DANIEL IGOR MENDOZA QUIÑONES

CONTROLE DE ROBÔ MÓVEL AUTÔNOMO PARA COLETAR LIXO

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para a obtenção do título de Mestre em
Engenharia

São Paulo
2007

DANIEL IGOR MENDOZA QUIÑONES

CONTROLE DE ROBÔ MÓVEL AUTÔNOMO PARA COLETAR LIXO

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para a obtenção do título de Mestre em
Engenharia

Área de Concentração:
Engenharia de Controle e Automação
Mecânica

Orientador:
Prof. Dr. Eduardo Lobo Lustosa Cabral

São Paulo
2007

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de outubro de 2007.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Mendoza Quiñones, Daniel Igor

**Controle de robô móvel autônomo para coletar lixo / D.I.
Mendoza Quiñones. -- ed.rev. -- São Paulo, 2007.
p.**

**Dissertação (Mestrado) - Escola Politécnica da Universidade
de São Paulo. Departamento de Engenharia Mecatrônica e de
Sistemas Mecânicos.**

**1. Robôs 2. Robótica I. Universidade de São Paulo. Escola
Politécnica. Departamento de Engenharia Mecatrônica e de
Sistemas Mecânicos II. t.**

**A meus pais Manuel e Detti
meus irmãos
e meu Amore**

AGRADECIMENTOS

Ao professor Eduardo Lobo Cabral por sua orientação e apoio

Aos meus amigos: Rocio, José, Demóstenes e Coco Max, por seu apoio incondicional.

RESUMO

Este trabalho apresenta o desenvolvimento dos algoritmos de controle de um robô móvel autônomo para coleta de lixo. O objetivo do robô é coletar latas de refrigerante espalhadas pelo chão. O sistema de navegação do robô foi implementado utilizando a arquitetura denominada “Motor-Schema”. Essa arquitetura fornece um método para projetar comportamentos primitivos que atuam em forma paralela para realizar ações robóticas inteligentes em resposta aos estímulos do ambiente. O sistema de controle apresentado foi constituído por vários comportamentos primitivos que, coordenados, permitiram ao robô explorar de forma segura um ambiente desconhecido, detectar e coletar o lixo e levá-lo num depósito determinado. Os algoritmos desenvolvidos foram testados utilizando uma ferramenta de simulação 2D denominada Player/Stage. Os resultados obtidos mostraram que a solução apresentada é adequada para resolver a aplicação robótica de coleta de lixo.

ABSTRACT

This work presents the control system for an autonomous mobile robot for soda can collection. The navigation system is implemented using a reactive architecture called “Motor-Schema”. This architecture provides a methodology to design primitive behaviors that can act in a distributed and parallel manner to yield intelligent robotic actions in response to environmental stimuli. The control system is composed of several primitive behaviors, which enable the robot explore an unknown environment, detect and collect the soda cans and navigate toward a soda can reservoir. The algorithms are tested using Player/Stage, a software for 2D simulation. The results show that the solution is suitable for soda can collection.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. Introdução.....	1
1.1 Introdução.....	1
1.2 Objetivo.....	2
1.3 Justificativas.....	3
1.4 Metodologia do Projeto.....	5
1.5 Organização da Dissertação.....	7
2. Fundamentos Teóricos.....	8
2.1 Controle de Robôs Móveis Autônomos.....	8
2.1.1 Estratégia Deliberativa.....	8
2.1.2 Estratégia Reativa.....	9
2.1.3 Estratégia Híbrida.....	10
2.1.4 Estratégia Baseado em Comportamento.....	11
2.2 “MOTOR-SCHEMA”.....	14
2.3 Ferramentas de Simulação Utilizadas.....	21
2.3.1 PLAYER/STAGE.....	21
2.3.2 Arquitetura PLAYER/STAGE.....	24
3. Desenvolvimento da aplicação robótica de coleta de lixo.....	27
3.1 Robô escolhido para realizar a tarefa de coleta de lixo.....	27
3.1.1 Modelo do Robô Pioneer 2AT.....	30
3.1.2 Interfaces de Controle.....	31
3.2 Comportamentos Desenvolvidos.....	36
3.3 Formalização dos Comportamentos Primitivos.....	38
3.3.1 Comportamentos AVANÇAR e RETROCEDER.....	39
3.3.2 Comportamento AVOID_OBSTACLE.....	40
3.3.3 Comportamento NOISE.....	42
3.3.4 Comportamento ATRAÇÃO_VERMELHO.....	42
3.3.5 Comportamento AVOID_RED_OBSTACLE.....	44
3.3.6 Comportamento MOVE_TO_GOAL.....	45

3.3.7 Comportamento TANGENCIAL_HORÁRIO e TANGENCIAL_ANTI-HORÁRIO.....	46
3.4 Testes dos Comportamentos Abstratos.....	49
3.4.1 Controle de Baixo Nível.....	49
3.4.2 Comportamento EXPLORAR PROCURANDO LIXO.....	52
3.4.3 Comportamento COLETAR LATA.....	55
3.4.4 Comportamento PEGAR LATA.....	58
3.4.5 Comportamento IR AO DEPÓSITO.....	61
3.4.6 Comportamento DEIXAR LATA.....	66
3.5 Seqüência de Controle dos Comportamentos Abstratos.....	69
3.6 Teste Final da Aplicação.....	73
4. Conclusão.....	81
5. Referências Bibliográficas.....	83
6. Anexos.....	87

LISTA DE FIGURAS

Figura 2.1 Decomposição de tarefas na arquitetura “Subsumption”	13
Figura 2.2 Campos potenciais básicos.....	18
Figura 2.3 Navegação de um robô desenvolvido seguindo a metodologia “Motor-Schema”	19
Figura 2.4 Diagrama de blocos da arquitetura “Motor-Schema”	21
Figura 2.5:Plataforma robótica Pioneer 2-DX.....	22
Figura 2.6 Visualização do ambiente de simulação do programa de simulação Stage.....	23
Figura 2.7 Visualização da simulação de um sistema multi-robô com o programa de simulação Stage.....	24
Figura 2.8 Arquitetura completa do Servidor Player.....	25
Figura 2.9: Interação cliente-servidor no Player/Stage.....	26
Figura 3.1. Robô Pioneer 2AT da companhia Mobile Robots.....	28
Figura 3.2. Modelo do robô Pioneer 2AT utilizado nesse trabalho.....	31
Figura 3.3. Modelo da comunicação do servidor Player com o simulador Stage e com plataformas robóticas reais.....	32
Figura 3.4. Posição do sistema de coordenadas odométrico.....	33
Figura 3.5. Distribuição dos anéis de sonares no modelo do robô Pioneer 2AT.....	34
Figura 3.6. Exemplo de utilização da interface Blobfinder com o Robô Pioneer.....	36
Figura 3.7. Esquema da posição dos sonares do anel frontal ao robô.....	40
Figura 3.8. Parâmetros utilizados no cálculo do vetor de repulsão.....	41
Figura 3.9. Parâmetros do comportamento ATRAÇÃO_VERMELHO.....	43
Figura 3.10. Parâmetros do comportamento AVOID_RED_OBSTACLE.....	44
Figura 3.11. Parâmetros do comportamento MOVE_TO_GOAL.....	46
Figura 3.12. Parâmetros dos comportamentos TANGENCIAL_HORÁRIO e TANGENCIAL_ANTI-HORÁRIO.....	48
Figura 3.13. Determinação das velocidades de rotação e translação para o controle de baixo nível.....	50

Figura 3.14. Comportamento do Robô dentro de um campo vetorial uniforme.....	51
Figura 3.15. Trajetória do Robô no primeiro teste do comportamento abstrato EXPLORAR.....	54
Figura 3.16. Trajetória do robô no segundo teste do comportamento abstrato EXPLORAR.....	55
Figura 3.17. Dados da posição do centróide fornecidos pelo simulador Stage.....	56
Figura 3.18 Ação do comportamento de COLETAR LATA.....	58
Figura 3.19. Robô controlado pelo comportamento PEGAR LATA.....	60
Figura 3.20. Campo vetorial do comportamento primitivo “Docking”.....	61
Figura 3.21. Campos vetoriais presentes no comportamento primitivo “Docking”.....	64
Figura 3.22. Robô controlado pelo comportamento IR AO DEPÓSITO.....	65
Figura 3.23. Autômato finito correspondente ao comportamento DEIXAR LATA.....	66
Figura 3.24. Resultado do teste do comportamento DEIXAR LATA.....	68
Figura 3.25. Sequência de controle da tarefa de coleta de lixo.....	69
Figura 3.26: Ação do estado de INICIO.....	70
Figura 3.27. Resultado do teste do esquema de INTENTOS.....	72
Figura 3.28. Cenário de simulação.....	74
Figura 3.29. Teste dos algoritmos de controle.....	77

LISTA DE TABELAS

Tabela 3.1. Variáveis correspondentes à interface Gripper.....	35
Tabela 3.2. Comportamentos abstratos e os comportamentos primitivos que os constituem.....	38
Tabela 3.3. Parâmetros determinados para o comportamento abstrato Explorar.....	53
Tabela 3.4. Parâmetros determinados para o comportamento abstrato COLETAR LATA.....	57
Tabela 3.5. Correspondência dos estados do manipulador e as ações efetuadas pelo comportamento PEGAR LATA.....	59
Tabela 3.6. Parâmetros determinados para o comportamento abstrato PEGAR LATA.....	60
Tabela 3.7. Parâmetros determinados para o comportamento abstrato IR AO DEPÓSITO.....	63
Tabela 3.8. Parâmetros determinados para o comportamento abstrato DEIXAR LATA.....	67
Tabela 3.9 Ativadores da seqüência de controle dos comportamentos abstratos.....	71
Tabela 3.10 Parâmetros dos comportamentos abstratos utilizados nas provas.....	75

1. Introdução

1.1 Introdução

Há cerca de 20 anos, uma nova abordagem para navegação e controle de robôs móveis autônomos começou a ser desenvolvida em diversos laboratórios (BROOKS, 1991c). Ao invés de perceber, modelar o ambiente, planejar a trajetória e executar uma ação, essa nova abordagem implementa sistemas de controle inteligentes através da ativação de um conjunto de módulos de controle de baixo nível denominados comportamentos. Um comportamento é uma regra de controle que tem como missão atingir e manter um objetivo. Ações inteligentes “emergem” da interação desses comportamentos e a complexidade do ambiente em que o robô trabalha. Essa nova abordagem é atualmente conhecida como arquitetura baseada em comportamento. Os robôs móveis desenvolvidos com essa arquitetura têm demonstrado serem eficientes em ambientes desconhecidos e dinâmicos. Essas características fazem com que esse tipo de arquitetura possa ser implementada com sucesso em diversas aplicações, desde navegação de robôs em ambientes dinâmicos, até em robôs de exploração planetária, como a sonda de exploração de Marte.

O tema desta dissertação é o desenvolvimento de algoritmos de controle para robôs móveis usando uma arquitetura baseada em comportamentos.

1.2 Objetivo

O presente trabalho teve como objetivo desenvolver o controle de um robô móvel autônomo com arquitetura comportamental reativa para realizar uma tarefa de coleta lixo. O lixo em questão foram latas de refrigerante espalhadas pelo chão de um ambiente com obstáculos. Nessa tarefa o robô teve que procurar as latas pelo ambiente, encontrá-las e levá-las a um depósito em local pré-determinado.

Os algoritmos de controle foram desenvolvidos e testados com o simulador para robôs moveis Player/Stage.

Para alcançar o objetivo proposto utilizou-se uma arquitetura de controle denominada “Motor-Schema”. Motor-Schema é uma arquitetura comportamental reativa desenvolvida por Ronald C. Arkin (ARKIN 1998) que utiliza uma variação do método de campos potenciais (KROGH, 1984) para determinar a ação dos comportamentos. Essa arquitetura apresenta modularidade, robustez e flexibilidade, e tem sido muito utilizada em diversas aplicações robóticas reais.

Os comportamentos reativos desenvolvidos neste trabalho permitem que um robô móvel realize as seguintes tarefas:

- **Navegação segura:** o robô navega pelo ambiente evitando obstáculos que estejam no caminho;
- **Exploração:** o robô procura por latas em todo o ambiente e não fica limitado a uma área específica do ambiente.
- **Deteção de objetos:** o robô detecta as latas de refrigerantes espalhadas pelo ambiente;
- **Manipulação de objetos:** o robô segura a lata dentro do manipulador
- **Atingir um objetivo:** o robô leva a lata a um depósito em local pré-determinado.

1.3. Justificativas

Um dos primeiros e mais populares projetos desenvolvido por pesquisadores de robótica móvel foi coletar latas de refrigerante (MURPHY, 2000). Coletar latas de refrigerante é um ciclo de tarefas que consiste em várias sub-tarefas. Essas sub-tarefas obrigam ao projetista procurar soluções adequadas para seus sistemas robóticos nos quatro temas básicos da pesquisa em robótica móvel: reconhecimento, manipulação, evitar obstáculos e navegação (FLYNN e BROOKS, 1988). Em razão disso, essa tarefa e outras similares são escolhidas como desafio nos eventos de competição e exibição de Robôs (“Robot Competition & Exhibition”) organizados pela AAAI (“American Association of Artificial Intelligence”) entre os anos de 1993 e 1998.

Em 1993 o tema da competição foi “Encontrar e Entregar” (“Find and Deliver”). Os robôs participantes deviam navegar em um ambiente de escritório para encontrar um determinado objeto e trazê-lo de volta (AAAI, 1993).

Nos anos 1994 e 1995 o tema foi “Limpar o Escritório” (“Clean Up the Office”). Nesse evento os robôs ficavam em um quarto fechado e tinham que procurar, no chão, copos de café de isopor, latas de refrigerante e pedaços de papel amassados para depois levá-los a depósitos colocados nos cantos do quarto. Existiam depósitos de lixo específicos para cada objeto. Os depósitos de lixo eram de metal e circulares, com 0,3 m. de altura e 0,3 m. de diâmetro. O ganhador era o robô que colocava mais objetos nos depósitos de lixo correspondentes em um tempo máximo de 20 min. Existiam penalidades por modificar o ambiente ou colidir com obstáculos (AAAI, 1994; HINKLE et al., 1995; e MURPHY, 2000).

Já em 1996 o objetivo variou um pouco. O evento teve como tema “Limpar a Quadra de Tênis” (“Clean Up The Tennis Court”). O ambiente de trabalho do robô era um quarto fechado aonde existia um pequeno depósito. O objetivo era coletar

bolas de tênis espalhadas pelo chão, além de uma bola motorizada chamada "squiggle ball", (<http://www.harttoys.com/swball.html>) que se movimentava em direções aleatórias, e depositá-las no depósito. As equipes podiam projetar o depósito para as bolas e marcá-lo de alguma forma, mas deviam considerar que seriam penalizados se as "squiggle balls" escapavam do depósito. Também podiam levar suas próprias bolas de tênis, da cor de sua escolha. Ganhava o robô que no final do tempo determinado coletasse e colocasse mais bolas no depósito. Pontos adicionais eram dados pela quantidade de "squiggle balls" que permanecessem no depósito no final do jogo e aos robôs que demonstrassem que intencionalmente seguiram e tocaram pelo menos uma vez as "squiggle balls" (AAAI, 1996).

Em 1997 e 1998 o evento ficou mais complexo. A tarefa foi "Encontrar Vida em Marte" ("Find life in Mars"). Nessa tarefa os robôs tinham que explorar uma longa área de Marte, um quarto de aproximadamente 1000 ft², procurando por sinais de vida passada e atual, e após isso retornar à nave base com essas formas de vida. Assumia-se que o robô tinha aterrissado em uma cratera. O quarto era ambientado com pedras e com paredes que delimitavam o espaço. As formas de vida passada eram objetos com forma esférica e cúbica, de cores diferentes, do tamanho de bolas de tênis e colocadas no chão. As formas de vida atual eram "squiggle balls", todas da mesma cor, movimentando-se pelo chão. Além disso, existiam zonas de perigo nas quais os robôs não deviam atravessar. Existiam também dois tipos de prisões para colocar as formas de vida, uma para as formas de vida passada e outra para as formas de vida atual. Como nas competições anteriores, ganhava o robô que fizesse mais pontos e tivesse menos penalidades no tempo determinado para a missão (5 a 10 min.). Existiam penalidades se o robô misturasse, nas prisões, as formas de vida passada e atual, modificasse o solo marciano ou navegasse pela zona de perigo

(AAAI, 1997; AAAI, 1998).

Essas competições constituíram o ambiente de provas de diversos robôs e tecnologias, como: a equipe de robôs Ganymede, Lo e Callisto, desenvolvidos pela equipe da Geórgia Tech e ganhadores da competição “Clean the Office” de 1994; e o robô Lola, um robô Nomad 200 da equipe da North Carolina State University, ganhadores da “Clean the Office” de 1995.

Como se pode constatar, o projeto de um sistema robótico para coleta de lixo apresenta suficientes desafios que exigem explorar diversas metodologias de controle de robôs autônomos móveis em procura de soluções eficientes. As soluções encontradas podem, no futuro, ser aplicadas em inúmeras situações, como por exemplo, robôs para procura e resgate, robôs de exploração planetária ou robôs assistentes de escritório, entre outras.

1.4 Metodologia de projeto

Para o desenvolvimento do projeto foi seguida, como referência, a metodologia proposta por Murphy e Arkin (MURPHY, 2000; ARKIN, 2000). Assim, o processo de projeto consistiu de quatro etapas:

- 1) Especificação da tarefa, do robô e avaliação do ambiente. Nessa etapa se descreve o que o robô tem que fazer. As tarefas específicas dessa etapa foram:
 - Descrição da tarefa: consistiu em descrever com detalhe a tarefa que o robô deve executar;
 - Descrição do robô: consistiu em descrever a configuração física do robô, sensores e atuadores utilizados. Essa tarefa serviu para conhecer as vantagens e limitações do robô;

- Descrição do ambiente de trabalho: consistiu em avaliar o ambiente de trabalho do robô para saber o seu grau de dinamismo e se é possível encontrar oportunidades perceptuais que possam ser aproveitadas para ativar comportamentos.
- 2) Projeto. Nessa etapa se estudaram as informações definidas na etapa anterior para determinar os comportamentos que permitem ao robô realizar sua tarefa. As tarefas específicas dessa etapa foram:
- Descrever a tarefa em termos de comportamentos e assim determinar os comportamentos primitivos necessários para cumprir a tarefa;
 - Construir uma tabela de comportamentos. A tabela permite visualizar todos os comportamentos do robô, indicando o nome do comportamento, seu “perceptual schema” (esquema de percepção), seu “motor schema” (esquema motor), a entrada de sensores e o ativador do comportamento. A vantagem de ter essa tabela é poder visualizar se existem conflitos entre os diversos comportamentos.
- 3) Implementação e provas de simulação. Nessa etapa se programaram e simularam cada comportamento por separado. A tarefa específica dessa etapa foi a implementação, teste e refinamento, usando simulação, de cada comportamento independentemente.
- 4) Testes do sistema. Nessa etapa se realizaram os testes de todos os comportamentos em conjunto.

1.5 Organização da dissertação

Esta dissertação inicia-se com essa introdução. O capítulo 2 apresenta os tipos de controle de robôs autônomos móveis e a descrição da arquitetura comportamental reativa escolhida para o desenvolvimento do controle do robô. Além disso, apresenta também a ferramenta de simulação utilizada para testar os algoritmos de controle. O capítulo 3 descreve detalhadamente o desenvolvimento e testes dos algoritmos de controle do robô autônomo para coleta de lixo. O capítulo 4 apresenta as conclusões deste trabalho.

2. Fundamentos teóricos

2.1. Controle de robôs móveis autônomos

Maja Mataric em (MATARIC, 2002) fornece uma boa revisão das estratégias existentes para o controle de robôs móveis autônomos. As principais estratégias e as suas principais características são as seguintes:

- Deliberativa: “pensa muito, logo atua”;
- Reativa: “não pense, reage”;
- Híbrida: “pensa e atua independentemente, em paralelo”;
- Behavior Based: “pensa na forma em que atua”.

A seguir apresenta-se uma breve descrição dessas estratégias de controle.

2.1.1 Estratégia Deliberativa

Essa estratégia usa um modelo centralizado do ambiente para planificar e gerar as ações do robô (MURPHY, 2000). O robô processa todas as informações disponíveis dos sensores e toda a informação gravada internamente para gerar um plano de ação. Gerar um plano de ação (planificar) requer a verificação de todos os planos de ação possíveis até encontrar um com o qual o objetivo possa ser atingido. Planificar é um dos processamentos que consome mais tempo. Se o robô tem que reagir rapidamente, planificar não é uma forma prática. Porém, se tem tempo suficiente, planificar permite ao robô atuar estrategicamente (PAGOAGA, 2001).

Outra característica dessa estratégia é que a planificação é feita sobre um modelo do entorno. Esse modelo tem que ser mantido o mais atualizado possível, já que se o entorno muda antes da resposta do robô o plano será feito com informação

desatualizada dando lugar a erros de operação.

2.1.2 Estratégia Reativa

Essa estratégia implementa o sistema de controle do robô como uma coleção de regras do tipo condição-ação pré-programadas com um número de estados mínimo (PAGOAGA, 2001).

O controlador desses sistemas consiste em uma coleção de regras que mapeiam situações específicas com ações específicas. Assim, esses sistemas conectam diretamente os sensores e os atuadores.

Outra característica importante dos sistemas reativos puros é que não usam nenhum sistema interno de representação do entorno e não realizam planificação. Eles simplesmente reagem de acordo com a informação atual dos sensores. Isso permite que o robô responda muito rápido a mudanças do entorno, típicas nos ambientes não estruturados.

Usualmente é difícil poder antecipar todas as possíveis situações que o robô pode encontrar. Em lugar disso, situações específicas disparam ações específicas e essas ações são usadas para solucionar todos os outros casos. Assim o projetista pode reduzir efetivamente o espaço sensorial a somente às situações de entrada que interessam, mapeando estas com as ações apropriadas, simplificando, assim, o sistema de controle.

As limitações dessa estratégia são que os robôs não têm informação do ambiente, não tem memória e não tem a habilidade de apreender.

Estratégias puramente reativas têm provado efetividade em uma variedade de problemas que têm como característica estarem bem definidos no momento do projeto (MILLER et al., 1992), mas são inflexíveis nas suas ações devido a sua

incapacidade de adaptação (MATARIC, 2002).

2.1.3 Estratégia Híbrida

Essa estratégia combina o melhor que tem as estratégias de controle reativo e deliberativo. A parte deliberativa encarrega-se das planificações enquanto a parte reativa encarrega-se das ações que exigem reagir rapidamente, como evitar obstáculos ou manter-se seguindo uma parede a uma distância fixa.

O desafio dessa estratégia de controle é fazer trabalhar as duas partes sincronizadas. A dificuldade de sincronização ocorre em razão das duas estratégias trabalharem com escalas de tempo diferentes. Assim, o sistema requer um componente intermediário entre os níveis de ação deliberativa e reativa, por isso são chamados de sistemas de três níveis (“three-layer systems”) (MATARIC, 2002).

Nas arquiteturas de controle híbrido, o nível reativo é controlado por um planejador ou seqüenciador. Isso permite ao robô ter capacidades de navegação sofisticadas. Ao invés de somente seguir paredes e evitar obstáculos aleatoriamente, os sistemas híbridos permitem ao sistema de controle realizar comportamentos em forma seqüencial (PAGOAGA, 2001). Um comportamento seqüencial típico para navegar até uma determinada sala seria: “seguir a parede, dar a volta no corredor e entrar na sala MZ-08”. Este tipo de seqüência é difícil de ser implementada em um sistema puramente reativo. Nos sistemas híbridos os comportamentos podem ser ativados e desativados para conseguir objetivos mais abstratos.

Ronald Arkin (ARKIN, 1998) explica como as principais arquiteturas híbridas abordam a planificação:

- **Seleção:** o planejador faz a seleção de diferentes configurações do sistema de controle reativo de acordo com o que o robô precisa fazer;

- **Aconselhar:** o planejador sugere mudanças que o sistema reativo pode usar ou não;
- **Adaptação:** o planejador modifica continuamente a configuração do sistema de controle reativo;
- **Adiamento (“Postponing”):** o planejador toma a decisão somente no último momento possível, permitindo que as informações sensoriais mais recentes afetem as decisões.

Essa estratégia tem provado efetividade em ambientes e tarefas nas quais modelos internos e planificação podem ser usados e que as demandas em tempo real sejam poucas ou suficientemente independentes da lógica de alto nível. Porém, se o entorno é muito mutável, um sistema puramente reativo pode ser uma solução melhor (MATARIC, 2002).

As principais arquiteturas que ilustram os princípios da abordagem híbrida são: AuRA (ARKIN, 1986), Atlantis (GAT, 1991), Planner-Reator (LYONS; HENDRIKS, 1992) e Procedural Reasoning System (GEORGEFF; LANSKY, 1987).

2.1.4 Estratégia Baseado em Comportamento (“behavior-based”)

O controle baseado em comportamento (“behavior-based”) é uma extensão dos sistemas de controle reativos (PAGOAGA, 2001). Pode-se dizer que esse tipo de estratégia fica entre as estratégias puramente reativa e deliberativa.

Segundo Mataric (MATARIC, 1999), a robótica baseada em comportamento (“Behavior Based robotics – RBB”) integra os campos da inteligência artificial, engenharia e ciência cognitiva. Essa abordagem é uma metodologia para projetar agentes autônomos e robôs, sendo também uma espécie de arquitetura de agentes inteligentes. As arquiteturas fornecem uma estrutura e impõem limitações na forma

em que os problemas de controle de robôs são resolvidos. A metodologia baseada em comportamento impõe uma filosofia geral, inspirada na biologia, de abaixo acima, permitindo certa liberdade de interpretação. Seu objetivo é desenvolver métodos para controlar sistemas artificiais (usualmente robôs reais, mas também robôs simulados e outros agentes de software autônomos) e usar a robótica para modelar e entender melhor os sistemas biológicos (usualmente animais, desde insetos até humanos).

Fundamentalmente, um controlador desenvolvido utilizando essa estratégia é composto por um conjunto de componentes modulares denominados comportamentos. Um comportamento é uma regra de controle que tem como missão atingir e manter um objetivo. Os comportamentos são executados em paralelo, e utilizam os dados de entradas dos sensores e/ou informações de outros comportamentos, e os transformam em comandos de saída que são enviados aos atuadores do robô e/ou outros comportamentos (MATARIC; JONES 2005).

As bases dos conceitos da abordagem baseada em comportamento estão na robótica reativa, especialmente na arquitetura “Subsumption” (BROOKS, 1986). Na arquitetura “Subsumption” a tarefa é decomposta em módulos de comportamentos que trabalham em paralelo realizando tarefas específicas como evitar obstáculos e identificar objetos (figura 2.1).

Porem, apesar de serem usualmente confundidos na literatura, a estratégia baseada em comportamento é muito mais poderosa do que a abordagem puramente reativa (MATARIC 2002). Em quanto que os sistemas baseados em comportamento apresentam algumas características dos sistemas reativos e podem conter módulos (comportamentos) reativos, o processamento da informação não é limitado a só perceber e reagir. Esses sistemas podem ter estados internos, memória e

aprendizagem ao longo prazo. Além disso, podem usar diferentes formas de representação interna (MATARIC 1992b) e realizar processamento computacional nessas representações para decidir que ação executar (MATARIC 1992a). Como foi mencionado anteriormente, essas características constituem limitações na abordagem reativa.

Figura 2.1. Decomposição de tarefas na arquitetura “Subsumption” (BROOKS, 1986).

O desafio de projeto para construir robôs com controle baseado em comportamento é selecionar a composição correta de comportamentos e reflexos para gerar o comportamento global desejado. Usualmente não existe uma conexão visível entre as estruturas de comportamentos internos e o comportamento global produzido. Por exemplo, um robô que limpa oficinas pode não ter um comportamento específico ‘limpar’, ao invés disso, sua interação com o entorno pode resultar em limpar oficinas. Isso é conhecido como comportamento emergente e é uma característica destes sistemas (STEELS, 1994).

2.2 “MOTOR-SCHEMA”

O “Motor-Schema” é uma metodologia desenvolvida para especificar e projetar sistemas robóticos baseados em comportamento (“behavior based”) (ARKIN, 1998). Foi apresentada por Ronald C. Arkin em (ARKIN, 1987). Esse método permite projetar comportamentos que atuam de forma paralela e distribuída para realizar ações robóticas inteligentes em resposta a um estímulo do ambiente.

O método é baseado na teoria psicológica de esquemas (“schema theory”). O uso de esquemas como um modelo filosófico para a explicação do comportamento psicológico humano data da época de Immanuel Kant no século XVIII (ARKIN, 1998). Arkin em (ARKIN, 1998) define o termo esquema como sendo a unidade básica do comportamento a partir do qual ações complexas podem ser construídas. Citando Arkin (ARKIN, 1988), um esquema é a unidade básica do comportamento a partir do qual ações complexas podem ser realizadas, ele consiste do conhecimento de como agir e perceber, bem como no processo computacional pelo qual o comportamento é implementado.

Na metodologia “Motor-Schema” um comportamento é constituído, basicamente, por dois componentes: um esquema perceptivo (“perceptual schema”) e um esquema motor (“motor schema”).

O esquema perceptivo engloba as percepções. A filosofia da metodologia “Motor-Schema” para o desenvolvimento da percepção é conhecida como “action oriented perception”. Esse princípio diz que as atividades da percepção sempre devem ser vistas do ponto de vista das necessidades motoras (ARKIN, 1998). Assim, o esquema perceptivo fornece as informações do ambiente exclusivamente necessárias para aquele determinado comportamento, permitindo ao robô reagir rapidamente às mudanças do mesmo. Outra característica importante dos esquemas

perceptivos é que podem ser definidos recursivamente, isto é, diferentes sub-esquemas perceptivos podem extrair pedaços de informação que serão subsequenteiramente processados por outro esquema perceptivo, gerando como resultado uma informação mais significativa para o comportamento (ARKIN, 1998).

O esquema motor representa a forma da atividade física. Usa a informação fornecida pelo esquema perceptivo para gerar o comando de controle do robô. Esse comando define a forma que o robô deve responder ao estímulo percebido. A resposta do esquema motor é codificada em forma de vetores e usa o método de navegação em campos potenciais.

O método de campos potenciais foi apresentado por Khatib (KHATIB, 1985) e Krogh (KROGH, 1984) como a base de geração de trajetórias sem obstáculos para sistemas robóticos móveis e robôs manipuladores. Essa metodologia utiliza uma função potencial arbitrária para gerar um campo de forças que representa a velocidade e direção com as quais o robô deve navegar pelo ambiente. Os objetivos são associados a campos de atração e os obstáculos a campos de repulsão. O método consiste em construir campos potenciais separados para representar a relação entre o robô e cada objeto dentro do campo sensorial do robô. Esses campos são combinados, mediante superposição, para gerar um único campo potencial. A partir do gradiente do campo potencial resultante, gera-se um campo de forças, representado por vetores com componentes de magnitude (velocidade) e direção, com o qual se pode planificar trajetórias de navegação para o robôs móveis e manipuladores robóticos. Dessa forma o robô pode ser visto como uma partícula em um campo de forças. Uma característica importante desse método é que codifica a ação de forma contínua, isto é, não interessa quão pequeno seja o elemento do

ambiente, em cada ponto do espaço sempre se tem um vetor de força associado a ele (ARKIN, 1998).

Existem cinco tipos de campos potenciais básicos, que podem ser combinados para gerar campos mais complexos (MURPHY, 2000):

- a) **Campo Uniforme:** nesse caso a força tem a mesma direção e magnitude em todo o campo. Dentro desse campo o robô, não importando onde nem em que posição esteja orientado, sentirá a necessidade de alinhar-se à direção do campo e navegar com uma velocidade proporcional à magnitude da força. O campo uniforme é usado para gerar o comportamento “ir na direção X”;
- b) **Campo Perpendicular:** esse campo é orientado perpendicularmente a algum objeto ou parede.
- c) **Campo de Atração:** nesse campo o objeto gera uma força de atração em sua direção. Esse campo geralmente é associado ao objetivo do robô;
- d) **Campo de Repulsão:** é o oposto ao campo de atração, nesse caso o objeto gera uma força de repulsão que faz o robô afastar-se dele; está associado aos obstáculos presentes no ambiente;
- e) **Campo Tangencial:** esse campo é tangencial ao objeto. Pode se pensar nele como sendo perpendicular às linhas radiais que se originam a partir do objeto. O campo tangencial pode girar em sentido horário ou anti-horário. Esses campos são úteis para dirigir o robô ao redor do obstáculo ou fazer que o robô investigue alguma coisa.

A figura 2.2 mostra estes cinco campos básicos.

O esquema motor utiliza os campos potenciais para gerar a ação motora do

comportamento, porem só calcula a contribuição de cada campo na posição instantânea onde o robô se encontra (ARKIN, 1989). Assim, não existe planificação de trajetória, a reação do robô ao ambiente é calculada tão rápido quanto o processamento sensorial permite (ARKIN, 1998).

A figura 2.3 mostra um exemplo de navegação de um robô desenvolvido utilizando a metodologia “Motor-Schema”. O campo de força completo é apresentado somente para ilustrar a reação do robô em cada ponto, mas o agente somente calcula a força na posição instantânea onde o robô se encontra.

Na metodologia “Motor-Schema”, diversos esquemas motores foram definidos (ARKIN, 1998):

- **“Move-ahead”**: mover-se em uma direção particular;
- **“Move-to-goal”**: mover-se em direção do objetivo detectado. Existem duas versões deste esquema: balística e controlada.
- **“Avoid-static-obstacle”**: afastar-se de obstáculos passivos;
- **“Dodge”**: afastar-se de um obstáculo em movimento que esteja se aproximando;
- **“Escape”**: afastar-se do ponto de interceptação calculado entre o robô e um predador se aproximando;
- **“Stay-on-path”**: mover-se em direção ao centro de um caminho: estrada ou corredor;
- **“Noise”**: mover-se em direção aleatória por um determinado tempo;
- **“Probe”**: mover-se em direção a áreas abertas;

(a)

(b)

(c)

(d)

(e)

Figura 2.2. Campos potenciais básicos: (a) campo uniforme (b); campo perpendicular; (c) campo de atração; (d) campo de repulsão; e (e) campo tangencial.

Figura 2.3. Navegação de um robô desenvolvido seguindo a metodologia “Motor Schema”

(ARKIN, 1987).

- **“Dock”**: aproximar-se de um objeto com uma determinada direção;
- **“Avoid-past”**: afastar-se de áreas visitadas recentemente;
- **“Teleautonomy”**: permite que um operador humano forneça comandos de controle no mesmo nível de qualquer outro esquema motor.

Esses esquemas motores são os blocos básicos para navegação de robôs autônomos na metodologia “Motor-Schema”.

A metodologia de campos potenciais fornece uma forma eficiente de planejar trajetórias de navegação e que pode ser calculada em tempo real (CHOSSET et al., 2005). Contudo, essa metodologia apresenta vários problemas. O mais característico é o fenômeno conhecido como mínimo local. Um mínimo local é um ponto do campo onde a resultante das forças é zero. Nesse ponto o robô não tem nenhuma força que gera movimento e, assim, fica preso no ponto. Existem diversos métodos desenvolvidos para corrigir este problema. Em particular a metodologia “Motor-Schema” utiliza o esquema motor “Noise” que injeta um ruído aleatório no campo para evitar que o robô fique detido num ponto diferente do objetivo (ARKIN, 1998).

O método de coordenação de comportamentos da metodologia “Motor-Schema” é a somatória de vetores. Cada vetor de saída de um esquema motor é multiplicado por um ganho associado e é somado aos outros vetores de saída. O resultado é um vetor que engloba a contribuição de todos os comportamentos ativos. Esse vetor deve ser normalizado para garantir que seja executável pelo robô. O vetor resultante normalizado é enviado ao robô para sua execução. Esse processo de percepção-ação é repetido tão rápido quanto seja possível (ARKIN, 1998).

A figura 2.4 mostra o diagrama de blocos de um controlador desenvolvido usando a metodologia “Motor-Schema”.

Figura 2.4. Diagrama de blocos de um controlador desenvolvido usando a metodologia "Motor Schema".

2.3 Ferramentas de simulação utilizadas

Para simular aplicação robótica desenvolvida nesse trabalho foram utilizadas as plataformas de simulação de robôs Player/Stage. Estas plataformas são distribuídas gratuitamente na página da internet <http://playerstage.sourceforge.net>.

As características apresentadas por estas ferramentas permitem o desenvolvimento de projetos de robôs móveis de forma rápida e eficiente e facilitam a transferência do projeto da simulação ao hardware real.

2.3.1 PLAYER/STAGE

A ferramenta Player/Stage foi desenvolvida nos laboratórios de pesquisa de robôs da Universidade de Southern Califórnia (USC Robotics Research Lab) em 1999 (GERKEY et al, 2003). A ferramenta consiste de dois componentes: o Player,

que é um servidor de dispositivos que permite o controle dos diversos sensores e atuadores de um robô; e o Stage, que é um simulador 2D para robôs moveis.

O servidor Player é uma aplicação para controle de robôs através de uma conexão de rede IP. Segue um protocolo cliente-servidor e fornece uma interface de comunicação com sensores e atuadores do robô. O programa de controle (cliente) conversa com o Player mediante uma interface TCP, podendo ler dados dos sensores, enviar comandos aos atuadores e configurar os dispositivos do robô em tempo real. Esta ferramenta dispõe de uma variedade de interfaces de comunicação para hardware, como câmeras, sonares ou “Laser Rangers” além de livrarias com de métodos sofisticados de sensoriamento e algoritmos de controle do tipo seguimento de marcos do ambiente (“landmark tracking”) e algoritmos de localização probabilística. Além disso, permite controlar diversas plataformas de robôs comerciais, em particular os robôs móveis Pioneer da empresa ActiveMedia (ver figura 2.5).

Figura 2.5: Plataforma robótica Pioneer 2-DX: O Player pode controlar muitas plataformas de robôs comerciais e seus periféricos (GERKEY, 2003).

O Stage é um simulador 2D que permite simular uma população de robôs moveis, sensores e objetos em um ambiente 2D (ver figura 2.6).

Figura 2.6: Visualização do ambiente de simulação do programa de simulação Stage.

Os dispositivos simulados são acessados através do servidor Player. Os usuários escrevem algoritmos de controle para robôs como clientes do servidor Player. Uma das vantagens desta forma de comunicação é que os programas clientes se comunicam com mesma interface, fornecida pelo servidor Player, seja para controlar o robô no mundo real ou no ambiente de simulação Stage. Essa característica faz com que a transferência do código de controle desde a simulação ao robô real seja realizada de forma muito rápida.

O Stage foi especificamente projetado para fornecer suporte à pesquisa de sistemas multi-robôs. Ele permite simular uma grande população de robôs, facilitando o desenvolvimento desse tipo de sistemas na medida em que seria

economicamente proibitivo desenvolver o sistema fisicamente antes de ser testado. A figura 2.7 apresenta como exemplo um ambiente de simulação com uma população de robôs. Atualmente o numero de robôs que podem ser simulados pelo Stage é de dezenas a centenas.

Figura 2.7: Visualização da simulação de um sistema multi-robô com o programa de simulação Stage.

2.3.2 Arquitetura PLAYER/STAGE

A figura 2.8 apresenta a arquitetura do Player. A arquitetura do Player consiste basicamente dos seguintes componentes: dispositivos, o servidor Player e aplicações clientes.

Figura 2.8. Arquitetura completa do Servidor Player (GERKEY et al. 2001).

Os dispositivos, como por exemplo, uma câmera, um sonar ou um robô completo, são o hardware real ou o simulado pelo módulo Stage.

O servidor Player fornece a interface entre o robô e qualquer programa que solicite ou envie informação para o robô. Usualmente, pode ser executado localmente no computador onde o dispositivo está conectado, como por exemplo, um sensor de distância baseado em laser conectado a um computador, ou no robô mesmo servindo como Sistema Operacional.

As aplicações cliente são os programas de controle do robô. O cliente pode ser executado em qualquer computador que tenha conectividade de rede ligado à máquina que contém o servidor Player. No início, o cliente estabelece uma conexão tipo TCP com o servidor Player. No passo seguinte o cliente envia algumas mensagens ao servidor para ativar os dispositivos que vai controlar. Depois disso, o

servidor continua enviando dados desses dispositivos ao cliente e o cliente executa o controle enviando os comandos apropriados para o servidor (ver figura 2.9).

Figura 2.9: Interação cliente-servidor no Player/Stage.

3. Desenvolvimento da aplicação robótica de coleta de lixo

O projeto da aplicação robótica de coleta de lixo implicou no desenvolvimento de comportamentos reativos que permitem ao robô navegar de forma segura, explorando um ambiente desconhecido em procura de objetos que reconhecidos com lixo são coletados e levados a um depósito determinado.

Como foi mencionado no capítulo 1, a metodologia utilizada para atingir o objetivo proposto é constituída pelas seguintes etapas:

- 1) Determinação dos comportamentos necessários para realizar a tarefa de coleta de lixo;
- 2) Expressar esses comportamentos como esquemas motores (“motor-schemas”);
- 3) Desenvolver os comportamentos;
- 4) Coordenação dos comportamentos;
- 5) Testes de simulação da aplicação.

Adicionalmente foi necessário aprender a utilizar a ferramenta de simulação Player/Stage. Esta ferramenta é muito usada em diversos centros de pesquisa e universidades de todo o mundo.

A seguir apresentam-se os trabalhos realizados para o desenvolvimento da tarefa de coleta de lixo.

3.1 Robô escolhido para realizar a tarefa de coleta de lixo

O robô escolhido para realizar a tarefa de coleta de lixo deve ter características que lhe permitam cumprir a tarefa. Seus sensores devem permitir

detectar as latas espalhadas pelo chão, detectar objetos para evitar colisões e conhecer sua posição no espaço.

Nesse trabalho foi escolhido o robô Pioneer 2AT da companhia Mobile Robots (<http://www.activrobots.com/index.html>) apresentado na figura 3.1. Esse robô foi escolhido por apresentar as características adequadas para atingir o objetivo proposto.

Figura 3.1. Robô Pioneer 2AT da companhia Mobile Robots (disponível em: <http://www.activrobots.com/ROBOTS/p2at.html>).

As especificações técnicas desse robô, fornecidas pelo fabricante, encontram-se no Anexo 1. Dessas especificações pode-se deduzir o seguinte:

- O robô tem quatro motores com controle independente. Deve-se considerar essa característica para o projeto do sistema de controle de locomoção do robô.
- O robô Pioneer 2AT vem com anéis de sonares frontais e traseiros incluídos. O anel de sonares são utilizados para detectar obstáculos. Esses sonares têm um alcance de 15 cm. a 5 m. de distância. Contudo, pela posição dos sonares no robô tem-se uma limitação de altura mínima

dos obstáculos para que o robô possa detectá-los. Na figura 3.1 pode observar-se que o anel de sonares fica na parte superior da plataforma do robô. Dos dados do manual de operações do Pioneer 2AT tem-se que os sonares ficam a cerca de 26 cm. do chão.

- O robô é equipado com um sistema de visão com câmera colorida. Isso permite detectar objetos pela cor. Assim, aproveitando uma das características principais da lata de refrigerante coca cola, que é a cor vermelha, essas latas são detectadas.
- O robô Pioneer 2AT tem a opção de ser adquirido com um manipulador equipado com uma garra. Da figura 3.1 observa-se que este manipulador cumpre perfeitamente com o objetivo desejado de pegar latas no chão. O Apêndice 1 apresenta as especificações desse manipulador.

Além de possuir as características desejadas, desenvolver os algoritmos de controle para o robô Pioneer 2AT apresenta as seguintes vantagens:

- O simulador Stage conta com um modelo pré-programado do robô Pioneer 2AT.
- O servidor Player conta com um “driver” especial, denominado p2os que é totalmente compatível com o sistema operacional P2OS, que é o sistema de controle do robô Pioneer 2AT.

Assim, se for desejado, após o desenvolvimento dos algoritmos de controle com o simulador Stage, pode-se transferi-los para o robô real sem precisar de mudanças significativas facilitando, assim, a evolução do projeto.

3.1.1. Modelo do Robô Pioneer 2AT

O modelo do robô Pioneer 2AT encontrado no simulador Stage consiste de um modelo geométrico que lembra a forma do robô Pioneer real. As dimensões do modelo no simulador são 44 cm. por 33 cm., igual ao robô real. O modelo possui dois anéis de sonares distribuídos no perímetro do robô. Além disso, podem-se adicionar modelos de outros dispositivos robóticos compatíveis com o robô Pioneer, tais como, sensor de distância a laser, manipulador com garras, sistema de visão e chaves de “bumpers”. Essas características do simulador do robô Pioneer 2AT presentes no Stage facilita o desenvolvimento de projetos para esse tipo de robô.

Nesse trabalho é utilizado o modelo do robô Pioneer 2AT que inclui as seguintes características:

- Controle de baixo nível em modo diferencial;
- Arranjo de 16 sonares, sendo 8 frontais e 8 traseiros, que permitem detectar objetos;
- Um manipulador com garras e com sensores infravermelhos que permitem detectar objetos dentro do manipulador;
- Um sistema de visão com câmera colorida.

Desta forma, o modelo utilizado possui todas as características necessárias para desenvolver e testar a aplicação desejada. A figura 3.2 apresenta o modelo do robô Pioneer 2AT incluindo o manipulador e o sistema de visão.

Figura 3.2. Modelo do robô Pioneer 2AT utilizado nesse trabalho.

3.1.2. Interfaces de Controle

Como foi mencionado no Capítulo 2 a comunicação do programa de controle com o robô simulado é feita através do servidor Player. O servidor Player proporciona interfaces de controle que permitem enviar comandos ou ler dados dos sensores e atuadores do robô. As mesmas interfaces são utilizadas com robôs reais ou robôs simulados. Essa comunicação é ilustrada na figura 3.3. Essa característica permite que os mesmos algoritmos desenvolvidos e testados mediante o simulador possam ser usados em robôs reais sem mudanças significativas.

As interfaces de controle utilizadas no desenvolvimento dos algoritmos apresentados neste trabalho são as seguintes:

- Position2d;
- Sonar;
- Blobfinder;
- Gripper.

Figura 3.3. Modelo da comunicação do servidor Player com o simulador Stage e com plataformas robóticas reais.

Position2d. Essa interface permite controlar robôs móveis como o robô Pioneer. Mediante essa interface controla-se o movimento do robô e obtém-se a informação da odometria obtida pelos encoders dos motores do robô.

No simulador Stage o sistema de coordenadas odométrico utiliza a convenção da mão esquerda. A posição no espaço desse sistema é a posição aonde o robô inicia a contagem odométrico, que geralmente é a posição inicial da aplicação robótica. A origem do sistema odométrico encontra-se na posição do centro de massa do robô, o eixo “x” coincide com a orientação da parte frontal do robô e o eixo “y” encontra-se orientado á esquerda do robô, perpendicular ao eixo “x”. A figura 3.4 mostra a posição deste sistema.

Figura 3.4. Posição do sistema de coordenadas odométrico.

As informações de odometria fornecidas por essa interface são as coordenadas de posição do centro de massa do robô, x e y , e a orientação do robô, Φ , descritas em relação ao sistema de coordenadas odométrico do robô.

A interface `position2d` fornece suporte para o controle de robôs com locomoção em modo diferencial (robôs com capacidade de translação para frente e rotação) ou robôs omnidirecionais (robôs com capacidade de translação para frente, para os lados e rotação). Existem comandos de controle de posição, de velocidade (translação e rotação), ou de posição e velocidade. Porém, o simulador Stage suporta somente comandos de controle de velocidade. Os algoritmos de controle desenvolvidos nesse trabalho controlam o movimento do robô em modo diferencial.

Sonar. A interface de controle dos sonares permite obter a informação da distância, em metros, dos objetos detectados mediante os sonares. A figura 3.5 mostra a distribuição dos anéis de sonares. Cada sonar tem um alcance máximo de 5 m e apresentam a mesma distribuição que no robô Pioneer real.

Figura 3.5. Distribuição dos anéis de sonares no modelo do robô Pioneer 2AT.

Gripper. A interface de controle do manipulador do robô é utilizada para conhecer o estado do manipulador e a posição de um objeto dentro do manipulador. O manipulador possui dois sensores infravermelhos que permitem conhecer a posição do objeto no manipulador. O estado do manipulador é obtido mediante a variável “state” e a posição do objeto dentro do manipulador mediante a variável “beam”. A tabela 3.1 apresenta os possíveis valores das variáveis state e beam.

Os comandos de controle desta interface são compatíveis com o manipulador fornecido com o robô Pioneer. Porém, o simulador Stage suporta somente dois comandos: abrir manipulador (OPEN) e fechar manipulador (CLOSE).

Tabela 3.1. Variáveis correspondentes à interface Gripper.

VARIÁVEL	VALOR	SIGNIFICADO
Beams	0	Manipulador vazio
	1	Objeto interfere o primeiro sensor infravermelho
	2	Objeto interfere o segundo sensor infravermelho
	3	Objeto interfere o dois sensores infravermelhos
State	1	Manipulador Aberto
	2	Manipulador Fechado
	3	Manipulador em movimento
	4	Erro no manipulador

Blobfinder. Essa interface permite controlar sistemas de detecção de objetos mediante a cor (“color blob detectors”), tal como o sistema ACTS fornecido com os robôs Pioneers. Com essa interface é possível obter a localização e a área de um objeto de cor em uma imagem. Além disso, pode-se configurar o sistema para detectar objetos de cores diferentes, associando um canal a cada cor, assim, os objetos de cor vermelha são agrupados no canal 1, os de cor azul no canal 2 etc.

As informações obtidas mediante essa interface são:

- Número de objetos detectados;
- A cor de cada objeto detectado;
- Área do objeto em pixels;
- Posição x e y do centróide do objeto na imagem.

A figura 3.6 mostra um exemplo da interface Blobfinder em operação.

Figura 3.6. Exemplo de utilização da interface Blobfinder com o Robô Pioneer.

3.2 Comportamentos desenvolvidos

Existem dois tipos de comportamentos que podem ser desenvolvidos para controlar um robô móvel na arquitetura “Motor-Schema”: comportamentos primitivos e comportamentos abstratos (MURPHY 2000). Um comportamento primitivo é composto unicamente por um esquema perceptivo e um esquema motor e não está inserido em um sistema de coordenação de comportamentos. Um comportamento abstrato é um comportamento composto por diversos comportamentos primitivos ou tem múltiplos esquemas perceptuais e motores (MURPHY 2000).

A tarefa de coleta de lixo implica que o robô navegue de forma segura por um ambiente desconhecido procurando por objetos reconhecidos como lixo. Nesse projeto é considerado como lixo as latas de refrigerante coca-cola. Ao serem encontradas as latas, o robô deve segurá-las com o manipulador e transportá-las para um depósito pré-determinado.

Uma forma de solucionar esse problema é o seguinte:

1. O robô utiliza uma seqüência de comportamentos para atingir o objetivo;
2. Esses comportamentos são do tipo abstrato.

Seguindo essas premissas, foram determinados os seguintes comportamentos abstratos para realizar a tarefa de coleta de lixo:

- **EXPLORAR PROCURANDO LIXO.** O objetivo desse comportamento é fazer o robô navegar de forma segura explorando um ambiente desconhecido. Nesse projeto essa exploração tem como missão procurar por objetos de atração no ambiente, nesse caso, as latas de refrigerante coca-cola. A detecção de latas é feita mediante o sistema de visão do robô aproveitando uma das características principais das latas, sua cor vermelha. Define-se para esse trabalho que as latas de refrigerante são os únicos objetos existentes no chão de cor vermelha, assim, a sua cor é uma boa forma de detectá-las. Esse comportamento comanda o robô até que uma ou mais latas sejam detectadas.
- **COLETAR LATA.** Esse comportamento tem como objetivo aproximar o robô da lata detectada até colocá-la ao alcance do manipulador. Em caso de mais de uma lata ser detectada, o comportamento aproximará o robô à lata mais próxima dele.
- **PEGAR LATA.** O objetivo desse comportamento é movimentar o robô até colocar a lata na melhor posição dentro do manipulador para que seja segura.
- **IR AO DEPÓSITO.** Esse comportamento faz com que o robô carregue a lata até a entrada do depósito de latas. O robô deve realizar essa tarefa evitando colidir com obstáculos ou com outras latas espalhadas pelo chão.

- **DEIXAR A LATA.** Esse comportamento faz com que o robô deixe a lata dentro do depósito.

Cada um dos comportamentos abstratos definidos anteriormente é constituído por comportamentos primitivos que, devidamente coordenados, fazem o comportamento abstrato atingir o seu objetivo. Na tabela 3.2 apresentam-se os comportamentos primitivos desenvolvidos para construir os comportamentos abstratos.

Tabela 3.2. Comportamentos abstratos e os comportamentos primitivos que os constituem.

COMPORTAMENTOS ABSTRATOS	COMPORTAMENTOS PRIMITIVOS
EXPLORAR PROCURANDO LIXO	Avoid_Obstacle
	Noise
COLETAR LATA	Atração_Vermelho
	Avoid_Obstacle
	Noise
PEGAR LATA	Avançar
	Retroceder
IR AO DEPÓSITO (Docking)	Move_To_Goal
	Tangencial_Horário
	Tangencial_Anti-Horário
	Avoid_Red_Obstacle
	Avoid_Obstacle
DEIXAR LATA	Noise
	Avançar
	Retroceder

Na seção a seguir descreve-se a formalização de cada comportamento primitivo.

3.3 Formalização dos comportamentos primitivos

Esse item descreve os comportamentos primitivos utilizados para construir os comportamentos abstratos definidos para a realização da tarefa coletar latas.

3.3.1. Comportamentos AVANÇAR e RETROCEDER

Os comportamentos de avançar e retroceder geram os movimentos básicos do robô. Esses comportamentos são definidos como se segue:

- **Avançar:** gera um campo vetorial constante que induz o robô a avançar em linha reta;
- **Retroceder:** gera um campo vetorial constante que induz o robô a navegar em marcha ré.

A codificação destes esquemas motores é composta pelas seguintes expressões:

Avançar:

$$V_{mag} = 1,0;$$

$$\Phi_{orient.} = 0^\circ.$$

Retroceder:

$$V_{mag} = 1,0;$$

$$\Phi_{orient.} = 180^\circ.$$

Nessas expressões V_{mag} representa a magnitude do vetor velocidade de movimento do robô e Φ_{orient} representa a orientação do vetor velocidade. Observa-se que V_{mag} igual a 1,0 significa velocidade máxima e V_{mag} igual a 0,0 significa ausência de velocidade.

3.3.2. Comportamento AVOID_OBSTACLE

Este comportamento tem como objetivo evitar que o robô colida com os obstáculos presentes no ambiente.

Como visto, o robô Pioneer 2AT possui dois anéis de sonares idênticos, um frontal e um traseiro. Eles são usados para detectar o obstáculo ao redor do robô. Na figura 3.7 pode se observar a posição de cada sonar do anel frontal.

Figura 3.7. Esquema da posição dos sonares do anel frontal ao robô. Extraído do manual de operações do robô Pioneer (MOBILEROBOTS, 2006).

O esquema perceptivo fornece ao esquema motor a leitura de cada sonar e o seu ângulo relativo à base robótica. Com esses dados, o esquema motor gera, se algum objeto for detectado, um vetor de repulsão com direção oposta à posição de cada sonar. A figura 3.8 mostra os parâmetros usados para o cálculo do vetor de repulsão.

A magnitude do vetor de repulsão varia lineamente com a distância do objeto detectado. É definida uma esfera de influência do objeto. Fora dessa esfera de influência (S) não existe repulsão. Dentro da esfera, a magnitude do vetor aumenta

de forma linear até a borda da margem de segurança. Dentro da margem de segurança (M) a magnitude da repulsão é máxima (1.0).

Figura 3.8. Parâmetros utilizados no cálculo do vetor de repulsão. O objeto detectado é representado pelo círculo preto no centro do círculo.

A magnitude do vetor de repulsão é dada por:

$$V_{mag}(d) = 0 \quad d > S;$$

$$V_{mag}(d) = \frac{S - d}{S - M} \quad M \leq d \leq S;$$

$$V_{mag}(d) = 1,0 \quad d \leq M;$$

$$\Phi_{orient} = \alpha_{sonar} + 180^\circ.$$

Finalmente, o vetor resultante do comportamento corresponde à somatória vetorial dos vetores repulsivos gerados para cada sonar.

3.3.3. Comportamento NOISE

O comportamento “NOISE” é um comportamento que gera um vetor de magnitude constante e orientação aleatória que permanece ativo durante um tempo determinado em algumas condições. A codificação deste esquema motor utiliza as seguintes equações:

$$V_{mag} = 1,0;$$

$$\Phi_{orient}(i = \text{insistência}) = \text{ângulo aleatorio (graus)}.$$

O parâmetro denominado “insistência” representa períodos de amostragem do controle em que a orientação do vetor permanece constante.

3.3.4. Comportamento ATRAÇÃO_VERMELHO

O objetivo deste comportamento é gerar um vetor de atração a objetos vermelhos. Os parâmetros utilizados para determinar o vetor resultante são apresentados na figura 3.9.

O esquema perceptivo deste comportamento utiliza a informação fornecida pelo sistema de visão do robô para determinar a área e o centróide do objeto vermelho encontrado na imagem.

O esquema motor desse comportamento utiliza o centróide do objeto para calcular o vetor resultante. A magnitude do vetor é controlada mediante a área do objeto detectado. Quando o robô se aproxima do objeto a área percebida aumenta. A magnitude do vetor varia de forma linear de 1,0, ao detectar o objeto, para 0,0 quando a área percebida atinge um valor denominado Área Máxima de Segurança.

Figura 3.9. Parâmetros do comportamento ATRAÇÃO_VERMELHO.

A Área Máxima de Segurança é determinada experimentalmente e serve para evitar que o robô colida com o objeto detectado. A orientação do vetor é dada pelo ângulo de posição do objeto em relação ao robô. As equações que descrevem este comportamento são as seguintes:

$$V_{mag}(a) = 1 - \frac{1}{A}a \quad a < A = \text{Área Máxima de Segurança};$$

$$V_{mag}(a) = 0.0 \quad a \geq A;$$

$$\Phi_{orient} = \alpha_{centroide} \quad (\text{graus}).$$

No caso em que o robô detecte em uma única imagem vários objetos vermelhos, o comportamento só gera o vetor de atração para o objeto que tiver maior área em pixels.

3.3.5. Comportamento AVOID_RED_OBSTACLE

O objetivo desse comportamento é evitar que o robô colida com as latas de refrigerante coca-cola espalhadas pelo chão. Essas latas não têm a altura necessária para serem detectadas pelos sonares do robô. Assim, nesse caso é necessário utilizar outro tipo de sensor, como o sistema de visão do robô.

O esquema perceptivo desse comportamento determina a área e o centróide de cada objeto vermelho detectado. O esquema motor calcula um vetor de repulsão para cada um desses objetos. Os parâmetros usados apresentam-se na figura 3.10.

Figura 3.10. Parâmetros do comportamento AVOID_RED_OBSTACLE.

A magnitude do vetor resultante aumenta linearmente em função da área do objeto percebida. No momento que o objeto é percebido a magnitude é 0,0, aumentando até 1,0 quando a área percebida atinge um valor denominado Área Máxima de Segurança. Essa Área Máxima de Segurança representa a Margem de

Segurança do comportamento AVOID_OBSTACLE. A magnitude do vetor continua igual ao valor máximo (1,0) para áreas percebidas maiores do que a Área Máxima de Segurança.

A orientação do vetor é o ângulo oposto da orientação do objeto com respeito ao robô. As equações são:

$$V_{mag}(a) = 0,1 + \frac{0,9}{A}a \quad a \leq A = \text{Área Máxima de Segurança};$$

$$V_{mag}(a) = 1,0 \quad a > A;$$

$$\Phi_{orient} = \alpha_{centroide} + 180^\circ.$$

O vetor final resultante é a soma vetorial dos vetores gerados por cada objeto detectado na imagem.

3.3.6. Comportamento MOVE_TO_GOAL

O esquema motor Move_to_Goal gera um campo vetorial de atração para uma localização determinada no sistema de referência global. Os parâmetros utilizados para codificar o esquema motor são apresentados na figura 3.11.

Fora da zona controlada a magnitude de atração é máxima (1,0). Dentro da zona controlada a magnitude decai linearmente de 1,0 a 0,0. O valor zero representa a posição objetivo. O ângulo de direção do vetor resultante segue a reta que une a posição do robô com o alvo, em direção ao alvo.

Figura 3.11. Parâmetros do comportamento MOVE_TO_GOAL.

O vetor resultante é expresso da seguinte forma:

$$V_{mag}(d) = 1 \quad d > r;$$

$$V_{mag}(d) = \frac{1}{r}d \quad d \leq r;$$

$$\Phi_{orient} = \alpha_{objetivo} \text{ (graus).}$$

3.3.7. Comportamentos TANGENCIAL_HORÁRIO E TANGENCIAL_ANTI-HORÁRIO

Esses comportamentos têm como objetivo gerar um campo vetorial tangencial à localização de um objetivo no sistema de coordenadas global. O resultado desse comportamento observado no robô é uma navegação rotativa ao redor do objetivo em sentido horário ou anti-horário.

A magnitude dos campos é constante e a orientação é tangente ao círculo que tem como raio a distância do objetivo ao robô, como mostra a figura 3.12. Assim, as equações que codificam esses campos tangenciais são as seguintes:

TANGENCIAL_HORÁRIO:

$$V_{mag}(d) = 1,0;$$

$$\Phi_{orient} = \alpha_{objetivo} + 90^{\circ}.$$

TANGENCIAL_ANTI-HORÁRIO:

$$V_{mag}(d) = 1,0;$$

$$\Phi_{orient} = \alpha_{objetivo} - 90^{\circ}.$$

Figura 3.12. Parâmetros dos comportamentos (a) TANGENCIAL_HORÁRIO e (b) TANGENCIAL_ANTI-HORÁRIO.

3.4 Testes dos comportamentos abstratos

Para a definição dos comportamentos abstratos foi necessário realizar testes com cada um desses comportamentos para determinar os parâmetros e ganhos necessários para a coordenação dos comportamentos primitivos que os constituem. A seguir apresentam-se os resultados dos testes realizados.

3.4.1. Controle de baixo nível

O movimento do robô se dá ao longo do vetor resultante dos comportamentos ativos. Esse vetor resultante é calculado pela soma vetorial dos vetores dos comportamentos ativos. Antes de realizar a soma vetorial, o vetor de cada comportamento ativo é multiplicado por um ganho.

Após calcular o vetor resultante dos comportamentos ativos, esse vetor deve ser traduzido em comandos de controle que possam ser executados pelo robô. Existem diversas formas de fazer isso dependendo do tipo de controle de baixo nível que o robô suporta. Nesse trabalho o robô simulado se movimenta em modo diferencial, portanto, tem-se a opção de comandar as velocidades de translação e rotação do robô.

O vetor resultante de cada comportamento primitivo é fornecido em relação ao sistema fixo no robô e sua magnitude varia de 0,0 (valor mínimo) a 1,0 (valor máximo). Após o cálculo desses vetores, uma somatória vetorial é realizada com todos eles. O vetor resultante dessa soma representa o comando de controle do comportamento abstrato, e pode ter magnitude maior que 1,0. Nos casos em que a magnitude do vetor resultante do comportamento abstrato seja maior que 1,0, essa magnitude é limitada a 1,0 para manter o robô dentro das velocidades máximas, de translação e rotação, estabelecidas na etapa de projeto.

Para obter as velocidades de controle do robô (figura 3.13) o vetor resultante do comportamento abstrato, é decomposto em suas componentes “x” e “y” no sistema fixo no robô. A componente “x” representa a velocidade de translação e a componente “y” a velocidade de rotação. Finalmente essas componentes são multiplicadas pelas velocidades máximas de translação e de rotação definidas no projeto.

Figura 3.13. Determinação das velocidades de rotação e translação para o controle de baixo nível.

As expressões matemáticas para as velocidades de translação e rotação são dadas por:

$$\text{Velocidade de Translação} = V_{TRANSL,MAX} V_{mag} \cos(\Phi_{orient}) \text{ (m/s);}$$

$$\text{Velocidade de Rotação} = V_{ROT,MAX} V_{mag} \sin(\Phi_{orient}) \text{ (graus/s).}$$

Essas duas velocidades são enviadas concomitantemente ao sistema de controle de baixo nível do robô.

A forma mais utilizada de controle de baixo nível em aplicações de robôs móveis é executar o vetor resultante em dois tempos. Primeiro o robô acerta o ângulo de orientação e depois executa o comando de translação. Após o robô executar o comando de translação o ciclo de controle inicia novamente. Nesse trabalho executa-se o controle da velocidade de translação e o ângulo de orientação simultaneamente. A vantagem desse método utilizado é que o programa de controle não precisa esperar que o robô acerte o ângulo de orientação para continuar seu ciclo. Com este método de controle de baixo nível a frequência com que o programa de controle obtém os dados dos sensores e gera as ações de movimento do robô é maior e, assim, o robô pode reagir ao ambiente mais rapidamente.

Na figura 3.14 observa-se como o robô se comporta dentro de um campo vetorial uniforme, quando está orientado inicialmente a 90° desse campo de velocidade.

Figura 3.14. Comportamento do Robô dentro de um campo vetorial uniforme.

Nos testes realizados verificou-se que para alguns comportamentos é preciso acertar o ângulo de orientação com o ângulo do campo vetorial antes de executar o movimento de translação do robô. Isso permite um melhor controle da orientação final do robô ao chegar ao seu objetivo. Desenvolveram-se duas funções vetoriais que fazem o robô girar no seu próprio eixo: Giro Horário e Giro Anti-Horário. As expressões dessas funções são as seguintes:

Giro Horário:

$$V_{mag} = 0,5;$$

$$\Phi_{orient.} = -90^\circ.$$

Giro Anti-Horário:

$$V_{mag} = 0,5;$$

$$\Phi_{orient.} = 90^\circ.$$

3.4.2. Comportamento EXPLORAR PROCURANDO LIXO

O primeiro comportamento a ser testado é o EXPLORAR. A finalidade dos testes é acertar os parâmetros dos comportamentos primitivos que constituem o comportamento EXPLORAR para fazer com que o robô navegue pelo ambiente, sem ficar preso em uma determinada região do espaço e sem colidir com obstáculos. Os parâmetros determinados a partir desses testes estão apresentados na tabela 3.3.

Em razão de o robô poder tanto avançar como retroceder, dependendo do ângulo de orientação do vetor resultante, é necessário impor uma restrição no ângulo de orientação gerado pelo comportamento primitivo NOISE.

Tabela 3.3. Parâmetros determinados para o comportamento abstrato Explorar.

COMPORTAMENTO ABSTRATO	COMPORTAMENTOS PRIMITIVOS	PARÂMETROS
EXPLORAR PROCURANDO LIXO	AVOID_OBSTACLE	Ganho=1,0
		S=1,5 m
		M=0,45 m
	NOISE	insistência=75 ciclos
		Ganho=0,5
		$-90^{\circ} \leq \Phi_{orient} \leq 90^{\circ}$

Com a imposição do ângulo de orientação estar sempre entre -90° e 90° o robô é forçado a nunca navegar em retrocesso, de forma que o sistema de visão esteja sempre orientado no sentido de movimentação do robô.

O primeiro teste consistiu em fazer o robô navegar em um ambiente sem obstáculos. O ambiente é uma oficina virtual com dimensões de 5m X 5m. A figura 3.15 mostra a trajetória do robô durante esse primeiro teste.

Observa-se que o robô consegue explorar todo o ambiente. Porém, o tempo que o robô leva para navegar por todo o ambiente é imprevisível em razão de que o comportamento primitivo NOISE gera vetores aleatoriamente e não possui memória das posições que o robô visitou anteriormente. Assim, o vetor resultante do comportamento primitivo NOISE pode fazer o robô voltar a uma região do ambiente já explorada, mesmo tendo regiões do ambiente ainda por explorar. Esse comportamento foi considerado satisfatório mesmo com esse problema em razão de que o tempo de exploração nesse trabalho não é uma restrição de projeto.

O segundo teste consistiu em testar o comportamento no mesmo ambiente com obstáculos. Na figura 3.16 observa-se o resultado desse teste. O robô consegue navegar eficientemente por todo o ambiente sem colidir com os obstáculos presentes.

1

2

3

4

5

6

Figura 3.15. Trajetória do Robô no primeiro teste do comportamento abstrato EXPLORAR

Figura 3.16. Trajetória do robô no segundo teste do comportamento abstrato EXPLORAR.

3.4.3. Comportamento COLETAR LATA

O objetivo do comportamento COLETAR LATA é gerar um campo vetorial que faz com que o robô se aproxime da lata até que a mesma fique ao alcance do manipulador.

O cálculo do vetor resultante do comportamento primitivo ATRAÇÃO_VERMELHO é feito com as informações obtidas da interface Blobfinder. Os valores para a área de segurança, obtidos mediante testes experimentais simulados, variam entre 1500 e 2500 pixels, dependendo da posição da lata. Essa área de segurança ia ser utilizada para parar o robô quando a lata estiver ao alcance do manipulador. Porém, para obter um melhor desempenho desse comportamento, utilizou-se como indicador para o fim do comportamento o detector infravermelho encontrado na garra do manipulador. Determinou-se que a magnitude mínima do vetor de velocidade resultante é 0,1 quando a área de cor vermelha percebida é de 900 pixels. Após a magnitude do vetor resultante atingir esse valor mínimo, ela

permaneça nesse valor até que a lata seja detectada pelo primeiro sensor infravermelho da garra do manipulador.

O cálculo do ângulo de orientação do vetor resultante é feito mediante a posição do centróide da área de cor vermelha detectada. O simulador Stage fornece essa posição como se fosse uma câmera adquirindo imagens em um ambiente de duas dimensões (HARALICK; SHAPIRO 1993). Na figura 3.17 observa-se um exemplo de como o simulador Stage fornece os dados da posição do centróide.

Figura 3.17. Dados da posição do centróide fornecidos pelo simulador Stage.

Assim, quando o robô se aproxima da lata, o valor da componente “x” do centróide varia, mas a componente “y” permanece em um valor constante, que no caso é 30 pixels. O ângulo de orientação do vetor resultante, que corresponde ao ângulo da posição da lata com respeito ao robô é calculado mediante a seguinte equação:

$$\Phi_{orient} = \arctg \left(\frac{40 - x}{30} \right).$$

Para controlar melhor o ângulo de aproximação do robô à lata o comportamento é executado em dois movimentos:

1. Girar o robô ate que o ângulo de orientação do vetor resultante encontre-se entre $+10^\circ$ e -10° graus;
2. Aplicar o campo vetorial ao robô.

Os parâmetros determinados para a coordenação dos comportamentos primitivos que constituem o comportamento COLETAR LATA estão listados na tabela 3.4.

Tabela 3.4. Parâmetros determinados para o comportamento abstrato COLETAR LATA.

COMPORTAMENTO ABSTRATO	COMPORTAMENTOS PRIMITIVOS	PARÂMETROS
COLETAR LATA	AVOID_OBSTACLE	Ganho=0,5
		S=0,7 mt
		M=0,45 mt
	NOISE	insistência=75 ciclos
		Ganho=0,5
		$-90^\circ \leq \Phi_{\text{orient}} \leq 90^\circ$
	ATRAÇÃO VERMELHO	Ganho=1,0
	GIRO HORARIO	Ganho=1,0
	GIRO ANTI-HORARIO	Ganho=1,0

No comportamento COLETAR LATA o parâmetro S, que determina a zona controlada do comportamento primitivo AVOID_OBSTACLE, é igual a 0,7 m., enquanto que para o comportamento EXPLORAR esse parâmetro S é igual a 1,5 m. Esse valor de S significa que as latas que o robô pode coletar devem ficar a uma distância maior do que 0,7 m. de qualquer outro obstáculo presente no ambiente. A ação dos comportamentos primitivos AVOID_OBSTACLE e NOISE no momento em que o robô se aproxima de uma lata não permite ao robô atingir o ângulo certo para

colocar a lata dentro do manipulador se outro obstáculo estiver muito perto da lata a ser coletada.

Na figura 3.18 pode se observar a ação do comportamento COLETAR LATA. Com os parâmetros determinados o robô consegue se aproximar da lata detectada com o ângulo certo para colocar a mesma no centro da garra do manipulador.

Figura 3.18: Ação do comportamento de COLETAR LATA: (a) detecção da Lata; (b) acertando ângulo de orientação do robô; e (c) robô atraído pelo campo de ATRAÇÃO_VERMELHO.

3.4.4. Comportamento PEGAR LATA

O comportamento PEGAR LATA tem a função de colocar a lata na posição adequada para o manipulador segurá-la. Esse comportamento utiliza a interface

Gripper para obter a posição da lata e o estado do manipulador em cada instante de tempo. Além disso, determina que ações devem ser realizadas para deixar a lata no meio da garra do manipulador. Os possíveis estados do manipulador e as ações correspondentes encontram-se listados na tabela 3.5.

Tabela 3.5. Correspondência dos estados do manipulador e as ações efetuadas pelo comportamento PEGAR LATA.

ESTADOS DO MANIPULADOR		AÇÕES
BEAM	STATE	
0	1 (OPEN)	-----
0	2 (CLOSE)	Abrir Manipulador
0	3 (MOVING)	-----
0	4 (ERROR)	Imprimir erro em janela
1	1 (OPEN)	Ativar comportamento Avançar
1	2 (CLOSE)	Abrir Manipulador. Novo intento
1	3 (MOVING)	-----
1	4 (ERROR)	Imprimir erro em janela
2	1 (OPEN)	Ativar comportamento Retroceder
2	2 (CLOSE)	Abrir Manipulador. Novo intento
2	3 (MOVING)	-----
2	4 (ERROR)	Imprimir erro em janela
3	1 (OPEN)	Fechar Manipulador (lata na posição correta)
3	2 (CLOSE)	Fim do comportamento
3	3 (MOVING)	-----
3	4 (ERROR)	Imprimir erro em janela

Nesse comportamento, além de outros, são utilizados os comportamentos primitivos AVANÇAR e RETROCEDER para colocar a lata no meio da garra do manipulador. Os parâmetros determinados para a coordenação dos comportamentos primitivos são listados na tabela 3.6.

Tabela 3.6. Parâmetros determinados para o comportamento abstrato PEGAR LATA.

COMPORTAMENTO ABSTRATO	COMPORTAMENTOS PRIMITIVOS	PARÂMETROS
PEGAR LATA	AVANÇAR	Ganho=0,1
	RETROCEDER	Ganho=0,1

Da tabela 3.6 pode se observar que os comportamentos primitivos AVOID OBSTACLE e NOISE estão desativados. Isso ocorre em razão de que assume-se que não existem objetos em movimento no ambiente que possam colidir com o robô durante a execução do comportamento PEGAR LATA.

Na figura 3.19 observa-se o robô executando o comportamento PEGAR LATA.

Figura 3.19. Robô controlado pelo comportamento PEGAR LATA. (a) lata interfere no primeiro sensor infravermelho do manipulador; (b) robô se aproxima da lata até posicioná-la no meio da garra do manipulador; (c) robô segurando a lata.

3.4.5. Comportamento IR AO DEPÓSITO

O comportamento IR AO DEPÓSITO encarrega-se de levar o robô ao depósito para deixar a lata coletada. Para executar essa tarefa utiliza-se um comportamento primitivo denominado “Docking” (ARKIN; MURPHY 1990). A figura 3.20 mostra o campo vetorial resultante do comportamento “Docking”.

Figura 3.20. Campo vetorial do comportamento primitivo “Docking”.

O comportamento primitivo “Docking” permite que o robô chegue ao depósito de latas na posição e orientação corretas para o seu ingresso no mesmo.

O comportamento IR AO DEPÓSITO desenvolvido nesse trabalho é composto pelos seguintes comportamentos primitivos:

- Move_To_Goal;
- Tangencial_Horário;
- Tangencial_Anti-Horário;
- Avoid_Obstacle;
- Noise;
- Avoid_Red_Obstacle.

Observa-se que a navegação do robô é efetuada pelo comportamento primitivo Docking. Esse comportamento por sua vez é composto de uma seqüência de controle que ativa os comportamentos primitivos Move_to_Goal, Tangencial_Horario e Tangencial_Anti-Horario, dependendo da posição em que o robô se encontra no ambiente. Em razão de que o esquema perceptivo desse comportamento utiliza unicamente informação da odometria para gerar os campos vetoriais, o robô deve conhecer sua posição relativa à localização do depósito. A aplicação foi desenvolvida considerando-se que a posição inicial do robô é a entrada do depósito e que a orientação do robô na entrada do depósito é tal que o sistema de coordenadas odométrico coincide com o sistema de coordenadas global do ambiente. Assim, a posição final desejada no comportamento “Docking” é o ponto do espaço com coordenadas (0,0) e orientação 0° . As zonas de ativação dos comportamentos MOVE_TO_GOAL, TANGENCIAL_HORÁRIO e TANGENCIAL_ANTI-HORÁRIO são mostradas na figura 3.21.

Para evitar colidir com obstáculos no ambiente, o comportamento IR AO DEPÓSITO utiliza AVOID_OBSTACLE e NOISE. Porém, no caminho de volta ao depósito, o robô pode encontrar latas ainda não coletadas. Para evitar colidir com elas, o comportamento primitivo AVOID_RED_OBSTACLE é também ativado.

Os parâmetros utilizados no desenvolvimento do comportamento primitivo “Docking” são mostrados na tabela 3.7.

O valor A (Área de Segurança Máxima) igual a 300 pixels foi determinado de forma experimental e fornece ao robô o espaço suficiente para evitar colidir com outras latas no ambiente.

A figura 3.22 mostra o robô controlado mediante o comportamento IR AO DEPÓSITO.

Tabela 3.7. Parâmetros determinados para o comportamento abstrato IR AO DEPÓSITO

COMPORTAMENTO ABSTRATO	COMPORTAMENTOS PRIMITIVOS	PARÂMETROS
IR AO DEPÓSITO	AVOID_OBSTACLE	Ganho=1,0
		S=1,5 mt
		M=0,45 mt
	NOISE	i=insistência=75 ciclos
		Ganho=0,5
		$-90^{\circ} \leq \Phi_{orient} \leq 90^{\circ}$
	MOVE_TO_GOAL	Ganho=1,0
		R=1,5
	TANGENCIAL_HORÁRIO	Ganho=1,0
		Distancia de ação=4 mt
	TANGENCIAL_ANTI-HORÁRIO	Ganho=1,0
		Distancia de ação=4 mt
	AVOID_RED_OBSTACLE	Ganho=1,0
		A=300 pixels

Figura 3.21. Campos vetoriais presentes no comportamento primitivo "Docking".

(a)

(b)

(c)

Figura 3.22. Robô controlado pelo comportamento IR AO DEPÓSITO. (a) Comportamento IR AO DEPÓSITO com as direções dos campos vetoriais do comportamento primitivo "Docking"; (b) e (c) Ação dos comportamentos primitivos `AVOID_RED_OBSTACLE` e `AVOID-OBSTACLE` inseridos no comportamento IR AO DEPÓSITO.

3.4.6. Comportamento DEIXAR LATA

Este comportamento segue uma sequência de ações com o propósito de deixar a lata dentro do depósito. Em razão do reconhecimento de latas ser feito mediante a cor vermelha, as latas já encontradas devem ficar fora do alcance do sistema de visão para que não interfiram com a coleta das latas faltantes. Assim, o robô deve ingressar no depósito, deixar a lata encontrada e sair do depósito para continuar a coleta. Observa-se que as latas colocadas no depósito ficam fora do alcance do sistema de visão.

O autômato finito que controla a sequência de ações efetuada pelo comportamento DEIXAR LATA é apresentado na figura 3.23.

Figura 3.23. Autômato finito correspondente ao comportamento DEIXAR LATA.

O estado “acertar ângulo de entrada” é um comportamento que gira o robô no sentido horário ou anti-horário de modo a fazer com que o ângulo de orientação do

robô seja 0°. Esse comportamento utiliza as funções GIRO_HORÁRIO ou GIRO_ANTI-HORÁRIO com os ganhos indicados na tabela 3.8.

Os parâmetros determinados para o comportamento DEIXAR LATA estão listados na tabela 3.8.

Tabela 3.8. Parâmetros determinados para o comportamento abstrato DEIXAR LATA.

COMPORTAMENTO ABSTRATO	COMPORTAMENTOS PRIMITIVOS	PARÂMETROS
DEIXAR LATA	AVANÇAR	Ganho=0,1
	RETROCEDER	Ganho=0,1
	GIRO_HORÁRIO	Ganho=0,5
	GIRO_ANTI-HORÁRIO	Ganho=0,5

O teste desse comportamento consistiu em colocar o robô na entrada do depósito com uma lata no manipulador e ativar o comportamento. A figura 3.24 mostra o resultado obtido.

Figura 3.24. Resultado do teste do comportamento DEIXAR LATA. (a) acertando ângulo de entrada; (b) AVANÇAR; (c) GIRO_ANTI-HORARIO; (d) AVANÇAR; (e) ABRIR MANIPULADOR; (f) RETROCEDER; (g) GIRO_ANTI-HORARIO; (h) AVANÇAR e ativação do comportamento EXPLORAR.

3.5 Seqüência de controle dos comportamentos abstratos

O programa de controle da aplicação encarrega-se de determinar e controlar a seqüência em que os comportamentos abstractos trabalham.

A seqüência de controle determinada para executar a missão do robô apresenta-se na figura 3.25.

Figura 3.25. Seqüência de controle da tarefa de coleta de lixo.

Podemos observar que os comportamentos abstratos correspondem aos estados do autômato finito. Além dos estados comportamentais, temos agregado um estado de INICIO e um estado denominado INTENTOS.

Como foi dito na seção anterior, a posição inicial do robô é a entrada do depósito, que permite igualar o sistema de coordenadas global com o sistema de coordenadas odométrico. O estado de INICIO faz o robô girar 180° em forma anti-

horária o que permite o robô iniciar a exploração. Na figura 3.26 observa-se este comportamento.

Figura 3.26. Ação do estado de INICIO

O estado INTENTOS encarrega-se de gerenciar os intentos do robô de coletar uma lata e levar-la no depósito. Em caso de que a lata escorregue do manipulador, o robô retrocede abrindo o manipulador para tentar coletar a lata. Esta operação é repetida 3 vezes. Se não consegue coletar-la em 3 intentos, o robô

passa ao estado de INICIO; girando 180° e iniciando a exploração novamente. A figura 3.27 mostra esta operação.

O autômato finito apresentado não amostra os comportamentos AVOID_OBSTACLE e NOISE, porem, eles estão ativos sempre, correspondendo aos comportamentos abstratos inibir suas ações em caso seja necessário.

Os ativadores dos comportamentos abstratos são determinados mediante o sistema de visão, os sensores infravermelhos do manipulador e o sistema odometrico. A tabela 3.9 mostra os valores definidos como ativadores dos comportamentos.

Tabela 3.9 Ativadores da seqüência de controle dos comportamentos abstratos.

ATIVADOR	SIGNIFICADO	VARIAVEIS	INTERFACE
MAN. VAZIO	Manipulador Vazio	Beams=0	Gripper
		State=1 OR State=2	
VERMELHO	Um ou mais objetos vermelhos detectados	Blob count =N	Blobfinder
OBJETO NO MANIPULADOR	Objeto interfere o primeiro sensor infravermelho	Beams= 1	Gripper
MAN. CHEIO	Objeto interfere os dois sensores infravermelhos do manipulador	Beam=3	Gripper
MAN. FECHADO	Manipulado Fechado	State=2	Gripper
EM DEPÓSITO	Robô na posição (0,0) do sistema de coordenadas odometrico (sistema de coordenadas global)	X=0 Y=0 Erro=0,04 mt.	Position2d
INTENTOS	Variável do programa de controle para gerenciar os intentos de coletar a lata encontrada	INTENTOS=3

Figura 3.27. Resultado do teste do esquema de INTENTOS. (a) robô carregando uma lata; (b) o robô perde a lata; (c) e (d) robô recua para tentar novamente; (e) e (f) após 3 intentos o robô gira 180°.

3.6 Teste final da aplicação

Após de determinar a seqüência de controle foram realizados testes da aplicação para verificar seu comportamento.

Para os testes foi elaborado um cenário simulado aonde o robô é obrigado a passar por vários tipos de configurações de obstáculos para poder coletar as latas.

O cenário simula uma sala retangular de 8 x 8 m. onde foram colocados diversos obstáculos. As latas de refrigerante foram representadas mediante blocos retangulares de cor vermelho de 0,065 x 0,065 m. Na figura 3.28 pode observar-se o cenário utilizado.

Diversos testes foram realizados com o objetivo de acertar os parâmetros utilizados em cada comportamento abstrato. Como resultados alguns desses parâmetros mudaram para melhorar a eficiência do robô. Na tabela 3.10 apresentam-se os parâmetros utilizados para os testes.

As velocidades máximas de translação e rotação utilizadas nos testes foram:

Velocidade de Translação = $V_{MAX} TRANSL = 0,50 \text{ m/seg}$;

Velocidade de Rotação = $V_{MAX} ROT = 20 \text{ graus/seg}$.

Figura 3.28. Cenário de simulação

Tabela 3.10. Parâmetros dos comportamentos abstratos utilizados nas provas.

COMPORTAMENTO ABSTRATO	COMPORTAMENTOS PRIMITIVOS	PARÂMETROS
EXPLORAR PROCURANDO LIXO	AVOID_OBSTACLE	Ganho=1,0
		S=1,0 mt
		M=0,45 mt
	NOISE	i=insistência=75 ciclos Ganho=0,5 $-90^{\circ} \leq \Phi_{orient} \leq 90^{\circ}$
COLETAR LATA	AVOID_OBSTACLE	Ganho=0,5 S=0,5 mt M=0,45 mt
		i=insistência=75 ciclos Ganho=0,5 $-90^{\circ} \leq \Phi_{orient} \leq 90^{\circ}$
		Ganho=1,0 Distancia Máxima do Sistema de Visão = 3 mt
	GIRO HORARIO	Ganho=1,0
	GIRO ANTI-HORARIO	Ganho=1,0
PEGAR LATA	AVANÇAR	Ganho=0,1
	RETROCEDER	Ganho=0,5
IR AO DEPÓSITO	AVOID_OBSTACLE	Ganho=1,0 fora da zona controlada Ganho=0,0 dentro da zona controlada a 1,5 mt do ponto (0,0) S=1,0 mt M=0,45 mt
		i=insistência=75 ciclos Ganho=0,5 fora da zona controlada Ganho=0,0 dentro da zona controlada a 1,5 mt do ponto (0,0) $-90^{\circ} \leq \Phi_{orient} \leq 90^{\circ}$
		Ganho=1,0 R=1,5
		Ganho=1,0 Distancia de ação=2,5mt
	TANGENCIAL_HORÁRIO	Ganho=1,0 Distancia de ação=2,5mt
	TANGENCIAL_ANTI-HORÁRIO	Ganho=1,0 Distancia de ação=2,5mt
	AVOID_RED_OBSTACLE	Ganho=1,0 A=300 pixels
DEIXAR LATA	AVANÇAR	Ganho=0,1
	RETROCEDER	Ganho=0,1
	GIRO HORÁRIO	Ganho=0,5
	GIRO_ANTI-HORÁRIO	Ganho=0,5

A figura 3.29 mostra uma sequência de imagens correspondentes a um dos testes realizados. No teste, o robô explora o ambiente em procura de 3 latas colocadas em diferentes posições do ambiente. Neste teste observa-se as principais propriedades dos algoritmos desenvolvidos, navegação segura, detectar e coletar as latas, volta ao depósito evitando os obstáculos do entorno e as latas encontradas na

trajetória do robô, ingresso ao depósito para deixar a lata no fim do depósito e reinício do ciclo de controle com o que se consegue encontrar todas as latas espalhadas pelo ambiente.

(a)

(b)

Figura 3.29. Teste dos algoritmos de controle: (a) Posição Inicial; (b) Ação do estado INICIO.

(c)

(d)

Figura 3.29. cont. (c) Exploração do ambiente e coleta da primeira lata; (d) Volta ao depósito evitando colidir com os obstáculos do ambiente e latas na trajetória do robô.

(e)

(f)

Figura 3.29. cont. (e) Exploração do ambiente e coleta da segunda lata; (f) Volta ao depósito.

(g)

(h)

Figura 3.29 cont. (g) Exploração do ambiente e coleta da terceira lata; (h) Volta ao depósito e fim do teste.

4. Conclusão

Este trabalho apresentou o desenvolvimento dos algoritmos de controle de um robô móvel para realizar uma tarefa de coleta de lixo. A solução proposta para esta aplicação robótica foi utilizar uma arquitetura baseada em comportamentos reativos.

A arquitetura escolhida foi a metodologia “Motor-Schema”, que se mostrou eficiente no desenvolvimento e integração de comportamentos reativos. Os testes de simulação realizados mostraram que a solução apresentada é adequada para resolver a aplicação robótica de coleta de lixo. O robô consegue explorar um ambiente desconhecido de forma segura, detectar e coletar objetos espalhados pelo ambiente e levá-los a um depósito determinado.

Os testes mostraram também as limitações do uso de uma solução comportamental reativa aplicada a essa tarefa. O comportamento EXPLORAR PROCURANDO LIXO consegue fazer o robô explorar um ambiente desconhecido, porém o tempo que demora em explorar todo o ambiente e detectar algum objeto é muito grande. Nos testes o tempo médio que o robô leva para encontrar uma lata é de cerca de 6 min. Se o objetivo for encontrar 8 latas, o tempo necessário pode chegar a ser cerca de 50 min. Isso ocorre em razão da política utilizada para explorar o ambiente. Uma solução possível para resolver este problema seria dotar o comportamento de memória de curta duração. Assim, se o robô encontra um grupo de latas no ambiente, poderia pegar a mais próxima dele e após deixá-la no depósito, voltar ao ponto onde encontrou o grupo de latas. Porém, os comportamentos desenvolvidos são puramente reativos e, assim, não possuem memória.

O comportamento IR AO DEPÓSITO também mostrou ser muito eficiente para guiar ao robô à localização do depósito e com a orientação correta. Porém é um comportamento baseado exclusivamente em informação odométrica e seu funcionamento está condicionado aos erros característicos dos sistemas de odometria.

Outra limitação encontrada na solução proposta foi o tempo gasto para determinar experimentalmente os parâmetros e os ganhos de coordenação correspondente a cada comportamento primitivo. Como a quantidade de comportamentos concorrentes existentes nos comportamentos abstratos desenvolvidos pode ser grande, acertar o valor dos parâmetros requer um tempo longo. Uma solução para este problema poderia ser utilizar algum método de aprendizagem de máquina para determinar os parâmetros e ganhos de coordenação corretos.

Por último, o passo seguinte imediato desse projeto é testar os algoritmos em um robô real. Os algoritmos desenvolvidos foram testados em um ambiente de simulação que não considera os erros nas medidas dos sensores. É imprescindível estudar o funcionamento da solução apresentada na presença de erros de medida, para verificar sua efetividade na tarefa de coleta de lixo real. A escolha do pacote de simulação Player/Stage para desenvolver a aplicação foi feita visando esta etapa futura, ou seja, a implementação dos algoritmos propostos em um robô real.

5. Referências bibliográficas

- AAAI, 1993.** AAAI - AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE (Estados Unidos) (Org.). **Registration Brochure of the AAAI-93 Eleventh National Conference on Artificial Intelligence.** Disponível em: <<http://www.aaai.org/Conferences/AAAI/aaai93.php>>. Acesso em: 15 July 1993.
- AAAI, 1994.** AAAI - AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE (Estados Unidos) (Org.). **Program & Exhibit Guide of the AAAI-94 Twelfth National Conference on Artificial Intelligence.** Disponível em: <<http://www.aaai.org/Conferences/AAAI/aaai94.php>>. Acesso em: 31 July 1994.
- AAAI, 1996.** AAAI - AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE (Estados Unidos) (Org.). **Program & Exhibit Guide of the AAAI-96 Thirteenth National Conference on Artificial Intelligence.** Disponível em: <<http://www.aaai.org/Conferences/AAAI/aaai96.php>>. Acesso em: 4 Aug. 1996.
- AAAI, 1997.** AAAI - AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE (Estados Unidos) (Org.). **Program & Exhibit Guide of the AAAI-97 Fourteenth National Conference on Artificial Intelligence.** Disponível em: <<http://www.aaai.org/Conferences/AAAI/aaai97.php>>. Acesso em: 27 Jul. 1997.
- AAAI, 1998.** AAAI - AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE (Estados Unidos) (Org.). **Program & Exhibit Guide of the AAAI-98 Fifteenth National Conference on Artificial Intelligence.** Disponível em: <<http://www.aaai.org/Conferences/AAAI/aaai97.php>>. Acesso em: 27 Jul. 1997.
- MOBILEROBOTS, 2006.** MOBILEROBOTS. Pioneer 3 Operation Manual. Disponível em: <http://www.mobilerobots.com/>.
- AGRE; CHAPMAN, 1987.** AGRE, P. E.; CHAPMAN, D. Pengi: an Implementation of a Theory of Activity. In: **Proceedings of the Sixth National Conference on Artificial Intelligence**, Seattle, Washington, 1987. v. 2, p. 268 - 271.
- ARKIN, 1986.** ARKIN, Ronald C.. Path Planning for a Vision-Based Autonomous Robot. In: **Proceedings of the SPIE Conference on Mobile Robots**, Cambridge, Massachusetts: SPIE, 1986. p. 240 - 249.
- ARKIN, 1987.** ARKIN, Ronald C.. **Motor Schema Based Navigation for a Mobile robot: An Approach to Programming by Behavior.** Disponível em: <http://www.cc.gatech.edu/ai/robot-lab/publications.html>.
- ARKIN, 1989.** ARKIN, Ronald C.. **Motor Schema-Based Mobile Robot Navigation.** Disponível em: <<http://www.cc.gatech.edu/ai/robot-lab/publications.html>>.
- ARKIN, 1998.** ARKIN, Ronald C.. **Behavior-Based Robotics.** 3.ed. Massachusetts: MIT Press, 491 p.

- ARKIN; MURPHY 1990.** Arkin, Ronald C.; Murphy, Robin R.. Autonomous Navigations in a Manufacturing Environment. **IEEE Transactions On Robotics and Automation**, v. 6, n. 4, pp.445-454, Aug. 1990
- BORENSTEIN et al. 1997.** Borenstein, J.; Everett,H. R.; Feng, L.; Wehe, D.. **Mobile Robot Positioning: Sensors and Techniques**. Disponível em: < <http://www-personal.umich.edu/~johannb/mypapers.htm> >. Acesso em: 15 Nov. 2006.
- BROOKS, 1986.** BROOKS, Rodney A.. A Robust Layered Control System for a Mobile Robot. **IEEE Journal Of Robotics And Automation**, pp. 14-23, march.
- BROOKS, 1990.** BROOKS, Rodney A.. Elephants don't play chess. In: MAES, Pattie (Ed.). **Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back**. Cambridge, Massachusetts: MIT/Elsevier, pp. 3-15.
- BROOKS, 1991a.** BROOKS, Rodney A.. Intelligence Without Reason. In: **Proceedings of 12th Int. Conf. on Artificial Intelligence**, Sydney, Australia: IJCAI, v. 1, p. 569 - 595.
- BROOKS, 1991b.** BROOKS, Rodney A.. Intelligence Without Representation. **Artificial Intelligence 47**, Amsterdam, Holanda, pp. 139-159, Jan.
- BROOKS, 1991c.** BROOKS, Rodney A.. **New Approaches to Robotics**. Science, v. 253, n. 13, p.1227-1232, set.
- BROOKS e CONNELL, 1986.** BROOKS, Rodney A.; CONNELL, J. H.. Asynchronous Distributed Control System for a Mobile Robot. In: **Proceedings of SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering**, Cambridge, Massachusetts: SPIE, pp. 77 - 84.
- CHATILA e LAUMOND, 1985.** CHATILA, Raja; LAUMOND, Jean-paul. Position Referencing and Consistent World Modeling for Mobile Robots. In: **Proceedings of the IEEE International Conference on Robotics and Automation**, Cambridge, Massachusetts: IEEE, v. 2, pp. 138 - 145.
- CHOSSET et al., 2005.** CHOSSET, Howie; LYNCH, Kevin; HUTCHINSON Seth; KANTOR, George; BURGARD Wolfram; KAVRAKI, Lydia; THRUN, Sebastian. **Principles of Robot Motion: Theory, Algorithms, and Implementations**. Cambridge, Massachusetts: Mit Press, 603 p.
- CONNELL, 1990.** CONNELL, Jonatham H.. **Minimalist Mobile Robotics : A colony-style architecture for an artificial creature**. 1th ed. Boston: Academic Press, 175 p.
- FLYNN e BROOKS, 1988.** FLYNN, Anita M.; BROOKS, Rodney A.. MIT Mobile Robots - What's Next? In: **Proceedings of the 1988 IEEE International Conference on Robotics and Automation**, Philadelphia, Pa: IEEE, p. 611 - 617.

- GAT, 1991.** GAT, Erann. **Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.** Tese (Doutorado) - Virginia Polytechnic Institute And State University, Blacksburg, Virginia.
- GEORGEFF; LANSKY, 1987.** GEORGEFF, Michael P.; LANSKY, Amy L.. Reactive Reasoning and Planning. In: **Proceedings of the Sixth National Conference on Artificial Intelligence**, Seattle, Washington: AAAI, pp. 677 - 682.
- GERKEY et al. 2001.** Brian P Gerkey; Richard T. Vaughan; Kasper Stoy; Andrew Howard; Gaurav S. Sukhatmé; Maja J. Mataric . **Most Valuable Player: A Robot Device Server for Distributed Control.** Disponível em: <<http://ai.stanford.edu/~gerkey/research/papers/index.html>>. Acesso em: 5 Dec. 2006.
- GERKEY, 2003.** Brian P Gerkey; Richard T. Vaughan; Andrew Howard. The Player/Stage Project: Tools for Multi-Robot and distributed Sensor System. In: **Proc. 11th International Conference on Advanced Robotics (ICAR)**, Coimbra, Portugal, p. 317 - 323.
- HARALICK; SHAPIRO, 1993.** HARALICK, Robert; SHAPIRO, Linda. **Computer and Robot Vision.** Massachusetts: Addison-Wesley, 1993. 630 p.
- HINKLE et al. 1995.** HINKLE, David; KORTENKAMP, David; MILLER, David. **The 1995 IJCAI Robot Competition and Exhibition.** Disponível em: <http://www.traclabs.com/~korten/publications/ai_mag96.pdf>. Acesso em: 5 Dec. 1995.
- KHATIB, 1985.** KHATIB, O.. Real-time obstacle Avoidance for Manipulators and Mobile Robots. In: **Proceedings of the IEEE International Conference on Robotics and Automation**, Cambridge, Massachusetts: IEEE, p. 338 - 349.
- KOENIG; HOWARD, 2004.** Nathan Koenig; Andrew Howard. Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. In: **Proc. of 2004 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS, 2004)**, Sendai, Japan, v.4,p.2149-2154.
- KROGH, 1984.** KROGH, B. H.. A generalized Potencial Field Approach to Obstacle Avoidance Control. In: **ROBOTICS RESEARCH CONFERENCE**, 1., 1984, Bethlehem, Pennsylvania. **Proceedings of International Robotics Research Conference.** Bethlehem, Pennsylvania, 1984.
- MAES, 1989.** MAES, Pattie. The Dynamics of Action Selection. In: **Proceedings of the International Joint Conference on Artificial Intelligence**, Morgan Kaufmann, Detroit: IEEE, p. 991 - 997.
- MATARIC, 1992a.** MATARIC, Maja J.. **Behavior-Based Control: Main Properties and Implications.** Disponível em: <<http://www-robotics.usc.edu/%7Emaja/publications.html>>. Acesso em: 20 nov. 2006.

- MATARIC, 1992b.** MATARIC, Maja. J.. Integration of Representation into Goal-Driven Behavior-Based Robots. **IEEE Transactions On Robotics And Automation**, v. 8, n. 3, p.304-312, jun.
- MATARIC, 1999.** MATARIC, Maja J.. Behavior-Based Robotics. In: WILSON, Robert A.; KEIL, Frank C. (Eds.). **The MIT Encyclopedia of the Cognitive Sciences**: Mit Press, p. 74-77.
- MATARIC, 2002.** MATARIC, Maja J.. **Situated Robotics**. Disponível em: <<http://www-robotics.usc.edu/%7Emaja/publications.html>>. Acesso em: 20 nov. 2006.
- MATARIC; JONES 2005.** MATARIC, Maja J.; JONES, Chris. **Behavior-Based Coordination in Multi-Robot Systems**. Disponível em: <<http://www-robotics.usc.edu/%7Emaja/publications.html>>. Acesso em: 25 nov. 2006.
- MILLER et al., 1992.** MILLER, David P.; DESAI, Rajiv S.; GAT, Erann; IVLEV, Robert; LOCH, John. Reactive Navigation Through Rough Terrein: Experimental Results. In: **Proceedings of the Tenth National Conference on Artificial Intelligence**, San Jose, California: AAAI, p. 823 - 828.
- MORAVEC; ELFES, 1985.** Moravec, Hans P.; Elfes, Alberto. High Resolution Maps from wide Angle Sonar. In: **Proc. of 1985 IEEE International Conference on Robotics and Automation**. v.2,pp.116-121
- MURPHY, 2000.** MURPHY, Robin R. **Introduction to AI Robotics**. Massachusetts: MIT Press, 466 p.
- PAGOAGA, 2001.** PAGOAGA, Aistol A. **Agent-Based Control Architecture for a Mobile Robot**. Disponível em: http://www.mech.kuleuven.be/mlr/publications/default_en.phtml
- STEELS, 1994.** STEELS, L.. **The Artificial Life Roots of Artificial Intelligence**. **Artificial Life Journal**, Cambridge, Massachusetts, p. 89-125. mar.

6. ANEXOS

1. Especificações da plataforma pionner 2 AT e do manipulador

PLATAFORMA PIONNER 2 AT	
CARACTERÍSTICAS	ROBÔ PIONNER 3-AT
Length	50cm
Width	49cm
Height (body)	26cm
Body clearance	8cm
Weight (with min. battery capacity)	12kg
Payload of base platform with included battery (other drive ratios available on large orders)	Painted concrete: 20kg Asphalt: 7kg Grass / dirt: 13.5kg
POWER	
Battery	12V sealed, lead-acid
Charge	252 watt-hr
Run time, base platform	4-8 hrs
Recharge time, standard charger	12 hrs
Recharge time, 1 high-capacity charger	2.4 hrs
MOBILITY	
Drive	4-wheel drive
Dr. Wheel diam.	21.5cm
Dr. Wheel width	8.8cm
Steering	Skid-steer
Gear ratio	38.3:1 (Gearbox) 49.8:1 (Final)
Pushing force	18kg
Swing radius	40cm
Turn radius	0cm
Translate speed max	1.2 m/sec
Traversable step max	10cm
Traversable gap max	15.2cm
Traversable slope max	40% grade
Traversable terrains	Asphalt, sand, dirt -- not carpet!
SENSING & MANIPULATION (not requiring onboard computing)	
Front sonar ring	8 optional, 1 each side, 6 forward @ 15° intervals
Rear sonar ring	8 optional, 1 each side, 6 rear @ 15° intervals
Sonar range	15cm - 5m (opt)
Std. Position encoders	500 tick encoders
Gripper option	Yes
Wireless Ethernet Operation option	Yes

ONBOARD COMPUTING, SENSORS & ACCESSORIES (requiring onboard computing)	
Optional onboard computer	Embedded size
Max. no. Cards & ports	2 PC104+ ; 2 USB ; 4 serial
Speaker	Piezo
Laser option	Yes
Gyro option	Yes
GPS option	Yes
Vision/ACTS Color- Tracking options	Yes
Speech/Voice option	Yes
StereoCam Rangefinder option	Yes
ELECTRONICS	
Processor	Hitachi H8S
Sonar inputs	16 max
Custom I/O connections	8-bit external I/O bus w/ up to 16 devices + PC104 I/O boards
Analog/Digital	5 @ 0-5 VDC
Communications ports	3 RS-232 serial ports on microcontroller, 4 RS-232 and 1 Ethernet on optional embedded computer
Wireless Communications options	Radio modem pair without embedded computer; Ethernet station adapter & access point with
Flash Memory	1 mB
Power switches	1 main; 2 auxiliary
CONTROLS, PORTS AND INDICATORS (side or top panels)	
LCD display	-na-
Reset pushbutton	Warm reboot
Function pushbutton	Self-test
Main power switch	Robot power; 12VDC; red LED indicator
Radio power switch	Radio modem or other 12VDC option
Aux power switch	12VDC
Motors pushbuttons	Single enable/disable
Serial comm ports	9-pin RS232 with Rcv and Xmt LED indicators
Joy drive port	Off & opt. onbd.
Charging	12VDC charge port

ESPECIFICAÇÕES DO MANIPULADOR	
CONSTRUCTION	anodized, CNC fabricated, and painted aluminum, with foam-covered gripper paddles
MOTION	2-axis--horizontal and vertical--with 1 degree of freedom
POWER	+5 and +12 VDC supplied by Pioneer robot
DIMENSIONS	10.5" x 2" x 1.75" (25.5 cm x 4.4 cm x 3.5 cm)
LIFT WIDTH	Grippers part to 9" (19.2 cm)
LIFT HEIGHT	Two vertical stops: one about 3/4 inch off the ground (so tall objects won't block robot sonar) and one at the top of the gripper
PAYLOAD	5.5 lb (2.5 kg) lift capability, lateral pressure software controlled
SENSORS	Front and rear break beams between gripper paddles. Pressure sensors inside each gripper paddle
SPEED	3 seconds from fully open to fully closed position