente maior que aquele visto a partir de uma posição vantajosa (DUDEK; JENKIN, 2010).

A locomoção inteligente em um ambiente de larga escala envolve a aquisição de conhecimento sobre o mundo e o reconhecimento de objetos e lugares dentro dele, para navegar de forma segura e precisa.

Portanto, um robô móvel pode ser considerado um sistema autônomo, que existe no mundo físico, é capaz de sentir, agir e navegar de forma harmoniosa e segura sobre ele para encontrar algum objetivo.

A autonomia, por sua vez, pode ser entendida sob diferentes contextos, como a Medicina, Filosofia, Educação, dentre várias outras. No contexto da robótica, a autonomia é geralmente entendida como a habilidade de relacionar os dados sensoriais ao acionamento dos atuadores de tal forma que seus objetivos sejam atendidos com sucesso sem intervenção externa (MAES, 1994) e por longos períodos de tempo (BROOKS, R. A., 1985). Neste sentido, a autonomia é percebida e mensurada através da eficiência e robustez com as quais um robô realiza suas tarefas em diferentes condições ambientais (ALAMI *et al.*, 1998), reagindo a eventos inesperados tanto no tempo quanto no espaço (FERGUSON, 1994).

Por outro lado, os robôs são muitas vezes comparados aos sistemas biológicos vivos, pois compartilham dos mesmos desafios no desenvolvimento de suas tarefas (BEKEY, 2005). Esta comparação têm inspirado trabalhos e gerado discussões tanto dentro da Robótica (FLOREANO; MATTIUSSI, 2008). No que toca a autonomia dos robôs, autores como Steels (1995) e Varela (1994) postulam que a autonomia não se limita apenas à habilidade de reagir e sobreviver ao ambiente para cumprir uma missão; ela engloba também a capacidade fundamental de existir no ambiente, de perceber, entender e compreender o ambiente e situar-se nele, e de conceber um mundo significante e pertinente sem a necessidade de informações pré-estabelecidas. Esta é uma definição forte e nenhum robô ou agente computacional alcançou esta autonomia.

Apesar de diferentes, ambas as definições de autonomia não são exclusivas. Pelo contrário, elas se complementam e propiciam uma autonomia plena. Neste sentido, a definição de autonomia utilizada neste trabalho divide-se em Autonomia Operacional e Autonomia Cognitiva. A Autonomia Operacional engloba os aspectos da Engenharia e dita que o robô

deve sentir e agir sobre o ambiente, reagir a eventos esperados e inesperados no tempo e no espaço, bem como operar sem intervenção humana, por longos períodos de tempo e com seus próprios recursos para alcançar seus objetivos. Já a Autonomia Cognitiva segue o pensamento das Ciências Cognitivas e é dada através da capacidade do robô de aprender sobre o ambiente através dos seus sentidos e ações, e modificar suas estruturas de processamento para desenvolver este aprendizado sem a necessidade de informações pré-concebidas sobre o mundo. A capacidade de aprendizado resulta em sistemas imprevisíveis, pois o robô pode modificar sua própria programação de tal forma que ele venha a mostrar comportamentos que seu desenvolvedor não pode conceber durante a criação do robô (STEELS, 1995).

Finalmente, não há na literatura um robô que possa ser chamada do completamente autônomo. Assim, um robô que implemente a totalidade, ou a maior parte, dos aspectos da Autonomia Operacional alcança o chamado Alto grau de autonomia.

3.2 Elementos de um Robô Móvel

Tais quais os animais, um robô móvel deve ser capaz de sobreviver no mundo real ao mesmo tempo em que cumpre sua missão. Seu corpo deve estar precisamente alinhado tanto a sua aplicação quanto ao ambiente onde ele existirá, permitindo-lhe atingir seus objetivos enquanto navega harmoniosamente pelo ambiente que o cerca. Alcançar um corpo que atenda a esta definição é uma tarefa complexa e criativa que envolve diferentes áreas do conhecimento humano. Seu processo de concepção deve levar em conta as estruturas necessárias para realizar as tarefas, para se locomover de forma segura, e para colaborar com outros robôs ou com pessoas. Além disso, como o desenvolvimento do robô depende aspectos tecnológicos e econômicos, é imprescindível que sejam consideradas também as limitações tecnológicas e o custo total do robô. Em outras palavras, o processo de criação do corpo deve alcançar o projeto melhor adaptado aos requisitos da aplicação e do mundo. De forma análoga, este processo deve conceber num curto espaço de tempo algo que a nature-

za tomou bilhões de anos para criar. Neste sentido, não é surpresa que muitos projetos de corpo de robô se inspirem nos sistemas vivos encontrados na natureza.

O primeiro passo no desenvolvimento do corpo de um robô móvel é determinar qual, ou quais, os ambientes em que ele existirá. Em geral, os ambientes dividem-se em terrestre, aquático e aéreo. Robôs móveis terrestres são aqueles capazes de se locomover sobre o solo, seja em ambientes internos, como residências ou indústrias, ou em ambientes externos, como aplicações de agricultura ou exploração de planetas. Dentro desta categoria encontram-se também os robôs móveis subterrâneos, aplicados na exploração de cavernas. Os robôs móveis aquáticos são aqueles que vivem sobre a água, de forma semelhante aos navios, enquanto aqueles que vivem sob a água são chamados de subaquáticos. Finalmente, os robôs móveis aéreos são aqueles capazes de mover pelo ar em ambientes fechados ou abertos.

Uma vez determinado o ambiente, suas características e complexidade devem ser estudadas para definir os sistemas básicos que compõe corpo do robô. O sistema de locomoção, responsável pela mobilidade do robô no ambiente, pode se apresentar de diferentes formas e está intimamente ligado ao ambiente (BEKEY, 2005). O uso de rodas, por exemplo, pode apresentar vantagens sobre as pernas robóticas em uma rodovia, mas não apresenta bons resultados em solos irregulares, ao contrário das pernas. Intimamente ligado ao sistema de locomoção, encontra-se o sistema de atuadores, que propicia a movimentação das juntas do robô móvel (SICILIANO; KHATIB, 2008). Por outro lado, o sistema de sensoriamento deve fornecer sensores capazes de extrair do ambiente os parâmetros relevantes tanto para a navegação (EVERETT, 1995) quanto para a aplicação, utilizando técnicas para o tratamento da incerteza associada aos sensores. Adicionalmente, o sistema de computação proporciona o planejamento e a tomada de decisão para os robôs móveis, estando diretamente ligado com sua autonomia. Em geral, o sistema de computação está atrelado a um sistema de comunicação que lhe confere a capacidade de implementar uma arquitetura distribuída (DUDEK; JENKIN, 2010). Ou seja, a comunicação permite que o robô tenha uma base computacional embarcada de pequenas dimensões e baixa potência, o que reduz o peso e o consumo do robô, mas que ainda possa acessar bases de processamento mais poten-

tes para tratar algoritmos mais complexos. Além disso, ela permite a comunicação entre o robô e os demais agentes automatizados presentes no ambiente.

Desenvolver os sistemas que compõe um robô é uma tarefa interativa, pois há uma interdependência entre eles. Quanto mais complexo o sistema de locomoção, por exemplo, mais atuadores e sensores de juntas serão utilizados, o que resultará no aumento do consumo de energia pelo robô. Se sua alimentação elétrica se der através de baterias, um número maior de células de energia será necessário e, consequentemente, maior será o peso do corpo, o que reflete no redimensionamento dos motores e, recursivamente, da fonte de energia. Além disso, o projeto do corpo de um robô esbarra em aspectos: tecnológicos, que ditam sua exequibilidade em termos de Engenharia; econômicos, no que diz respeito ao custo de produção; segurança, garantindo que o robô não causará danos estruturais ou ao ser humano; ou ambientais, como a conformidade com a norma ISO 14000 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, [S.d.]), que orienta o desenvolvimento de produtos e serviços no intuito de reduzir o impacto ambiental. Ainda devem ser considerados os aspectos sociais do robô quando este estará inserido num ambiente habitado por pessoas. Neste sentido, seu corpo deve ser concebido para inspirar confiança e segurança nas pessoas e, portanto, deve incorporar uma forma agradável e possuir uma interface simpática para interagir com elas.

3.3 Navegação

A capacidade de se locomover pelo ambiente é a habilidade primordial dos robôs móveis e os separa dos manipuladores robóticos. A esta capacidade de locomoção, damos o nome navegação, que pode ser entendida como um sistema que responde às perguntas levantadas por Leonard e Durrant-Whyte (1991): “onde estou?”, “para onde vou?” e “como chego lá?”.

A localização responde à primeira pergunta: “onde estou”. Em outras palavras, uma técnica de localização deve retornar a posição do robô no mundo, o que pode se dar tanto

de forma relativa quanto absoluta (BORENSTEIN *et al.*, 1995). Os sistemas de navegação relativa fornecem dados de posição em relação ao ponto de partida do robô, ao contrário dos sistemas absolutos que fornecem dados de posição em relação a um modelo global, como latitude e longitude, por exemplo. Borenstein (1995) divide os sistemas de posicionamento relativo em:

1. **Odometria** é, talvez, a forma de localização mais comumente encontrada nos robôs móveis, pois sensores que são naturalmente utilizados para medir a velocidade dos motores que movem rodas. Além disso, é uma técnica matematicamente simples. Entretanto, os dados fornecidos pela odometria são imprecisos, pois sua natureza integrativa resulta numa taxa crescente de erros através do tempo.

2. **Navegação inercial** utiliza acelerômetros e giroscópios para medir a variação de movimento executada por um robô móvel. Tal qual a odometria, são de natureza integrativa, o que ao longo do tempo resulta em crescente taxa de erros.

Quanto às técnicas de posicionamento absoluto, Borenstein (1995) as divide em:

1. **Bússolas magnéticas** são capazes de informar a orientação de um robô móvel por virtualmente todo o globo terrestre, graças a seu campo eletromagnético natural. Entretanto, não apresenta boa precisão em ambientes internos pois o campo eletromagnético terrestre é distorcido próximo a estruturas metálicas e linhas de força.

2. **Marcadores ativos** (*active beacons*) são artefatos que emitem um sinal interferente no mundo que é reconhecido pelo robô, como sinais de rádio frequência ou luz estruturada. Ao reconhecer três ou mais marcadores, um robô é capaz de estimar sua posição através da triangulação ou trilateração.

3. **Sistema de Posicionamento Global** (GPS) é um tipo de marcador ativo que utiliza uma rede de satélites artificiais. . É o método mais comumente adotado

de localização, pois permite estimar a posição tridimensional do veículo, bem como a data e hora atual, em qualquer ponto da superfície da Terra.

4. **Navegação por marcos** (*landmarks*) utiliza características do ambiente que são detectadas e reconhecidas pelo sensor do robô, não sendo necessariamente reconhecidas por seres humanos (NEHMZOW; OWEN, 2000). Estes marcos podem ser: naturais, como plantas, rochas, prédios, portas, lâmpadas no teto; ou artificiais, caracterizando-se por objetos cuja função primária é servir como ponto de referencia, sendo ajustados para facilitar seu reconhecimento e prover informações adicionais.

5. **Casamento de modelos** (*model matching*) utiliza características do ambiente para construir um mapa ou para reconhecer um ambiente dentro de um mapa conhecido previamente. A construção de um mapa envolve problemas de exploração autônoma e modelagem do ambiente. Já os desafios do reconhecimento de ambiente é encontrar a correspondência entre o mapa local (ambiente) e o mapa global conhecido.

Esta divisão das técnicas de localização apresentada por Borenstein (1995) é resumida pelo diagrama da Figura 3.1.

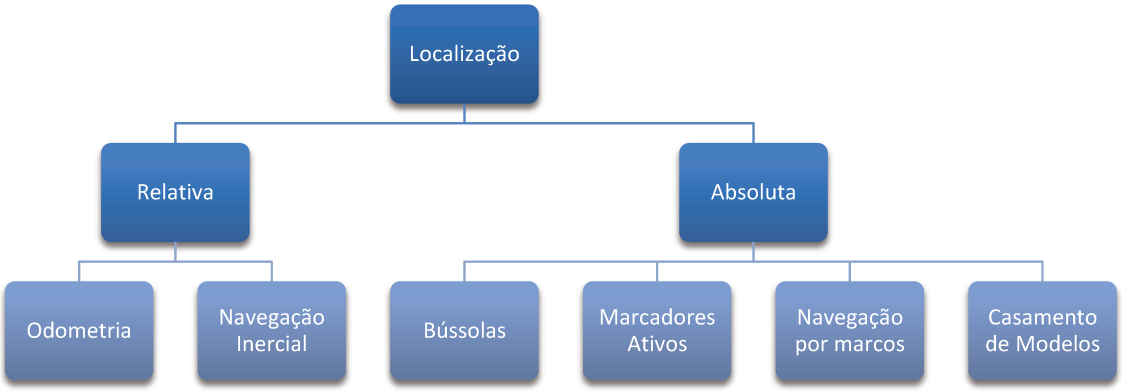


Figura 3.1 – Divisão das técnicas de localização apresentada por Borenstein (1995).

Adicionalmente, no intuito de melhorar os dados fornecidos pelos sistemas de localização, são aplicadas técnicas probabilísticas, como o filtro de Kalman, cadeias de Markov, e o método de Monte Carlo (DELLAERT *et al.*, 1999).

A segunda pergunta, “para onde vou?” é respondida pela aplicação do robô. Como o objetivo deste trabalho é desenvolver uma plataforma de software que forneça suporte a técnicas de navegação e colaboração que independam da aplicação do robô móvel, a discussão da aplicação do robô móvel foge do escopo do trabalho.

Finalmente, a terceira questão, “como chego lá?”, diz respeito ao planejamento de caminho, encarregado de traçar a rota que o robô móvel deve seguir para alcançar seu objetivo. Em geral, esta discussão se inicia com o problema da representação do mundo e dos objetos e seres que nele existem – inclusive o robô móvel. Esta representação se dá através do uso de mapas, que em geral podem ser classificados em mapas geométricos ou topológicos. Em mapas geométricos, o ambiente é representado através das geometrias dos obstáculos nele presentes, como mostra a Figura 3.2 (b). O problema da representação do robô em mapas geométricos é resolvido, por exemplo, com diagramas de Voronoi generalizados, que reduzem o robô a um ponto, enquanto aumentam a geometria dos obstáculos com o tamanho do robô (SICILIANO; KHATIB, 2008). Esta técnica permite verificar facilmente os lugares onde um robô não é capaz de passar, e propicia técnicas de planejamento de caminho como os campos potenciais (DUDEK; JENKIN, 2010). Por outro lado, os mapas topológicos não utilizam uma medida precisa, baseando-se principalmente em navegação por marcos. Este tipo de mapa, conforme mostra a Figura 3.2 (c), é representado tipicamente por grafos (BEKEY, 2005) e, portanto, pode utilizar os métodos de busca da teoria de grafos para encontrar o melhor caminho. Complementarmente, os mapas geométricos podem ser discretizados em mapas de grade de ocupação, o que permite utilizar, também, os métodos de busca da teoria dos grafos.

De forma complementar, Murphy (MURPHY, 2000) propõe uma quarta pergunta, que é “por onde estive?”, que remete ao problema resolvido pela técnica de Localização e Mapeamento Simultâneos (SLAM). A SLAM remete ao problema de adquirir o mapa do ambiente onde um robô móvel está situado enquanto, simultaneamente, localiza o robô em relação a este mapa. Ele envolve técnicas de posicionamento relativo e de posicionamento absoluto (principalmente o casamento de modelos). O SLAM é considerado um dos principais desafios para a construção de um robô verdadeiramente autônomo. É um campo relativamente

jovem que apresenta um progresso enorme, mas que deixa aberto uma série de questões a serem solucionadas (SICILIANO; KHATIB, 2008).

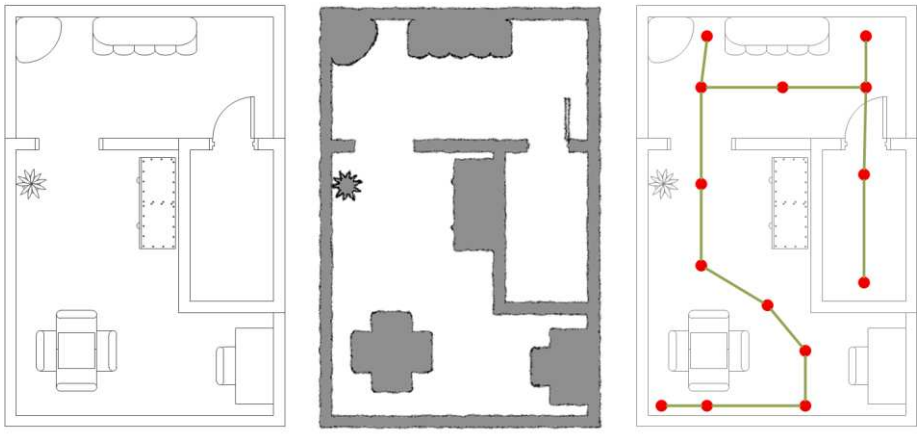


Figura 3.2 – Exemplos de mapa: (a) planta original do ambiente; (b) mapa métrico da planta; (c) mapa topológico sobreposto à planta.

Contudo, as técnicas de navegação clássica[1] desenvolvidas nas ultimas décadas ainda não alcançaram a flexibilidade de desempenho dos mecanismos de navegação de formigas e abelhas. Isto motivou pesquisadores a estudarem e implementarem comportamentos de navegação observados em agentes biológicos, principalmente insetos (FRANZ; MALLOT, H. A., 2000)FRANZ *et al.*, 2003)

(FRANZ et al., 2003)(FRANZ et al., 2003)(FRANZ et al., 2003).

Franz e Mallot (2000) elaboraram um relatório que descreve os fenômenos da navegação baseados na literatura biológica recente. Os autores dividem as técnicas de navegação biomiméticas em dois grupos: navegação local e descoberta de caminho. A navegação local trata dos problemas básicos da navegação, como o desvio de obstáculos e seguimento de trilhas, para deslocar o robô do ponto de partida (conhecido ou não) a um destino conhecido e dentro do campo de visão robô. Já a descoberta de caminho trata de estratégias que utilizam a navegação local para guiar o robô do ponto de partida até um destino conhecido, porém fora do campo de visão do robô. Um exemplo que ilustra ambos os grupos é encontrar uma sala de aula dentro de uma universidade: a descoberta de caminho elege uma se-

[1] O termo "navegação clássica" foi utilizado aqui para distinguir as técnicas de navegação que buscam responder as três questões apresentadas por Durrant-Whyte.

quencia de destinos que levam até a sala (geralmente baseados em marcos), enquanto a navegação local se encarrega do deslocamento até estes destinos.

3.4 Colaboração entre Robôs Móveis

A colaboração é um comportamento observado na natureza de diversas formas. Colônias de formigas, vespas, abelhas e outros insetos distribuem o trabalho de encontrar, trazer e estocar alimento. Em aves migratórias, pode-se observar o voo em formação que não apenas mantém o grupo reunido, mas também reduz o gasto de energia resultante do atrito com o ar e facilita a orientação e comunicação entre os pássaros. A colaboração também é identificada em seres humanos, que se organizam em grupos para realizar as mais diversas atividades.

São vários os exemplos encontrados na natureza que demonstram as vantagens potenciais de trabalhar cooperativamente. Examinando estes exemplos é possível verificar que robôs colaborativos (ou distribuídos) oferecem diversas vantagens, como a redução do tempo gasto na realização de tarefas e até mesmo a diminuição o preço total do sistema robótico (PARKER, LYNNE E, 1998). A adoção de robôs distribuídos pode aumentar a confiabilidade, flexibilidade, robustez e eficiência de soluções automatizadas graças à redundância e paralelismo (PARKER, LYNNE E, 2000).

Quanto à composição, um sistema robótico colaborativo é classificado de acordo com as características intrínsecas dos robôs que o compõe. Neste sentido, há duas categorias de sistemas robóticos colaborativos (CAO *et al.*, 1997): homogêneos e heterogêneos.

Sistemas homogêneos pressupõem que todos os robôs do grupo possuem as mesmas características e funcionalidades, propiciando sistemas redundantes, o que lhes confere alta tolerancia a falhas, pois um robô defeituoso pode ser substituído por qualquer outro robô do grupo sem qualquer perda de desempenho (QUINN *et al.*, 2003). Um caso particular de sistema homogêneo é os robôs de enxame, que são geralmente pequenos e possuem uma quantidade limitada de sensores e atuadores (HETTIARACHCHI, 2007), e são utilizados

principalmente como plataforma de testes para arquiteturas biomiméticas (NOUYAN; DORIGO, 2006).

Sistemas heterogêneos constituem-se por robôs com características e especialidades distintas. Seu estudo é importante por duas razões: primeiro, a especialização dos robôs pode reduzir o custo total do sistema, uma vez que construir vários robôs especializados pode ser mais simples e barato que construir um (ou vários) robô completo e genérico (PARKER, LYNNE E, 1998). Segundo, construir robôs verdadeiramente homogêneos tem se mostrado quase impossível no mundo real, pois que cópias do mesmo robô derivam para a heterogeneidade conforme o passar do tempo devido à regulagem dos sensores, desgaste mecânico, etc. (SICILIANO; KHATIB, 2008).

No que toca a comunicação, é possível identificar dois grupos de técnicas (ARAI *et al.*, 2002): comunicação implícita e comunicação explicita.

Comunicação implícita é realizada através da percepção do mundo e dos agentes que nele existem. A comunicação implícita pode ser por "mundividência" (stigmergy), onde os robôs são capazes de ver a modificação causada no mundo por outros robôs, ou por reconhecimento passivo de ação, onde os robôs reconhecem os membros de sua equipe e quais tarefas estão desenvolvendo através de seus sensores (MATARIĆ, 1995).

Ao contrário da comunicação implícita, a comunicação explícita envolve mecanismos criados especificamente para a comunicação entre os robôs da equipe. Vários métodos de comunicação são encontrados na literatura, como por exemplo, os baseados em redes de computadores (ASAMA *et al.*, 1989), conexão por rádio-frequência (PARKER, LYNNE E, 1998) e baseado em modelos biológicos (SCHMICKL; CRAILSHEIM, 2006). Destes métodos, as redes de computadores tem particular importância pois permite que o robô se comunique com outros sistemas não-robóticos, inclusive com seres humanos, através da rede (SICILIANO; KHATIB, 2008).

Um dos desafios da comunicação entre robôs móveis está em definir quais as informações que devem ser trocadas entre os robôs. Estudos revelam que o desempenho de um time de robôs não é diretamente proporcional à quantidade de informações trocadas. Na

verdade, o desempenho é prejudicado quando a quantidade de dados transmitidos alcança ou ultrapassa o limite de processamento de mensagens do robô (SICILIANO; KHATIB, 2008). Os métodos de comunicação adotados por um sistema robótico e as informações que devem ser trocadas ou percebidas pelos robôs dependem da aplicação, de modo que não há fórmulas para defini-las.

3.5 Arquiteturas de Controle

A arquitetura de controle é um módulo imprescindível de um robô móvel. Ela é responsável por fornecer um método formal para a organização de um sistema de controle, impondo regras e restrições no modo como o problema do controle é resolvido (MATARIĆ, 1992). Consequentemente, é a arquitetura de controle que define os componentes arquiteturais de um robô móvel e como tais componentes interagem entre si (DEAN, WELLMAN, 1991 apud MURPHY, 2000). Em outras palavras, a arquitetura de controle determina como se dá o fluxo de informação dentro de um sistema robótico móvel (MURPHY, 2000).

Em seu trabalho, Murphy (2000) classifica as arquiteturas de controle para robôs móveis em paradigmas de organização da inteligência que são entendidos sob duas visões. A primeira visão diz respeito a como se relacionam as três primitivas comumente aceitas na robótica: “Sentir”, que engloba as funções que produzem uma informação relevante a partir de um sensor do robô; “Planejar”, que é composto de funções que utilizam informações oriundas dos sensores ou de um banco de dados para orquestrar a execução de uma ou mais tarefas do robô; e “Agir”, que abrange as funções que geram ação no mundo físico através do acionamento de atuadores. Já a segunda visão contempla a forma com a qual os dados sensoriais são processados e distribuídos através do sistema, pois este processamento pode ser local – onde cada sensor é atribuído a uma única função – ou global – onde todos os sensores são direcionados para uma função global antes de serem processados por funções locais.

Utilizando as primitivas da primeira visão, é possível distinguir os três paradigmas apresentados por Murphy (2000): hierárquico; comportamental; e híbrido. No paradigma hierárquico, o fluxo de informação se dá sempre na ordem "Sentir-Planejar-Agir", como mostra a Figura 3.3 (a). Ou seja, os dados sensoriais alimentam um sistema de planejamento que coordena as ações do robô, que finalmente são sentidas pelos sensores do robô, fechando assim o ciclo de controle. Em outras palavras, neste paradigma as ações do robô se dão de modo deliberado. Numa linha contrária ao paradigma hierárquico se encontra o paradigma comportamental, mostrado pela Figura 3.3 (b), onde o fluxo de informação se dá entre "Sentir-Agir", sem a necessidade de um módulo de planejamento, ou seja, as ações do robô se dão de forma reativa. Finalmente, o paradigma híbrido combina a reatividade do paradigma comportamental com a deliberação do paradigma hierárquico. No paradigma híbrido, apresentado pela Figura 3.3 (c), o fluxo de informação se dá na ordem "Planejar, Sentir-Agir", que significa que o planejamento é feito separado, enquanto o sensoriamento e a ação são feitos concomitantemente. É importante lembrar que no paradigma híbrido ainda há uma relação entre as primitivas "Sentir-Planejar", permitindo que o planejamento leve em conta informações atualizadas sobre o ambiente.

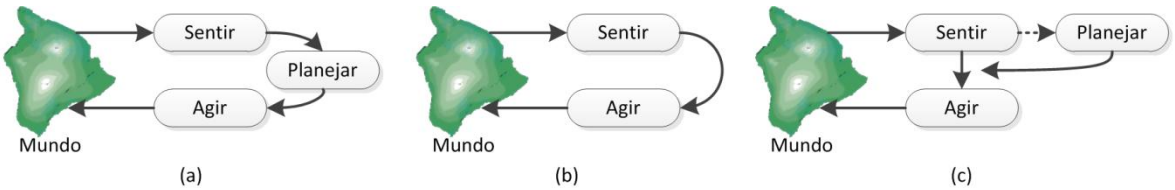


Figura 3.3 – Paradigmas das arquiteturas de controle para robôs móveis: (a) Hierárquico; (b) Comportamental; e (c) Híbrido.

Algumas arquiteturas de controle para robôs móveis representativas de cada um dos três paradigmas de Murphy são as arquiteturas NASREM (*NASA/NBS Standard Reference Model*), de subordinação (*subsumption*), TCA (*Task Control Architecture*) e LAAS. A arquitetura NASREM segue o paradigma hierárquico e é baseada em vários estágios de processamento que, sucessivamente, transformam um dado do ambiente em uma decisão de controle (ALBUS, 1991). Já arquitetura de subordinação desenvolvida por Brooks (1985), inspirada nos comportamentos observados em animais, se enquadra no paradigma comportamental. A arquitetura TCA proposta por Simmons (1990), identificada pelo paradigma híbrido, adota uma estrutura semelhante a de um sistema operacional, com um controlador central

para o qual os módulos funcionais se reportam. Por fim, a arquitetura LASS, também do paradigma híbrido, é dividida em três níveis, sendo eles duas camadas de decisão e um nível funcional (CHATILA, RAJA, 1995).

Os três paradigmas apresentados por Murphy bem classificam as arquiteturas de controle para um único robô móvel. Entretanto, quando tratamos um time composto por vários robôs, é necessária uma arquitetura de controle que seja capaz de orquestrar as ações dos diferentes robôs para que eles alcancem o objetivo desejado. Tais arquiteturas são chamadas de arquiteturas de controle para robôs colaborativos e adicionam o problema da multiplicidade de agentes robóticos à divisão proposta por Murphy.

As arquiteturas de controle para robôs colaborativos tratam tanto do problema da aplicação quanto da alocação de tarefas. O problema da aplicação diz respeito aos papéis que os robôs podem desempenhar para alcançarem o objetivo global do sistema. Já a alocação de tarefas remete a distribuição dos papéis para cada robô levando em conta sua disponibilidade, especialidade e localização (ARAI *et al.*, 2002).

Assim como a arquitetura de controle para um robô determina o fluxo de informação na rede de módulos que compõem o robô, a arquitetura de controle para robôs colaborativos determina o fluxo de informação na rede de robôs que compõem um time de robôs. De certa forma, a definição de ambos os tipos de arquiteturas são semelhantes, mas diferem na complexidade dos nós da rede que elas tratam: para um único robô, nem todo nó (neste caso, um módulo) é capaz de produzir uma ação sozinho; ao contrário, para um time de robôs, cada nó (ou robô) é capaz de produzir uma ação sozinho. É por conta desta diferença que o estudo das arquiteturas de controle para robôs colaborativos é necessário.

As arquiteturas de controle para robôs colaborativos podem ser divididas em quatro modalidades: arquiteturas centralizadas, hierárquicas, descentralizadas e hibridas. Em arquiteturas centralizadas, a coordenação do time de robôs é realizada a partir de um único ponto, como exemplifica a Figura 3.4 (a). Esta arquitetura é dificilmente implementada na prática devido à vulnerabilidade oriunda da falha em um único ponto e da dificuldade em comunicar o estado de todo o sistema de forma a possibilitar o controle de tempo real (SI-

CILIANO; KHATIB, 2008). Um exemplo desta arquitetura é encontrado no time de futebol de robôs Carrossel Caipira, a UNESP de Bauru (BORTHOLIN *et al.*, 2006).

Em arquiteturas hierárquicas, há um controle centralizado local, onde um robô controla a ação de um grupo de robôs, capazes de controlar ação de outro grupo e robôs e assim por adiante até o menor robô (CAO *et al.*, 1997), como mostra a Figura 3.4 (b). Esta arquitetura é um resquício da estrutura militar de controle.

Ao contrário das arquiteturas centralizadas e hierárquicas, onde a comunicação desempenha um papel fundamental na arquitetura de controle, nas arquiteturas descentralizadas não há qualquer troca de informações entre os robôs, conforme mostra a Figura 3.4 (c). Nestas arquiteturas, cada robô deve ter conhecimento local de sua situação para a tomada de decisão. Este sistema é robusto, pois a ação dos robôs independe do controle de outros robôs, porém torna-se difícil alcançar uma coerência global – isto é, é difícil coordenar os robôs, que tomam decisões locais, para alcançar um objetivo global em conjunto (SICILIANO; KHATIB, 2008).

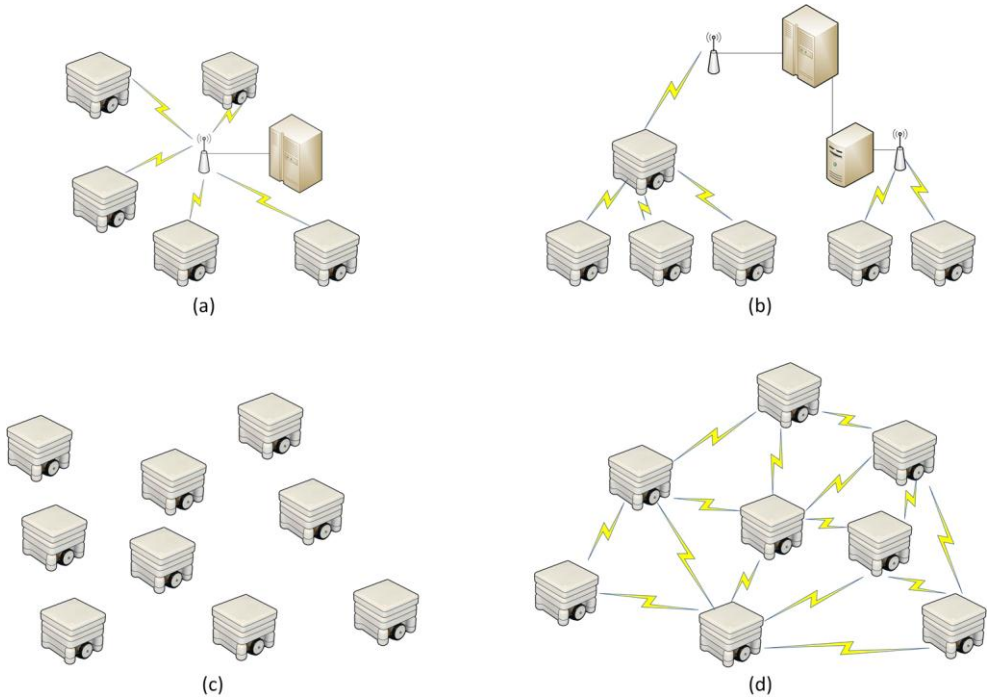


Figura 3.4 – Arquiteturas de controle para robôs colaborativos: (a) centralizada; (b) hierárquica; (c) descentralizada; e (d) híbrida.

Por fim, as arquiteturas híbridas combinam laços locais de controle com algum método de mais alto nível para criar uma arquitetura onde um robô possa tomar suas próprias decisões ao mesmo tempo em que influencia e é influenciado por outros robôs para alcançar um objetivo global (MATARIĆ, 1995). Ou seja, o robô móvel pode tomar suas próprias decisões, mas também pode se comunicar com outros robôs para adquirir informações que o auxiliem em sua tomada de decisão, como mostra a Figura 3.4 (d).

3.6 Arquiteturas de Software

Conforme discutido na seção 3.5, as arquiteturas de controle estabelecem o fluxo de informação em um sistema robótico móvel, fornecendo um método formal para a organização dos componentes de controle e estabelecendo como ocorre a interação entre eles. Entretanto, a arquitetura de controle não contempla as técnicas que podem ser empregadas para sua implementação. Em outras palavras, ela especifica "o que" será construído em termos de controle, mas não "como" tal controle será constituído. Esta lacuna é preenchida pela arquitetura de software, que é responsável por determinar quais os componentes computacionais comporão a arquitetura e como tais componentes serão organizados para suportar a arquitetura de controle proposta. Logo, as arquiteturas de software representam um aspecto importante dentro da robótica móvel, pois definem a forma como os dados são distribuídos e processados e impactam na facilidade de desenvolvimento e reuso de código. Uma tendência atual é a criação de arquiteturas abertas, cujo objetivo é fornecer uma interface comum para os componentes do robô (como sensores e atuadores, por exemplo) que seja capaz de terminar as funções e especificações destes componentes, análoga à capacidade de *plug-and-play* ("conectar e usar") dos computadores (BEKEY, 2005). Elas devem ser também flexíveis, de forma que um elemento específico é uma especialização de um elemento genérico (HOLLAND, 2003).

Na literatura, é possível encontrar diferentes abordagens para a composição de arquiteturas de software para robôs móveis. De forma genérica, elas podem ser classificadas como bibliotecas, sistemas operacionais (SO), *middleware* ou baseadas em redes.

As bibliotecas de software são constituídas por código fonte reusável. Em sua forma mais simples, uma biblioteca é compilada junto ao aplicativo do usuário, sendo integrada ao binário do aplicativo. Neste caso, as interfaces são escritas em uma única linguagem de programação, o que engessa o desenvolvimento de software. As bibliotecas também se apresentam na forma de código externo, o que oferece vantagens como modularidade e não-replicação de código, mas continuam dependentes de uma única linguagem. Esta dependência é vencida através de técnicas de interface de funções estrangeiras ou ligação de linguagens, como fornecido pelo projeto SWIG[2] (*Simplified Wrapper and Interface Generator*, ou Gerador de Interface e Embalador Simplificado). É claro que o suporte ao SO é importante, pois o binário gerado em cada SO é diferente. Neste sentido, as bibliotecas podem, ou não, tomar o cuidado de gerar um código que possa ser compilado em vários SOs – também conhecidas como *cross-platform*. Como exemplos desta arquitetura, pode-se citar o projeto CLARATy (NESNAS *et al.*, 2003) da NASA com vários algoritmos para a robótica móvel, o V-Clip (MIRTICH, 1998) que fornece algoritmos rápidos de detecção de colisão, e o OpenCV (*Open Computer Vision*) (BRADSKI; KAEHLER, 2008) dedicado a processamento de imagens.

Dentro da abordagem de sistema operacional, encontramos o ROS (*Robot Operating System*, ou Sistema Operacional para Robô) (QUIGLEY *et al.*, 2009), um sistema operacional livre e de código aberto voltado para a robótica. Nesta abordagem, as diferentes funcionalidades que o robô pode ter (ex.: motores, algoritmos de localização, etc.) não são vistas como bibliotecas, mas como módulos do sistema operacional, que podem ser invocadas por diferentes aplicativos em variadas situações. Por ser um SO, seus módulos funcionam apenas dentro dele, não sendo possível executá-los num SO diferente – no caso específico do ROS, é possível instalá-lo em um sistema GNU/Linux compatível. Finalmente, o acesso aos módulos também dependem da interface de programação disponível para as linguagens, de forma que o ROS suporta apenas quatro linguagens: C++, Python, Octave e LISP.

Por outro lado, as arquiteturas baseadas em *middleware* entendem os diferentes componentes de software como aplicativos independentes e em execução na mesma máquina ou através da rede. A comunicação entre cada módulo é realizada por uma interface inter-

[2] Simplified Wrapper and Interface Generator: <http://www.swig.org/>. Acessado em 6 de abril de 2011.

mediária: o *middleware* (daí seu nome: "*middle*", meio; "-*ware*", soft-ware). A maior parte dos robôs móveis baseados em *middleware* utiliza uma solução comercial, tal qual o CORBA[3] (*Common Object Request Broker*, ou Intermediário Comum para Requisições de Objetos), que é utilizado pelo MIRO (*Middleware for Mobile Robots*, ou *Middleware* para Robôs Móveis) (UTZ *et al.*, 2002). O CORBA fornece ferramentas que facilitam a criação de software distribuído, entretanto não oferece uma implementação padrão do *middleware*, além de ter uma documentação extensa e complexa (HENNING, MICHI, 2006). Como resposta a estes problemas, Henning (2004) desenvolveu um novo *middleware* baseado no CORBA, batizado de ICE (*Internet Communication Engine*, ou Motor para Comunicação via Internet), que tem uma implementação padrão do *middleware* e uma documentação simples.

Finalmente, as arquiteturas baseadas em redes utilizam-se da popularização da Internet e de seus protocolos para propor suas infraestruturas. Impulsionados pelo conceito de *web services* (serviços de rede), surgiram padrões como o UPnP (*Universal Plug-and-Play*, ou Plug-and-Play Universal), que dentro da robótica é visto como uma forma eficaz de criar módulos reusáveis e de fácil manutenção (MOK; WU, C.-HAUR, 2006). Sua característica de descoberta dinâmica de serviços disponíveis na rede é citada como uma das mais relevantes desta tecnologia, de modo que Ahn et al. (2005) avaliam que os robôs serão ou devem ser um ambiente de computação distribuído no futuro, que se comunica por uma rede de computadores tal qual o nosso corpo se comunica através de uma rede neural. Outro padrão é o WS4D (*Web Services for Devices*, ou Serviços de Rede para Dispositivos) (CHAN *et al.*, 2006), cujas premissas são as mesmas do UPnP. Contudo, ambas as implementações não foram desenvolvidas especialmente para a robótica e, portanto, algumas modificações devem ser feitas no protocolo para que operem de forma satisfatória. Além disso, ainda há pouco material de referência disponível para auxiliar no desenvolvimento e não há uma diretriz que indique qual das tecnologias será adotada – fato este que levou Nain et al. (2008) a desenvolverem um dispositivo capaz de compreender tanto o UPnP quando o WS4D, que o autor batizou de esquizofrênico.

[3] Common Object Request Broker: <http://www.corba.org/>. Acessado em 6 de abril de 2011.

3.7 Conclusão

Este capítulo apresentou uma breve discussão da definição de robôs móveis autônomos e do que os compõe, bem como uma breve revisão da literatura referente aos principais aspectos da robótica móvel: navegação, arquiteturas de controle, colaboração, e arquiteturas de software. Nele, é possível notar a extensão de cada um dos tópicos envolvidos pela robótica móvel e qual a contribuição de cada um deles na construção de um dispositivo robótico móvel.

Uma vez que o objetivo deste trabalho é desenvolver e avaliar uma plataforma de software voltada para navegação e colaboração de robôs móveis, foi dada maior atenção aos tópicos de navegação, colaboração e arquitetura de software. Cada um destes tópicos será discutido com maior profundidade nos capítulos 4 e 5, respectivamente. Nestes capítulos, são discutidos quais técnicas e métodos foram contemplados e desenvolvidos por este trabalho.

4 NAVEGAÇÃO E COLABORAÇÃO

Em um sistema de navegação clássico, o robô móvel deve seguir um caminho no plano através do tempo, seja ele conhecido previamente ou calculado em tempo real. Para o robô móvel, seguir este dado caminho é um dos maiores desafios, pois ele está sujeito às perturbações do ambiente, à imprecisão dos seus sensores e atuadores, e à presença de obstáculos conhecidos e desconhecidos no tempo e no espaço, conforme discutido na seção 3.3. Consequentemente, o controle para robôs móveis é atualmente o foco de numerosas pesquisas e tem motivado o desenvolvimento de técnicas altamente não-lineares de controle (SICILIANO; KHATIB, 2008).

4.1 Navegação

De modo geral, o controle do caminho ou trajetória é implementado com uma malha fechada ou feedback. Isso ocorre porque, para a robótica móvel, os controles de laço aberto não apresentam desempenho satisfatório, pois não permitem que o robô possa adaptar ou corrigir a trajetória para responder a mudanças dinâmicas do ambiente (SIEGWART; NOURBAKHSH, 2004). Dentro dos controladores de trajetórias, podemos destacar aqueles que utilizam uma trajetória contínua (SICILIANO; KHATIB, 2008) ou discreta (SIEGWART; NOURBAKHSH, 2004). Trajetórias contínuas têm a vantagem de permitirem maior domínio sobre o movimento do robô, entretanto exigem maior poder computacional para serem geradas e controladores mais complexos. Por outro lado, trajetórias discretas não permitem descrever a trajetória entre cada um dos pontos que a compõem, contudo exigem menor processamento e utilizam controladores mais simples. Ainda dentro da classificação dos controladores de trajetória, pode-se dividi-los em cinemáticos ou dinâmicos (MARTINS, 2010). Os controladores cinemáticos são os mais comumente empregados na robótica móvel, pois o modelo cinemático envolve uma matemática mais simples e, também, devido ao fato dos robôs móveis geralmente utilizarem motores elétricos com controle de velocidade que, caso tenham um laço eficiente de controle, permitem desacoplar a cinemática da dinâ-

mica (SICILIANO; KHATIB, 2008). Todavia, a suposição de que os motores desenvolveram as velocidades requisitadas não se mantém na prática, o que impulsiona o interesse no desenvolvimento de controladores dinâmicos, que levam em conta os torques desenvolvidos pelas estruturas do robô móvel (MARTINS, 2010).

Além do controlador de trajetória, é imprescindível a existência de um sistema de localização capaz de realimentar o controlador com dados precisos para reduzir os erros de posicionamento. Normalmente, um sistema de navegação utiliza dois ou mais métodos ou técnicas diferentes de localização para reduzir a incerteza e aumentar sua robustez (BORENSTEIN *et al.*, 1995). Os métodos de navegação são discutidos com maior profundidade na seção 3.3.

A seguir, serão apresentadas as técnicas adotadas para a elaboração da navegação dos robôs móveis utilizados no trabalho.

4.2 Construção de Mapas e Planejamento de Trajetória

A navegação de um robô móvel depende fortemente do planejamento de trajetória, que está intimamente ligado ao mapa do ambiente. Nesta seção serão apresentados o mapa e o processo de geração de trajetória adotados neste trabalho.

4.2.1 Mapa

Dentre as abordagens discutidas na seção 3.3, foi adotado neste trabalho o mapa em grade de ocupação de tamanho fixo. Este tipo de mapa oferece um meio razoável para a representação de obstáculos fixos e móveis, além de ser facilmente implementado computacionalmente, pois consiste em uma matriz de duas dimensões.

Neste projeto, cada grade do mapa corresponde a um quadrado cujas medidas dependem do robô em uso. Preferencialmente, a medida do lado do grade deve ser igual ou maior

que a medida do maior lado do robô. Quando não há qualquer obstáculo na célula, ela é considerada "vazia". Por outro lado, quando há um obstáculo fixo em qualquer região da celula, ela é considerada "ocupada". Neste trabalho, não será considerada a presença de obstáculos móveis, logo não haverá uma representação para eles neste mapa.

Um exemplo deste mapa é mostrado pela Figura 4.1, que compara um mapa geométrico hipotético de uma sala com seu correspondente mapa em grade de ocupação. Sua conversão em grades de ocupação é obtida através da superposição das grades (Figura 4.1 b) e, em seguida, pela atribuição do conteúdo da grade, ou seja, se ela está ocupada ou vazia (Figura 4.1 c). O resultado é um mapa de grade de ocupação que representa os espaços por onde o robô é capaz de navegar.

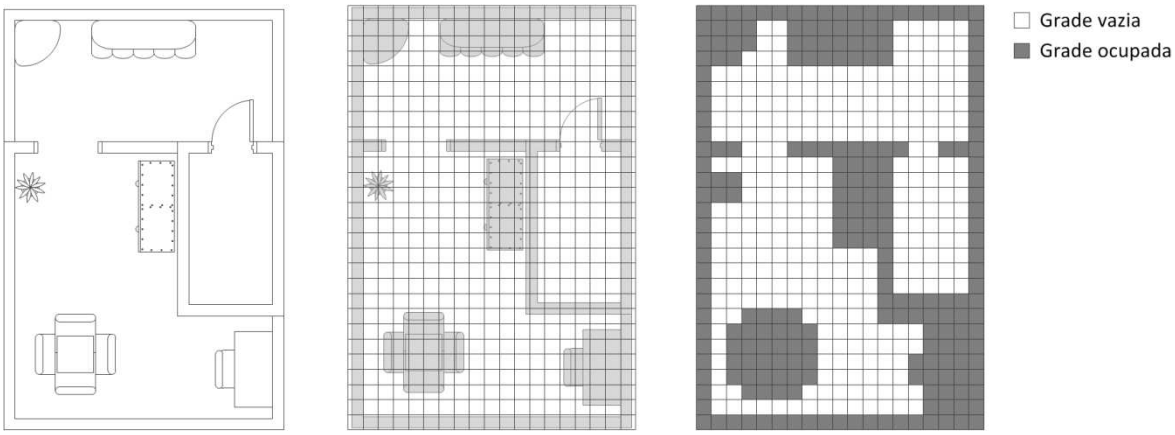


Figura 4.1 – Exemplo de (a) mapa geométrico; (b) processo de discretização em grade de ocupação fixa; e (c) o mapa discretizado.

4.2.2 Planejamento de Trajetória

O planejamento de caminho ou planejamento de trajetória é um campo vasto dentro da navegação de robôs móveis e pode ser tratado com o uso de diferentes técnicas. Seu principal objetivo é encontrar uma trajetória $\mathcal{T}$ no espaço livre do plano que leve o robô de uma posição inicial q_0 até a posição final q_1. Uma trajetória pode ser elaborada de modo a atender diferentes tipos de requisitos – as vezes conflitantes entre si –, como menor distân-

cia do trajeto, menor tempo decorrido para percorrer o trajeto, menor consumo de recursos, maior segurança, dentre outros.

Em mapas discretizados, como o de grades de ocupação de tamanho fixo aqui adotado, é possível utilizar técnicas de busca de grafos. Tais técnicas englobam algoritmos como busca por largura, busca por profundidade, algoritmo de Dijkstra, A^*, D^*, dentre outros. O caminho traçado por estes algoritmos pode ser convertido facilmente em trajetórias através do uso de técnicas como Splines, β-Splines ou Curvas de Dubins.

Neste trabalho, será adotado o algoritmo A^* que, apesar de ser computacionalmente custoso, bem se adapta aos requisitos do trabalho. Seu custo computacional pode ser restringido limitando-se o tamanho do mapa e das grades do mapa, visto que ele é diretamente proporcional à quantidade de grades do mapa. Para a composição final do caminho, será utilizada uma técnica semelhante às Curvas de Dubins, que será descrita no decorrer deste capítulo.

4.2.3 Algoritmo A^*

Suponha que $f(n) = g(n) + h(n)$ é uma função de custo onde: $g(n)$ é o custo real de deslocamento do nó (ou grade, no caso deste trabalho) inicial para o nó n; $h(n)$ é o custo estimado do nó n ao nó objetivo; e que $g(n)$ seja um *upper bound* em $g^*(n)$, e $h(n)$ seja um *lower bound* em $h^*(n)$. Este é conhecido como o algoritmo A^*, capaz de encontrar o melhor caminho de um ponto inicial ao ponto final (DUDEK; JENKIN, 2010). No algoritmo adotado neste trabalho, as funções $g(n)$ e $h(n)$ são dadas pelas distâncias euclidianas entre os nós envolvidos.

Lester (2005) exemplifica o uso do algoritmo A^* em um mapa de grade de ocupação de tamanho fixo e fornece um pseudocódigo que explica seu funcionamento. Este pseudocódigo foi adaptado a terminologia adotada neste trabalho e é reproduzido pela Figura 4.2.

Defina a grade inicial e final;
Adicione a grade inicial na lista aberta;
Defina a grade inicial como grade atual;
Repita:
    Procure pelo menor custo $f(n)$ na lista aberta;
    Mova-o para a lista fechada;
    Para cada *um das 8 grades adjacentes*, **faça**:
        Se *a grade está vazia e não está na lista fechada*, **faça**:
            Se *a grade não está na lista aberta*, **faça**:
                Mova a grade para a lista aberta;
                Defina a grade atual como mãe desta grade;
                Grave os custos $f(n)$, $g(n)$ e $h(n)$ desta grade;
                Se *este caminho estiver na lista aberta*, **faça**:
                    Verifique se o caminho é o melhor, usando $g(n)$ como medida;
                    Se *um menor custo $g(n)$ é encontrado*, **faça**:
                        Mude a grade mãe para a grade atual;
                        Recalcule os custos $g(n)$ e $f(n)$ da grade;
                    Fim Se;
                Fim Se;
            Fim Se;
        Fim Se;
    Fim Para Cada;
Até (*adicionar a grade objetivo na lista aberta*) **ou** (*falhar em encontrar a grade objetivo* **e** *a lista aberta está vazia*);

Figura 4.2 – Pseudocódigo de busca A^* para o algoritmo proposto por Lester (2005).

Para entender o algoritmo, será utilizado um exemplo. Considere o mapa da Figura 4.3, onde o nó inicial encontra-se na grade (1,1) e o nó final na grade (5,1).

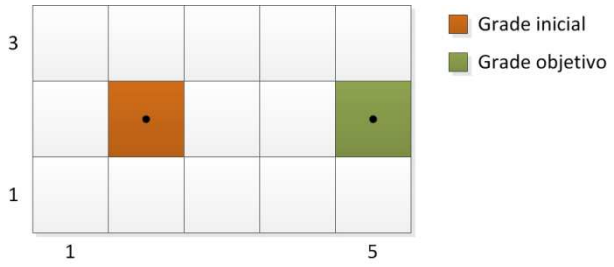


Figura 4.3 – Exemplo de mapa com os nós inicial e final.

O algoritmo inicia-se determinando quais são os nós próximos ao nó inicial, que neste caso são as grades (1,1), (1,2), (1,3), (2,3), (3,3), (3,2), (3,1) e (2,1), e para cada uma destas grades será calculada a função de custo $f(n)$. Para o caso da grade (1,1), o custo $g(1,1)$ é dado pela distância euclidiana entre (2,2) e (1,1), ou seja, $g(1,1) = \sqrt[2]{2}$, e o custo $h(1,1)$ é

dado pela distância entre (1,1) e (5,2), ou seja, $h(1,1) = \sqrt[2]{17}$. A Figura 4.4 (a) ilustra o cálculo de $g(1,1)$ e $h(1,1)$. Logo, a função de custo para a grade (1,1) é $f(1,1) = \sqrt[2]{2} + \sqrt[2]{17} \cong 5{,}5$. Este processo é repetido para todas as sete grades restantes, conforme mostra a Figura 4.4 (b), onde o custo $f(n)$ de cada grade é mostrado no canto superior esquerdo da grade.

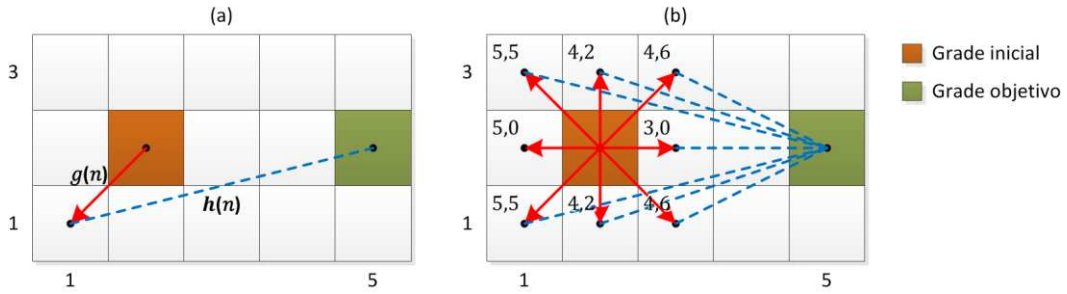


Figura 4.4 – Exemplo do algoritmo A^*, onde: (a) mostra o cálculo das funções $g(n)$ e $h(n)$, e (b) mostra o custo $f(n)$ de cada uma das grades ao redor do nó inicial.

Conforme pode ser visto na Figura 4.4 (b), a grade com menor custo é a grade (3,2), logo ela é escolhida como novo nó de origem. Consequentemente, as grades ao redor de (3,2) cujos custos não foram calculados são selecionadas e computadas, conforme mostra a Figura 4.5 (a). No final desta iteração, tem-se que a grade com menor custo é a (4,2), que é selecionada para a próxima iteração. Entretanto, na próxima iteração a grade objetivo é encontrada, encerrando, portanto, a execução do algoritmo. O caminho encontrado é mostrado pela Figura 4.5 (b).

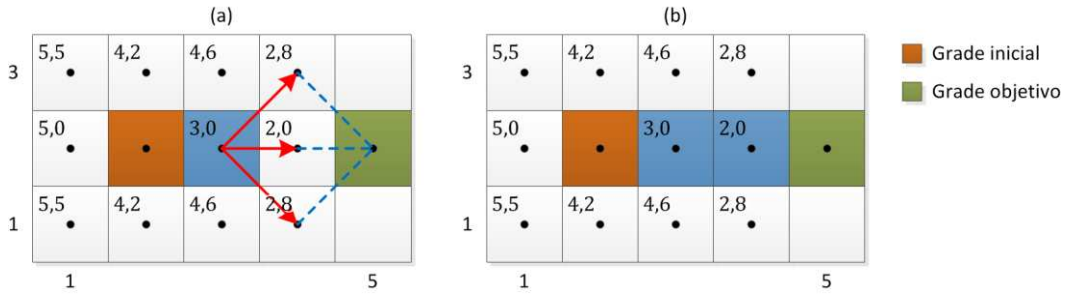


Figura 4.5 – Exemplo do algoritmo A^*, onde: (a) mostra o cálculo do custo de cada uma das grades da iteração; e (b) mostra o caminho encontrado pelo algoritmo.

Neste trabalho, será adotada a implementação do algoritmo A^* apresentada por Rotkiewicz (2008), que oferece as rotinas necessárias para o cálculo do melhor caminho dentro de um mapa de grade de ocupação de tamanho fixo. O resultado da execução das rotinas é uma lista de grades pelas quais o robô deve passar. Um exemplo de uso deste algoritmo é

mostrado pela Figura 4.6, que mostra a grade inicial do robô, a grade final e o caminho encontrado pelo algoritmo. Nesta versão do algoritmo, não é considerado o movimento na diagonal. Esta restrição foi feita para reduzir a complexidade do algoritmo, pois caso contrário ele deveria testar se há um obstáculo que impediria a movimentação do robô na diagonal.

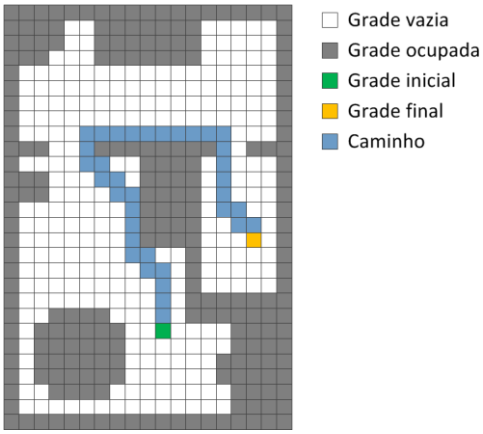


Figura 4.6 – Exemplo de caminho calculado utilizando-se o algoritmo A^*.

4.2.4 Geração de Trajetória

Com o caminho fornecido pelo algoritmo A^* apresentado na seção 4.2.3, é possível criar facilmente a trajetória em termos de velocidades linear v e angular ω. Para iniciar a geração da trajetória, é necessário determinar a posição inicial e a orientação do robô. A posição inicial, em termos de (x, y) é dada pelo centro da grade inicial. Já a orientação inicial θ do robô depende da relação entre a posição da grade inicial com a grade subsequente do caminho, pois ambas devem estar alinhadas, conforme indica a Figura 4.7.

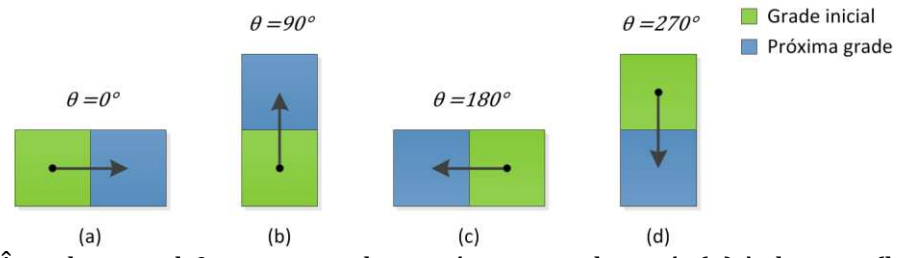


Figura 4.7 – Ângulo inicial θ para quando a próxima grade está: (a) à direita; (b) acima; (c) à esquerda; e (d) abaixo.

Com a pose inicial do robô (x, y, θ), é possível descrever a trajetória em termo das velocidades desenvolvidas por ele. Consideremos v a velocidade linear do robô; ω a velocidade angular do robô; k_v a constante de velocidade linear arbitrária; e l a largura da grade.

Para que o robô se desloque do centro da grade atual até o centro da próxima grade, ele deve descrever uma velocidade $v = k_v$ e $\omega = 0$ por um tempo t_{reta} que é dado pela equação:

$$t_{\text{reta}} = \frac{l}{v} \tag{4.1}$$

Quando há uma curva o robô deve descrever um quarto de circunferência entre a grade atual, onde é posicionado o ponto inicial q_i, e a próxima grade depois da curva, onde se encontra o ponto final q_f, conforme exemplifica a Figura 4.8. A circunferência descrita tem raio l.

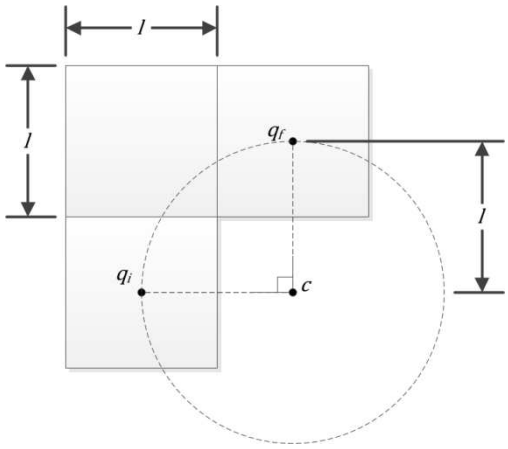


Figura 4.8 – Procedimento para uma curva.

Para executar a curva da Figura 4.8, o robô deve descrever uma velocidade linear $v = k_v$ e uma velocidade angular $\omega = k_\omega$ por um tempo t_{curva}, sabendo que:

$$t_{\text{curva}} = \frac{2\pi l}{v} \tag{4.2}$$

$$k_\omega = \frac{\pi}{2t_{\text{curva}}} = \frac{v}{4l} \tag{4.3}$$

Para exemplificar este processo, será considerado o caminho hipotético da Figura 4.9 (a), da qual é gerada a trajetória da Figura 4.9 (b) cujos valores de velocidade são dados pela equação (4.4) e tem duração $t, 0 < t < t_{\text{reta}} + t_{\text{curva}}$.

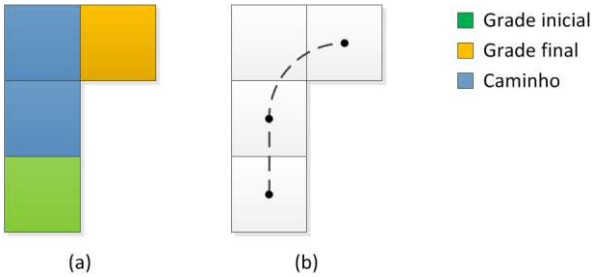


Figura 4.9 – Exemplo do processo de geração de trajetória, mostrando: (a) o caminho resultante da aplicação do método A^*; e (b) a trajetória gerada.

$$\begin{cases} v = k_v, \omega = 0, & 0 < t \leq t_{reta} \\ v = k_v, \omega = k_\omega, & t_{reta} < t < t_{reta} + t_{curva} \end{cases} \tag{4.4}$$

Assim, sendo capaz de gerar as trajetórias do robô, faz-se necessário a elaboração de um controlador de trajetória, que será discutido na seção seguinte.

4.3 Sistema de Localização

Conforme discutido na seção 3.3, um Sistema de Localização envolve as técnicas e métodos que permitem ao robô móvel determinar sua posição no plano ou no espaço. A presente seção apresenta os dois sistemas que serão utilizados neste trabalho, que são a odometria e o sistema de Visão Global.

4.3.1 Odometria

A odometria é a técnica de localização local mais comumente adotada nos robôs móveis. Ela combina a informação do deslocamento das rodas ao modelo cinemático do robô para calcular sua movimentação no plano. O deslocamento das rodas é medido através de encoders acoplados ao eixo do motor ou da roda. Os modelos cinemáticos dos robôs diferencial e omnidirecional serão apresentados na seção 4.4.1 e 4.4.2.

O robô Robotino® dispõe deste método de localização, que é disponibilizado através de sua API (*Application Programming Interface*, ou Interface de Programação de Aplicativos), conforme apresenta a seção 6.1.3.

4.3.2 Visão Global

A Visão Global corresponde a uma técnica de localização global que se utiliza das imagens capturadas por uma câmera suspensa paralela à área de trabalho para identificar a posição e orientação dos robôs ali presentes, conforme mostra a Figura 4.10. Para facilitar o processo seu reconhecimento, cada robô possui uma etiqueta com duas cores diferentes entre si e contrastantes com o chão e o robô. Este sistema é utilizado em competições de futebol de robôs devido sua versatilidade.

Figura 4.10 – Sistema de Visão Global.

O processo reconhecimento da posição dos robôs se dá da seguinte forma: primeiro, a imagem é capturada pela câmera e decodificada pelo computador. Esta imagem decodificada passa, então, por um processo de segmentação de cor cujo objetivo é reconhecer as cores predefinidas das etiquetas, agrupando-as em regiões. Em sequência, é calculado os centros geométricos das regiões reconhecidas, que são utilizadas para determinar o centro geométrico do robô e sua orientação. Suponha que a etiqueta reconhecida seja representada pelas regiões de centros C_0 e C_1. O centro geométrico C do robô pode ser determinado como o ponto médio entre C_0 e C_1, já sua orientação θ é dada por:

$$\theta = \alpha - 45° \tag{4.5}$$

onde α é o ângulo entre o segmento $\overline{C_0C_1}$ e a abscissa do plano, conforme visto na Figura 4.11.

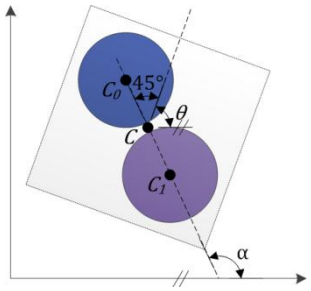


Figura 4.11 – Cálculo da posição e orientação de um robô no plano.

O AEDROMO tem implementado um sistema de Visão Global que fornece a posição dos robôs e de objetos reconhecidos pela câmera (FERASOLI FILHO *et al.*, 2006). Este sistema disponibiliza estas posições através de uma rede UDP/IP, o que facilita a integração de softwares escritos em diferentes linguagens de programação ou ainda sendo executados em máquinas diferentes. A Figura 4.12 mostra a interface do sistema do AEDROMO, onde pode ser vista (a) a imagem capturada pela câmera e (b) os objetos reconhecidos.

Este sistema será adaptado, também, para ser utilizado com os robôs 14-bis, Roburguer e Robotino® a fim de fornecer dados mais precisos sobre seu posicionamento no plano.

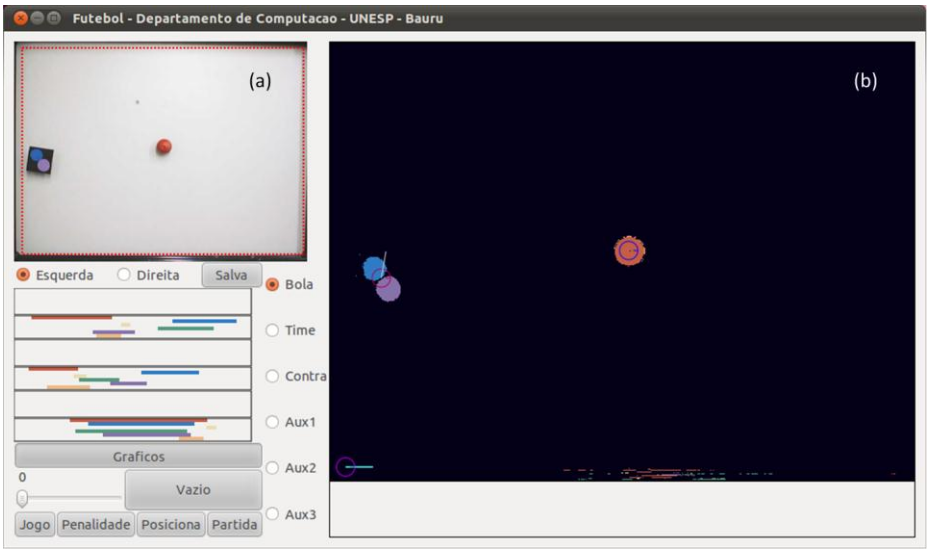


Figura 4.12 – Sistema de Visão Global do AEDROMO, mostrando (a) a imagem capturada e (b) os objetos reconhecidos.

4.4 Modelagem e Controle Cinemático

Neste trabalho foram adotados dois controladores para a navegação dos robôs móveis. O primeiro corresponde a um controlador de posição discreto, capaz de dirigir um robô automaticamente a uma dada posição no plano. O segundo baseia-se no trabalho desenvolvido por Siciliano e Khatib (SICILIANO; KHATIB, 2008), que consiste em um estabilizador de trajetória cinemático. Estes controladores foram escolhidos devido à possibilidade de desacoplar a cinemática da dinâmica do robô e pela facilidade de desenvolvimento. Para simplificar as características do controle, o ambiente onde o robô móvel será inserido será estático e conhecido, ou seja, não haverão obstáculos móveis ou desconhecidos. Além disso, o solo do ambiente será pouco acidentado, o que reduz a deterioração do controle do robô, pois diminui, na medida do possível, as incertezas oriundas da interação do robô com o piso.

A arquitetura do controlador de posição é mostrada pela Figura 4.13 e utiliza a Postura de referência, o Sistema de Localização, o Estabilizador de Posição, e o Acionamento dos Motores. Nesta arquitetura, o Estabilizador de Posição é responsável por comparar a postura de referência p_r com a postura atual do robô p –ou seja, o erro de postura e_p – e gerar os perfis de velocidade de controle $\dot{\varphi}_{i,c}$ para cada uma das i rodas do robô móvel. As velocida-

des $\dot{\varphi}_{i,c}$ são utilizadas como o parâmetro de entrada para o acionamento do motor correspondente a i-ésima roda, cujo controlador, implementado em hardware, trabalha para manter o erro $e_{\dot{\varphi}}$.em zero.

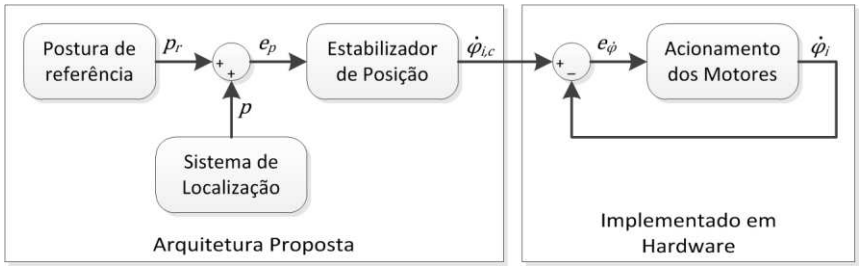


Figura 4.13 – Arquitetura do controlador de posição.

A arquitetura do controlador de trajetória, mostrada pela Figura 4.14, é semelhante à arquitetura do controlador de posição. Nesta arquitetura, porém, o Estabilizador de Trajetória é responsável por comparar a postura p_r, obtida a partir da trajetória de referência, com a postura atual do robô p e gerar os perfis de velocidade $\dot{\varphi}_{i,c}$ para cada uma das i rodas do robô móvel.

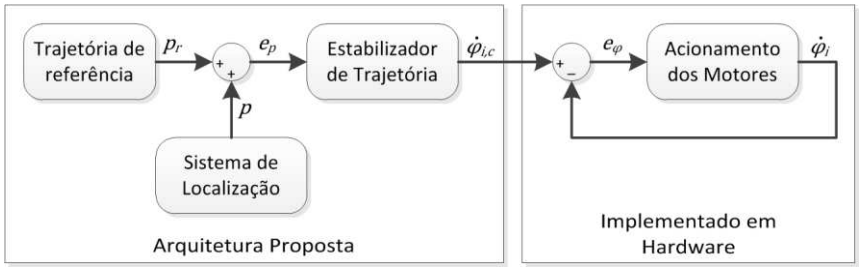
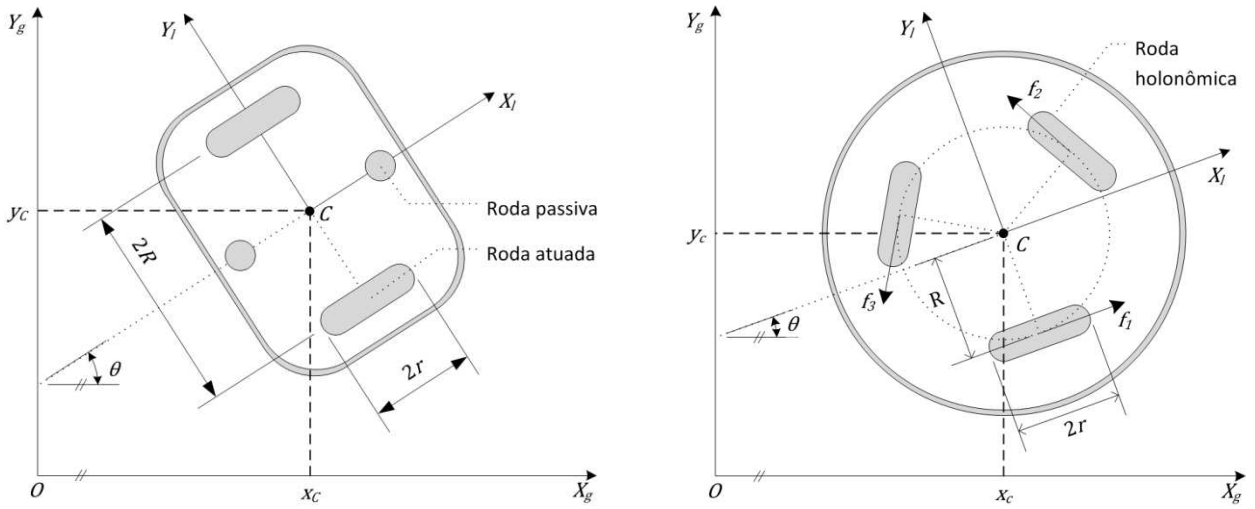


Figura 4.14 – Arquitetura do controlador de trajetória.

Neste trabalho, serão desenvolvidos o Estabilizador de Posição, o Estabilizador de Trajetória e o Sistema de Localização, conforme mostra a Figura 4.14 e Figura 4.14. O acionamento dos motores, por outro lado, será estudado neste capítulo, porém não será desenvolvido neste trabalho, pois já está implementado em hardware pelos robôs que serão utilizados.

Para desenvolvê-los, os modelos cinemáticos dos robôs holonômicos e não-holonômicos foram estudados. O modelo cinemático diz respeito às características geométricas do robô é obtido através do estudo de seu deslocamento provocado pelas velocidades

de seus atuadores. Neste capítulo são apresentados os modelos cinemático para robôs com configurações holonômicas e não-holonômicas. Os modelos matemáticos de ambas configurações são apresentados pela Figura 4.15.



(a) (b)

Figura 4.15 – Modelo matemático dos robôs móveis com (a) sistema de tração diferencial e (b) sistema de tração holonômico

Na notação utilizada no estudo dos robôs apresentados na Figura 4.15: OX_gY_g é o sistema de coordenadas global ou inercial $\mathcal{F}_g$, e CX_lY_l é o sistema de coordenadas local $\mathcal{F}_l$ do robô; C é o centro de massa ou ponto de guiamento do robô, que neste caso coincide com o ponto P de interseção entre o eixo de simetria e o eixo das rodas, ou seja, $C = P$; r é o raio das rodas, sendo que os raios das rodas de cada robô são iguais; R é a distância entre as rodas e o eixo de simetria; m_c é a massa do robô sem rodas e motores; I_c é o momento de inércia do robô sem rodas e motores em respeito ao eixo vertical através do ponto P; $\dot{\varphi}_i$ e v_i correspondem respectivamente às velocidades angular e linear para a roda i; v e ω são as velocidades linear e angular do robô diferencial, tendo $\mathcal{V} = (v, \omega)^{\mathrm{T}}$; e q é o vetor de coordenadas generalizadas sendo $q = [q_1\ q_2 \dots q_n]^{\mathrm{T}} \in \Re^n$. A postura do robô é dada pela tripla (x_c, y_c, θ), onde x_c e y_c são as coordenadas de C em $\mathcal{F}_g$ e θ é o ângulo de $\mathcal{F}_l$ em relação à $\mathcal{F}_l$, sabendo que $(x_c, y_c) = (x_g, y_g)$.

4.4.1 Modelo Cinemático do Robô Diferencial

O modelo cinemático foi desenvolvido em coordenadas retangulares, conforme apresentado por Martins (2010). Como neste robô móvel o ponto de guiamento é nulo (ou seja, $C = P$), ele apresenta três restrições cinemáticas (YAMAMOTO; YUN, 1994), sendo que a primeira define que o robô pode mover-se apenas na direção normal ao eixo das rodas, conforme é visto na equação (4.6), e a segunda e a terceira dizem respeito à restrição de rolamento puro das rodas esquerda e direita respectivamente, conforme definido nas equações (4.7) e (4.8).

$$\dot{y}_c \cos\theta - \dot{x}_c \operatorname{sen}\theta = 0 \tag{4.6}$$

$$\dot{x}_c \cos\theta + \dot{y}_c \operatorname{sen}\theta + R\dot{\theta} - r\dot{\varphi}_d = 0 \tag{4.7}$$

$$\dot{x}_c \cos\theta + \dot{y}_c \operatorname{sen}\theta + R\dot{\theta} - r\dot{\varphi}_e = 0 \tag{4.8}$$

Sabendo que $v = \dot{x}_c \cos\theta + \dot{y}_c \operatorname{sen}\theta$ e que $\dot{\theta} = \omega$, pode-se escrever as equações (4.6) e (4.7) para obterem-se as velocidades linear v e angular ω do centro de massa C do robô, resultando em:

$$\begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{R}{r} \\ \frac{1}{r} & -\frac{R}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.9}$$

Outra forma de descrever o modelo cinemático é através das velocidades nos eixos do plano $\mathcal{F}_l$ e de rotação, ou seja, $\dot{x}_c$, $\dot{y}_c$ e $\dot{\theta}$, conforme mostra a equação (4.10).

$$\begin{aligned} \dot{x}_c &= v\cos\theta \\ \dot{y}_c &= v\,\mathrm{sen}\,\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{4.10}$$

Também é possível expressar em termos de coordenadas generalizadas q, conforme a equação (4.11).

$$\dot{q} = S(q)v(t) \tag{4.11}$$

onde

$$S(q) = \begin{bmatrix} \cos(\theta) & 0 \\ \mathrm{sen}(\theta) & 0 \\ 0 & 1 \end{bmatrix} \tag{4.12}$$

4.4.2 Modelo Cinemático do Robô Omnidirecional

O modelo cinemático foi desenvolvido em coordenadas retangulares conforme apresentado por Batlle e Barjau (2009). O modelo pode ser descrito no eixo global $\mathcal{F}_g$, conforme a equação (4.13), ou no eixo local $\mathcal{F}_l$, conforme a equação (4.14),

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} -\mathrm{sen}(\theta+\varphi_1) & \cos(\theta+\varphi_1) & R \\ -\mathrm{sen}(\theta+\varphi_2) & \cos(\theta+\varphi_2) & R \\ -\mathrm{sen}(\theta+\varphi_3) & \cos(\theta+\varphi_3) & R \end{bmatrix}\begin{bmatrix} \dot{x}_g \\ \dot{y}_g \\ \dot{\theta} \end{bmatrix} \tag{4.13}$$

$$\begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} -\mathrm{sen}(\varphi_1) & \cos(\varphi_1) & R \\ -\mathrm{sen}(\varphi_2) & \cos(\varphi_2) & R \\ -\mathrm{sen}(\varphi_3) & \cos(\varphi_3) & R \end{bmatrix}\begin{bmatrix} \dot{x}_l \\ \dot{y}_l \\ \dot{\theta} \end{bmatrix} \tag{4.14}$$

onde φ_1, φ_2 e φ_3 são as posições angulares de cada uma das três rodas do robô; $\dot{\varphi}_1$, $\dot{\varphi}_2$ e $\dot{\varphi}_3$ são as velocidades angulares das respectivas rodas; e v_1, v_2 e v_3 são as velocidades lineares das respectivas rodas.

4.4.3 Estabilizador de Posição

O estabilizador de posição é um controlador capaz de levar de uma posição C qualquer da área de trabalho até uma meta, dada pelo ponto $M = (x_M, y_M)$ no plano $\mathcal{F}_g$. Conforme mostra a Figura 4.16, sabendo-se a orientação do robô móvel e as coordenadas de C e M, é possível determinar o erro de distância d_e e o erro angular θ_e. Tais erros são determinados através da equação (4.15).

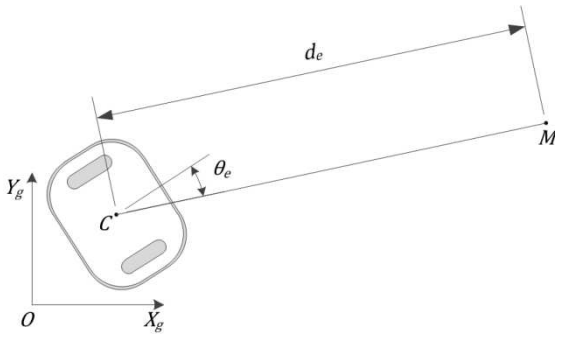


Figura 4.16 – Cinemática do robô em relação aos pontos de interesse.

$$\begin{aligned} \Delta_x &= x_M - x_C \\ \Delta_y &= y_M - y_C \\ d_e &= \sqrt{\Delta_x^2 + \Delta_y^2} \\ \theta_e &= -\theta + \text{atan2}(\Delta_y, \Delta_x) \end{aligned} \tag{4.15}$$

Considerando que o robô apresenta uma velocidade linear máxima $v_{\max}$ e é capaz de desacelerar com uma aceleração a, é possível determinar tanto a velocidade com que o robô deve se deslocar para alcançar a meta, quanto a distância d_a com a qual ele deve começar a desacelerar. A distância d_a é dada pela equação (4.16).

$$d_a = -\frac{v_{\max}^2}{2a}, \qquad a < 0 \tag{4.16}$$

Para determinar um controle (v_c, ω_c) que estabilize os erros d_e e $\theta_e$em zero, é utilizada a seguinte lei de controle:

$$\begin{aligned} v_c &= \begin{cases} v_{\max}, & d_e > d_a, \ |\theta_e| < k_1 \\ v_{\max} \cdot \dfrac{d_e}{d_a}, & d_e \leq d_a, \ |\theta_e| < k_1 \\ 0, & |\theta_e| \geq k_1 \end{cases} \\ \omega_c &= k_2 \cdot \theta_e \end{aligned} \tag{4.17}$$

Onde k_1 é o ângulo de erro mínimo aceito para que o robô possa se mover linearmente e k_2 é o ganho da velocidade angular. Com este controle, o robô se moverá apenas quando o erro θ_e for inferior a k_1, o que impede o robô de se mover mais que o necessário para alcançar sua meta.

Para avaliar este controlador, foi realizada uma simulação onde a posição inicial do robô era $C_0 = (0{,}1\ ,\ 0{,}1)$, com $\theta_0 = 0$, e a meta é dada por $M = (0{,}4\ ,\ 0{,}3)$. Os valores de das constantes e ganhos são: $v_{\max} = 0.1$ m/s, $a = -0{,}05$ m/s^2, $k_1 = \pi/16$ rad, e $k_2 = 0{,}2$.

Conforme pode ser visto nos gráficos da Figura 4.17, o robô foi capaz de sair de sua posição inicial e se dirigir até a meta em 7,62 s, com θ_e se estabilizando aos 3 s. Assim, pode-se concluir que o controlador proposto funciona de acordo com o esperado.

Na seção 6.2.1, são apresentados os experimentos práticos utilizando o Estabilizador de Posição, onde o mesmo é avaliado com diferentes posições iniciais.

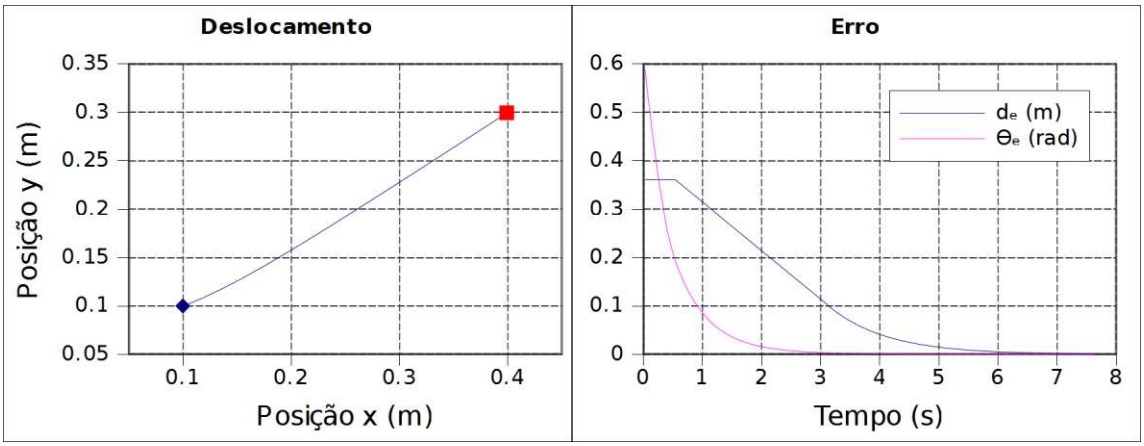


Figura 4.17 – Resultados da simulação, onde pode-se visualizar o deslocamento real do robô e os erros d_e e θ_e.

4.4.4 Estabilizador de Trajetória

Conforme já discutido na abertura deste capítulo, foi adotado o Estabilizador de Trajetória apresentado por Siciliano e Khatib (SICILIANO; KHATIB, 2008). Tal qual o trabalho original, o controlador aqui desenvolvido faz uso do modelo cinemático do robô não-holonômico diferencial, apesar da existência de um robô holonômico. Esta escolha se deu pelo fato de que um robô holonômico pode desenvolver as trajetórias de um robô não-holonômico, mas o contrário não é verdadeiro. Logo, optou-se por implementar um Estabilizador capaz de controlar ambas as classes de robô, ou seja, um Estabilizador baseado na cinemática do robô não-holonômico.

O Estabilizador, conforme pode ser visto na Figura 4.18, é composto por um Controlador de Trajetória baseado no Modelo Cinemático do robô não-holonômico e pelo Modelo Cinemático do robô que será controlado. O Controlador gera as velocidades de controle linear (v_c) e angular (ω_c), que são convertidas pelo Modelo Cinemático as velocidades de controle $\dot{\varphi}_{i,c}$ para cada uma das i rodas do robô.

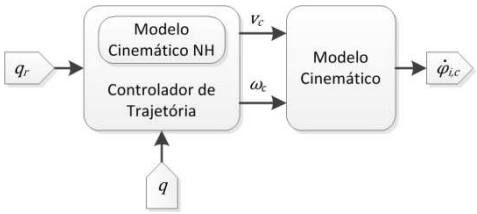


Figura 4.18 – Partes integrantes do Estabilizador de Trajetória.

Dada uma trajetória de referência em $\mathcal{F}_g$ definida no tempo por $t \longmapsto \big(x_r(t), y_r(t), \theta_r(t)\big)$ e a posição do ponto C do robô definida no tempo por $t \longmapsto \Big(x_g(t), y_g(t), \theta(t)\Big)$, o objetivo do controlador é estabilizar o erro de postura $e_p = \Big(x_g(t) - x_r(t),\ y_g(t) - y_r(t), \theta(t) - \theta_r(t)\Big)$ em zero. Para alcançar tal objetivo, o controlador necessita de uma entrada de referência de controle, que é aqui denotada por $t \longmapsto \mathcal{V}_r(t) = \big(v_r(t), \omega_r(t)\big)^{\mathrm{T}}$. Através da equação (4.10), é possível definir para um robô diferencial os valores:

$$\begin{aligned} \dot{x}_r &= v_r \cos(\theta_r) \\ \dot{y}_r &= v_r \operatorname{sen}(\theta_r) \\ \dot{\theta}_r &= \omega_r \end{aligned} \tag{4.18}$$

Supondo o sistema de coordenadas de referência $\mathcal{F}_r$ dado por $C_r X_r Y_r$, conforme mostra a Figura 4.19, é possível escrever o erro de posicionamento $\big(x_g - x_r,\ y_g - y_r\big)$ em respeito a $\mathcal{F}_r$, conforme mostra a equação (4.19).

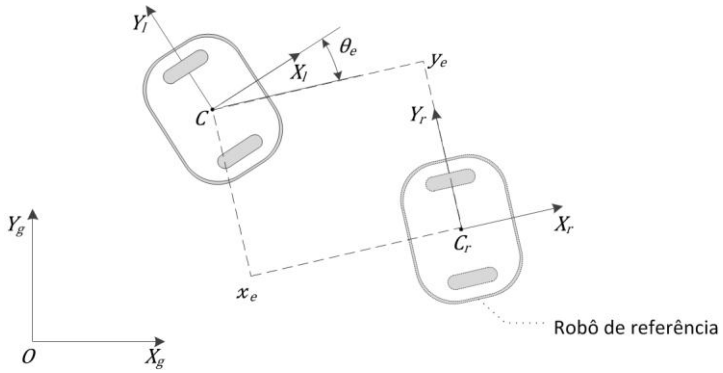


Figura 4.19 – Robô de referência e coordenadas de erro.

$$\begin{aligned}\dot{x}_e &= \omega_r y_e + v\cos(\theta_e) - v_r \\ \dot{y}_e &= -\omega_r x_e + v\,\mathrm{sen}(\theta_e) \\ \dot{\theta}_e &= \omega - \omega_r\end{aligned} \tag{4.19}$$

Para determinar um controle (v_c, ω_c) que assintoticamente estabilize o erro (x_e, y_e, θ_e) em zero, é necessário considerar a seguinte troca das coordenadas e variáveis de controle:

$$(x_e, y_e, \theta_e, v_c, \omega_c) \longmapsto (z_1, z_2, z_3, w_1, w_2) \tag{4.20}$$

onde:

$$\begin{aligned}z_1 &= x_e \\ z_2 &= ye \\ z_3 &= \tan(\theta_e) \\ w_1 &= v_c\cos(\theta_e) - v_r \\ w_2 &= \frac{\omega_c - \omega_r}{\cos^2(\theta_e)}\end{aligned} \tag{4.21}$$

Devido à $z_3 = \tan(\theta_e)$, este mapeamento só é definido quando $\theta_e \in (-\pi/2, \pi/2)$, ou seja, o erro de orientação entre o robô e sua referência deve ser menor que $\pi/2$.

Com a equação (4.20), é possível reescrever o sistema para:

$$\begin{aligned}\dot{z}_1 &= \omega_r z_2 + w_1 \\ \dot{z}_2 &= -\omega_r z_1 + v_r z_3 + w_1 z_3 \\ \dot{z}_3 &= \omega_c\end{aligned} \tag{4.22}$$

resultando, finalmente, na lei de controle:

$$
\begin{aligned}
w_1 &= -k_1|v_r|(z_1 + z_2 z_3), \qquad k_1 > 0 \\
w_2 &= -k_2 v_r z_2 - k_3|u_r|z_3, \qquad k_2, k_3 > 0
\end{aligned}
\tag{4.23}
$$

que gera a origem do sistema da equação (4.22) globalmente estável se v_r for uma função diferenciável restrita cuja derivada é restrita e não tende a zero conforme t tende ao infinito.

Para investigar a eficiência do controlador desenvolvido, foi utilizado o software Simulink® que permitiu a visualização da trajetória realizada pelo robô móvel e os erros. Para a simulação apresentada pela Figura 4.20, foi utilizada a trajetória de referência $\mathcal{V}(t) = (0{,}5\ ,\ 0{,}1)^{\mathrm{T}}$ partindo da origem $(0{,}0)$. O robô móvel foi posicionada na posição inicial $(-1, -1)$ com orientação $\theta = -1$ rad. O Sistema de Localização utilizado foi a odometria, devido a fácil implementação.

A simulação durou 62 segundos, e, conforme pode ser visto na Figura 4.20 (a), em $t = 25$ a configuração do robô coincidiu com a configuração de referência. A Figura 4.20 (b) mostra o movimento do robô no plano cartesiano.

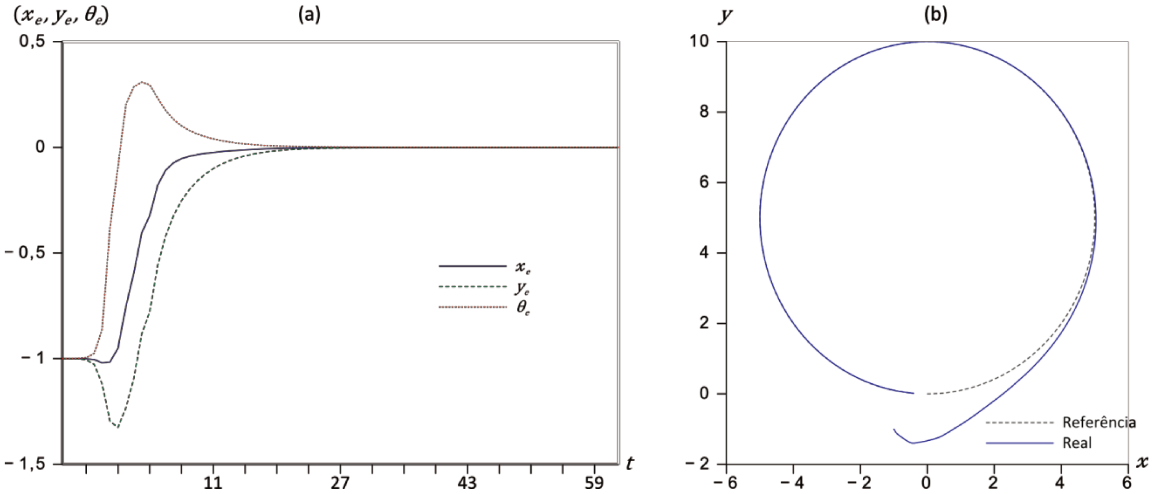


Figura 4.20 – Resultados da simulação do estabilizador de trajetórias, onde (a) apresenta as coordenadas de erro através do tempo e (b) apresenta o movimento cartesiano

4.5 Colaboração entre Robôs Móveis

A colaboração entre robôs móveis, conforme discutido na seção 3.4, é um assunto atual e de grande interesse para a comunidade. Entretanto desenvolver um repositório de módulos de software genéricos para a colaboração entre robôs móveis não é uma tarefa simples. A colaboração envolve não apenas as características do robô, ela leva em conta os aspectos funcionais da tarefa que o robô deve executar, seus objetivos e a dinâmica de interação de um robô para com os seus pares e para o ambiente. Esta forte relação de dependência entre as técnicas de colaboração, os robôs móveis, as tarefas e o ambiente dificultam a concepção de um repositório padronizado que contemple os métodos de colaboração existentes.

A seguir, serão abordadas as técnicas de colaboração escolhidas para serem desenvolvidas durante o trabalho.

4.6 Comunicação Explícita através de uma Rede de Computadores

Neste trabalho, os métodos de colaboração entre robôs móveis limitam-se a implementação de técnicas de comunicação, mais especificamente à comunicação explícita, que se refere a mecanismos desenvolvidos com o intuito de permitir o intercambio de dados entre robôs móveis. Esta técnica foi escolhida porque pode ser facilmente implementada com protocolos de rede e porque os robôs disponíveis apresentam uma interface de rede de computadores IP (*Internet Protocol*, ou Protocolo de Internet) através de computadores remotos ou embarcados.

A implementação da comunicação explícita através de uma rede de computadores requer a definição de um protocolo de comunicação sobre os protocolos de rede como o TCP (*Transmission Control Protocol*, ou Protocolo com Controle de Transmissão) ou o UDP (*User Datagram Protocol*, ou Protocolo de Datagrama de Usuário). O TCP é um protocolo interessante pois permite a entrega de pacotes com confiabilidade, porém seu mecanismo de en-

trega de pacotes o torna lento. Por outro lado, o UDP não oferece nenhuma confiabilidade pois não implementa nenhum mecanismo que garanta a entrega da mensagem, entretanto, apresenta maior velocidade que o TCP (KUROSE; ROSS, 2005). Outros protocolos podem ser utilizados, como o HTTP (*Hypertext Transfer Protocol*, ou Protocolo de Transferência de Hipertexto) baseado em TCP, o XMPP (*Extensible Messaging and Presence Protocol*, ou Protocolo Extensível de Mensagem e Presença) também baseado em TCP, dentre outros.

Neste trabalho, será adotado o protocolo UDP/IP (UDP sobre IP) devido à velocidade da entrega da mensagem. Sobre ele, será descrito um protocolo que permita aos robôs trocar mensagens através da rede. Para isto, cada robô deverá conhecer anteriormente o endereço IP dos demais robôs presentes na rede, bem como o porto aberto para a comunicação e seu número de identificação. A plataforma não especifica um padrão para os dados transferidos, uma vez que, conforme discutido na abertura deste capítulo, os tipos de dados que serão trocados dependem do tipo de robô, da aplicação e do ambiente.

O sistema aqui descrito é mostrado pela Figura 4.21, onde são exemplificados três robôs hipotéticos com IPs diferentes, ouvindo o porto 5000, conectados a um ponto de acesso sem fio comum. Estes três robôs são capazes de trocar entre si mensagens escritas sobre o protocolo UDP.

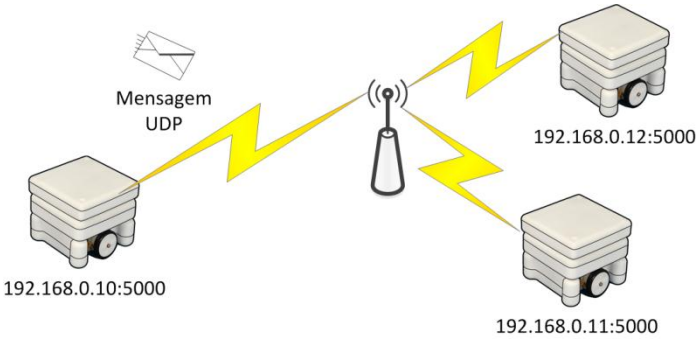


Figura 4.21 – Sistema de Comunicação Explícita proposto.

4.7 Conclusão

Neste capítulo foram apresentados o Mapa, o Planejamento de Trajetória, o Estabilizador de Trajetória, e o Sistema de Comunicação utilizado pela plataforma. Conforme mostraram as simulações, tanto o Estabilizador quanto o Controlador obtiveram resultados satisfatórios. O próximo passo se dará na implementação real do algoritmo para testar seu desempenho em um caso real.

O sistema de comunicação apresentado é simples, porém permite a troca de diferentes tipos de dados através de uma rede *loopback*, IEEE 802.3 (Ethernet, cabeada), ou IEEE 802.11 (Wi-Fi, sem fio). Ele se baseia em um protocolo próprio sobre UDP/IP, o que garante rápida resposta ao sistema. Como os testes serão realizados utilizando uma rede *loopback*, a taxa de erro de pacotes UDP é nula, logo não há preocupação com a inconfiabilidade do protocolo.

Este sistema também permite a troca de dados através da Internet, entretanto este caso não será considerado, pois acrescentaria um nível de complexidade indesejado para este trabalho. Isto se deve porque ao expor um software à Internet, este deve levar em consideração aspectos que podem ser negligenciados em redes de curto alcance e controladas. Dentre estes aspectos, o mais significativo é a segurança dos dados, que diz respeito tanto a integridade dos dados transmitidos quanto a sua veracidade.

5 PLATAFORMA DE SOFTWARE

De forma geral, os computadores constituem hoje o módulo responsável pela tomada de decisão do robô e pela arbitração dos demais módulos. Isso se deve a diversos fatores, dos quais os principais são: a redução contínua de custo, consumo energético e tamanho dos equipamentos; o aumento do poder de processamento e memória; o surgimento de linguagens de programação com maior grau de abstração; e a popularização do computador em nossa sociedade. Para ser adaptado às necessidades do projeto, o computador necessita ser programado, o que envolve as linguagens de programação para a composição do software de controle. Portanto, é razoável supor que a adoção da arquitetura de software é um momento crucial para o projeto do software de controle, pois definirá não apenas o modelo de processamento adotado, como também a facilidade de implementação, a manutenção e o reuso de código, conforme discutido na seção 3.6.

O objetivo principal deste trabalho é o desenvolvimento de uma plataforma de software que facilite o desenvolvimento de aplicativos de controle para robôs móveis através da padronização das interfaces de software para a Navegação e a Colaboração entre Robôs Móveis apresentadas nos itens 4 e 4.5. Neste sentido, o desenvolvimento da plataforma contemplou três características para o software, baseadas nos Fatores de Qualidade de McCall (MCCALL, 2002):

- **Flexibilidade**, que diz respeito à facilidade com a qual a plataforma pode ser modificada para atender às necessidades do projeto;
- **Reusabilidade**, que remete ao problema do reaproveitamento de código para diferentes projetos;
- **Portabilidade**, que está associada ao esforço necessário para transferir a plataforma para robôs com diferentes hardwares.

Para alcançar estas características, a plataforma desenvolvida – batizada de TAO – faz uso de interfaces padronizadas de software que implementam diferentes módulos da nave-

gação e colaboração entre robôs móveis. Os módulos contemplados atualmente pela plataforma são: localização, mapas, busca de caminho, controle e comunicação, conforme mostra o diagrama da Figura 5.1.

Figura 5.1 – Plataforma TAO.

As interfaces padronizadas, conforme será discutido a seguir na seção 5.1, permitem o reuso de código através da implementação dos métodos comuns aos módulos de mesma categoria. Além disso, elas aumentam a flexibilidade do software, pois permitem sua modularização.

Finalmente, foi escolhida uma linguagem de programação que permitiu tanto implementar tais interfaces de forma eficiente, quanto executar a plataforma em diferentes hardwares de robôs móveis – seja de forma remota ou embarcada – ou ainda em ambientes de simulação, conforme ilustra a Figura 5.2.

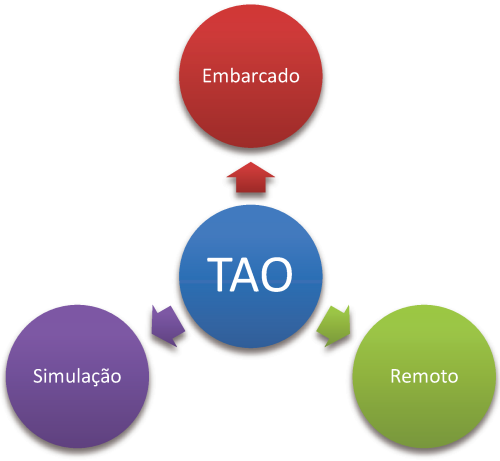


Figura 5.2 – Formas de execução da plataforma TAO.

Neste capítulo, serão apresentadas as decisões arquiteturais, as escolhas tecnológicas e os módulos desenvolvidos para a plataforma TAO.

5.1 Arquitetura e Tecnologia

Dentre as arquiteturas de software introduzidas na seção 3.6, optou-se pela arquitetura baseada em bibliotecas. Arquiteturas baseadas em bibliotecas não requerem comunicação entre processos (IPC – *Inter-Process Communication*), o que reduz a complexidade do algoritmo. Além disso, as bibliotecas são compiladas junto ao arquivo executável do software de controle, ou seja, suas rotinas estão no escopo de execução do software do usuário, o que resulta em maior velocidade na chamada destas rotinas. Uma desvantagem, porém, é o fato das bibliotecas de software requererem que o código fonte seja escrito na mesma linguagem da biblioteca, inibindo o uso de diferentes linguagens de programação.

Conforme discutido, uma biblioteca é escrita em uma linguagem de programação. No caso deste trabalho, foi escolhida a linguagem C++ para o desenvolvimento da arquitetura, pois é uma linguagem robusta, aberta, multiplataforma e com suporte a Orientação a Objetos (OO). É sabido que o uso de OO é menos eficiente, em termos de desempenho, que a programação estruturada devido ao uso demasiado de tabelas de endereços que dão suporte à implementação das classes. Por outro lado, o uso de OO confere maior flexibilidade ao promover o reuso de código, característica muito utilizada no desenvolvimento desta arquitetura. Conforme será visto no decorrer deste capítulo, a característica mais importante do C++ para esta arquitetura é o polimorfismo, pois permite que classes descendentes sejam apontadas por ponteiros do tipo da classe ancestral (PRATA, 2004). Como a perda de desempenho proveniente da abordagem OO não é significativa, esta linguagem foi escolhida em detrimento do C.

Outra escolha tecnológica diz respeito ao SO utilizado. Apesar de ser possível criar bibliotecas multiplataforma, capazes de funcionar em diferentes tipos de SO, neste trabalho foi contemplada apenas uma família de SO, o GNU/Linux, com o uso de bibliotecas POSIX

(*Portable Operating System Interface*, ou Interface Portável ente Sistemas Operacionais) para melhorar sua portabilidade. O GNU/Linux designa uma família de SOs de código fonte aberto, com várias distribuições voltadas a diferentes aplicações, e que contempla diferentes plataformas de hardware (LINUX KERNEL ORGANIZATION, 2011). Em decorrência desta escolha, a TAO pode ser executada em qualquer plataforma robótica que forneça suporte a uma distribuição do GNU/Linux, seja de forma remota ou embarcada. Para o desenvolvimento deste trabalho, foi adotada a distribuição Ubuntu em sua versão 10.04 de 32 bits, por ser uma distribuição popular e madura.

Neste trabalho, cada módulo de um sistema robótico móvel foi mapeado por uma classe, que serve como interface padronizada para tais módulos. Esta decisão foi tomada com o intuito de padronizar as interfaces para facilitar e incentivar o reuso de código. Uma interface consiste em uma classe que contém definições, ou "rascunhos", dos métodos que devem ser desenvolvidos nas classes descendentes que implementam tal interface. Em outras palavras, todas as classes descendentes de uma interface possuem métodos com o mesmo nome, mesmos parâmetros e mesma funcionalidade, porém os códigos destes métodos são diferentes para cada classe. Junto ao polimorfismo, estas interfaces permitem que suas classes descendentes sejam trocadas ou reusadas por outras classes sem a necessidade de reescrever o código fonte.

Para exemplificar como as interfaces foram utilizadas, suponha duas interfaces: *Tração* e *Localização*. A interface *Tração*, mostrada pela Figura 5.3 (a), contém os métodos necessários para acionar um sistema de tração qualquer, seja ele com rodas ou não, através das velocidades descritas pelo robô. Já a interface *Localização*, mostrada pela Figura 5.3 (b), fornece os métodos necessários para verificar qual a localização estimada por um sistema de localização qualquer. Considere, também, quatro classes hipotéticas: *Diferencial* e *Holonômico*, mostradas respectivamente pela Figura 5.3 (c) e Figura 5.3 (d), que representam dois sistemas de tração diferentes e que implementam a interface *Tração*; Odometria e *Visão Global*, mostradas respectivamente pela Figura 5.3 (e) e Figura 5.3 (f), que representam dois sistemas de localização diferentes e que implementam a interface *Localização*.

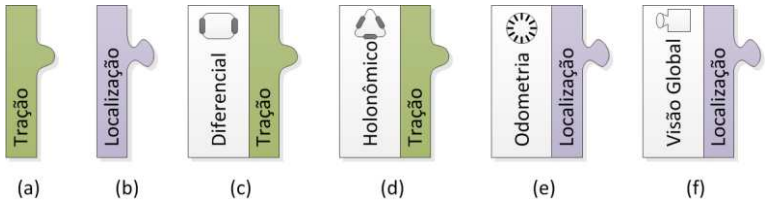


Figura 5.3 – Exemplo do uso de interfaces, onde: (a) é a interface para os sistemas de tração – *Tração*; (b) é a interface para sistemas de localização – *Localização*; (c) é uma classe que implementa a interface *Tração* para um robô diferencial; (d) é uma classe que implementa a interface *Tração* para um robô holonômico; (e) é uma classe que implementa a interface *Localização* para um sistema de odometria; e (f) é uma classe que implementa a interface *Localização* para um sistema de visão global;

Por implementarem a mesma interface *Tração*, tanto a classe *Diferencial* quanto a classe *Holonômico* fornecem os métodos padrão de *Tração*, ao passo que *Odometria* e *Visão Global* fornecem os métodos padrão de *Localização*. Logo, qualquer objeto de *Diferencial* pode ser substituído por um objeto de *Holonômico*, da mesma forma que um objeto de *Odometria* pode ser substituído por um objeto de *Visão Global*. Contudo, um objeto de *Diferencial* não pode, por exemplo, substituir um objeto de *Odometria*, pois implementam diferentes tipos de interface.

Com as interfaces *Tração* e *Localização*, é possível escrever códigos que fazem uso dos métodos por elas fornecidos. Por exemplo, suponha a classe *Controlador*, mostrada pela Figura 5.4 (a), responsável por controlar a locomoção de um robô móvel. A classe *Controlador* utiliza um sistema de localização – fornecido pela interface *Localização* – para recuperar a posição atual do robô móvel, e um sistema de tração – fornecido pela interface *Tração* – para locomover o robô. Por usar as interfaces padrão, a classe *Controlador* pode utilizar as classes *Diferencial*, *Holonômico*, *Odometria* e *Visão Global* sem necessitar a adequação de seu código. Para utilizá-las, a classe Controlador requer apenas que as classes desejadas sejam referenciadas nas interfaces corretas. Por exemplo, a Figura 5.4 (b) mostra a classe *Controlador* utilizando as classes *Diferencial* e *Odometria* referenciadas em suas respectivas interfaces, enquanto a Figura 5.4 (c) mostra a mesma classe Controlador utilizando as classes *Holonômico* e *Visão Global*. Para que houvesse a modificação do sistema de tração e localização, não foi necessário modificar o código fonte da classe *Controlador*. Para modifica-lo, foi necessário apenas trocar as classes referenciadas pelas interfaces.

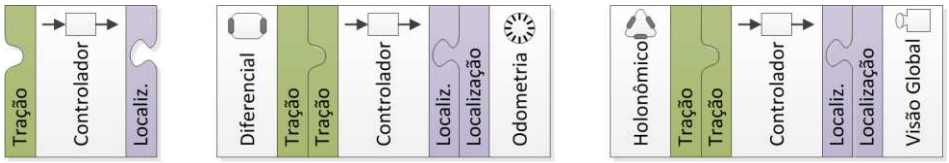


Figura 5.4 – Exemplo do uso de interfaces, onde: (a) é a classe *Controlador* que controla a locomoção de um robô móvel e, para este fim, utiliza as interfaces *Tração* e *Localização*; (b) mostra a classe *Controlador* utilizando as classes *Diferencial* e *Odometria*; e (c) mostra a classe *Controlador* utilizando as classes *Holonômico* e *Visão Global*.

Portanto, a arquitetura de software adotada facilita o desenvolvimento de aplicativos para robôs móveis ao promover flexibilidade e reusabilidade. A portabilidade é obtida através da adoção do SO GNU/Linux, que oferece suporte a diferentes plataformas de hardware, e das bibliotecas POSIX, que garantem que a plataforma possa ser executada em qualquer uma destas distribuições.

As classes desta arquitetura foram divididas em 2 pacotes que serão apresentados a seguir: *Navigation2D* (Navegação 2D), com as classes que implementam a navegação no plano apresentada no capítulo 4; e *Collaboration* (Colaboração), com as classes para comunicação explicita através da rede também descrita no capítulo 4.

5.2 Pacote Navigation 2D

O pacote *Navigation2D* é composto por duas estruturas básicas e quatro subpacotes. As estruturas básicas, mostradas pela Figura 5.5, são: *Pose* (Pose), que grava as informações sobre a pose do robô com a tripla (x, y, θ); e *Speed* (Velocidade), que grava informações de velocidade em termos de $(\dot{x}, \dot{y}, \dot{\theta})$. Estas estruturas são utilizadas pelas demais classes do pacote *Navigation2D*, presentes nos subpacotes, para representar as poses e velocidades do robô. Já os quatro subpacotes são: *Localization* (Localização), que fornece as interfaces para implementar as técnicas de localização; *Locomotion* (Locomoção), com as interfaces necessárias para implementar a locomoção do robô; *Map* (Mapa), que contém as classes que representam diferentes tipos de mapa; e *Path* (Caminho), que oferece suporte a construção de caminhos.

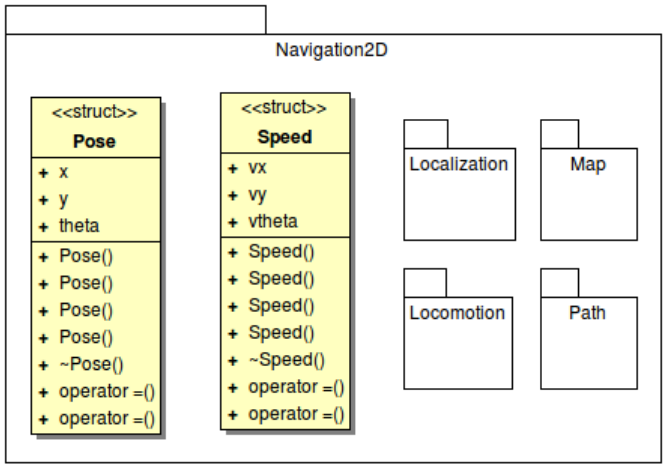


Figura 5.5 – Classes básicas do pacote *Navigation2D*.

O pacote *Localization* (Localização) contém a classe de interface *LocalizationTechnique* (Técnica de Localização) que serve como classe ancestral para todas as técnicas de localização implementadas dentro desta arquitetura. Portanto, ela oferece métodos padronizados para ler a posição e velocidade do robô. Há também um método para configurar a posição do robô, que é utilizado por sistemas de localização inercial ou para calibrar o offset de sistemas de localização absolutos. A Figura 5.6 mostra o pacote *Localization* e a interface *LocalizationTechnique*.

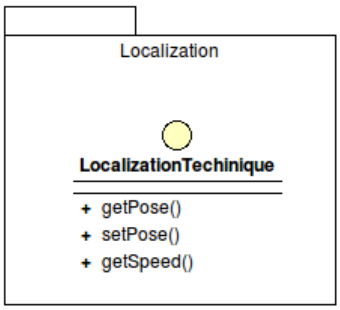


Figura 5.6 – Pacote *Localization* e sua classe *LocalizationTechnique*.

As classes que representam os mapas são encontradas no pacote *Map* (Mapa). Neste momento, apenas o mapa de grade de ocupação, descrito na seção 4.2.1, foi implementado, logo o pacote *Map* conta apenas com a classe *GridMap*, conforme mostra a Figura 5.7.

O pacote *Path* (Caminho), mostrado pela Figura 5.8, é responsável por implementar as classes e interfaces referentes à geração de caminhos ou trajetórias. Para isso, foram desen-

volvidas duas estruturas para representar caminhos – uma em termos de posição, *PathPose*, e outra em termos de velocidade, *PathSpeed* – e duas classes para algoritmos de busca de melhor caminho. A classe *PathFinder* é a interface que dita os métodos que uma função de busca de melhor caminho deve oferecer. Para exemplificar o uso desta interface e também fornecer um algoritmo de busca para a arquitetura, foi criada a classe AStar, descendente de *PathFinder*, que implementa o algoritmo A^*. É importante notar que a classe *PathFinder* – e, consequentemente, a classe *AStar* – recebem como parâmetro um mapa do tipo *GridMap* e retornam um caminho descrito pela classe *PathPose* ou *PathSpeed*.

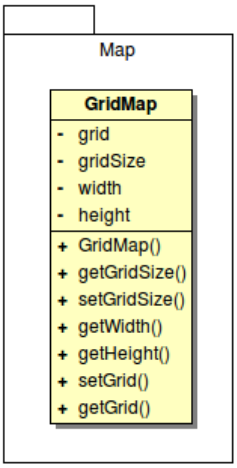


Figura 5.7 – Pacote *Map* com a classe *GridMap*.

Finalmente, o pacote *Locomotion* (Locomoção), que pode ser visto na Figura 5.9, fornece uma interface para o sistema de tração de um robô móvel, uma interface para sistemas de controle de locomoção, e duas classes para controle de locomoção. A interface para o sistema de tração do robô, *DriveSystem*, fornece os métodos necessários para acionar os motores do robô através das velocidades $(\dot{x}, \dot{y}, \dot{\theta})$. Todo robô que utiliza esta arquitetura deve fornecer uma classe descendente de *DriveSystem* para o acionamento dos motores do robô, que implemente, também, os métodos definidos pela classe ancestral. Já a interface *DriveControl* serve como classe ancestral para todas as classes que implementam um controle de locomoção e, portanto, as classes *TrajectoryStabilizer* e *PositionStabilizer* dela descendem. Para ser capaz de dirigir o robô, as classes descendentes de *DriveControl* devem receber um ponteiro para uma classe descendente de *DriveSystem*, que fornece o acionamento do robô, e um ponteiro para uma classe descendente de *LocalizationTechnique*, que

fornece um método de localização. Isso permite que a mesma classe descendente de *DriveControl* controle diferentes robôs utilizando diferentes sistemas de localização graças ao polimorfismo brevemente discutido no seção 5.1. Por fim, a classe *TrajectoryStabilizer* implementa o estabilizador de trajetória apresentado na seção 4.4.4, enquanto a classe *PositionStabilizer* implementa o estabilizador de caminho apresentado na seção 4.4.3.

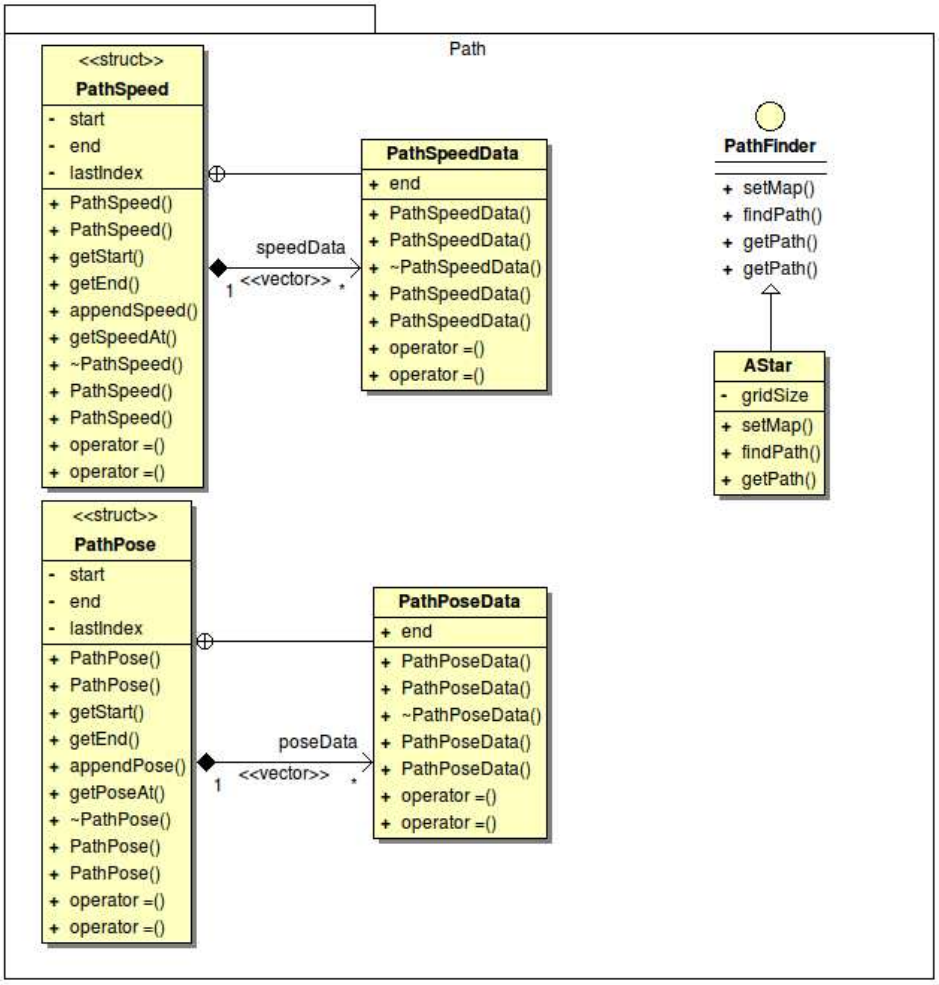


Figura 5.8 – Pacote *Path* e suas estruturas e classes.

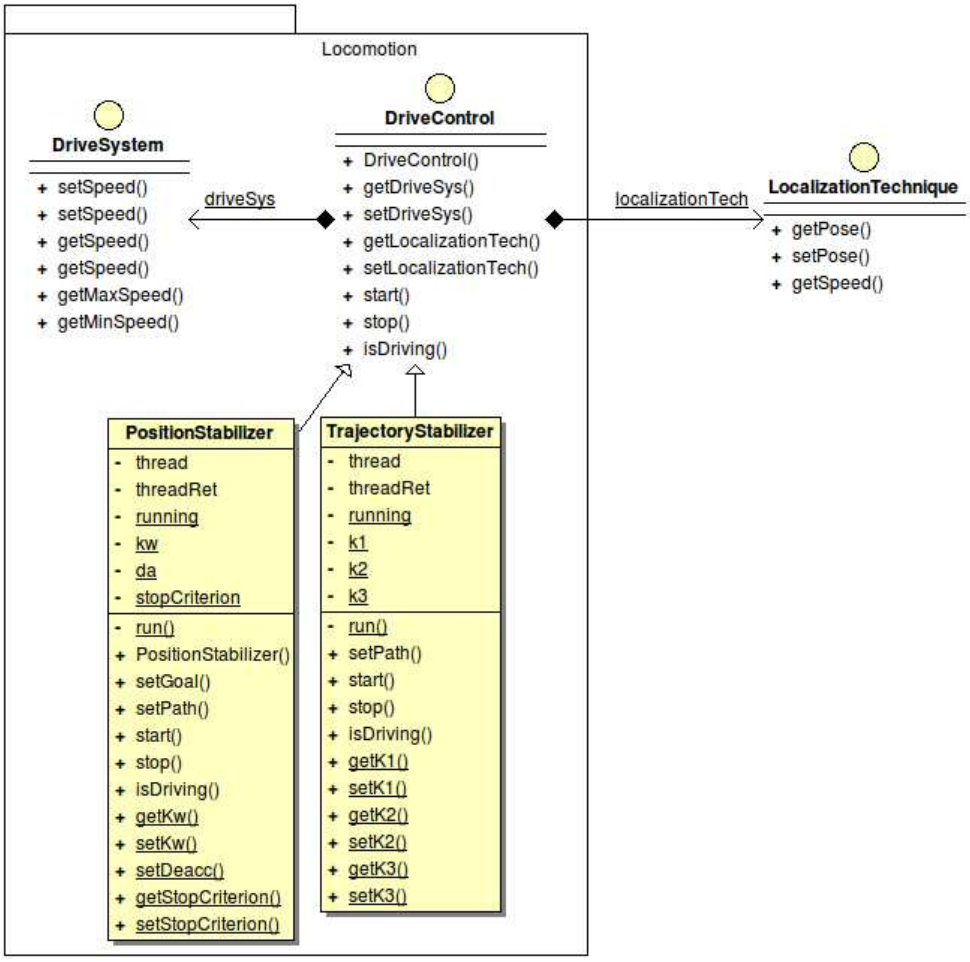


Figura 5.9 – Pacote *Locomotion*, suas classes e dependências.

5.3 Pacote Collaboration

O pacote *Collaboration* é onde as técnicas de colaboração se encontram. Neste trabalho, foi implementado apenas o pacote *Communication* que oferece a classe *UDPComm*. A classe *UDPComm* fornece um sistema simples de IPC através do protocolo UDP/IP, descrito na seção 4.6.

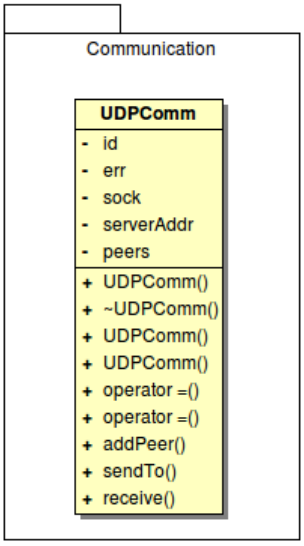


Figura 5.10 – Pacote *Communication* com sua classe *UDPComm*.

5.4 Pacotes desenvolvidos para os robôs móveis utilizados

Para que os robôs 14-bis, Roburguer, AEDROMO e Robotino® - que serão apresentados na seção 6.1 – fossem capazes de usufruir da arquitetura TAO, foi necessário criar classes de acionamento para os robôs descendentes da classe *DriveSystem*. Também foi criada uma classe descendente de *LocalizationTechnique* para o sistema de Visão Global descrito na seção 4.3.2.

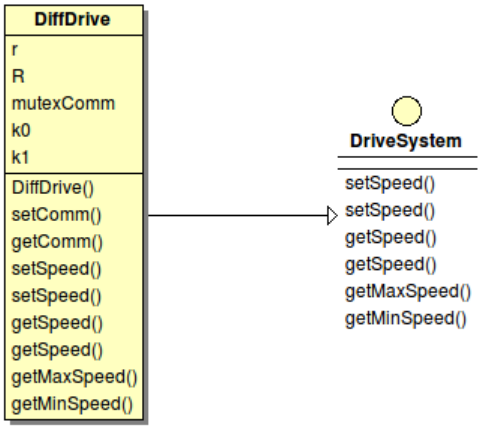


Figura 5.11 – Classe *DiffDrive*, descendente de *DriveSystem*, que implementa o sistema de tração diferencial que aciona os robôs da BER.

Como todos os robôs da BER utilizam sistema de tração diferencial, foi crida a classe *DiffDrive*, descendente de *DriveSystem*, para cada um dos robôs. Estas classes se baseiam na API de cada um dos robôs e sua representação UML é mostrada pela Figura 5.11. Para o Robotino®, que utiliza tração omnidirecional, foi criada a classe *OmniDrive*, também descendente de *DriveSystem*, conforme mostra a Figura 5.12.

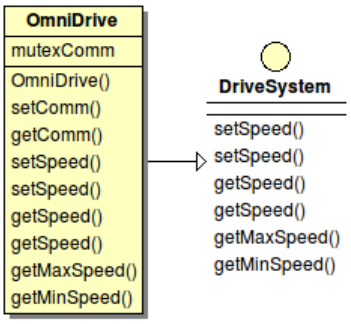


Figura 5.12 – Classe *OmniDrive*, descendente de *DriveSystem*, que implementa o sistema de tração omnidirecional para o Robotino®.

Para o sistema de Visão Global, foi criada a classe *GlobalVision*, descendente de *LocalizationTechnique*, que faz a interface com o software criado para o AEDROMO. Esta classe oferece as posições de dois robôs e de um objeto presente na área de trabalho.

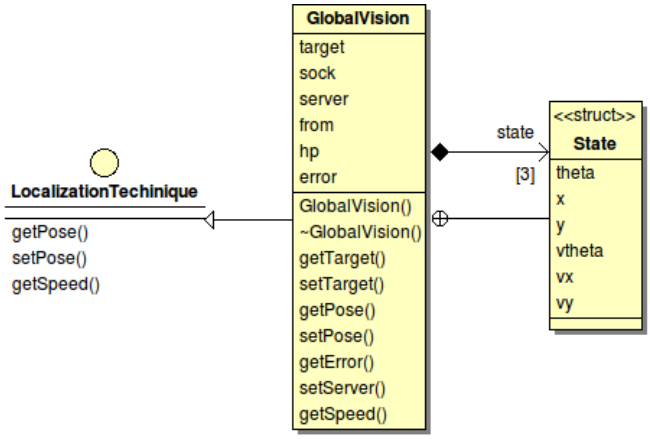


Figura 5.13 – Classe *GlobalVision*, descendente de *LocalizationTechnique*, que fornece os dados adquiridos um sistema de Visão Global.

Para fins de teste, também foi criada uma classe *Simulator*, que implementa um simulador. Esta classe descende tanto de *DriveSystem* quanto de *LocalizationTechnique*, o que permite que as demais classes da TAO modifiquem a velocidade do robô simulado e obte-

nham dados sobre sua posição e velocidade de forma natural. A classe *Simulator* é mostrada pela Figura 5.14.

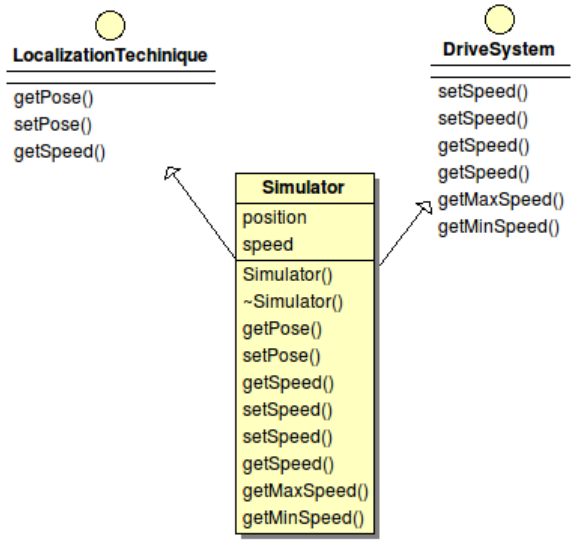


Figura 5.14 – Classe *Simulator*, que implementa um simulador.

Para o software do usuário, a classe *Simulator* funciona da mesma forma que uma classe escrita para um robô real. Desta forma, o código utilizado na simulação é o mesmo utilizado nos testes reais, não requerendo, portanto, que sejam criadas duas versões diferentes do software em ambientes de desenvolvimento diferentes. Esta flexibilidade acelera o desenvolvimento do software e facilita sua manutenção por não necessitar de versões diferentes.

5.5 Conclusão

Neste capítulo foi apresentada a plataforma de software desenvolvida. As classes associadas aos diferentes módulos de um robô móvel foram divididas em pacotes para organizar a arquitetura. Tais classes foram propostas no intuito de permitir a implementação de algumas das técnicas de navegação e colaboração num padrão predefinido que incentiva o reuso de código e fornece flexibilidade ao sistema. Neste sentido, estas classes fazem uso do polimorfismo permitido pela linguagem C++ para reutilizar classes descendentes das interfaces definidas pela arquitetura.

A plataforma resultante, TAO, apesar de estar atualmente em um estágio experimental, apresenta-se como uma alternativa interessante de arquitetura de software para a robótica por sua flexibilidade. Os testes realizados com a TAO serão discutidos a seguir no capítulo 6.

O código fonte da plataforma TAO, exemplos e o manual do usuário podem ser encontrados no site http://www2.fc.unesp.br/gisdi/TAO/.

6 EXPERIMENTOS

Para demonstrar a arquitetura TAO, foram realizados quatro experimentos, com robôs diferentes, que avaliam o desempenho das diferentes classes da arquitetura. Cada experimento é responsável por demonstrar um aspecto diferente da TAO. Neste sentido, os quatro testes são:

- **Estabilizador de Posição**: avalia as classes *PositionStabilizer*, *DriveSystem* e *LocalizationTechnique* do pacote *Navigation2D*;
- **Estabilizador de Trajetória**: avalia as classes *TrajectoryStabilizer*, *DriveSystem* e *LocalizationTechnique* do pacote *Navigation2D*;
- **Criação de Mapa e Busca de Caminho**: avalia as classes *GridMap*, *PathFinder* e *AStar* do pacote *Navigation2D*;
- **Colaboração**: avalia as classes *UDPComm* do pacote *Collaboration*.

No decorrer deste capítulo, serão apresentados os robôs móveis utilizados nos experimentos, além da discussão e resultados de cada um destes experimentos.

6.1 Robôs Móveis Utilizados

No desenvolvimento deste trabalho, serão utilizados os robôs disponíveis no LAIR (Laboratório de Automação Integrada e Robótica) do Departamento de Projeto Mecânico da Faculdade de Engenharia Mecânica da UNICAMP, e no LISDI (Laboratório de Integração de Sistemas e Dispositivos Inteligentes) do Departamento de Computação da Faculdade de Ciências da UNESP de Bauru. Os robôs disponíveis para uso são aqueles que compõem a Base Experimental Robótica (BER) – os robôs 14-bis, Roburguer e Aedromo – e o Robotino®.

Neste capítulo serão descritas as arquiteturas de cada robô, os sensores e atuadores disponíveis em cada um deles e suas respectivas interfaces de programação.

6.1.1 14-bis e Roburguer

O 14-bis é um robô desenvolvido recentemente em um trabalho conjunto entre o GIS-DI (Grupo de Integração de Sistemas e Dispositivos Inteligentes)da UNESP de Bauru e o Laboratório de Automação Inteligente (LAI) da Universidade Federal do Espírito Santo (UFES). Ambos foram construídos no intuito de servirem como interface entre crianças com deficiência motora severa e o mundo real. Neste sentido, o robô 14-bis possui, além das rodas, uma caneta no centro do seu corpo que pode ser controlada remotamente pelas crianças. De forma semelhante, o robô Roburguer possui uma grade que lhe permite segurar e arrastar objetos pequenos. Os dois robôs são mostrados pela Figura 6.1

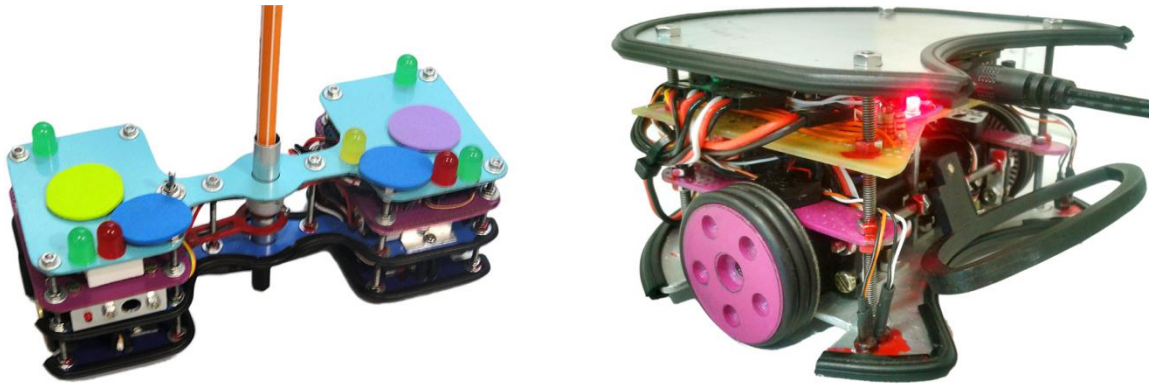


Figura 6.1 – 14-bis (a) e Roburguer (b).

Ambos utilizam a mesma arquitetura de acionamento, que é composta por: um Computador Remoto, responsável pela implementação da arquitetura de controle dos robôs; e pelos supervisores presentes em cada robô móvel, que se encarregam pelo acionamento das rodas e do servo motor que movimenta a caneta ou grade. Essa arquitetura é mostrada pela Figura 6.2.

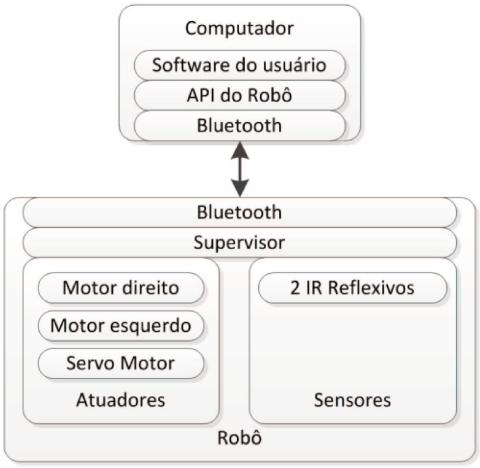


Figura 6.2 – Arquitetura de acionamento dos robôs 14-bis e Roburguer.

6.1.2 AEDROMO

O AEDROMO é um Ambiente Experimental Didático com Robôs Móveis, formado por dois robôs (ou mais), um computador, uma câmera global, do tipo *webcam*, e um transmissor, conforme mostra a Figura 6.3. A flexibilidade de adaptação e uso por várias disciplinas e ciclos de ensino, onde vários experimentos podem ser realizados e o baixo custo são as premissas primitivas na concepção e desenvolvimento deste ambiente. Com o intuito de facilitar e incentivar o uso deste ambiente, as suas dimensões também foram pontos norteadores desta pesquisa. Os experimentos, neste ambiente, podem ser motivados para fins de pesquisa, aprendizagem ou, meramente, entretenimento (FERASOLI FILHO *et al.*, 2006).

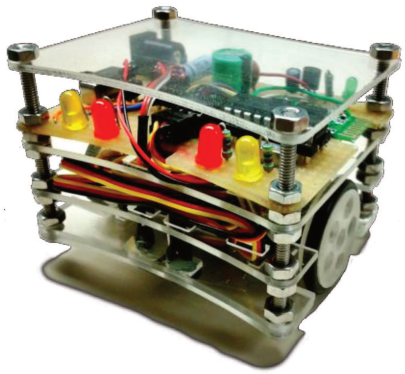


Figura 6.3 – Robô AEDROMO

Sua arquitetura de acionamento se baseia em: um sistema de Visão Global, composto por uma câmera posicionada sobre o campo onde os robôs estão situados; pelo Computador Remoto, responsável pela implementação da arquitetura de controle dos robôs; e pelos supervisores presentes em cada robô móvel que se encarrega pelo acionamento das rodas. Essa arquitetura é mostrada pela Figura 6.4.

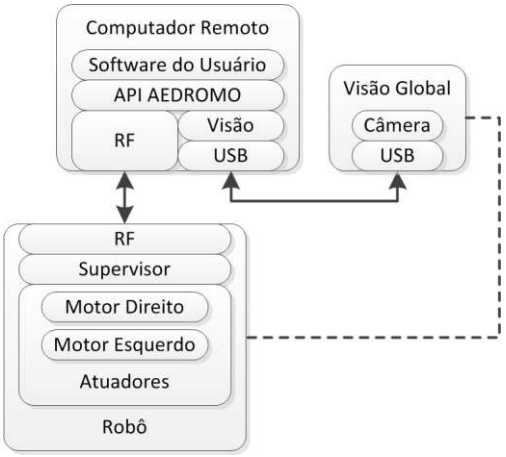


Figura 6.4 – Arquitetura de acionamento do AEDROMO.

As imagens capturadas pela câmera global são enviadas por um cabo ao computador com sistema operacional GNU/Linux. No computador as imagens são adquiridas e processadas (Sentir). O processamento determina as posições cartesianas (x, y, θ) individuais dos objetos (robôs, bolas, blocos, etc.) inseridos na área de trabalho. Estas informações são utilizadas pelo módulo de estratégia de acordo com as regras da aplicação em execução (Planejar). Então, comandos são enviados aos robôs para agirem no ambiente segundo estas regras (Atuar). Os movimentos ocorridos devido às ações são vistos pela câmera e o processo se repete numa taxa equivalente à velocidade de captura da câmera, que em geral varia de 15 a 30 fps (*frames per second*, ou quadros por segundo). Esse tipo de arquitetura usando visão global confere grande versatilidade ao sistema, uma vez que permite a pequenos robôs reagirem com o ambiente de forma complexa, sem a necessidade de inclusão de sensores sofisticados.

6.1.3 Robotino

O Robotino® é um robô móvel holonômico desenvolvido pela FESTO[4] para fins educacionais. O robô embarca um computador do tipo PC 104 com sistema operacional GNU/Linux, onde é executado continuamente um software servidor que oferece o controle do robô tanto localmente (ou seja, utilizando uma rede *loopback*) quanto remotamente através de uma conexão Wi-Fi. Ele permite a aquisição de dados de diferentes sensores e o acionamento dos atuadores. Sua programação é facilitada pela API OpenRobotino, que fornece as classes necessárias para o desenvolvimento do software de controle. Adicionalmente, ele oferece o software RobotinoView 2, que permite desenvolver o software de controle em uma linguagem visual amigável e intuitiva.

Sua arquitetura de acionamento baseia-se em um Computador Remoto, por onde o usuário pode controlar do robô pela rede Wi-Fi; pelo computador embarcado do robô, que pode ser programado localmente ou remotamente; e pelo Servidor que fornece o acesso ao robô através da rede. Esta arquitetura é mostrada na Figura 6.5.

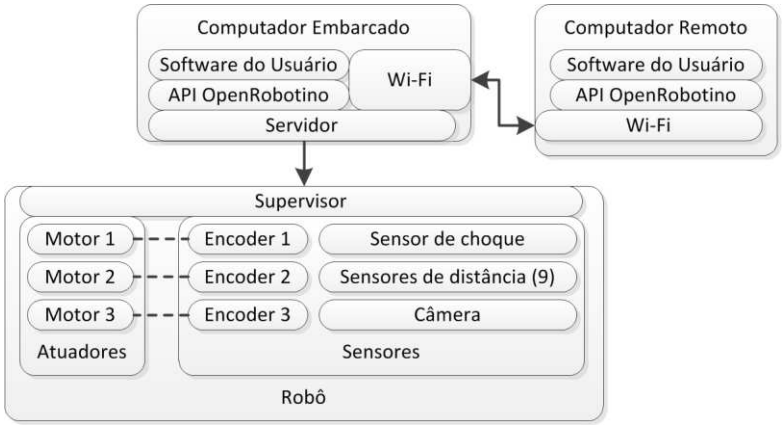


Figura 6.5 – Arquitetura de acionmento do Robotino®.

O Supervisor é responsável pela interface entre os atuadores e sensores do robô. Por ser um robô holonômico, o Robotino® dispõe de três rodas, cujas velocidades são monitoradas por três encoders. Ele também oferece 9 sensores de distância por infravermelho es-

[4] Education and Research Robots: Robotino < http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino/ >. Acesso em 06 de junho de 2011.

paçados em 40° para a detecção de obstáculos entre 5 e 40 cm. Além disso, ele fornece também um sensor de choque, usado para detectar colisões, e uma câmera USB, que pode ser utilizada como realimentação de um controle remoto ou para identificar objetos conhecidos dentro do ambiente.

O Supervisor é acionado pelo Servidor presente no Computador Embarcado. Este servidor oferece a interface do Supervisor através de protocolos de rede. Neste sentido, ele permite a execução do programa tanto dentro do Computador Embarcado (através de uma rede *loopback*) quanto dentro do Computador Remoto através de uma rede Wi-Fi. O servidor pode ser acessado através dos módulos de software fornecidos pela API OpenRobotino, disponível tanto embarcada quanto remotamente.

6.2 Experimentos

A seguir, será apresentado e discutido cada um dos quatro experimentos e seus respectivos resultados.

6.2.1 Estabilizador de Posição

Para testar o controlador discreto implementado pela classe *PositionStabilizer*, apresentada na seção 5.2, foi realizado um teste onde o robô deve se dirigir ao centro da área de trabalho, dado pelas coordenadas (0,4 , 0,3), partindo de um ponto qualquer. Na Figura 6.6, apresentam-se cinco repetições deste teste, onde a posição de partida do robô varia em cada uma das iterações. Conforme pode ser observado na mesma figura, o robô foi capaz de alcançar o centra da área de trabalho em todas as iterações do teste.

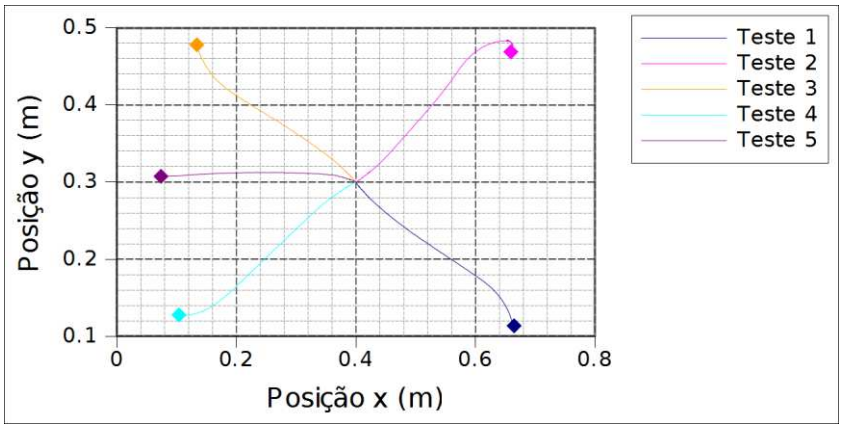


Figura 6.6 – Trajetória descrita pelo robô nos cinco testes.

O erro de posição, dado pela distância entre a posição atual do robô e o centro da área de trabalho, é apresentado pela Figura 6.7. Nela, pode-se observar que o controlador apresenta o mesmo comportamento em todas as iterações do teste. Assim que o controlador entra em ação, o erro se mantém constante até o momento em que o robô se alinha com o ponto central da área de trabalho. Uma vez alinhado, o controlador aciona o robô em sua velocidade máxima, causando a redução constante do erro até que este erro seja igual a 0,05 m. Quando o erro se torna menor que 0,05 m, o controlador passa a desacelerar o robô, o que diminui a taxa de redução do erro. No fim de cada iteração, o erro torna-se inferior ao critério de parada do controlador, que equivale a 0,001 m.

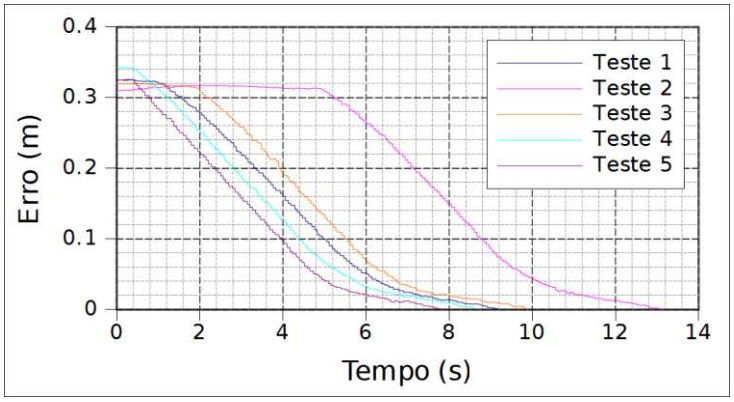


Figura 6.7 – Erro de posição em cada um dos cinco testes.

Portanto, pode-se concluir que o controlador discreto implementado pela classe *PositionStabilizer* obteve resultados satisfatórios nos experimentos.

6.2.2 Estabilizador de Trajetória

Para testar o controlador de trajetória contínuo implementado pela classe *TrajectoryStabilizer*, apresentada na seção 5.2, foi utilizada uma trajetória de referência descrita pela equação (6.1):

$$\mathcal{V}(t) = \begin{cases} (0{,}04\,,0)^{\mathrm{T}}, & 0 \leq t < 8 \\ \left(0{,}04\,,\frac{\pi}{10}\right)^{\mathrm{T}}, & 8 \leq t < 18 \\ (0{,}04\,,0)^{\mathrm{T}}, & 18 \leq t < 26 \\ \left(0{,}04\,,\frac{\pi}{10}\right)^{\mathrm{T}}, & 26 \leq t < 36 \end{cases} \tag{6.1}$$

Os ganhos do controlador foram os mesmos usados na simulação da seção 4.4.4, logo $k_1 = 1, k_2 = 1$ e $k_3 = 3$. A posição inicial do robô de referência é $(0{,}24\,,\ 0{,}15\,,\ 0)^{\mathrm{T}}$, enquanto a posição inicial do robô móvel real é dada por $(0{,}244\,,\ 0{,}127\,,\ -0{,}02)^{\mathrm{T}}$. O gráfico da Figura 6.8 mostra que o robô móvel sai de sua posição inicial e se estabiliza sobre a trajetória de referência. Comparado com a simulação da seção 4.4.4, o robô real apresentou maior erro durante a execução da trajetória. Conforme pode ser visto no gráfico, o robô real corrige a cada instante suas velocidades a fim de descrever uma trajetória que seja a mais próxima possível da trajetória de referência.

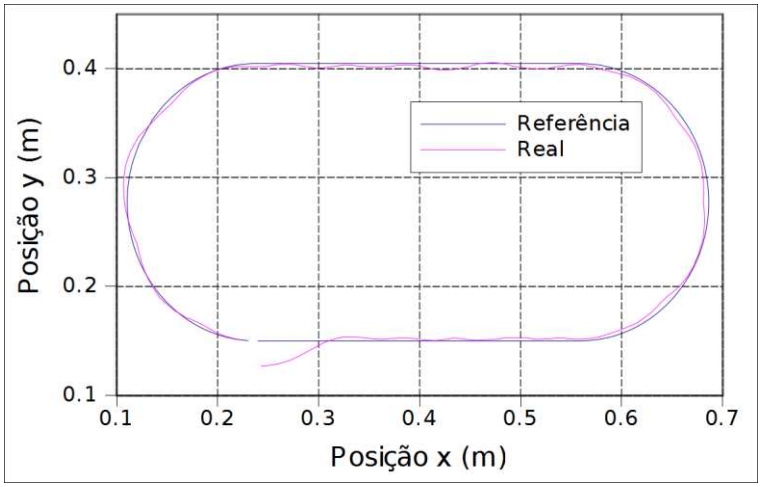


Figura 6.8 – Comparação entre a trajetória de referência data por $\mathcal{V}(t)$e a trajetória descrita pelo robô móvel.

As correções realizadas pelo robô móvel ficam mais claras nos gráficos da Figura 6.9, que traz os erros x_e, y_e e θ_e obtidos pelo experimento. Conforme pode ser visualizado, o robô desempenhou erros medianos de $-0{,}868$ para x_e, de 0,213 para y_e, e de $-0{,}107$ para θ_e, com flutuação. Estes erros são decorrentes da dificuldade do robô em manter as velocidades desejadas pelo controlador instantaneamente, do atraso de atualização do sistema de Visão Global, que é de cerca de 200 ms, e da necessidade de reajuste dos ganhos do controlador para o robô real.

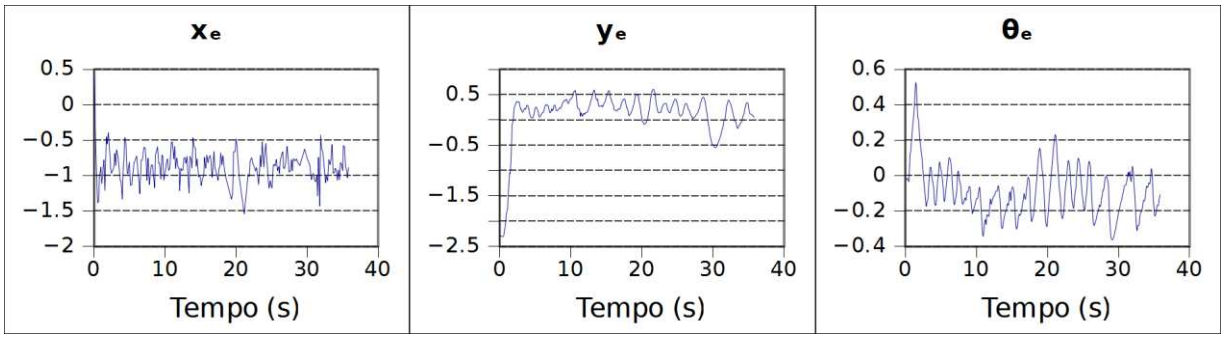


Figura 6.9 – Erros x_e, y_e e θ_e obtidos pelo experimento.

Embora os erros não se estabilizem em zero como desejado, o controlador foi capaz de dirigir o robô de forma razoável. Para melhorar os resultados, é necessário reduzir o atraso do sistema de Visão Global e reajustar os ganhos do controlador. Outra abordagem para melhorar o sistema é a aplicação do filtro de Kalman no intuito de reduzir a degradação das medidas devido a ruídos e incertezas.

Entretanto, visto que vários robôs são utilizados nesta arquitetura, optou-se por não utilizar este controlador nos demais experimentos devido à necessidade de reajustar os ganhos para cada robô. Por estas razões, foi escolhido o controlador discreto em detrimento do controlador contínuo.

6.2.3 Criação de Mapa e Busca de Caminho

Neste teste, foram avaliadas as classes *GridMap* para criação de mapas de grade de ocupação e *AStar* para busca de melhor caminho, ambas descritas na seção 5.2. Para isso, foi

elaborado um mapa de uma papelaria hipotética, que é usado para gerar o mapa de grade de ocupação a ser representado pela classe *GridMap*. Como a área de trabalho do robô mede 0,8 m por 0,6 m e o corpo do robô tem um diâmetro de 0,1 m, o mapa foi divido em grades quadradas de 0,1 m de lado. Tanto o mapa da papelaria quanto o mapa de grade de ocupação utilizados são mostrados pela Figura 6.10.

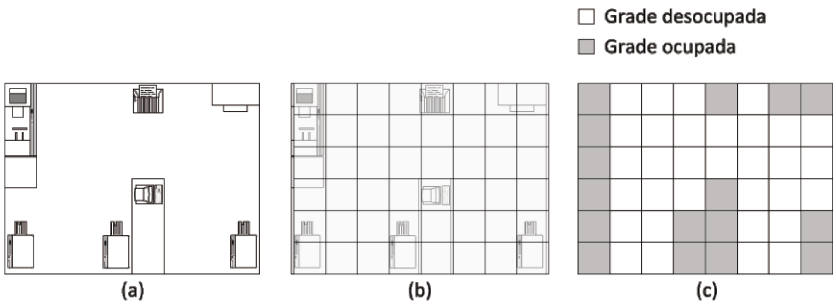


Figura 6.10 – Mapa utilizado pelo experimento, mostrando: (a) o mapa da papelaria; (b) o mapa da papelaria sobreposto pelas grades de ocupação; e (c) o mapa de grade de ocupação.

Com o mapa apresentado, estabeleceram-se as grades inicial e final para o robô, que são, respectivamente, (1, 1) e (6,1). Tendo o mapa e as grades inicial e final como parâmetros, foi utilizada a classe *AStar* que gerou o caminho mostrado pela Figura 6.11.

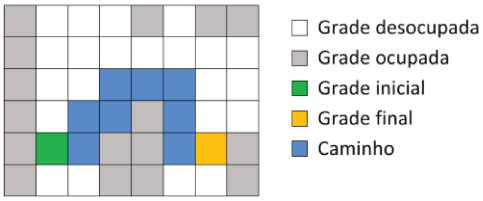


Figura 6.11 – Uso do algoritmo A^* para encontrar o caminho entre as grades inicial e final

Finalmente, com o caminho gerado, foi possível utilizar a classe *PositionStabilizer* para controlar o robô móvel dentro da papelaria hipotética, conforme mostra a Figura 6.12. Para completar o caminho da Figura 6.11, o robô levou 60,5 s.

Neste teste, o robô seguiu satisfatoriamente o caminho gerado pela classe *AStar*. O caminho evitou colisões com os obstáculos estáticos presentes no ambiente e completou a execução num tempo aceitável.

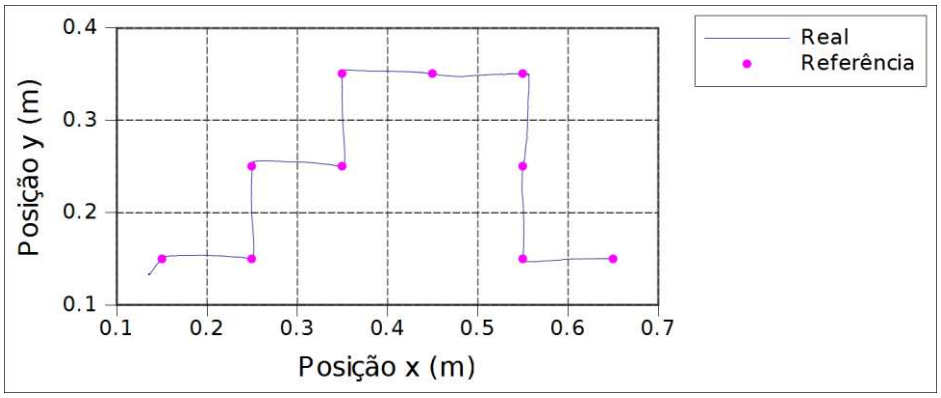


Figura 6.12 – Caminho descrito pelo robô móvel.

6.2.4 Colaboração

Para demonstrar a classe *UDPComm* do pacote *Collaboration*, foi desenvolvido um teste que envolve a ação coordenada de dois robôs móveis. Neste teste, um dos robôs desenha um círculo na área de trabalho, retorna à sua origem e finalmente notifica ao segundo que seu trabalho está pronto. Uma vez que o segundo robô recebe esta notificação, ele busca um objeto conhecido na área de trabalho e o leva ao centro do círculo desenhado pelo primeiro robô, e finalmente retorna a sua posição de origem. Este experimento é demonstrado pelo diagrama de sequência mostrado pela Figura 6.13.

Para este experimento, são usados os robôs 14-bis para desenhar, e Roburguer para mover o objeto. Foi utilizado, também, a classe *LocalizationTechnique* para identificar o objeto e a classe *PositionStabilizer* para dirigir os robôs. Foi escrito um software diferente para cada um dos robôs que é executado em paralelo. Para que ambos saibam qual o estado atual de execução de seu parceiro, foi utilizada a classe *UDPComm* que os permite trocar mensagens. Assim, é através da *UDPComm* que o robô 14-bis notifica a conclusão de sua tarefa ao robô Roburguer. Conforme ilustra a Figura 6.14, os robôs desenvolveram satisfatoriamente o experimento, colaborando com sucesso para atingir o objetivo proposto. A classe de comunicação *UDPComm* permitiu que ambos trocassem informação entre si, o que tornou possível a colaboração entre ambos.

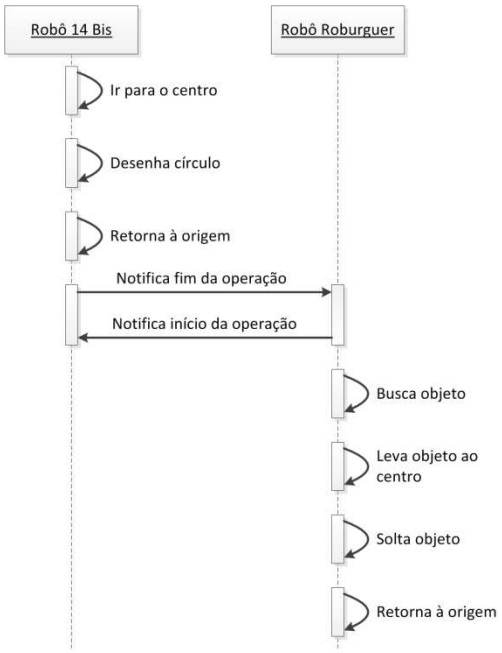


Figura 6.13 – Diagrama de sequência ilustrando o teste de colaboração.

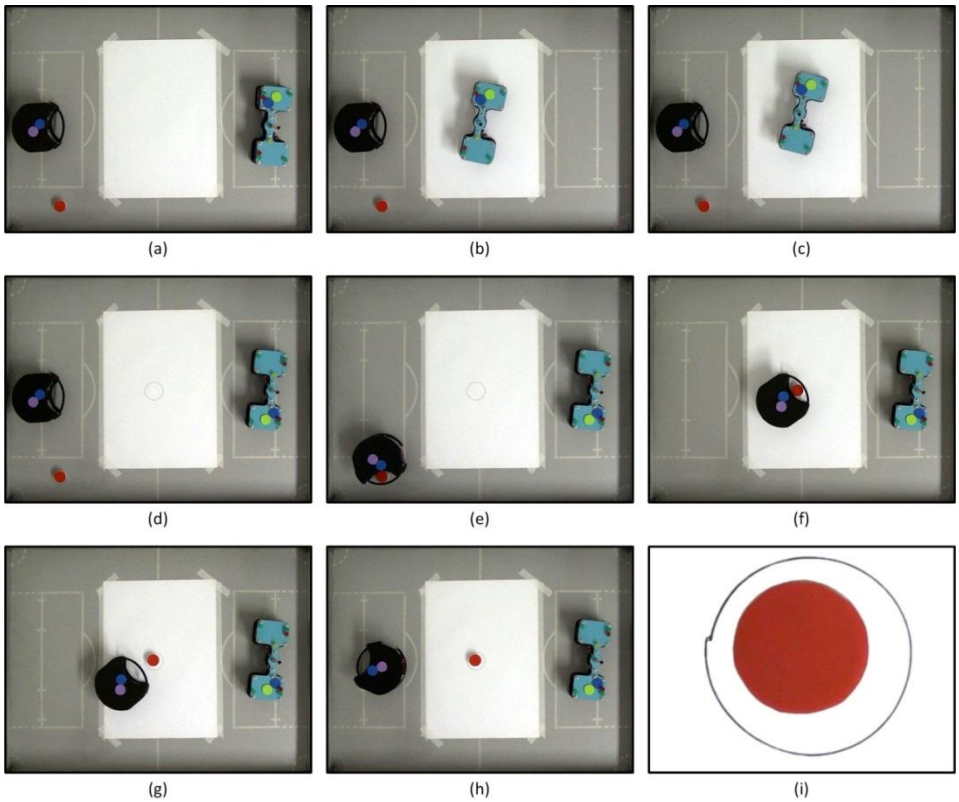


Figura 6.14 – Experimento visto por cima em diferentes momentos: (a) robôs nas posições de origem; (b) 14-bis vai ao centro da área de trabalho; (c) 14-bis desenha um círculo; (d) 14-bis volta à origem; (e) Roburguer busca o objeto; (f) Roburguer leva o objeto ao centro da área de trabalho; (g) Roburguer solta o objeto; (h) Roburguer retorna a posição de origem e encerra o teste; (i) *close-up* do círculo desenhado e do objeto nele posicionado.

6.3 Conclusão

Este capítulo apresentou os experimentos que demonstram o funcionamento da arquitetura TAO. Todas as classes desenvolvidas foram testadas e seu desempenho foi como o esperado.

Dentro de todos os experimentos, foi possível notar a vantagem de utilizar as classes padronizadas. Como as interfaces para os sistemas de tração dos robôs e o sistema de localização adotado foram escritas como classes descendentes das interfaces de software da TAO, foi possível adaptar o mesmo software de controle para diferentes robôs. O melhor exemplo disto é o experimento Colaboração, que demonstra o uso da mesma classe *PositionStabilizer* para dois robôs distintos.

O código fonte da plataforma TAO, exemplos e o manual do usuário podem ser encontrados no site http://www2.fc.unesp.br/gisdi/TAO/.

7 CONCLUSÃO E TRABALHOS FUTUROS

A robótica móvel se posiciona hoje como uma área que desperta grandes expectativas, pois há uma grande variedade de aplicações, desde domésticas até industriais, e seu custo vem caindo com o passar dos anos. Há um grande esforço no sentido de desenvolver tecnologias que permitam aos robôs móveis realizarem tarefas perigosas ou enfadonhas para os seres humanos e, também, habitarem no meio ambiente de forma harmoniosa. Prova disso é o surgimento de robôs domésticos como o Roomba, de exploração como o projeto Mars Rovers, de pesquisa e entretenimento como o Nao, carros inteligentes como os desenvolvidos para o desafio DARPA, dentre outros. Entretanto, apesar de todo o esforço da comunidade científica, não há ainda uma técnica refinada e robusta que permita a integração dos robôs móveis no âmbito social, o que significa que ainda há diversos problemas abertos para serem estudados dentro da robótica móvel.

Um dos problemas da robótica móvel diz respeito às arquiteturas de software que podem ser aplicadas no desenvolvimento do software de controle de um robô móvel. A opção por determinada arquitetura não define apenas a tecnologia que será adotada, mas causa impacto nos paradigmas de desenvolvimento, manutenção e reuso do software. Conforme discutido neste trabalho, existem diferentes abordagens na literatura que fornecem ferramentas para permitir o reuso e a fácil manutenção do código. Todavia, tais arquiteturas estão ainda em um estado embrionário e não oferecem uma solução para todos os problemas enfrentados durante o desenvolvimento de um software para robôs móveis. Foi neste contexto que se inseriu o trabalho apresentado, no qual foi proposta uma arquitetura de software flexível que forneça métodos de navegação e colaboração para o desenvolvimento de aplicações baseadas em robôs móveis ou para o auxílio na pesquisa.

A biblioteca de software desenvolvida, TAO, alcançou os objetivos do trabalho de pesquisa. Ela fornece classes que oferecem interfaces padronizadas para compor diferentes técnicas de navegação e colaboração de forma organizada. Além disso, ela fornece controladores de movimento para robôs móveis que podem ser usados com hardwares diferentes, contanto que sejam fornecidas classes de interface para o hardware do robô que estejam

dentro do padrão especificado pela TAO. Prova desta flexibilidade foi demonstrada pela implementação de diferentes interfaces de hardware para os robôs da BER e para o Robotino®.

Os controladores fornecidos pela TAO operam de forma adequada e obtiveram resultados dentro do esperado, conforme demonstram os experimentos apresentados nos itens 6.2.1 e 6.2.2. Estes controladores funcionam com qualquer robô móvel e sistema de localização que possuam interfaces concordantes com a TAO e podem ser utilizados tanto para desenvolver aplicações, quanto para avaliar o desempenho de outros controladores.

As interfaces para hardware dos robôs e a interface para sistemas de localização alcançaram seu objetivo de fornecer mecanismos que permitam o reuso de código e o fácil intercâmbio de módulos de software, conforme apresenta a seção 5.4. O mesmo ocorre para as classes de representação de mapas e de busca de melhor caminho, que obtiveram resultados satisfatórios, como mostra a seção 6.2.3.

Já a classe de comunicação fornecida pelo pacote de colaboração permite a troca efetiva de mensagens entre robôs através de uma rede de computadores. Esta troca de mensagens explícita confere aos robôs a capacidade de coordenar suas tarefas de forma conjunta para alcançar um objetivo em comum. O experimento mostrado pela seção 6.2.4 demonstra o uso desta classe.

Desta discussão, podemos concluir que a TAO é uma plataforma de software que facilita o desenvolvimento do aplicativo de controle de robôs móveis através da padronização dos componentes de software para controlar robôs diferentes e da abstração do hardware do robô e das técnicas de navegação e colaboração. Ela implementa algumas técnicas de navegação e colaboração que podem ser usadas tanto para fins de aplicação quanto de pesquisa. Além disso, novas técnicas de navegação e colaboração podem ser escritas dentro do padrão proposto pela TAO, o que permite a ampliação do arcabouço de software da plataforma sem comprometer sua flexibilidade e modularidade.

É claro que a TAO se encontra em um estágio inicial e fornece apenas uma pequena parcela das técnicas de navegação e colaboração. Entretanto, ela é proposta como uma pro-

va de que é possível alcançar a flexibilidade desejada para projetos de mundo real, através da padronização dos componentes de software que compõe um robô móvel, justificando, portanto, os esforços na busca por uma plataforma padrão de software modular e flexível para a robótica móvel.

O código fonte da plataforma TAO e dos experimentos, bem como os vídeos gravados dos experimentos e o manual do usuário, podem ser encontrados no site http://www2.fc.unesp.br/gisdi/TAO/.

7.1 Trabalhos Futuros

Diferentes aspectos da arquitetura podem ser melhorados em trabalhos futuros. As técnicas de localização e os controladores podem ter seus resultados melhorados através do uso da aplicação de filtros de Kalman ou do método de Monte Carlo. O algoritmo A^* pode ser melhorado para encontrar caminhos que considerem o movimento na diagonal, ou outros algoritmos de busca, como o D^*, podem ser implementados. Naturalmente, uma das propostas para trabalho futuro é expandir o arcabouço de software da TAO com outras técnicas de navegação e colaboração.

A biblioteca TAO pode ser substituída por uma arquitetura baseada em Redes de Computadores, como a UPnP ou a CORBA. Este modelo arquitetural permite que programas escritos em linguagens diferentes – ou ainda sendo executados em máquinas diferentes -, consigam intercambiar seus recursos de software como se estivessem sendo executados na mesma máquina. Este tipo de arquitetura tornaria ainda mais flexível a plataforma de software, entretanto acarretaria também atrasos na execução do software, o que requer estudos mais profundos neste sentido.

Finalmente, para que uma arquitetura de software para robôs móveis seja completa, é necessário desenvolver uma taxonomia para robôs móveis que permita identificar padrões nas diferentes partes que compõe um robô móvel. Tal taxonomia deve ser desenvolvida em conjunto com outras instituições de pesquisa e, também, com representantes da indústria,

pois apenas com um esforço em conjunto é possível criar uma plataforma de software robusta e confiável. Esta é uma tarefa complexa, mas alguns esforços semelhantes são mostrados pelo grupo promotor da UPnP. Neste sentido, propõe-se um estudo arquitetural em conjunto com outras instituições de pesquisa para melhorar a robustez da arquitetura proposta.

REFERÊNCIAS BIBLIOGRÁFICAS

AHN, S. C.; KIM, J. H.; LIM, K. *et al.* **UPnP Approach for Robot Middleware**. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. **Anais**... [S.l: s.n.]. , 2005

ALAMI, R.; CHATILA, R.; FLEURY, S.; GHALLAB, M.; INGRAND, F. An Architecture for Autonomy. **The International Journal of Robotics Research**, v. 17, n. 4, p. 315 -337, 1 abr 1998.

ALBUS, J. S. Outline for a theory of intelligence. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 21, n. 3, p. 473–509, 1991.

ARAI, T.; PAGELLO, E.; PARKER, L. E. Editorial: Advances in multi-robot systems. **IEEE Transactions on Robotics and Automation**, v. 18, n. 5, p. 655–661, 2002.

ASAMA, H.; MATSUMOTO, A.; ISHIDA, Y. **Design of an autonomous and distributed robot system: ACTRESS**. IEEE/RSJ IROS. **Anais**... [S.l: s.n.]. , 1989

BATLLE, J. A.; BARJAU, A. Holonomy in mobile robots. **Robotics and Autonomous Systems**, v. 57, n. 4, p. 433-440, 30 abr 2009.

BEKEY, G. A. **Autonomous robots: from biological inspiration to implementation and control**. [S.l.]: MIT Press, 2005.

BITENCOURT, A. C. P.; FRANCO, A. DA C. E S.; SOUZA, M. E. DE; FONTES, C. H. DE O.; COSTA, A. C. P. L. DA. Internal Model Control for Trajectory Tracking of an Omni-Directional Robot. **ABCM Symposium Series in Mechatronics**, v. 3, p. 363-372, 2008.

BORENSTEIN, J.; EVERETT, H. R.; FENG, L. **Where am I? Sensors and Techniques for Mobile Robot Positioning**. [S.l.]: AK Peters, Ltd., Wellesley, MA), 1st ed., Ch. 2, pp. 28-35 and pp. 71-72, 1995.

BORTHOLIN, R. C.; KANETO, P. H. U.; FERASOLI FILHO, H.; PEGORARO, R.; CALDEIRA, M. A. C. **Carrossel caipira - um time de futebol de robôs de baixo custo**. Anais do XXVI Congresso da SBC - Enri III Encontro de Robótica Inteligente. **Anais**... [S.l: s.n.]. , 2006

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer vision with the OpenCV library**. [S.l.]: O'Reilly Media, 2008.

BROOKS, R. **Visual map making for a mobile robot**. Robotics and Automation. Proceedings. 1985 IEEE International Conference on. **Anais**... [S.l: s.n.]. , 1985

BROOKS, R. A. **A robust layered control system for a mobile robot**. . [S.l.]: Massachusetts Institute of Technology. , 1985

CAO, Y. U.; FUKUNAGA, A. S.; KAHNG, A. B. Cooperative Mobile Robotics: Antecedents and Directions. **Autonomous Robots**, v. 4, p. 1–23, 1997.

CHAN, S.; CONTI, D.; KALER, C. *et al.* **Devices Profile for Web Services**. . [S.l: s.n.]. . Acesso em: 9 set. 2010. , fev 2006

CHATILA, RAJA. Deliberation and reactivity in autonomous mobile robots. **Robotics and Autonomous Systems**, v. 16, n. 2-4, p. 197-211, dez 1995.

COELHO, P.; NUNES, U. Lie Algebra Application to Mobile Robot Control: A Tutorial. **Robotica**, v. 21, n. 05, p. 483-493, 2003.

DELLAERT, F.; FOX, D.; BURGARD, W.; THRUN, S. **Monte Carlo Localization for Mobile Robots**. IEEE International Conference on Robotics and Automation (ICRA99). **Anais**... [S.l: s.n.]. , maio 1999

DUDEK, G.; JENKIN, M. **Computational Principles of Mobile Robotics**. [S.l.]: Cambridge University Press, 2010.

EVERETT, H. R. **Sensors for mobile robots: theory and application**. [S.l.]: A K Peters, Ltd., 1995.

FERASOLI FILHO, H.; PEGORARO, R.; CALDEIRA, M. A. C.; ROSÁRIO, J. M. AEDROMO- An Experimental and Didactic Environment with Mobile Robots. **Proceedings of The 3rd International Conference on Autonomous Robots and Agents**, 2006.

FERGUSON, I. Models and Behaviours: A Way Forward for Robotics. 1994.

FLOREANO, D.; MATTIUSSI, C. **Bio-inspired artificial intelligence: theories, methods, and technologies**. [S.l.]: MIT Press, 2008.

FRANZ, M. O.; MALLOT, H. A. Biomimetic robot navigation. **Robotics and autonomous Systems**, v. 30, n. 1-2, p. 133–154, 2000.

FRANZ, M. O.; STURZL, W.; HUBNER, W.; MALLOT, H. **A robot system for biomimetic navigation-from snapshots to metric embeddings of view graphs**. Proc Workshop on Cognitive and Robotics Approaches to Spatial Mapping. **Anais**... [S.l: s.n.]. , 2003

HENNING, M. A new approach to object-oriented middleware. **Internet Computing, IEEE**, v. 8, n. 1, p. 66-75, 2004.

HENNING, MICHI. The Rise and Fall of CORBA. **Queue**, v. 4, p. 28–34, jun 2006.

HETTIARACHCHI, S. D. Distributed evolution for swarm robotics. 2007.

HOLLAND, J. M. **Designing Autonomous Mobile Robots: Inside the Mind of an Intelligent Machine**. [S.l.]: Newnes, 2003.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 14000 – Environmental management**. Text. Disponível em: <http://www.iso.org/iso/iso_catalogue/management_and_leadership_standards/environmental_management/iso_14000_essentials.htm>. Acesso em: 1 abr. 2011.

KUROSE, J. F.; ROSS, K. W. **Computer networking: a top-down approach featuring the Internet**. [S.l.]: Pearson/Addison Wesley, 2005.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. **Robotics and Automation, IEEE Transactions on**, v. 7, n. 3, p. 376–382, 1991.

LESTER, P. **A* Pathfinding for Beginners**. Disponível em: <http://www.policyalmanac.org/games/aStarTutorial.htm>. Acesso em: 29 jun. 2011.

LINUX KERNEL ORGANIZATION. **The Linux Kernel Archives**. Disponível em: <http://www.kernel.org/>. Acesso em: 1 jan. 2012.

LIU, Y.; WU, X.; JIM ZHU, J.; JAE LEW. **Omni-directional mobile robot controller design by trajectory linearization**. American Control Conference, 2003. Proceedings of the 2003. **Anais**... [S.l: s.n.]. , 2003

MAES, P. Modeling Adaptive Autonomous Agents. **ARTIFICIAL LIFE**, v. 1, p. 135--162, 1994.

MARTINS, N. A. **Controle Adaptativo e Robusto de Robôs Móveis com Rodas**. Florianópolis: Universidade Federal de Santa Catarina, 2010.

MATARIĆ, M. J. Behavior-Based Control: Main Properties and Implications. **IN PROCEEDINGS, IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, WORKSHOP ON ARCHITECTURES FOR INTELLIGENT CONTROL SYSTEMS**, p. 46--54, 1992.

MATARIĆ, M. J. Issues and approaches in the design of collective autonomous agents. **Robotics and Autonomous Systems**, v. 16, n. 2, p. 321–332, 1995.

MATARIĆ, M. J. **The robotics primer**. [S.l.]: MIT Press, 2007.

MCCALL, J. A. Quality Factors. **Encyclopedia of Software Engineering**. [S.l.]: John Wiley & Sons, Inc., 2002. .

MIRTICH, B. V-Clip: Fast and robust polyhedral collision detection. **ACM Transactions on Graphics (TOG)**, v. 17, n. 3, p. 177–208, 1998.

MOK, S. M.; WU, C.-HAUR. **Automation integration with UPnP modules**. Electronic Design, Test and Applications, 2006. DELTA 2006. Third IEEE International Workshop on. **Anais**... [S.l: s.n.]. , 19 2006

MURPHY, R. **Introduction to AI robotics**. [S.l.]: MIT Press, 2000.

NAIN, G.; DAUBERT, E.; BARAIS, O.; EQUEL, J.-M. J. EZ´. Using MDE to Build a Schizophrenic Middleware for Home/Building Automation. **Towards a Service-Based Internet**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. v. 5377p. 49-61.

NEHMZOW, U.; OWEN, C. Robot Navigation in the Real World: Experiments with Manchester's Forty Two in Unmodified Large Environments. **Robotics and Autonomous systems**, v. 33, n. 4, p. 223–242, 2000.

NESNAS, I. A.; WRIGHT, A.; BAJRACHARYA, M. *et al.* **Claraty: An architecture for reusable robotic software**. SPIE Aerosense Conference. **Anais**... [S.l: s.n.]. , 2003

NOUYAN, S.; DORIGO, M. Chain based path formation in swarms of robots. **Lecture Notes in Computer Science**, v. 4150, p. 120–131, 2006.

PARKER, LYNNE E. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. **IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION**, v. 14, p. 220-240, 1998.

PARKER, LYNNE E. Lifelong Adaptation in Heterogeneous Multi-Robot Teams: Response to Continual Variation in Individual Robot Performance. **Autonomous Robots**, v. 8, n. 3, p. 239-267, 2000.

PRATA, S. **C++ Primer Plus**. 5. ed. Indianapolis, Indiana: Sams Publishing, 2004.

QUIGLEY, M.; GERKEY, B.; CONLEY, K. *et al.* **ROS: an open-source Robot Operating System**. ICRA Workshop on Open Source Software. **Anais**... [S.l: s.n.]. , 2009

QUINN, M.; SMITH, L.; MAYLEY, G.; HUSBANDS, P. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. **Philosophical Transactions: Mathematical, Physical and Engineering Sciences**, p. 2321–2343, 2003.

ROTKIEWICZ, M. **A-star Algorithm**. Disponível em: <http://michalhr.ehost.pl/blog/?cat=7&paged=5>. Acesso em: 29 jun. 2011.

SCHMICKL, T.; CRAILSHEIM, K. Trophallaxis among swarm-robots: A biologically inspired strategy for swarm robotics. **Proceedings of BioRob**, p. 1–4244, 2006.

SICILIANO, B.; KHATIB, O. (EDS.). **Springer Handbook of Robotics**. Heidelberg: Springer, 2008.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to autonomous mobile robots**. [S.l.]: MIT Press, 2004.

SIMMONS, R.; LIN, L.-J.; FEDOR, C. **Autonomous task control for mobile robots**. Intelligent Control, 1990. Proceedings., 5th IEEE International Symposium on. **Anais**... [S.l: s.n.]. , 1990

STEELS, L. When are robots intelligent autonomous agents? **Robotics and Autonomous Systems**, v. 15, n. 1-2, p. 3-9, jul 1995.

UTZ, H.; SABLATNOG, S.; ENDERLE, S.; KRAETZSCHMAR, G. Miro - middleware for mobile robot applications. **Robotics and Automation, IEEE Transactions on**, v. 18, n. 4, p. 493-497, 2002.

VARELA, F. J. **Toward a practice of autonomous systems: proceedings of the First European Conference on Artificial Life**. [S.l.]: MIT Press, 1994.

WATANABE, K. **Control of an omnidirectional mobile robot**. Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on. **Anais**... [S.l: s.n.]. , 1998

YAMAMOTO, Y.; YUN, X. Coordinating locomotion and manipulation of a mobile manipulator. **Automatic Control, IEEE Transactions on**, v. 39, n. 6, p. 1326-1332, 1994.

APÊNDICE A – ATIVIDADES DESENVOLVIDAS

Capítulos de Livro

1. ALVES, S. F. R. ; ROSÁRIO, J. M. ; FERASOLI FILHO, H. ; RINCON, Liz K. A. ; YAMASAKI, Rosana A. T. . Conceptual Bases of Robot Navigation Modeling, Control and Applications. In: Alejandra Barrera. (Org.). Advances in Robot Navigation. 1 ed. Rijeka: InTech - Open Access Publisher, 2011, v. , p. 3-28.

2. ALVES, S. F. R. ; FERASOLI FILHO, H. ; PEGORARO, R. ; CALDEIRA, M. A. C. ; ROSÁRIO, J. M. ; YONEZAWA, W. M. . Educational Environment for Robotic Applications in Engineering. In: David Obdr álek, Achim Gottscheber. (Org.). Research and Education in Robotics - EUROBOT 2011. 1 ed. Berlin: Springer-Verlag, 2011, v. 161, p. 17-28.

Artigos em Anais de Evento

1. FERASOLI FILHO, Humberto; CALDEIRA, Marco A. C.; PEGORARO, Rene; ALVES, Silas F. R.; VALADÃO, Carlos; BASTOS-FILHO, Teodiano F.. Use of Myoelectric Signals to Command Mobile Entertainment Robot by Disabled Children: Design and Control Architecture. ISSNIP Biosignals and Biorobotics Conference 2012 - ISSNIP Biosignals and Biorobotics Workshop 2012, 2012, Manaus. Aceito para publicação.

2. ONO, J. H. P. ; CALDEIRA, M. A. C. ; Pegoraro, R. ; FERASOLI FILHO, H. ; ALVES, S. F. R. . Ambiente Gráfico de Simulação Robótica para fins Educacionais. 2011.

3. ALVES, S. F. R. ; FERASOLI FILHO, H. ; PEGORARO, R. ; CALDEIRA, M. A. C. ; ROSÁRIO, J. M. ; YONEZAWA, W. M.. Educational Environment for Robotic Applications in Engineering. In: Eurobot Conference 2011 - 4th International Confer-

ence on Research and Education in Robotics, 2011, Praga - República Tcheca. Research and Education in Robotics - EUROBOT 2011. Berlin - Alemanha : Springer, 2011. v. 161. p. 17-28.

4. ALVES, S. F. R. ; FERASOLI FILHO, H. ; PEGORARO, R. ; CALDEIRA, M. A. C. ; ROSÁRIO, J. M. ; YONEZAWA, W. M.. Proposal of Educational Environments with Mobile Robots. In: 5th IEEE International Conference on Robotics, Automation and Mechatronics, 2011, Qingdao, China. Proceedings of the 5th IEEE International Conference on Robotics, Automation and Mechatronics, 2011. p. 264-269.

5. ALVES, S. F. R. ; FERASOLI FILHO, H. ; PEGORARO, R. ; CALDEIRA, M. A. C. ; YONEZAWA, W. M. ; ROSÁRIO, J. M. Ambiente Educacional de Robótica Direcionado a Aplicações em Engenharia. In: X Simpósio Brasileiro de Automação Inteligente, 2011, São João del-Rei. Anais do X Simpósio Brasileiro de Automação Inteligente, 2011.

6. URIBE, Alvaro ; ALVES, S. F. R. ; ROSÁRIO, J. M. ; FERASOLI FILHO, H. ; PÉREZ-GUTIÉRREZ, Byron . Mobile Robotic Teleoperation using Gesture-Based Human Interfaces. In: Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC), 2011, Bogota. Proceedings of Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC), 2011. p. 1-6.

7. ALVES, S. F. R. ; ROSÁRIO, J. M. ; FERASOLI FILHO, H. ; PEGORARO, R. . Environment for Teaching and Development of Mobile Robot Systems. In: Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010, 2010, Cuernavaca. Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010, 2010. p. 302-307.

8. ALVES, S. F. R. ; FERASOLI FILHO, H. ; PEGORARO, R. ; ROSÁRIO, J. M. . Ambiente de Ensino e Desenvolvimento de Sistemas Robóticos Móveis. In: XVIII Con-

gresso Brasileiro de Automática, 2010, Bonito. Anais do XVIII Congresso Brasileiro de Automática, 2010. p. 675-681.

9. ALVES, S. F. R. ; FERASOLI FILHO, H. ; YONEZAWA, W. M. ; ROSÁRIO, J. M. . Nova Abordagem para o Ensino de Computação: A Bancada Experimental Robótica. In: ROBOCONTROL 2010 - 4º Workshop de Robótica Aplicada e Automação, 2010, Bauru. Anais do Robocontrol 2010, 2010.

Organização de Evento

1. PORTO, Arthur Vieira; CORDERO, Arturo Forner; ROSÁRIO, João Maurício; SIVLA, Jose Rei-naldo; GUILHERME, Ivan Rizzo; RILLO, Márcio; CALDEIRA, Marco A. Corbucci; PEGORARO, Renê; ALVES, Silas Franco dos Reis; YONEZAWA, Wilson Massashiro. Robocontrol'10 - 4th Applied Robotics and Automation. 6-7 de maio de 2010.

2. PORTO, Arthur Vieira; FORNER, Arturo Cordero; FERASOLI FILHO, Humberto; GUILHERME, Ivan Rizzo; ROSÁRIO, João Mauricio Rosário; CASTANHO, José E. C.; SILVA, Jose Reinaldo; FRANCHIN, Marcelo N.; CALDEIRA, Marco A. C. Caldeira; PEGORARO, Renê; ALVES, Silas F. R.; YONEZAWA, Wilson M. Robocontrol 2012 - 5th Applied Robotics and Automation. 15-16 de junho de 2012.

Palestras Ministradas

1. ALVES, Silas; FERASOLI FILHO, Humberto; ROSÁRIO, João M. Robotino – Interface de programação em C++ e Java. In: ROBOCONTROL 2010 - 4º Workshop de Robótica Aplicada e Automação, 2010, Bauru.

2. ALVES, Silas; FERASOLI FILHO, Humberto; ROSÁRIO, João M. Robotino – RobotinoView, C++ e Java. In: XIII Semana da Engenharia Elétrica, 2010, Bauru.

Participação em Projetos de Pesquisa e Extensão

1. Ciência e Artes nas Férias – Aprendendo a Conhecer a Mecatrônica, UNICAMP, 2010
2. Oficina de Robótica, UNESP, 2011
3. Construção dos robôs 14-bis e Roburguer, UNESP/UFES, 2011

Cursos de Formação Complementar

1 Introduction to High Performance Computing

   a. Objetivo: Estudar bibliotecas de software voltadas para a programação em computadores paralelos de alto desempenho
   b. Professor: Dr. Leopold Grinberg, Divisão de Matemática Aplicada da Brown University
   c. Duração: 29/11/2010 a 03/12/2010
   d. Local: CENAPAD-SP

APÊNDICE B – MODELAGEM E CONTROLE DINÂMICO

O Acionamento dos Motores permite o desacoplamento do modelo cinemático do dinâmico. Para isto, o controle de velocidade do motor deve levar em conta os aspectos dinâmicos da estrutura do robô móvel e do próprio motor. Neste sentido, o controlador de velocidade pode valer-se dos modelos dinâmicos do robô móvel para implementar o método de *torque computado* (SICILIANO; KHATIB, 2008) ou *aceleração resolvida* (WATANABE, 1998) para assimilar a dinâmica do robô.

Neste apêndice serão apresentados os modelos dinâmicos para os robôs móveis holonômico e não-holonômico, e o controlador de velocidade do motor elétrico.

Robô Diferencial

A modelagem dinâmica adotada desconsidera as incertezas do modelo e as massas e momentos de inércia dos motores e das rodas. O modelo foi obtido através do formalismo de Euler-Lagrange (COELHO; NUNES, 2003; MARTINS, 2010; YAMAMOTO; YUN, 1994) para o caso do robô da Figura 4.15 (a) e é dado por:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} = -A^T(q)\rho + B(q)\tau \qquad (0.1)$$

onde $H(q) \in \Re^{n\times n}$ é a matriz de inércia simétrica definida positiva, $C(q,\dot{q}) \in \Re^{n\times n}$, é a matriz de torques de *Coriolis* e centrípetos, $A(q) \in \Re^{p\times n}$ é a matriz associada às restrições não-holonômicas, ρ é o vetor de forças de restrição, $B(q) \in \Re^{n\times m}$ é a matriz de transformação das entradas de controle, e $\tau \in \Re^{p}$ o vetor de torque.

$$
\begin{aligned}
H(q) &= \begin{bmatrix} m_c & 0 & m_c d\,\mathrm{sen}(\theta) \\ 0 & m_c & -m_c d\cos(\theta) \\ m_c d\,\mathrm{sen}(\theta) & -m_c d\cos(\theta) & I_c \end{bmatrix} \\
C(q,\dot{q})\dot{q} &= \begin{bmatrix} m_c d\dot{\theta}^2\cos(\theta) \\ m_c d\dot{\theta}^2\,\mathrm{sen}(\theta) \\ 0 \end{bmatrix} \\
A(q) &= [-\sin(\theta) \quad \cos(\theta) \quad 0] \\
B(q) &= \begin{bmatrix} \dfrac{\cos(\theta)}{r} & \dfrac{\cos(\theta)}{r} \\ \dfrac{\mathrm{sen}(\theta)}{r} & \dfrac{\mathrm{sen}(\theta)}{r} \\ \dfrac{R}{r} & -\dfrac{R}{r} \end{bmatrix} \\
\tau &= \begin{bmatrix} \tau_d \\ \tau_e \end{bmatrix} \\
q &= [x_c \quad y_c \quad \theta]
\end{aligned}
\tag{0.2}
$$

A equação (0.1) pode ser transformada para se adequar aos propósitos de controle (COELHO; NUNES, 2003), resultando na equação(0.3), onde $S(q)$ é a matriz jacobiana ou o modelo cinemático.

$$
S^T(q)\big(H(q)S(q)\dot{v}(t) + H(q)\dot{S}(q)v(t) + C(q,\dot{q})\dot{q}\big) = S^T(q)B(q)\tau \tag{0.3}
$$

Robô Omnidirecional

A modelagem desconsidera as incertezas do modelo e as massas e momentos de inércia dos motores e das rodas. O modelo foi obtido através do formalismo de Euler-Newton (BITENCOURT *et al.*, 2008; LIU *et al.*, 2003; WATANABE, 1998) para o caso do robô da Figura 4.15 (b) e é dado por:

$$
\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\text{sen}(\theta) \\ \text{sen}(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \end{bmatrix}
$$

$$
\dot{x}_l = \frac{1}{\dot{\theta}} \left(\ddot{y}_l - \sqrt[3]{3} b_1 (\tau_1 - \tau_2) \right)
$$

$$
\dot{y}_l = \frac{1}{\dot{\theta}} \left(\ddot{x}_l + b_1 (\tau_1 + \tau_2 - 2\tau_3) \right) \tag{0.4}
$$

$$
b_1 = \frac{k_\text{t}}{2Mr}
$$

onde x_l e y_l são as coordenadas do robô no plano $\mathcal{F}_l$; $\dot{x}_l$ e $\dot{y}_l$ são suas velocidades nos eixos X_l e Y_l, respectivamente; e $\ddot{x}_l$ e $\ddot{y}$ são suas acelerações, também nos eixos X_l e Y_l, respectivamente.

A equação (0.4) pode ser transformada para se adequar aos propósitos de controle (WATANABE, 1998), resultando na equação (0.5), com q correspondendo às variáveis de estado, tal que $q = [x_c, y_c, \theta, \dot{x}_c, \dot{y}_c]^\text{T}$, e q' às variáveis de estado de postura, tal que $q' = [\dot{x}_w, \dot{y}_w, \theta]^\text{T}$.

$$
\begin{aligned} q &= D(q)q + E(q)\tau \\ q' &= Fq \end{aligned} \tag{0.5}
$$

onde

$$
\begin{aligned}
D(q) &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
E(q) &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ b_1\beta_1 & b_1\beta_2 & 2b_1\cos(\theta) \\ b_1\beta_3 & b_1\beta_4 & 2b_1\sin(\theta) \\ b_2 & b_2 & b_2 \end{bmatrix} \\
F &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\
\tau &= \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \\
b_2 &= \frac{k_t L}{I_c r}
\end{aligned}
\tag{0.6}
$$

$$
\beta_1 = -\sqrt[3]{3}\,\text{sen}(\theta) - \cos(\theta), \quad \beta_2 = \sqrt[3]{3}\,\text{sen}(\theta) - \cos(\theta),
$$
$$
\beta_3 = \sqrt[3]{3}\cos(\theta) - \text{sen}(\theta), \quad \beta_4 = -\sqrt[3]{3}\cos(\theta) - \text{sen}(\theta)
$$

sabendo que k_t é o fator de ganho de tração.

Controle de Velocidade do Motor

O controle de velocidade do motor leva em conta os esforços e atritos oriundos da caixa de redução, do momento de inércia do robô e da interação entre a roda e o solo. É evidente a necessidade do estudo do modelo dinâmico do motor (LIU *et al.*, 2003), que é dado por:

$$L_a \frac{di_a}{dt} + R_a i_a + k_e \dot{\varphi}_{\mathrm{m}} = u \tag{0.7}$$

$$J_0 \dot{\omega}_m + b_0 \omega_m + \frac{Rf}{k_n} = K_\tau i_a \tag{0.8}$$

onde u é a tensão de armadura aplicada, i_a é a corrente de armadura, L_a é a impedância da armadura, R_a é a resistência da armadura k_e é a constante de fem (força eletromotriz), k_τ é a constante de torque do motor, J_0 é o momento combinado de inércia do motor, redução e roda em referência ao eixo do motor, b_0 é o coeficiente de atrito viscoso da combinação entre o motor, a redução e as rodas, e k_n é a taxa de redução e $\dot{\varphi}_{\mathrm{m}}$ é a velocidade angular do eixo do motor.

No modelo adotado, a dinâmica do circuito elétrico do motor é negligenciada, pois a constante de tempo do motor é muito pequena em comparação à constante de tempo mecânica (LIU *et al.*, 2003), o que nos leva a:

$$\frac{di_a}{dt} = 0, \qquad i_a = \frac{1}{R_a}(u - k_e \dot{\varphi}_{\mathrm{m}}) \tag{0.9}$$

Logo, substituindo as considerações da equação (0.9) em (0.8), obtém-se a equação dinâmica do motor dada pela equação (0.10):

$$J_0 \ddot{\varphi}_{\mathrm{m}} + \frac{R}{k_n} f = \frac{k_\tau}{R_a} u - \frac{k_\tau k_e}{R_a} \dot{\varphi}_{\mathrm{m}} \tag{0.10}$$

A equação (0.10) pode ser reescrita através da transformada de Laplace em termos de s, conforme mostra a Figura 0.1.

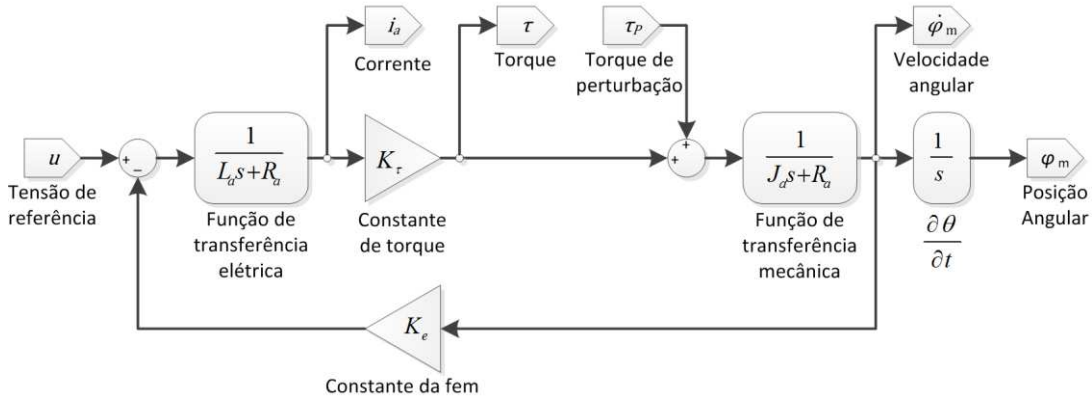


Figura 0.1 – Diagrama da Equação do Motor.

Através da função de transferência apresentada pela Figura 0.1, obtém-se o controlador PID da equação (0.11), onde k_p é o ganho proporcional, k_i é o ganho integrativo e k_d é o ganho derivativo.

$$k_p + \frac{k_i}{s} + k_d s = \frac{k_d s^2 + k_p s + k_i}{s} \tag{0.11}$$

Este controlador é acoplado ao modelo do motor, conforme exemplifica a Figura 0.2.

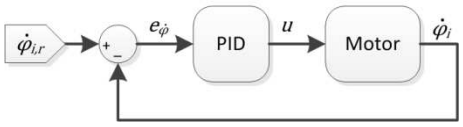


Figura 0.2 – Controlador de Velocidade do Motor.

Para demonstrar o controlador PID, foram realizados alguns ensaios através do MATLAB® com diferentes valores para suas constantes. Nestes ensaios, foram utilizados os valores $J_0 = 0{,}01$ kg m^2/s^2, $k_e = k_\tau = 0{,}01$ Nm/Amp, $R_a = 1$ ohm e $L_a = 0{,}5$ H. Os resultados são mostrados pela Figura 0.3, onde: (a) demonstra o comportamento do motor quando apenas o ganho proporcional é aplicado ($k_p = 100$); (b) demonstra-o quando os três ganhos são colocados sem ajuste ($k_p = 200$, $k_i = 200$, e $k_d = 10$); e, finalmente, (c) demonstra-o quando os três ganhos são corretamente calibrados ($k_p = 120$, $k_i = 230$, e $k_d = 10$). Pode-se ver nestes gráficos que o controlador com os ganhos corretamente calibrado convergiu mais rapidamente à velocidade desejada sem a presença de *overshoot*.

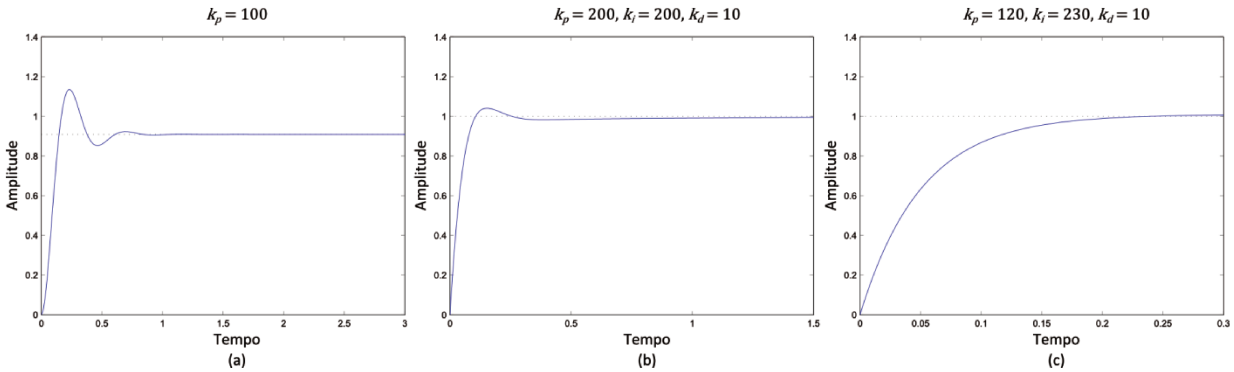


Figura 0.3 – Resultados dos ensaios do controlador do motor DC onde (a) é o ensaio para o controlador proporcional, (b) para o controlador PID desajustado e (c) para o PID ajustado.