UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Alexandre Teodoro Guimarães

IMPLEMENTAÇÃO DE UM SISTEMA ROBÓTICO DE COMPORTAMENTOS DINÂMICOS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. João Bosco da Mota Alves

Florianópolis, outubro de 2003

IMPLEMENTAÇÃO DE UM SISTEMA ROBÓTICO DE COMPORTAMENTOS DINÂMICOS

Alexandre Teodoro Guimarães

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Fernando A. Ostuni Gauthier, Dr.
Coordenador

Banca Examinadora

_____

Dr. João Bosco da Mota Alves (orientador)

_____

Dr. Benedito Renê Fischer

_____

Dr. João Cândido Lima Dovicchi

“Free your mind.”

Morpheus

Dedico este trabalho à minha mãe Marlene, ao meu pai Emílio, à minha esposa Teresa e ao meu filho Edgar pelo incentivo e compreensão diários.

Agradecimentos

Gostaria de agradecer primeiramente a minha família que me apoiou durante todo o processo, mesmo quando precisei deixa-los ao mudar de cidade para iniciar o processo de pós-graduação na UFSC.

Agradeço também ao amigo e orientador João Bosco da Mota Alves que me acolheu como filho e acreditou no meu trabalho.

Também é hora de lembrar dos amigos Maurício de Pelotas e Saulo de Chapecó que me ajudaram a percorrer este difícil caminho, vencendo problemas e proporcionando alegrias.

Lembrar ainda da experiência e de todo pessoal do RexLAB, os quais se tornaram amigos que nunca serão esquecidos.

Agradeço também aos professores Alfen, Elmo e Adriano de minha graduação em Uberlândia, pois sem o incentivo e apoio destes amigos, provavelmente não teria iniciado este trabalho.

Não esquecer ainda dos amigos Fernando e Arthur que ajudaram a tornar os dias e noites melhores.

Por fim, agradeço o carinho de todos os catarinenses que receberam tão bem este mineiro em sua capital e minha esposa Teresa e meu filho Edgar, que mesmo com apenas 3 meses de idade, de alguma forma compreendeu todo este processo.

SUMÁRIO

RESUMO ..... **x**

Abstract ..... **xi**

1 INTRODUÇÃO **1**

1.1 O Sistema Robótico *2*

1.2 Proposta do trabalho *3*

1.3 Descrição do trabalho *4*

2 ROBÔS BASEADOS EM COMPORTAMENTOS **6**

2.1 Robôs deliberativos e reativos *6*

2.2 Biblioteca de comportamentos *10*

2.3 Gerenciamento de comportamentos *12*

3 SENSORES **16**

3.1 Micro-Câmera *17*

3.2 Sensores de toque *18*

4 SISTEMA ROBÓTICO **20**

4.1 Visão geral do sistema *21*

4.2 Montagem *23*

4.2.1 Construção do decodificador 26

4.2.2 Construção do gerador de freqüências 30

5 SISTEMA OPERACIONAL DO ROBÔ **32**

5.1 Construindo comportamentos *34*

5.1.1 Biblioteca de procedimentos 34

5.1.2 Associando procedimentos 43

5.2 Um gerenciador de comportamentos *43*

6 Estudo de caso **46**

6.1 Comportamento vagar *46*

6.2 Comportamento seguir cor *46*

6.3 Comportamento desviar de obstáculos *49*

6.4 Utilização de comportamentos múltiplos no gerenciador de comportamentos ... *50*

7 Conclusões e trabalhos futuros **56**

REFERÊNCIAS BIBLIOGRÁFICAS 60
BIBLIOGRAFIA 61
GLOSSÁRIO 63

Lista de figuras

FIGURA 1.1 – O sistema robótico 2
FIGURA 2.1 – Comportamento segundo IA tradicional 7
FIGURA 2.2 - Técnicas e abordagens diferentes para controle de robôs 8
FIGURA 2.3 – Comportamento segundo BROOKS 9
FIGURA 2.4 – Escalonamento Round Robin. A) Processos prontos. B) A mesma lista, após o quantum do processo B ter expirado. 14
FIGURA 2.5 – Algoritmo de escalonamento com quatro classes de prioridade 14
FIGURA 3.1 - SHAKEY robô que reconhece objetos 17
FIGURA 4.1 – Esquema geral do sistema robótico 21
FIGURA 4.2 – Detalhes do carro rádio controlado 24
FIGURA 4.3 – Baterias do controle e do carro 24
FIGURA 4.4 – Micro-câmera utilizada no robô 24
FIGURA 4.5 – Vídeo-link transmissor de vídeo sem fio 25
FIGURA 4.6 – Esquema de conexões do robô 25
FIGURA 4.7 – Esquema padrão da porta paralela do computador 27
FIGURA 4.8 – Conexões do controle remoto e o decodificador 27
FIGURA 4.9 – Esquema do circuito do decodificador 29
FIGURA 4.10 – Circuito gerador de freqüências 31
FIGURA 5.1 – Sistema operacional do robô 33
FIGURA 5.2 – Biblioteca de procedimentos 35
FIGURA 5.3 – Variáveis que podem ser ajustadas nos filtros de eliminação de ruídos 36
FIGURA 5.4 - Variáveis que podem ser ajustadas no filtro de seleção de cores 38
FIGURA 5.5 – Exemplo do filtro de seleção de cores 39
FIGURA 5.6 – Variáveis da conversão para tons de cinza 39
FIGURA 5.7 – Exemplo de conversão da imagem para tons de cinza 40
FIGURA 5.8 – Variáveis do conversor para imagem binária 40

FIGURA 5.9 – Exemplo de conversão binária 41
FIGURA 5.10 – Exemplo da função que segue cores 42
FIGURA 5.11 – Função *OnAudio* 43
FIGURA 5.12 – Gerenciador de comportamentos 44
FIGURA 6.1 – (A) Imagem recebida com ruído. (B) Resultado da imagem sem o tratamento adequado com os filtros. (C) Resultado da imagem com a utilização adequada dos filtros 47
FIGURA 6.2 – Visão do robô ao executar o comportamento de seguir luminosidade 48
FIGURA 6.3 – Simulação de um comportamento de desvio de obstáculo 49
FIGURA 6.4 – Seqüência de imagens do robô efetuando desvio de obstáculo e atingindo um objeto luminoso 53
FIGURA 6.5 – Seqüência de imagens do robô executando comportamento de seguir luminosidade, desvio de obstáculo e vagar. 54

Lista de tabelas

TABELA 4.1 – Atuação do decodificador 28
TABELA 4.2 – Descrição das saídas do decodificador para o controle remoto 29

RESUMO

A utilização da arquitetura de *Subsumption* no controle de robôs baseada em comportamentos vem sendo analisada e implementada em vários centros de pesquisa no mundo. Seus métodos já se constataram eficientes no controle destes robôs. Entretanto, a implementação torna-se cara, assim como em outros sistemas robóticos. A procura por meios que tornem mais viável a implementação de robôs baseados nesta arquitetura é um grande desafio, bem como a redução da complexidade de construção de robôs para profissionais da computação com conhecimentos limitados em engenharia elétrica e mecânica. Este trabalho apresenta uma nova metodologia no desenvolvimento de sistemas robóticos, implementando algoritmos de sistemas operacionais para controle de robôs baseados em comportamentos e simulando a arquitetura de *Subsumption,* bem como a implementação do conceito de biblioteca de comportamentos descrito por MAES & BROOKS (1990). O trabalho visa ainda, utilizar, ao máximo, recursos existentes de produtos prontos, para viabilizar a construção do sistema robótico em nível de custos e de conhecimentos limitados em outras áreas diferentes da computação. O sistema aqui apresentado pode aumentar a difusão da robótica para a área da computação e aplicações de algoritmos pertinentes a gerenciamento de processos de sistemas operacionais, tratamento de imagens, Transformada Rápida de Fourier para identificação de freqüências distintas, dentre outros. Todos os detalhes de implementação estão apresentados no texto, incluindo descrições completas dos resultados.

Palavras-chave: Sistemas robóticos. Robôs baseados em comportamento. Biblioteca de comportamentos. Processamento de imagens. Processamento de áudio. *Wireless.*

Abstract

The use of the architecture of Subsumption and its use in the behaviors-based robots' control It's being analyzed and implemented in several research centers in the world. This methods already verified efficient in behaviors-based robots' control, however the implementation becomes expensive as well as in others robotics systems. The search for ways that turn possible implementation of robots' based in this architecture is a great challenge, such as the reduction of the complexity of robots' construction for professionals of the Computer Science, whom have limited knowledge in electric and mechanical engineering. This work presents a new methodology in the development of robotics systems, implementing algorithms of operating systems to control behavior-based robots' and simulating the architecture of Subsumption as well as the implementation of the library of behaviors concept described by MAES & BROOKS (1990). This work use a lot existent resources of ready products to make possible the robotic system construction in low budget and turn possible the robotic system construction to others areas different from Computer Science. The system here presented can increase the diffusion of the robotics in Computer Science area using applications of pertinent algorithms in processes management of operating systems, image processing, Fast Fourier Transform for identification of different frequencies and others. Every implementaton details are presented in the text, including complete results descriptions.

Words key: *Robotics systems. Behavior-based robots. Library of behaviors. Image processing. Audio processing. Wireless.*

1 INTRODUÇÃO

Como vem se percebendo ao longo do tempo, cada vez mais os robôs estão sendo utilizados na indústria, em missões espaciais, para explorar oceanos, em usinas nucleares e em diversos outros setores da tecnologia e desenvolvimento. O campo da tecnologia requer conhecimentos de Engenharia Elétrica, Engenharia Mecânica, Engenharia Industrial, Computação, Matemática dentre outras.

"O termo robô foi introduzido por Czech Playwright Karel Capek, em 1920, no trabalho *Rossum's Universal Robots*. Desde então, inúmeros mecanismos, tais como teleoperadores, veículos autônomos, veículos submarinos e qualquer sistema com um certo grau de autonomia construído sob uma plataforma computacional são comumente chamados de robôs".(FARIA, 2003. p.1).

Neste trabalho, o termo robô se refere a veículo terrestre controlado por computador sem fio *wireless,* mostrado na Fig.1.1. Este robô é essencialmente um carro rádio controlado por um computador equipado com câmera e sensores, movido por motores de corrente contínua (DC). Seu complexo sistema de controle requer o estudo de robôs baseados em comportamento, reconhecimento de padrões e técnicas de sistemas operacionais.

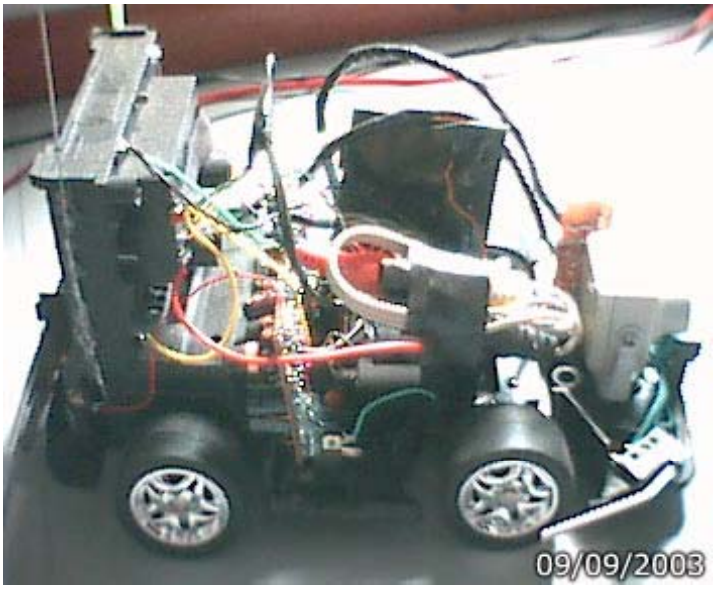

FIGURA 1.1 – O sistema robótico

Para isto, são utilizados: um carro rádio controlado em 27Mhz com motores DC, micro-câmera colorida NTSC, circuito gerador de freqüências baseado no circuito integrado NE555, vídeo-link transmissor de vídeo sem fio e um computador que conterá o sistema operacional do robô, que irá efetuar todo o processamento de dados e tomará as decisões para movimentar o robô.

1.1 O Sistema Robótico

Este trabalho tem por objetivo desenvolver um robô de comportamentos dinâmicos baseado em diversos fundamentos, tais como sistemas operacionais modernos, biblioteca de comportamentos, dentre outros. As características principais do sistema são:

- O robô possui uma micro-câmera que capta imagens coloridas e as envia sem a utilização de fios, pelo vídeo do canal VHF 13 com a ajuda do vídeo-link.

- O robô possui quatro sensores de toque posicionados em locais estratégicos, que captam colisões e envia freqüências distintas de cada sensor de toque pelo áudio do canal VHF 13 com a ajuda do vídeo-link.
- O computador captura os sinais de áudio e vídeo enviados pelo robô no canal VHF 13 com uma placa de captura de TV *PixelView PlayTV*.
- O computador utiliza a porta paralela para acionar um circuito, que envia os movimentos a serem tomados para o robô, por rádio freqüência.
- O robô e o computador se comunicam sem a utilização de fios.
- O computador possui um sistema operacional de controle de robô.
- O computador realiza a tarefa de analisar e processar os dados capturados pelo robô e enviar o movimento a ser executado após a tomada de decisão.
- O sistema operacional do robô permite criar diferentes comportamentos baseados em procedimentos (funções, conversores e filtros) e adicioná-los em uma biblioteca de comportamentos.
- O sistema operacional do robô é capaz de associar diferentes comportamentos da biblioteca de comportamentos e gerenciá-los de acordo com suas prioridades.

1.2 Proposta do trabalho

As contribuições deste trabalho são:

- Construir um robô baseado em comportamento dinâmico com custo relativamente baixo.
- Mostrar as vantagens de um sistema robótico, que executa comportamentos dinâmicos de forma autônoma.

- Criar um sistema operacional de controle de robô simples, de fácil operação e rápidas alterações.
- Criar um sistema operacional, que implemente a arquitetura de *Subsumption* e a idéia da biblioteca de comportamentos.
- Mostrar diversas utilidades para este sistema em indústrias, empresas, segurança, dentre outras.

1.3 Descrição do trabalho

No capítulo 2, será apresentada a introdução a robôs, os conceitos e história de robôs baseados em comportamentos e a idéia, proposta por MAES & BROOKS (1990), da biblioteca de comportamentos. Também será feita uma comparação entre a arquitetura de *Subsumption* e as técnicas de gerenciamento de processos em sistemas operacionais, descrita por TANENBAUM (1992).

No capítulo 3, serão demonstrados como os sensores estão inseridos em robótica e sua importância neste trabalho. O capítulo será concluído com enfoque no sensor de visão do robô e na detecção de obstáculos, descrevendo inclusive o circuito gerador de freqüências.

No capítulo 4, será demonstrado o funcionamento do sistema operacional criado para este robô de comportamentos dinâmicos e como o *software* capta os dados, processa e envia de volta ao robô.

O capítulo 5 será reservado para descrever em detalhes, toda a construção do robô de forma que possa ser replicado com fidelidade no futuro.

No capítulo 6, será feito um estudo de caso em um ambiente controlado, mostrando como o mesmo robô pode trabalhar com diferentes comportamentos e tomar

decisões completamente diferentes, mostrando o funcionamento da arquitetura de *Subsumption* e da biblioteca de comportamentos proposta por MAES & BROOKS (1990).

No capítulo 7, são feitas as conclusões, a apresentação de propostas de novas pesquisas e a contribuição deste trabalho.

2 ROBÔS BASEADOS EM COMPORTAMENTOS

Uma das definições, dada em 1985, segundo JABLONSKI E POSEY, da *Robotics Industry Association* - *RIA,* foi de que "um robô é um manipulador re-programável e multifuncional desenvolvido para mover materiais, partes, ferramentas ou dispositivos especiais através de uma variedade de movimentos programados, para executar uma variedade de tarefas". (JABLONSKI & POSEY ,1985 *apud* ARKIN,1999, p.1).

Nos tempos atuais, é possível dizer que um robô pode ser, simplesmente, uma máquina capaz de extrair informações de seu ambiente e usar o raciocínio sobre este mundo, para mover-se seguramente de maneira significativa e intencional.

2.1 Robôs deliberativos e reativos

Um comportamento pode ser simplesmente definido como uma reação a um estímulo. Segundo a Inteligência Artificial - IA tradicional, um robô deve obter dados de seu mundo, compreendê-los de forma cognitiva, executar uma ação, repetindo o ciclo novamente (Fig. 2.1). Em meados de 1986 e 1987, alguns pioneiros revelaram algumas idéias sobre novas metodologias de controle por comportamentos.

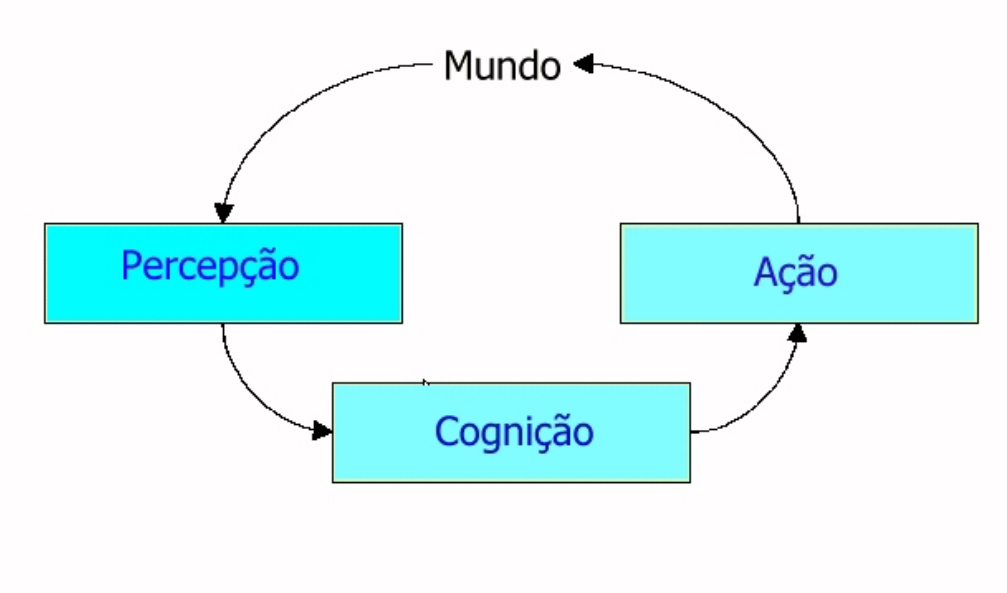

FIGURA 2.1 – Comportamento segundo IA tradicional
FONTE – ZAMBIASI et al 2001.

AGRE & CHAPMAN (1987) *apud* ZAMBIASI *et al.* (2001), propuseram que "as atividades cotidianas não são o resultado de uma planificação, e sim de atividades rotineiras". Já ROSENSCHEIN & KAELBLING (1986) *apud* ZAMBIASI *et al.* (2001) afirmaram que "é possível definir crenças e objetivos, e compilá-los em programas eficientes baseados em circuitos". BROOKS (1986) *apud* ZAMBIASI *et al.* (2001) viu a importância de "construir agentes com sensores reais, que operem no mundo real. Muitas das ações de um agente são separáveis e um comportamento coerente pode emergir da interação independente dos componentes com o mundo real".

Assim algumas técnicas e abordagens diferentes para controle de robôs têm sido criadas. A Fig. 2.2 retrata um espectro atual de estratégias de controle de robôs. O lado esquerdo representa métodos, que empregam reações deliberativas, que respondem a estímulos externos, mas analisa as informações e pode tomar iniciativas para cumprir os objetivos intencionais.

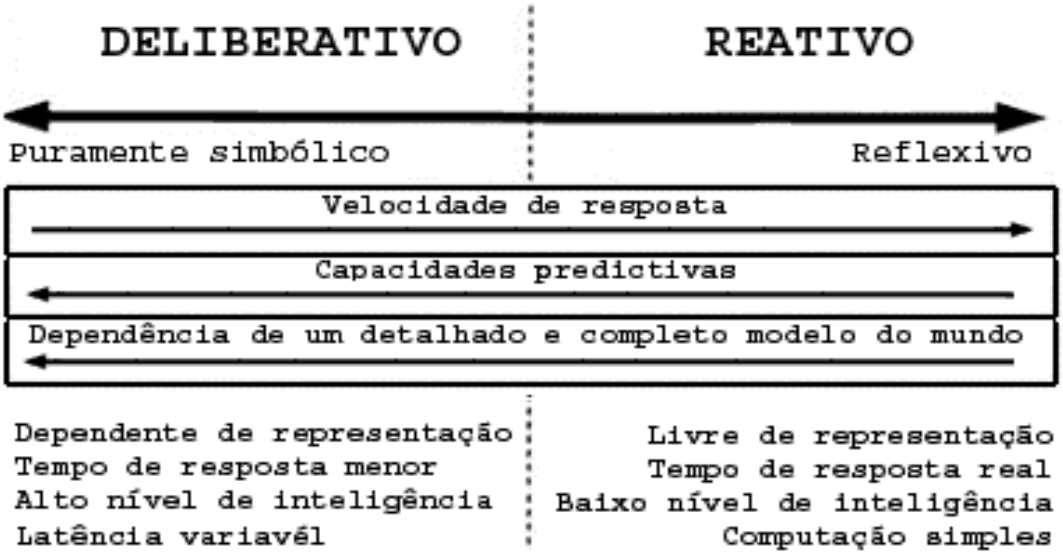

FIGURA 2.2 - Técnicas e abordagens diferentes para controle de robôs

FONTE – ARKIN, 1999. P.20.

O lado direito representa controle reativo, que apenas reagem a estímulos externos. Um robô que emprega razões deliberativas requer relativamente, completo conhecimento sobre o mundo e o uso do conhecimento para prever o resultado de suas ações, uma habilidade que o capacita para otimizar sua performance relativa ao modelo do seu mundo.

Reações deliberativas quase sempre requerem grande conhecimento sobre o seu modelo de mundo, para que o conhecimento no qual o raciocínio é baseado seja consistente, confiável e certo. Se a informação que o raciocínio usa for imprecisa ou sofrer alterações após ser obtida, o resultado do raciocínio pode ficar seriamente comprometido. Em um mundo dinâmico, onde objetos podem mudar arbitrariamente, como em um campo de batalha ou em um corredor lotado, é muito perigoso basear-se em informação que não mais seja válida. A representação dos modelos do mundo é, portanto, geralmente construída sobre ambas prioridades - o conhecimento do ambiente e a entrada do sensor de dados.

Porém, para BROOKS *apud* ZAMBIASI *et al.* (2001), o processamento cognitivo é desnecessário e consome um tempo relativamente grande, que atrapalha o desempenho dos robôs, sendo melhor construir os comportamentos de um robô sem este processamento, ou seja, um processamento totalmente reativo (Fig. 2.3). Esta nova técnica para a robótica inteligente é chamada de mecanismo de estímulo/resposta, também conhecido como arquitetura Subsumption.

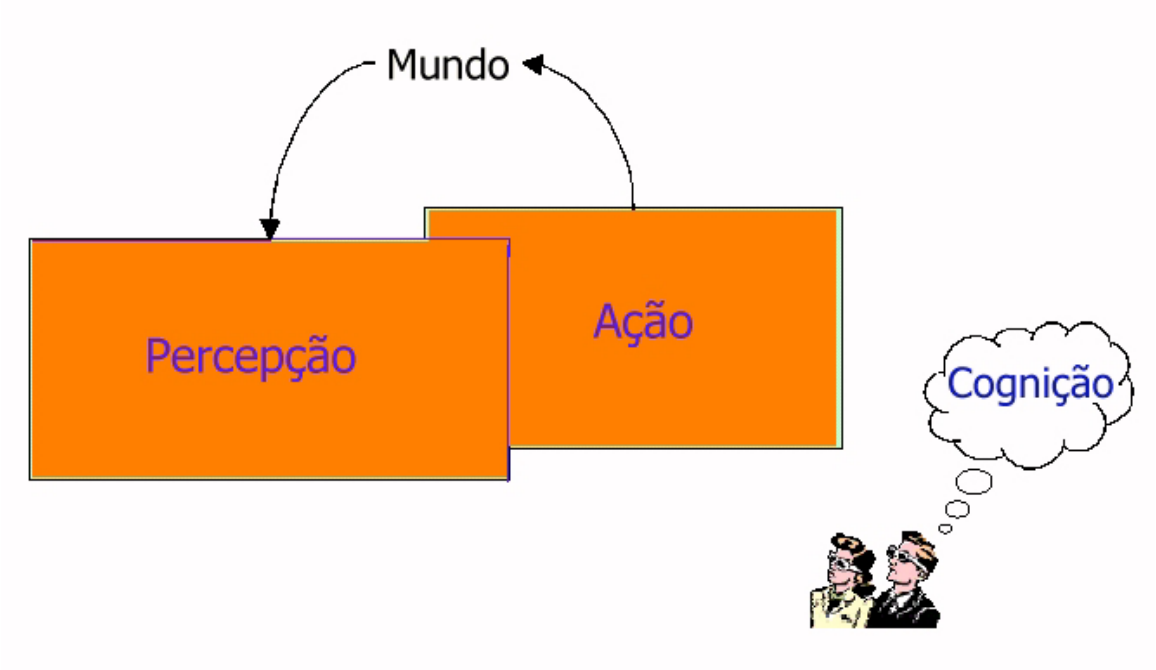

FIGURA 2.3 – Comportamento segundo BROOKS
FONTE – ZAMBIASI et al. 2001.

Em um robô que utiliza a técnica de estímulo/resposta, não existe memória e nem decisões lógicas para serem feitas, somente respostas diretas a estímulos. Por exemplo, conectando sensores de luz diretamente a motores, é possível construir um robô que segue uma fonte luminosa. Vários mecanismos de estímulo/resposta operando juntos em um robô podem criar comportamentos elaborados, que podem parecer inteligentes.

Um sistema reativo não contém nenhuma forma central de modelo do ambiente e não utiliza um raciocínio simbólico complexo. Ele é baseado no princípio da reatividade

dos agentes, ou seja, na suposição de que comportamentos inteligentes podem ser gerados sem nenhuma representação simbólica explícita e a suposição de que a inteligência é uma propriedade emergente de certos sistemas complexos. Agentes reativos são capazes de reagir ao ambiente sem utilizar um raciocínio complexo.

Um sistema reativo opera em um nível muito baixo de abstração, sem nenhum conhecimento prévio de seu ambiente. Esta característica é em si uma vantagem, mas também uma desvantagem. O tempo de resposta do agente é muito baixo. Isto conduz para uma comunicação eficiente entre dois agentes ou entre o agente e o ambiente.

A desvantagem é que o agente móvel não pode executar análises complexas de seus dados sensoriais. Isto também significa que ele não pode executar operações cognitivas de alto nível envolvendo crenças, desejos e intenções, a não ser que ele tenha acumulado conhecimento suficiente para executar tais funções.

A navegação baseada em comportamentos foi introduzida por BROOKS *apud* ZAMBIASI *et al.* (2001) e divide a tarefa de navegação em comportamentos básicos. Esta técnica gera caminhos que podem ser difíceis de prever, mas apresentam uma oportunidade de navegação reativa independente de um caminho geométrico baseado em modelos do ambiente. ARKIN (1999) *apud* ZAMBIASI *et al.* (2001), define este comportamento como senso-motor**.** O conceito de senso-motor originou-se na psicologia e na neurologia para descrever a interação entre percepção e ação dos seres vivos.

2.2 Biblioteca de comportamentos

Um dos artigos mais interessantes de Rodney A. Brooks e Pattie Maes, ambos do *Massachusetts Institute of Techonology-MIT,* intitulado *Learning to Coordinate*

Behaviors, descreve um algoritmo que permite, que robôs baseados em comportamento aprendam com base em resultados positivos ou negativos, gerados por suas próprias ações. O algoritmo foi testado com sucesso em um robô, que aprendeu a coordenar suas pernas.

Ao final da primeira parte, onde termina de ser especificada a situação do problema, MAES & BROOKS (1990) esperavam com seu trabalho, programar comportamentos do robô, selecionando-os de uma biblioteca de comportamentos, conectando-os aos sensores e ativadores do robô, definindo funções que retornariam respostas positivas ou negativas e fazendo que aprendam com estas experiências, quando os comportamentos deveriam ficar ativos.

A idéia de poder associar comportamentos diferentes através de uma biblioteca expande as possibilidades de um robô, tornando-o dinâmico mesmo que com ações simplesmente reativas, sendo capaz de executar diferentes tipos de comportamento.

Sem dúvida, este tipo de recurso seria de alta utilidade para robôs que deveriam executar comportamentos diversos em momentos diferentes. Um mesmo robô equipado com uma câmera de vídeo poderia ser utilizado para encontrar a saída de um túnel por meio do brilho do lado externo, ser utilizado para segurança em residências, ou encontrar garrafas verdes, dependendo do comportamento executado em determinado instante.

Porém, dificuldades surgem, quando se adicionam vários comportamentos criados a partir da biblioteca de comportamentos. Neste mesmo trabalho, os autores dizem que "à medida que cresce o número de comportamentos, o problema de controle e coordenação torna-se complexo. É bastante difícil, especificar detalhes do ambiente e tarefa para o robô atingir seus objetivos" (MAES & BROOKS, 1990).

Com base nos conceitos de gerenciamento de processos utilizados em sistemas operacionais e na arquitetura de *Subsumption* sugerida por BROOKS *apud* ARKIN (1999) p.112, as dificuldades se tornam menores.

2.3 Gerenciamento de comportamentos

É possível, com base no gerenciamento de processos dos sistemas operacionais, reutilizar a mesma idéia no gerenciamento de comportamentos, implementando a arquitetura de *Subsumption*.

Segundo TANENBAUM (1992)p.10, "um processo é basicamente um programa em execução, sendo constituído do código executável, dos dados referentes ao código, da pilha de execução, do valor do contador de programa, do valor do apontador de pilha, dos valores dos demais registradores de *hardware*, além de um conjunto de outras informações necessárias à execução do programa".

Um processo pode ocupar três estados diferentes:

- Poder estar rodando - usando o processador no momento.
- Pronto - em condições de rodar, mas bloqueado temporariamente para dar a vez a um outro processo.
- Bloqueado - impedido de rodar até que ocorra um determinado evento externo ao processo.

O gerenciador de processos possui uma tabela de processos, onde ficam armazenadas as informações de processos, que foram bloqueados por terem excedido seu tempo de utilização pelo processador, para dar lugar a um outro processo de maior prioridade ou por estar aguardando uma entrada ainda não disponível.

Desta forma, é possível imaginar um gerenciador de comportamentos, com uma tabela de comportamentos, onde ficam armazenadas informações de comportamentos que podem estar sendo executados (rodando) no momento, ou bloqueados esperando seu momento para executar, ou prontos quando já tiver sido executados e estiver esperando seu momento de executar novamente.

Assim, como um gerenciador de processos deve manter uma tabela chamada tabela de processos, com uma entrada por processo, o gerenciador de comportamento deve fazer o mesmo com cada comportamento.

Para replicar o funcionamento da arquitetura de *Subsumption* neste aspecto de gerenciamento de comportamento, pode-se utilizar os algoritmos de escalonamento de processos, que, no caso, atuará no gerenciamento de comportamentos do robô. O escalonador é a parte do sistema operacional, que decide, no caso de existir mais de um processo, qual deles deve rodar primeiro.

O escalonador *Round Robin* é um dos mais antigos, simples e justo; e, muito utilizado em sistemas operacionais. Cada processo possui um intervalo de tempo, chamado de *quantum*, para rodar.

No caso do processo, se após esgotar seu *quantum,* ainda necessitar de tempo para rodar, ele dará o lugar a um novo processo e irá para o fim da tabela de processos. Desta forma, todos os processos têm chance de rodar com tempos de processamento iguais para cada processo. (Fig. 2.4)

De forma diferente do escalonamento *Round Robin*, que assume que todos os processos são igualmente importantes, o escalonamento com prioridades considera fatores externos para a escolha do próximo processo a rodar.

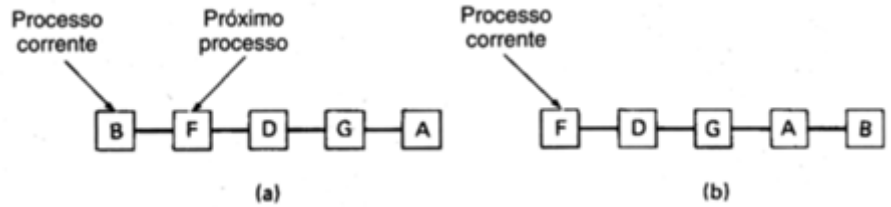

FIGURA 2.4 – Escalonamento Round Robin. A) Processos prontos. B) A mesma lista, após o quantum do processo B ter expirado.

FONTE – TANENBAUM, 1992. p.45.

A idéia é de que cada processo seja associado a uma prioridade e o processo pronto com maior prioridade será aquele que vai rodar primeiro.

Porém, TANENBAUM (1992)p.46 diz que, "algumas vezes pode ser conveniente agrupar processos em classes de prioridades e usar o escalonamento com prioridade entre as classes e o Round Robin dentro de cada classe".(Fig. 2.5).

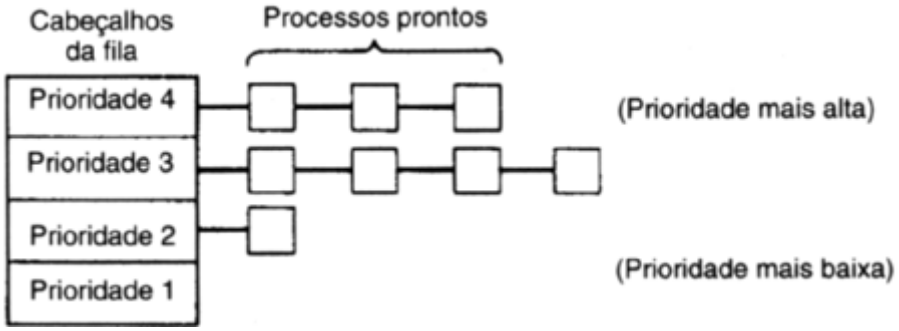

FIGURA 2.5 – Algoritmo de escalonamento com quatro classes de prioridade

FONTE – TANENBAUM, 1992. p.46.

Por meio desta figura, é possível perceber nitidamente a grande semelhança com a arquitetura de *Subsumption*, onde os processos seriam comportamentos, e cada comportamento pode ter prioridade igual ou diferente de outro comportamento.

Fica mais fácil visualizar isto, da seguinte forma: Imaginando que um robô possua sensores de ultra-som, que captem barreiras espalhadas na frente, dos lados e atrás, e que cada sensor possua um comportamento específico para cada sensor a ser acionado ou não. Neste caso, todos os comportamentos podem ter a mesma prioridade, visto que

a importância de cada sensor ao desviar de obstáculos é a mesma. Por outro lado, o sensor da frente poderia ter maior importância do que os das laterais, que por sua vez teriam mais importância do que os de trás, fazendo com que o robô tente sempre caminhar adiante em um determinado ambiente.

No próximo capítulo, será demonstrado como os sensores são parte indispensável para a compreensão do ambiente onde o robô se encontra, para que ele consiga tomar as decisões cabíveis.

3 SENSORES

Seja qual for a função a ser realizada por um robô autônomo, ou seja, que não necessite de alguém que controle seus movimentos, é preciso que ele de algum modo consiga extrair informações de seu mundo. Para isto, são equipados com os mais diversos tipos de sensores, que conseguem captar sinais do mundo exterior, transformando-os em dados digitais, de forma que ele os compreenda.

Dentre vários destes sensores, podemos citar o infravermelho para detectar fontes de calor, ultra-som ou sensores de toque para detectar barreiras, uma câmera que capte imagens e forneça informações sobre o mundo para o robô, além de vários outros tipos de sensores para as mais diversas funções.

Um dos primeiros robôs móveis foi construído na década de 60, no Instituto de Pesquisa de Stanford (Fig 3.1). O SHAKEY, como foi chamado, precisa estar em mundo inabitado e artificial, em um ambiente de trabalho no qual os objetos são especialmente coloridos e moldados para auxiliá-lo no reconhecimento de um objeto, utilizando a visão. Ele planeja uma ação do tipo empurrar um objeto reconhecido de um lugar para outro e, então, executar um plano.

FIGURA 3.1 - SHAKEY robô que reconhece objetos
FONTE – ARKIN, 1999. p.16.

Este robô foi construído sobre dois motores de passo independentes e tem uma câmera de televisão e um sensor de alcance ótico montados no topo. A câmera tem um motor que controla a inclinação, o foco e a íris.

Neste trabalho, o robô conta com uma micro-câmera colorida no formato NTSC e quatro sensores de toque para detecção de barreiras.

3.1 Micro-Câmera

A micro-câmera do robô é responsável por captar imagens a cores do ambiente onde o robô está inserido e enviar, através de sua saída de vídeo composto para o vídeo-link, o sinal de vídeo, que por sua vez transmite o sinal no canal VHF 13 em um raio de 50 metros. Ambos, micro-câmera e vídeo, são alimentados no robô com 12 volts, fornecidos por 8 pilhas pequenas AA recarregáveis de 850mA.

Até este momento, não houve nenhum processamento do que está sendo capturado pela câmera, que pode ser observado até mesmo com um televisor

sintonizado no canal VHF 13. O processamento começa realmente, somente após a captura do vídeo feita pela placa de captura instalada no computador, que será detalhado no capítulo cinco.

A câmera juntamente com os filtros e os conversores do sistema operacional construídos é de grande importância neste sistema robótico. O robô pode realizar operações como, ser atraído ou ser repelido pela luminosidade, imitando comportamentos de insetos e até comportamentos mais sofisticados, como ser atraído por uma determinada cor.

3.2 Sensores de toque

Algumas vezes, o robô pode se deparar com um obstáculo imperceptível para seus "olhos", como uma barreira de vidro ou um objeto fora de seu campo de visão.

Por isto, neste sistema robótico também foram inseridos sensores de toque, que poderiam ser substituídos por sensores de ultra-som, que detectariam as barreiras existentes antes mesmo da colisão. Porém, este sistema robótico visa economia de recursos para torná-lo o mais acessível possível.

Quatro sensores de toque estão estrategicamente posicionados no robô, sendo: um sensor na parte central traseira, um sensor na parte central da frente e os outros dois na parte frontal esquerda e direita.

Com este posicionamento dos sensores, o robô consegue detectar as barreiras que encontra pelo caminho e sabe exatamente em que parte ocorreu a colisão, encontrando a melhor solução para contornar o obstáculo e/ou executar uma operação cabível para a situação encontrada.

Os sensores se mantêm em posição aberta o tempo todo e se fecham no caso de uma colisão, gerando uma freqüência distinta[1] e ao perceber que a colisão já foi evitada os sensores voltam à posição aberta novamente, parando de transmitir a freqüência.

As freqüências geradas são enviadas para o vídeo-link, que transmite o sinal pelo áudio do canal VHF 13, que é capturado posteriormente pela placa de TV, no computador. O sinal de áudio é então processado pelo sistema operacional do robô, pela Transformada Rápida de Fourier. Daí então, o histograma da freqüência enviada é construído, possibilitando a comparação com as freqüências já analisadas previamente e a identificação, com precisão, do sensor que foi acionado.

No próximo capítulo, será mostrado com detalhes como foi construído o sistema robótico em questão, a fim de que a experiência possa ser replicada com fidelidade.

[1] Esta freqüência é gerada com a ajuda de um circuito produzido em conjunto com um grupo de alunos da Universidade Federal de Uberlândia em 2003, que utiliza o circuito integrado NE555 chamado de gerador de freqüências.

4 SISTEMA ROBÓTICO

Este capítulo foi escrito para possibilitar a reconstrução deste sistema robótico com fidelidade. É importante salientar que este sistema visa:

- Ser um sistema robótico de baixo custo.
- Tornar a alteração, criação e manipulação de comportamentos simples, rápida e de fácil acesso.
- Possibilitar a construção do robô sem a necessidade de importar componentes.
- Minimizar a construção de circuitos e conhecimentos em elétrica e mecânica, utilizando produtos já existentes de baixo custo que realizem as tarefas necessárias.
- Poder visualizar os comportamentos executados pelo robô e perceber como o funcionamento de um gerenciador de comportamentos simula exatamente a arquitetura de *Subsumption*.
- Utilizar o poder de processamento de um computador para executar o sistema operacional do robô.

Seguindo estes princípios, é possível retirar um projeto do papel e torná-lo real não só em laboratórios e grupos de pesquisa com escassez de recursos, mas torná-lo também um sistema robótico possível para estudantes e pesquisadores autônomos da área de Ciência da Computação, sem exigir deles conhecimentos profundos das áreas de engenharia elétrica e mecânica.

4.1 Visão geral do sistema

O robô foi construído sobre um chassi de um carro de rádio controle, que opera em rádio freqüência de 27Mhz. É equipado com uma micro-câmera padrão NTSC e sensores de toque conectados a um circuito gerador de freqüências. Os sinais de áudio e vídeo são enviados por meio de um transmissor de vídeo sem fio, em um raio de 50 metros, para o canal VHF 13 (Fig.4.1).

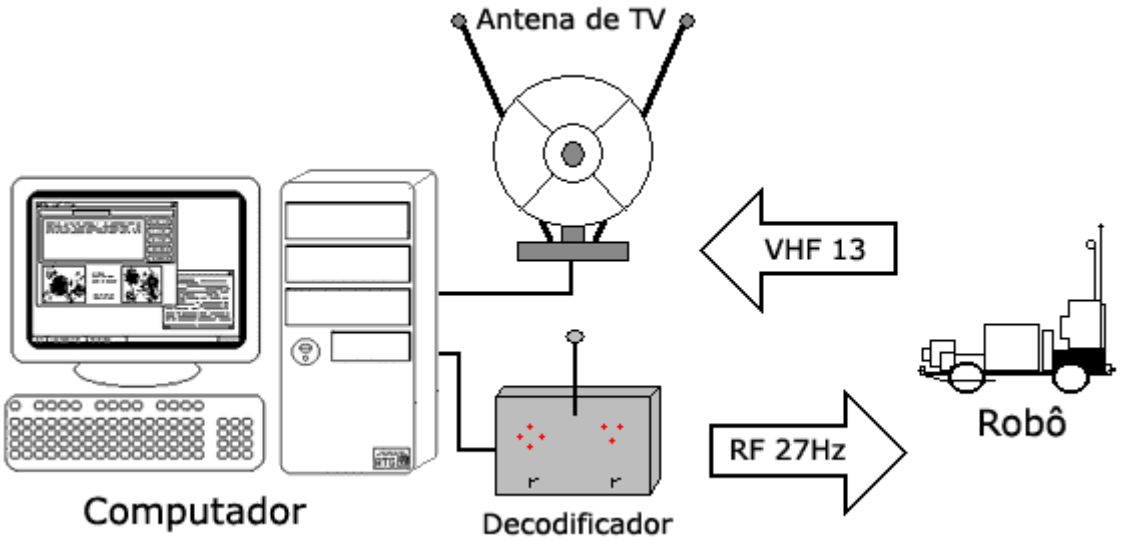

FIGURA 4.1 – Esquema geral do sistema robótico

Nas proximidades desse transmissor, um computador equipado com uma placa de captura de TV e uma antena de TV padrão, capta o sinal de áudio e vídeo enviado pelo robô, no canal VHF 13. O computador então processa o sinal de áudio e vídeo e envia

através da porta paralela, um comando para um circuito, que remete dados para um rádio controle, que por sua vez transmite para o robô o movimento a ser executado.

Os itens utilizados na construção deste sistema robótico foram:

- Um controle e um carro rádio controlado, de brinquedo.
    - Marca Newcon Farrari F50, modelo 6908 (figura 4.2).
    - Alimentação do carro 12 volts, providos por 8 pilhas pequenas AA recarregáveis de 850mA e do controle 9 volts (Fig.4.3).
    - Freqüência de 27Mhz.
    - Operação sem fio em raio aproximado de 13 metros.
- Quatro sensores de toque.
- Um gerador de freqüência baseado no NE555.
    - Alimentação 12 volts.
    - Um temporizador NE555
    - Um capacitor de cerâmica 152
    - Um potenciômetro
    - Cinco resistores (8200, 100K, 82K, 68K, 56K)
- Uma micro-câmera colorida padrão NTSC. (Fig. 4.4)
    - Modelo AIA-M-206CH.
    - Entrada de energia corrente contínua de 12 volts.
    - Consumo 50mA.
    - Saídas de áudio e vídeo composto.
    - Resolução de 380 linhas.
    - Lente de 3.6mm.
    - Abertura de 92^{o}.

- Um vídeo-link, transmissor de vídeo e áudio VHF (Fig. 4.5).
    - Canal 13.
    - Alimentação energia corrente contínua de 12 volts.
    - Consumo 30mA.
    - Operação sem fio, em raio aproximado de 50 metros.
- Um computador *desktop*
    - Processador Pentium II 400 e 128Mb PC100
    - Placa de TV PlayTV da Prolink PCI
    - Placa de Som CMI8738 PCI
    - Porta paralela 0x378h
- Antena de recepção de TV interna.
- Um decodificador de sinais
    - Alimentação 12 volts
    - Consumo aproximado 20mA.
    - Cinco reles 9 volts
    - Cinco resistores 1/8 200 ohms
    - Cinco diodos 1N4148
    - Uma placa de cobre 6x9cm

4.2 Montagem

A montagem do robô ocorreu de forma tranqüila, sem grandes modificações em nenhuma das partes adquiridas, com exceção do chassi do carro, que foi cortado para deixá-lo menor, apesar desta ser uma alteração desnecessária em se tratando do

funcionamento. Ao retirar a cobertura de plástico do carro rádio controlado e do vídeo-link, ocorreu a seguinte montagem demonstrada pela figura 4.6.

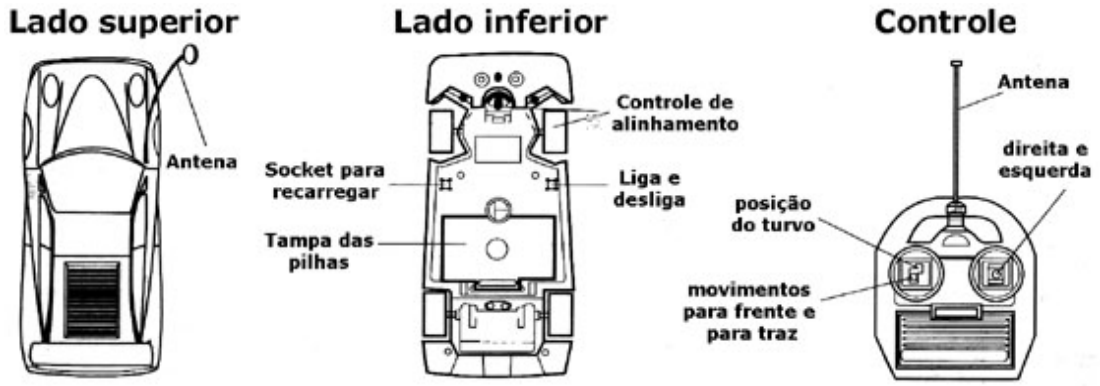

FIGURA 4.2 – Detalhes do carro rádio controlado

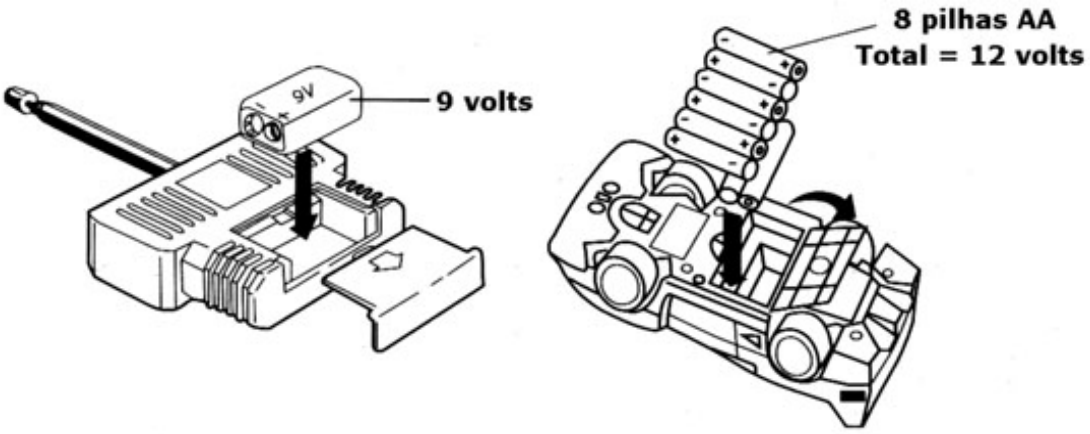

FIGURA 4.3 – Baterias do controle e do carro

FIGURA 4.4 – Micro-câmera utilizada no robô

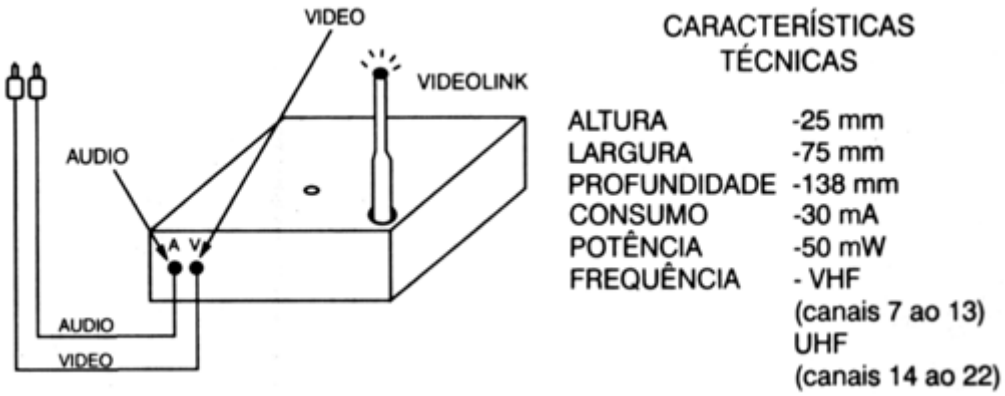

FIGURA 4.5 – Vídeo-link transmissor de vídeo sem fio

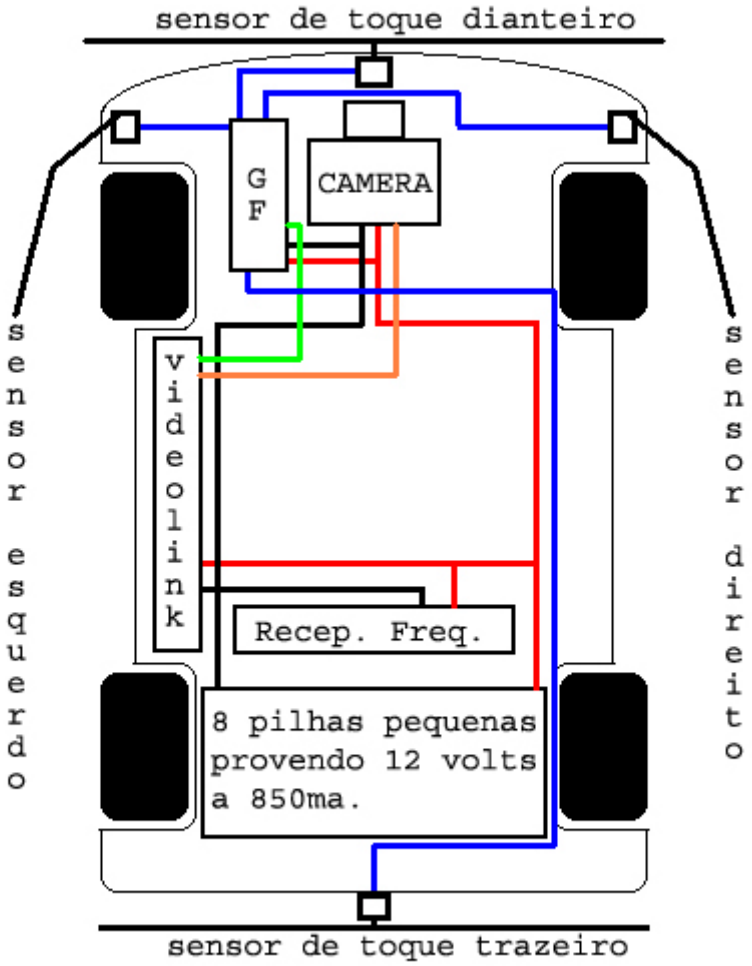

FIGURA 4.6 – Esquema de conexões do robô

A câmera foi colocada na frente do carro, sua saída de vídeo foi conectada ao vídeo-link (indicado em laranja) e sua saída de áudio não foi utilizada. A alimentação da câmera vem das próprias pilhas no carro (indicado em vermelho e preto).

Os quatro sensores foram dispostos na frente, nos lados e na parte traseira do carro, e foram conectados ao gerador de freqüências (indicado em azul). A saída das freqüências do gerador de freqüências foi conectada na entrada de áudio do vídeo-link (indicado em verde). Assim como a micro-câmera, o vídeo-link e o gerador de freqüências são alimentados pelas pilhas do carro (indicado em vermelho e preto).

Com estas alterações, o robô já está pronto para enviar os dados para o computador. O computador sintonizará o canal VHF 13 com a ajuda de uma antena e receberá os sinais de áudio e vídeo referentes às imagens do robô e aos sensores de toque. Entretanto, para que o computador consiga se comunicar com o robô é preciso, que seja construído o decodificador, que receberá os dados da porta paralela e os enviará para o controle do carro rádio controlado.

4.2.1 Construção do decodificador

O decodificador que recebe os dados da porta paralela e os envia para o controle de rádio freqüência funciona de maneira bastante simples. O sistema operacional do robô define a direção a ser tomada pelo robô e envia dados para os pinos D0 e D3 da porta paralela (Fig. 4.7). O pino D4 também foi utilizado, porque o controle remoto deste projeto precisa que, toda vez que uma direção for tomada, um comando comum a todas as direções também seja acionado simultaneamente.

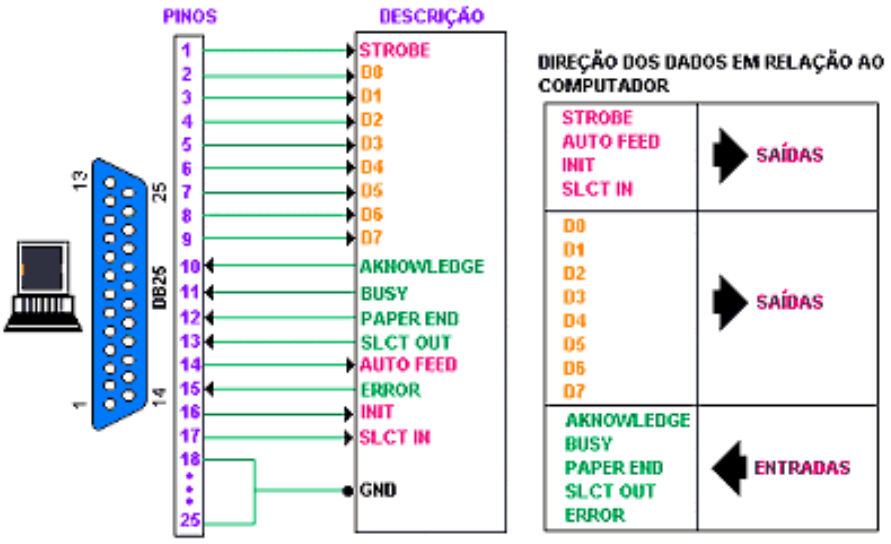

FIGURA 4.7 – Esquema padrão da porta paralela do computador

FONTE – MESSIAS, 2002.

O controle remoto foi aberto e todos os contatos de direção e também de alimentação foram soldados a um conector DB9, facilitando assim a conexão entre a porta paralela do computador, o decodificador e o controle remoto (Fig.4.8).

FIGURA 4.8 – Conexões do controle remoto e o decodificador

A função do decodificador construído é unicamente obter os dados recebidos da porta paralela e traduzir para o controle remoto, de forma que o robô execute os

movimentos. A tabela 4.1 representa o que cada pino da porta paralela significa para o controle remoto.

Bits da porta paralela (binário)	Significado para o controle remoto
10001	Virar para esquerda
10010	Virar para a direita
10100	Ir para trás
11000	Ir para frente
00000	Parar

TABELA 4.1 – Atuação do decodificador

Para ir para os lados é preciso combinar os valores. Assim, se o sistema operacional do robô definir que o movimento a ser tomado pelo robô seja, por exemplo, ir para frente à direita, ele irá enviar o valor binário 11010. O mesmo ocorre, por exemplo, se o movimento a ser tomado for para a esquerda e para trás, executando uma rotina de desvio de obstáculo, ele irá enviar o valor binário 10101.

O esquema de construção do decodificador (Fig.4.9) e o gerador de freqüências (Fig. 4.10) são os únicos circuitos construídos neste sistema robótico.

As entradas dos pinos da porta paralela estão representadas por D0 até D4 e GND, enquanto que as saídas do circuito para o controle remoto estão representadas por C0 até C6. A tabela 4.2 demonstra as ligações de C0 até C6 no controle.

A idéia geral do circuito é utilizar a energia da porta paralela de 5 volts, para acionar um transistor BC548, que libera 9 volts para um rele, que fecha os contatos diretamente no controle, através do conector DB9 previamente instalado,

movimentando o robô para quaisquer direções possíveis. O circuito é capaz ainda de prover alimentação para o controle remoto através das saídas C5 e C6.

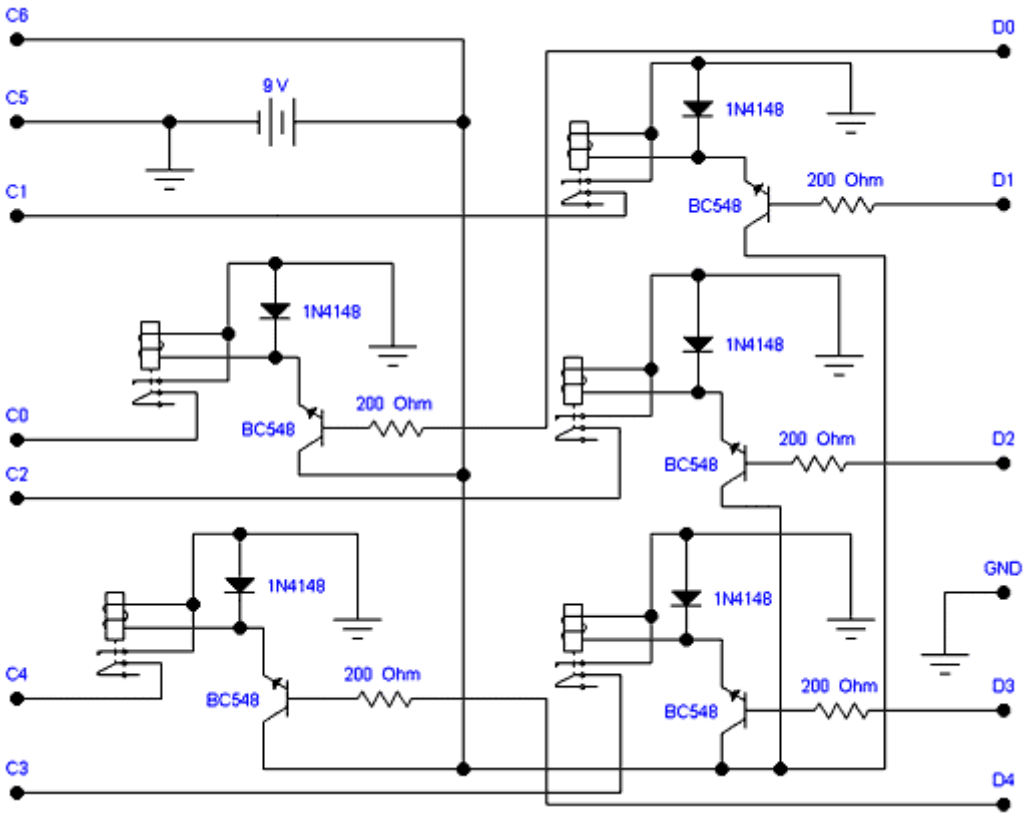

FIGURA 4.9 – Esquema do circuito do decodificador

Saídas do circuito	Descrição no controle remoto
C0	Virar para esquerda
C1	Virar para a direita
C2	Ir para baixo
C3	Ir para frente
C4	Comum
C5	Terra
C6	Alimentação (9 volts)

TABELA 4.2 – Descrição das saídas do decodificador para o controle remoto

4.2.2 Construção do gerador de freqüências

O circuito gerador de freqüências foi construído baseado no temporizador NE555, um oscilador monoestável, que estabelece intervalos regulares de tempo, para gerar freqüências de áudio distintas, que pudessem ser reconhecidas individualmente pelo sistema robótico desenvolvido neste trabalho.

A figura 4.10 mostra o esquema do circuito gerador de freqüências desenvolvido. Os sensores estão representados pelos números 1 a 4, respectivamente posicionados no robô como sensor frontal central, sensor frontal direito, sensor frontal esquerdo e sensor traseiro central. A saída de áudio do circuito é diretamente conectada ao vídeo-link, que envia o sinal de áudio pelo canal VHF 13, para que o sistema operacional do robô possa distinguir qual sensor foi acionado.

O gerador de freqüências pode operar entre 5 e 15 volts. Porém, neste sistema robótico, o limite é menor ficando entre 10 e 12 volts. Embora a voltagem de entrada seja extremamente importante para obter uma freqüência exata, neste sistema robótico isto não é necessário. Isto porque, os sensores podem ser calibrados novamente a qualquer momento, identificando a freqüência de saída para cada sensor acionado.

No próximo capítulo, será explicado em detalhes como foi feito o sistema operacional para gerenciar o sistema robótico criado. Será percebida, ao longo do capítulo, sua relação com a arquitetura de *Subsumption*, a biblioteca de comportamentos e o gerenciamento de processos de sistemas operacionais utilizado para o gerenciamento de comportamentos de robôs.

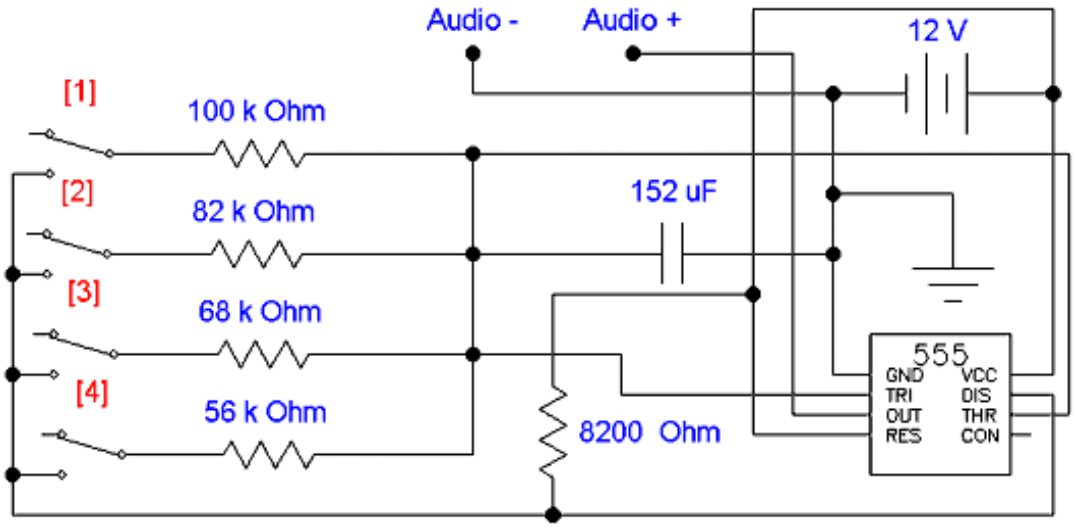

FIGURA 4.10 – Circuito gerador de freqüências

5 SISTEMA OPERACIONAL DO ROBÔ

Com o objetivo de controlar o robô e demonstrar o funcionamento da arquitetura de *Subsumption* utilizando técnicas de gerenciamento de processos, foi desenvolvido um sistema operacional, escrito no Borland Delphi 5.0 capaz de criar, modificar, excluir e controlar comportamentos do robô. A escolha da linguagem foi única e exclusivamente devido a conhecimentos prévios por parte do autor deste trabalho, na captura e tratamento do áudio e vídeo, aumentando consideravelmente, a produtividade no desenvolvimento deste sistema robótico.

Na figura 5.1, é possível visualizar o sistema operacional do robô em execução em uma máquina *desktop.* Percebe-se o vídeo e o áudio captados do robô, por meio da placa de sintonia de TV, em entrada de vídeo e no espectro calculados respectivamente, provenientes da micro-câmera instalada na frente do robô e do circuito gerador de freqüências, que emite freqüências distintas para cada sensor de toque instalado na frente, dos lados e atrás do robô. Ao lado da entrada de vídeo, está a janela de vídeo processado, que permite visualizar como o sistema operacional está interpretando as informações recebidas.

As setas representam a direção em que o robô deve se mover. Porém, os dados somente são enviados para o robô, se a opção **Enviar dados para LPT** for selecionada. As três opções de *delay*, logo abaixo, são para melhorar o controle do sistema operacional para com o robô. Estas opções devem ser modificadas de acordo com o poder de processamento do computador, que será utilizado para comandar o robô. Os valores de *delay* serão menores para um computador com maior poder de processamento.

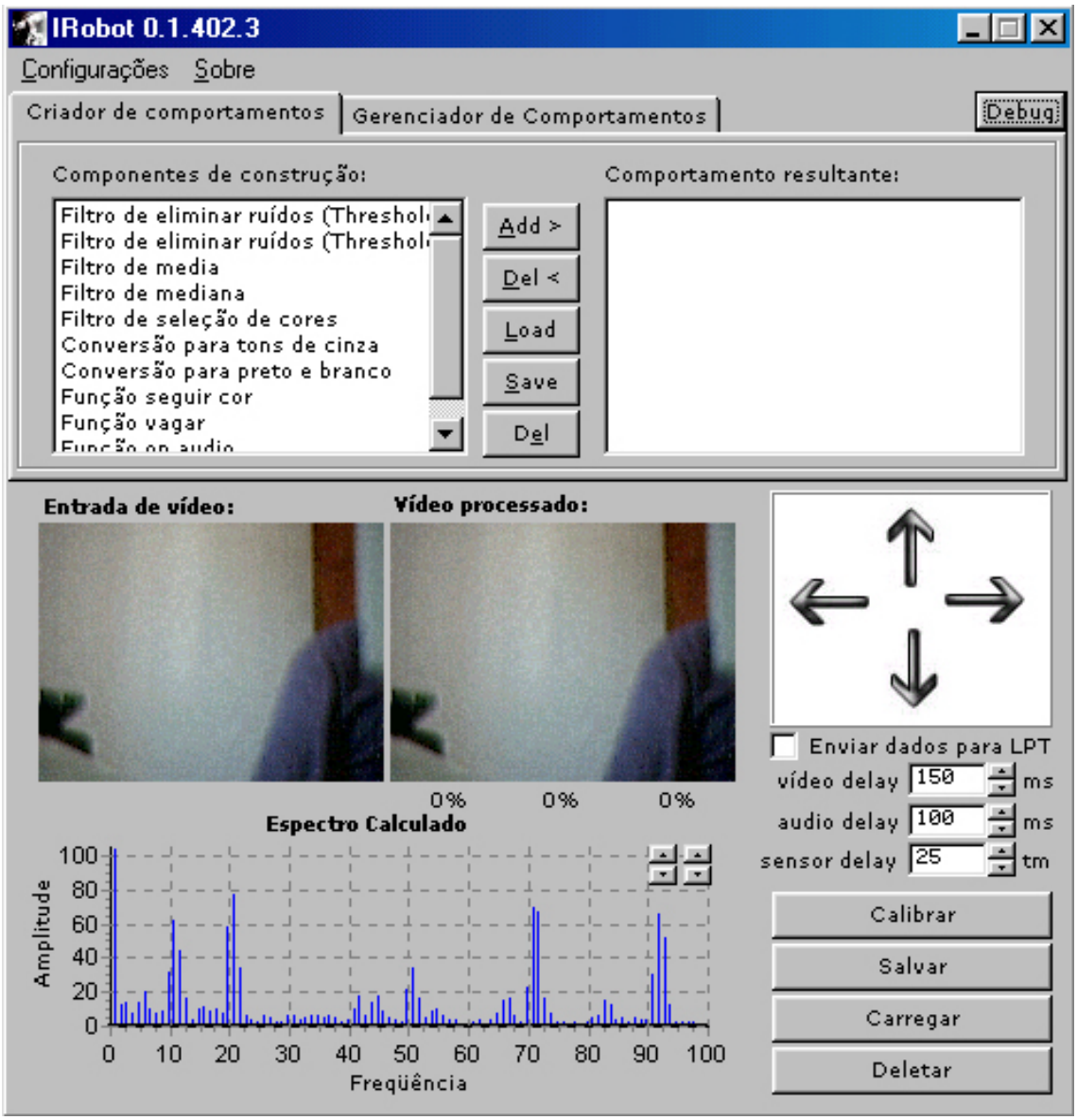

FIGURA 5.1 – Sistema operacional do robô

Após configurar corretamente estas opções, é preciso calibrar os sensores no robô. Para isto, é preciso acionar um determinado sensor de toque manualmente no robô, fazendo com que o robô transmita a freqüência deste sensor para o computador. Depois disto, basta clicar no botão calibrar, que irá capturar cinqüenta amostras desta freqüência, detectar a freqüência ideal e associá-la a um nome, como por exemplo, Sensor 1. Os botões abaixo - salvar, carregar e deletar - servem para que não seja necessário, a todo momento, calibrar os sensores de toque, bastando salvar suas configurações e carregá-las futuramente. Caso, por algum motivo, estas freqüências não estejam mais coincidindo com as configurações salvas, basta excluí-las e calibrar novamente as novas freqüências.

Na parte superior do sistema operacional existe duas opções: a primeira, o criador de comportamentos e a segunda, o gerenciador de comportamentos.

5.1 Construindo comportamentos

A partir da idéia de MAES & BROOKS (1990), este sistema operacional de controle de robôs possui uma biblioteca de procedimentos, de onde é possível, por meio da seleção de funções, filtros e conversores, criar comportamentos distintos, que podem ser armazenados em uma biblioteca de comportamentos e serem utilizados futuramente em conjunto ou não.

5.1.1 Biblioteca de procedimentos

A biblioteca de procedimentos é constituída por uma série de filtros, conversores e funções, que podem ser associados e terem suas variáveis alteradas de diferentes maneiras para obterem comportamentos distintos. (Fig. 5.2.)

A idéia principal é, que esta biblioteca de procedimentos pode vir a receber novos procedimentos a cada nova versão do sistema operacional, que permita sempre estar associando procedimentos diferentes, gerando mais combinações e assim comportamentos novos.

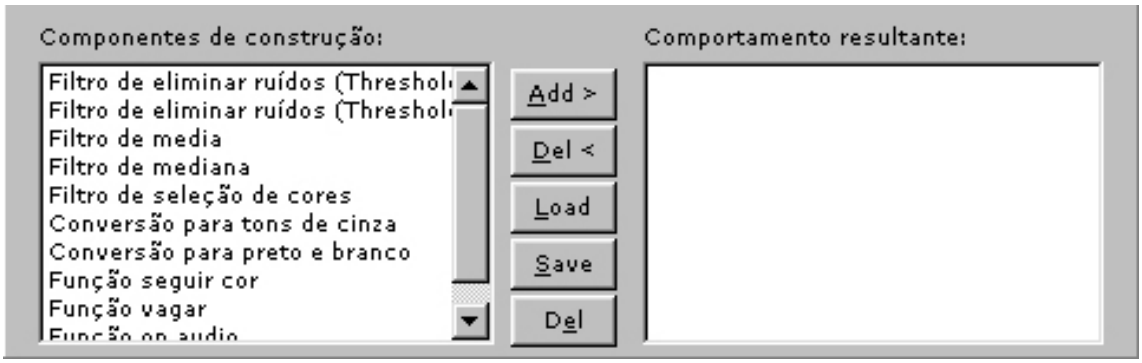

FIGURA 5.2 – Biblioteca de procedimentos

Até o momento em que esta dissertação foi escrita, existiam cinco filtros, dois conversores e três funções, que serão explicadas a seguir com mais detalhes.

5.1.1.1 Filtros

Os filtros foram criados com base em técnicas de processamento de imagens para auxiliar no tratamento e processamento das imagens recebidas do robô. A maioria deles, com intenção de melhorar a imagem recebida, devido à grande inconsistência das imagens provenientes do envio do sinal de vídeo sem a utilização de cabos, totalmente *wireless*.

O primeiro e o segundo filtro são variações do mesmo filtro, conhecido como filtro de eliminação de ruídos ou *threshold*. A idéia é capturar uma seqüência de imagens e fazer uma média entre os pontos capturados. Neste filtro, é possível escolher a quantidade de quadros a ser analisados, otimizando a performance do algoritmo para cada caso. (Fig. 5.3.)

Assim, quanto maior for o número de quadros analisados, melhor será o resultado final, já que a probabilidade de ocorrer ruídos em locais alternados da imagem é maior. O grande inconveniente deste filtro é a necessidade de o robô estar completamente parado, ou seja, a imagem não deve estar em movimento. Isto causa uma certa perda de performance na movimentação do robô.

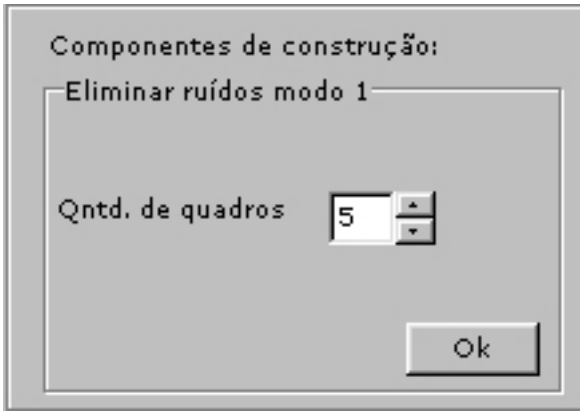

FIGURA 5.3 – Variáveis que podem ser ajustadas nos filtros de eliminação de ruídos

A diferença entre o primeiro e o segundo filtro de eliminação de ruídos é, que o primeiro dá maior peso na última imagem capturada, enquanto que no segundo todas têm o mesmo peso. O primeiro é mais recomendável em ocasiões onde a imagem não esteja totalmente estática, enquanto que no segundo, o ideal é que não exista qualquer movimento durante a análise. O terceiro filtro, denominado filtro de média, trabalha com uma matriz de 3x3, obtida a partir de uma parte da imagem. Este algoritmo soma todos os valores da matriz e o divide por nove, ou seja, pela quantidade de números somados. O resultado obtido substitui o ponto central da matriz que está sendo analisado.

O exemplo seguinte mostra à esquerda uma matriz 3x3 obtida de uma parte de uma imagem capturada pelo robô, e à direita o resultado obtido após a aplicação do

algoritmo. O valor do *pixel* central que era de 48, foi substituído por 70, que é o resultado da expressão: (34+45+46+48+58+89+65+120+122) / 9.

34	120	46
58	**48**	89
122	45	65

➔

34	120	46
58	**70**	89
122	45	65

O filtro de mediana é bem parecido com o filtro da média, pois seu objetivo, assim como os demais é de tentar melhorar a imagem recebida, causando a eliminação de pequenos ruídos na imagem. Neste caso, o filtro também trabalha com uma máscara 3x3. Este filtro ordena os nove valores da matriz 3x3, que representam as cores de cada *pixel* da imagem e obtém o de posição cinco, ou seja, o do *pixel* mediano, e o define como sendo o valor do *pixel* central desta matriz analisada.

Tomando como exemplo, a mesma matriz do exemplo anterior, neste caso seria obtido o valor 58, visto que, os valores da matriz em ordem seriam [34, 45, 46, 48, **58**, 65, 89, 120, 122].

34	120	46
58	**48**	89
122	45	65

➔

34	120	46
58	**58**	89
122	45	65

Os filtros de média e mediana provocam uma espécie de borramento da imagem, que melhora significativamente o resultado obtido para este projeto em especial, além de serem algoritmos extremamente rápidos para os dias de hoje.

O último filtro, chamado de filtro de seleção de cores, tem funcionamento bastante simples e é de grande utilidade, ao construir comportamentos para evitar ou seguir objetos que possuam uma determinada cor. Para utilizá-lo, basta clicar uma única vez na janela de vídeo processado e o valor do *pixel*, no padrão RGB, será associado às variáveis deste filtro (Fig. 5.4.)

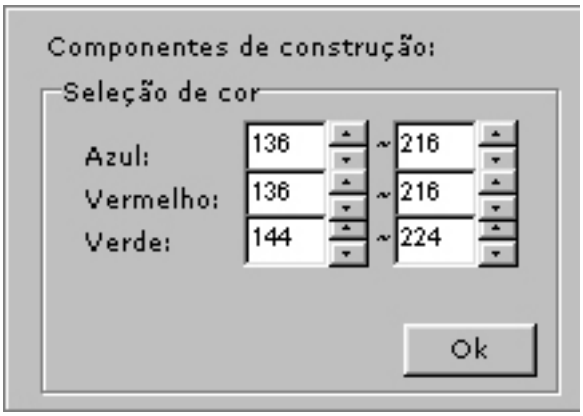

FIGURA 5.4 - Variáveis que podem ser ajustadas no filtro de seleção de cores

Os resultados são bastante satisfatórios com este filtro, que já prevê pequenas alterações de luminosidade, visto que o valor do *pixel* pode estar em uma faixa de cores ao contrário de um único valor. A figura 5.5 demonstra um exemplo deste filtro, onde apenas a cor azul é selecionada na imagem.

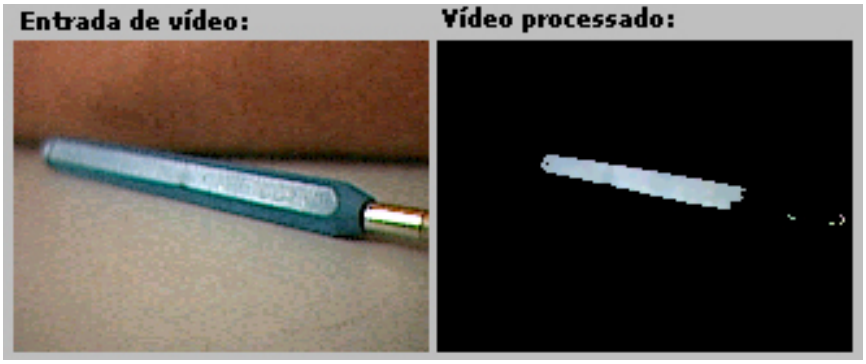

FIGURA 5.5 – Exemplo do filtro de seleção de cores

5.1.1.2 Conversores

Foram desenvolvidos dois conversores para este trabalho, que atuam na imagem capturada pelo robô. O primeiro deles faz a conversão da imagem para tons de cinza e o segundo para uma imagem binária, preto e branco.

Existem três modos disponíveis para se converter a imagem para tons de cinza (Fig. 5.6). No primeiro modo, a conversão é feita por intensidade, onde os três valores do *pixel* RGB são somados e divididos por três. A média obtida nesta equação é o novo valor a ser dado para os três canais de cores RGB.

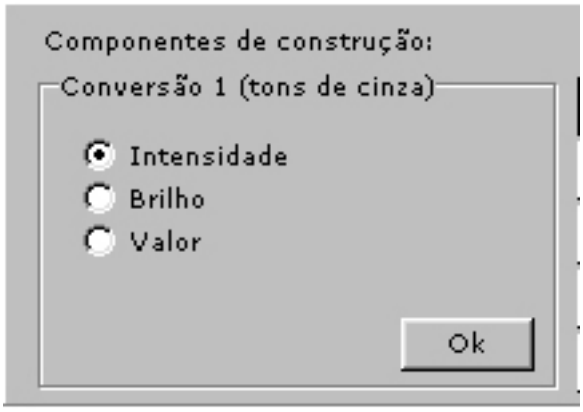

FIGURA 5.6 – Variáveis da conversão para tons de cinza

No segundo modo, a conversão é feita por brilho, onde os valores mais altos e mais baixos do RGB são somados e divididos por dois. O valor resultante é associado aos três canais de cores.

A terceira e última opção de conversão é a conversão por valor, onde o valor mais alto encontrado dos canais RGB é o que será associado ao outros dois canais. A diferença das conversões na prática, para o olho humano, é insignificante. Porém, para o robô pode definir entre uma ação ou outra (Fig. 5.7).

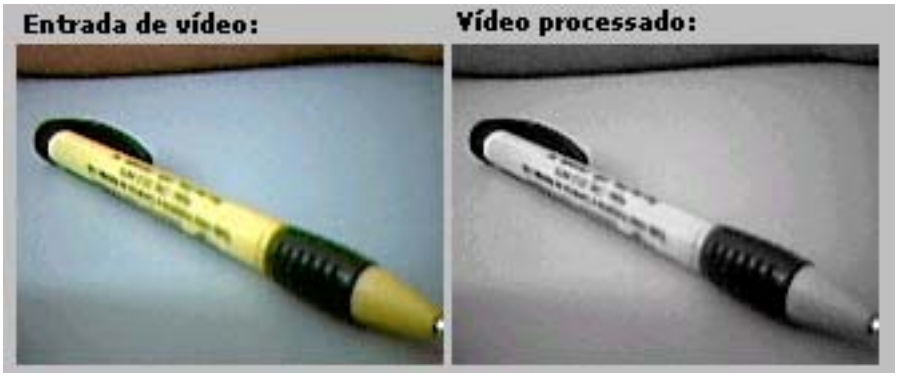

FIGURA 5.7 – Exemplo de conversão da imagem para tons de cinza

O outro conversor existente transforma a imagem RGB em uma imagem binária, preto e branco. Esta conversão possui três modos, assim como o anterior (figura 5.8).

O conversor para imagem binária ainda possui uma opção de inversão de cores. Esta inversão é muito útil, visto que, sua utilização ou não, pode resultar em um comportamento de seguir ou evitar objetos.

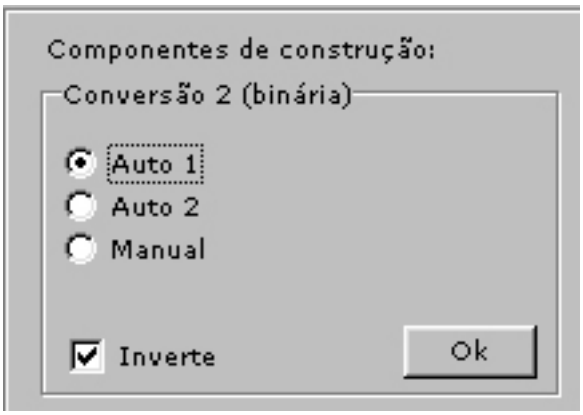

FIGURA 5.8 – Variáveis do conversor para imagem binária

Os dois primeiros modos encontram o ponto de corte, ou seja, qual a cor do *pixel* que define onde o branco e o preto se encontram, automaticamente. Assim, se o ponto de corte for definido como 75 todos os pontos abaixo dele serão pretos e os acima serão brancos (Fig. 5.9).

O primeiro algoritmo soma o valor de todos os pontos da imagem, e divide pela quantidade de pontos somados, encontrando o valor médio de corte, enquanto que o

segundo algoritmo soma o valor mais alto e o mais baixo encontrado, e divide este valor por dois, atribuindo-o ao ponto de corte.

Existe ainda um terceiro modo de conversão, que é a escolha manual, apontando a cor do ponto de corte, ou seja, qual a cor do *pixel*, que define onde o branco e o preto se encontram.

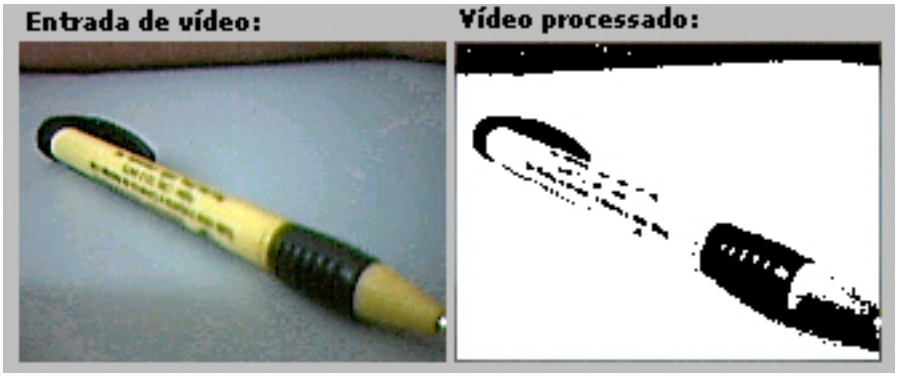

FIGURA 5.9 – Exemplo de conversão binária

5.1.1.3 Funções

Todo comportamento deve possuir uma função. São os resultados das funções que informam ao robô o que ele deve fazer. As funções retornam valores correspondentes a uma direção a ser tomada pelo robô ou um resultado de erro com valor –1.

O resultado de erro é de grande utilidade e informa ao gerenciador de comportamentos que um comportamento falhou, ou simplesmente que é hora de dar chance a um outro comportamento de executar.

O atual sistema operacional do robô conta com três funções distintas: função vagar, função de seguir cor e função *OnAudio.*

A função vagar é a mais simples e mais comum de ser encontrada. O resultado desta função pode ser quaisquer direções a serem tomadas com movimento para frente.

A escolha é feita aleatoriamente por uma função de escolha randômica, que pode inclusive escolher o valor de erro –1.

A segunda função, denominada função de seguir cor, está relacionada ao sistema de visão do robô. A imagem capturada pela câmera, depois de processada com filtros e conversores, é utilizada para definir o que deve ser perseguido pelo robô. Para isto, basta selecionar a cor na janela de vídeo processado, selecionar o mínimo de quadros que devem ser encontrados e ativar a função. Quando esta função não encontra o mínimo de pontos que foram selecionados, ela retorna também o valor de erro –1.

A janela de vídeo processado será dividida em três partes iguais e na parte inferior será informada a porcentagem de *pixels* obtida da cor selecionada (Fig. 5.10).

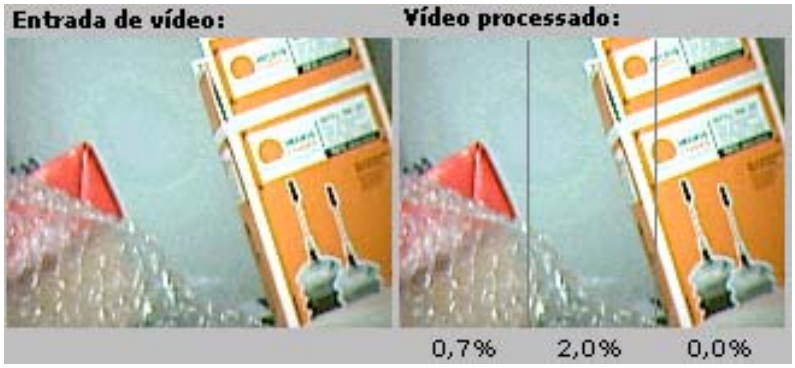

FIGURA 5.10 – Exemplo da função que segue cores

A última função, chamada de *OnAudio*, está relacionada aos sensores de toque que estão no robô. Esta função retorna o valor de erro –1 toda vez que o sensor não é detectado.

Como cada sensor do robô emite uma freqüência de som distinta, ao ser acionado, esta função utiliza a Transformada Rápida de Fourrier – FFT, para determinar qual freqüência pertence a qual sensor e executar uma seqüência de movimentos específicos. A figura 5.11 mostra um exemplo no qual, quando o sensor 1 for acionado no robô, ele irá se mover para trás durante 500 milissegundos.

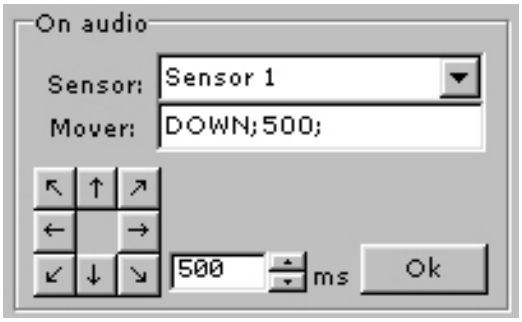

FIGURA 5.11 – Função *OnAudio*

5.1.2 Associando procedimentos

Diferentes respostas de comportamento podem ser obtidas com uma mesma função, isto é possível graças a diferentes associações entre os procedimentos existentes e os diferentes valores de suas variáveis.

Um comportamento pode ser formado por um único procedimento, no caso uma função, ou constituído por uma série de procedimentos para que o comportamento consiga um resultado qualquer desejado.

Um exemplo de um comportamento seria a função vagar, que não precisa que nenhum outro procedimento seja adicionado, para que execute sua função corretamente.

Por outro lado, um comportamento que deveria seguir um objeto de cor verde escuro, poderia obter melhores resultados com a utilização de outros procedimentos, tais como filtros de mediana, seleção de cor e outros.

5.2 Um gerenciador de comportamentos

Assim como em um gerenciador de processos em um computador, que é capaz de decidir qual e por quanto tempo um processo deve rodar, o mesmo deve acontecer com o robô, que deve saber tomar a decisão - se um comportamento deve dar lugar a outro ou continuar sendo utilizado.

Após vários comportamentos terem sido criados, é preciso poder encontrar uma forma de gerenciá-los de forma correta, para que realizem corretamente a arquitetura de *Subsumption,* tornando o robô capaz de realizar diferentes comportamentos para cada resultado obtido.

Desta forma, assim como o gerenciador de processos, o gerenciador de comportamentos deve utilizar algoritmos de escalonamento e cada comportamento deve possuir estados que vão definir se deve executar, aguardar ou dar lugar a outro comportamento (Fig. 5.12).

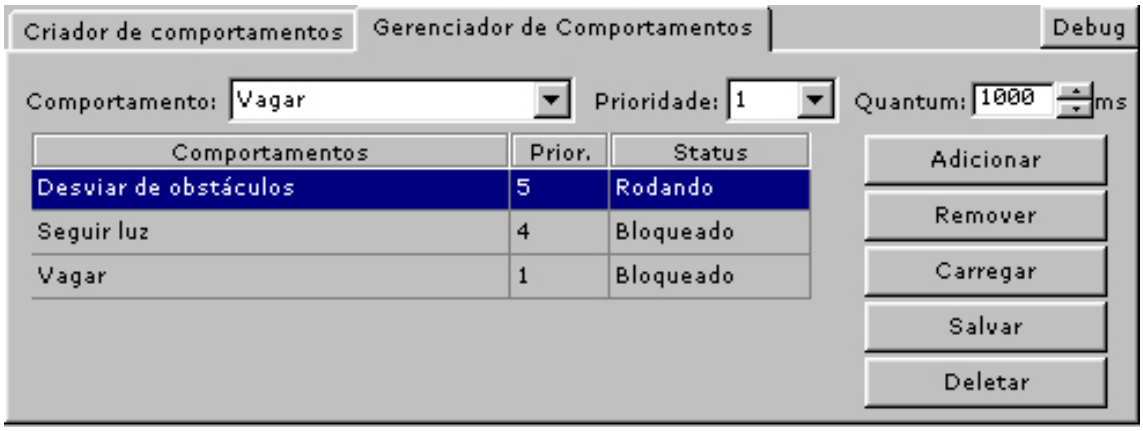

FIGURA 5.12 – Gerenciador de comportamentos

Utilizando-se as regras do escalonamento por prioridade, do mesmo modo que um processo, um comportamento com maior prioridade que outro é sempre executado primeiro. O escalonamento não-preemptivo, faz com que os outros comportamentos de prioridade menor devem fiquem aguardando, enquanto o comportamento de prioridade maior não é terminado.

O gerenciador de comportamentos também utiliza o algoritmo de escalonamento *Round Robin*. O gerenciador possui uma variável de ajuste de tempo chamada *quantum,* que define qual o intervalo o comportamento deve continuar rodando. Após passar o tempo *quantum*, o gerenciador passa a vez para outro comportamento de mesma

prioridade daquele que esteja rodando, caso exista. Se não existir, o gerenciador reinicia seu procedimento, fazendo com que o comportamento de maior prioridade seja executado. O comportamento que estiver rodando, não é interrompido e outro comportamento só irá iniciar caso o comportamento que estiver rodando assumir o estado de terminado, independente de retornar sucesso ou fracasso.

No próximo capítulo, será demonstrado, na prática, a utilização do sistema robótico criado e sua coerência de funcionamento de acordo com a arquitetura de *Subsumption.*

6 Estudo de caso

Este capítulo é dedicado a mostrar os resultados, na prática, da utilização do sistema operacional de controle de robôs que foi criado.

6.1 Comportamento vagar

Os resultados do comportamento vagar não sofrem grandes mudanças, uma vez que a função vagar não obtém dados variáveis de câmera ou sensores. Porém, o caminho que o robô toma, sempre é diferente devido à escolha aleatória de movimentos proporcionados pela função.

Se o robô colide com algum obstáculo, ele ficará preso a este até que, por ventura, algum de seus movimentos aleatórios o retire do lugar.

Este é um típico comportamento de exploração de ambientes, no qual o robô simplesmente se move sem destino ou objetivos.

6.2 Comportamento seguir cor

O comportamento de seguir cor obtém diversos resultados diferentes, dependendo do modo que suas variáveis são ajustadas. A utilização de filtros e conversores é imprescindível na eliminação de ruídos obtidos pela comunicação sem fio.

Percebe-se isto já na imagem captada, que por ser enviada sem a utilização de fios do robô para o computador, é comum chegar com uma grande quantidade de ruídos. Os ruídos obtidos durante esta transmissão de imagens podem resultar em ações errôneas, que poderiam ser evitadas com o tratamento da imagem com os filtros disponíveis.

A figura 6.1 mostra como a tomada de decisão de um comportamento pode ficar comprometida, se não for feito um tratamento adequado na imagem com a utilização dos filtros existentes. A imagem A é a imagem original com ruídos onde o robô deve seguir a faixa branca no chão. Em B e C percebe-se a diferença da imagem não tratada adequadamente e da imagem pré-processada pelos filtros existentes de média, mediana etc.

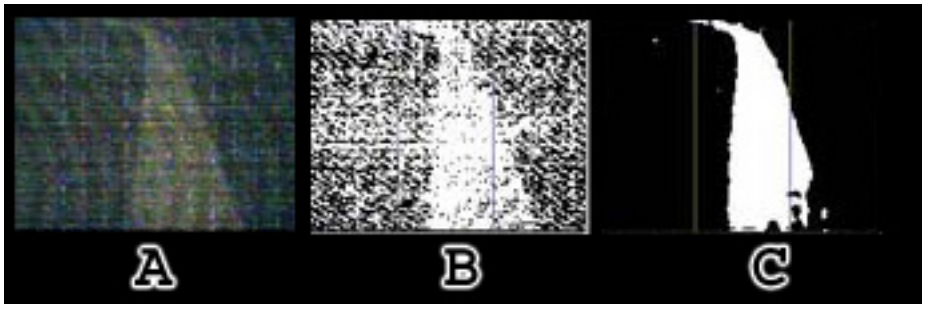

FIGURA 6.1 – (A) Imagem recebida com ruído. (B) Resultado da imagem sem o tratamento adequado com os filtros. (C) Resultado da imagem com a utilização adequada dos filtros

Este mesmo comportamento pode ser utilizado com outras finalidades como, por exemplo, atingir um objeto luminoso. Na figura 6.2, pode-se observar as imagens capturadas pelo robô, do início onde se encontrava até o final do processo, quando consegue atingir o objeto luminoso.

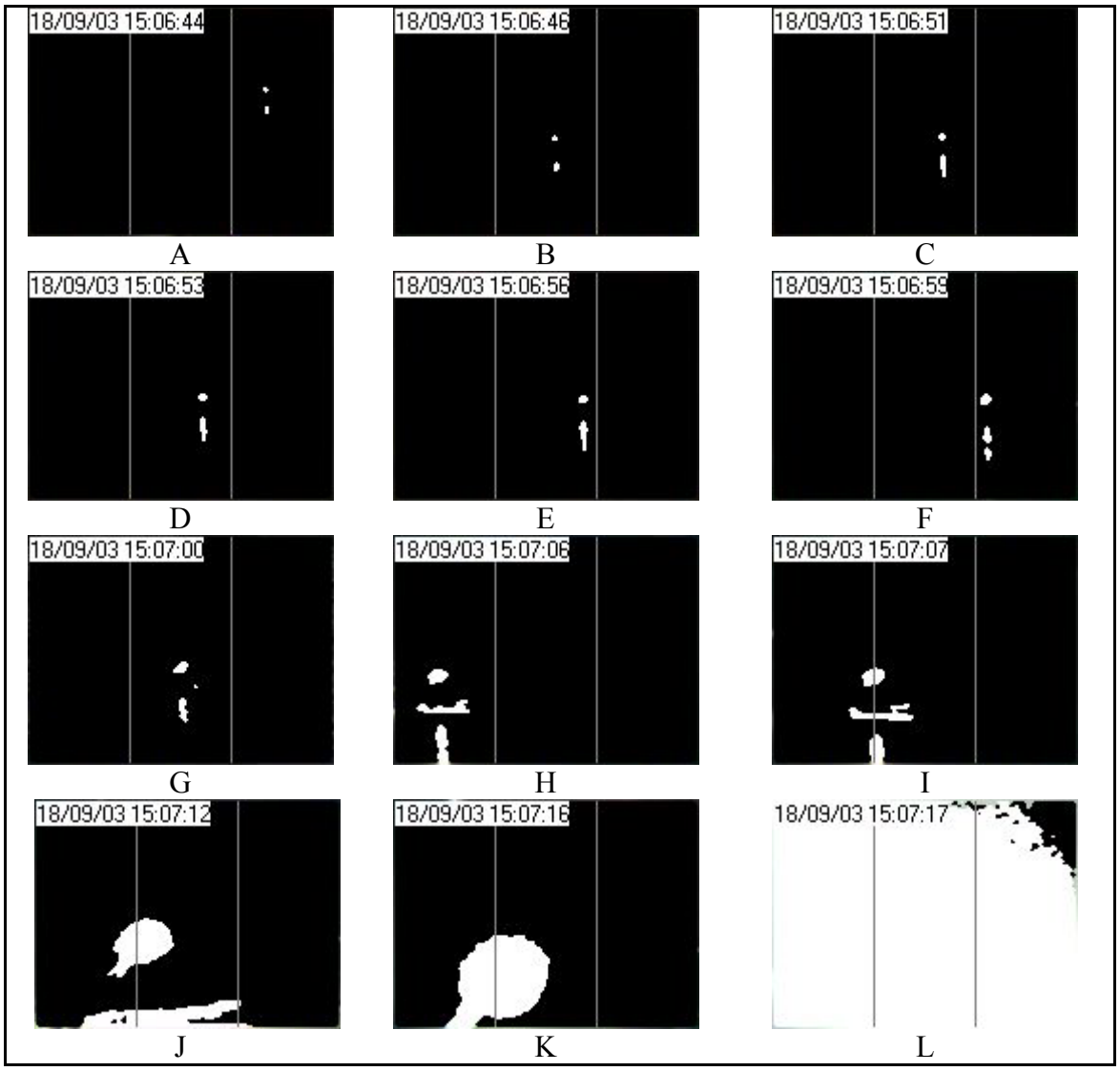

FIGURA 6.2 – Visão do robô ao executar o comportamento de seguir luminosidade

Pelas seqüências da figura 6.2, percebe-se o funcionamento do algoritmo que segue cores. No caso acima, a imagem capturada foi convertida para preto e branco, onde o branco significa os objetos mais luminosos. Sempre que o objeto luminoso fica fora do centro (Fig 6.2 a, f, h), o robô se movimenta para os lados para posicionar o objeto luminoso no centro novamente (Fig 6.2 b, g, i), conseguindo no final do seu percurso encontrar o objeto luminoso com grandes chances de sucesso (Fig 6.2 l).

6.3 Comportamento desviar de obstáculos

A utilização do comportamento *OnAudio* pode simular o comportamento desviar obstáculos perfeitamente. Quando um sensor fecha ao encostar-se a um objeto qualquer, uma determinada freqüência é enviada e reconhecida em sistema operacional do robô, que consegue identificar qual o sensor do robô foi acionado.

Selecionando, por exemplo, para o sensor frontal os movimentos para trás 300ms, esquerda à frente 100ms, direita à frente 100ms, simula exatamente um movimento de desvio de objeto. (Fig. 6.3)

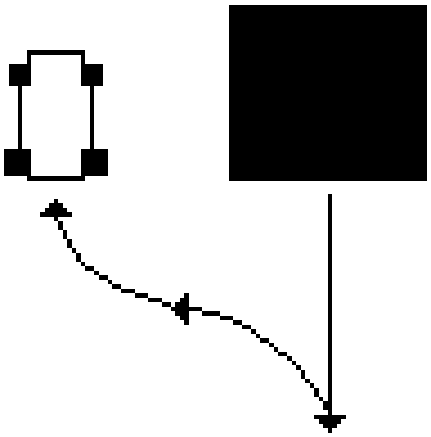

FIGURA 6.3 – Simulação de um comportamento de desvio de obstáculo

A rota de desvio pode ser planejada diferente para cada sensor do robô. Assim, se durante o desvio ele encontrar um outro obstáculo, uma nova rota de desvio é acionada fazendo o desvio de obstáculos mais eficaz.

O comportamento O*nAudio* poderia ser usado também para fazer seguir objetos, para isto bastaria trocar os sensores de toque por sensores de ultra-som e planejar rotas diferentes para cada um dos sensores. Por exemplo, se encontrar um objeto a frente, vá para frente, se encontrar ao lado esquerdo vá para esquerda e assim por diante.

6.4 Utilização de comportamentos múltiplos no gerenciador de comportamentos

Nesta parte do trabalho, é possível perceber como a implementação da biblioteca de comportamentos descrita por MAES & BROOKS, conceitos e fundamentos de gerenciamento de processos em sistemas operacionais e a arquitetura de *Subsumption* resultam em um sistema robótico de comportamentos dinâmicos.

Em um ambiente controlado foi inserido o robô em uma extremidade, uma fonte luminosa a ser atingida em outra extremidade e ao centro um obstáculo. Foram criados alguns comportamentos, para serem utilizados no gerenciador de comportamentos, para que o objetivo de chegar até o objeto luminoso fosse atingido.

O primeiro comportamento foi criado com apenas a função vagar e o objetivo dele é apenas vagar em um ambiente. Ele foi definido no gerenciador de comportamentos, como tendo prioridade 1, ou seja, a prioridade mais baixa de todos os comportamentos a serem adicionados. Assim, no caso dos outros comportamentos falharem, o robô começará a vagar pelo ambiente, até que outro comportamento de maior prioridade assuma o controle do robô.

O segundo comportamento criado foi de seguir luminosidade. Foram adicionados filtros de média e mediana para retirada de ruídos, o filtro de conversão para tons de cinza com conversão feita por valor, o filtro de conversão para preto e branco com conversão manual de corte em 220 e a função seguir cor com a variável azul, vermelho e verde em 255 e com o mínimo de 10 pontos para validar a função. Sua prioridade foi definida como 2, ou seja, logo acima do comportamento vagar. Assim, atingir o objeto luminoso é mais importante do que simplesmente vagar pelo ambiente.

Foi feita uma calibragem dos sensores, para que pudessem ser criados comportamentos de desvio de obstáculos, desta forma os sensores ficaram da seguinte forma:

- Sensor 1 – sensor frontal direito.
- Sensor 2 – sensor frontal.
- Sensor 3 – sensor frontal esquerdo.
- Sensor 4 – sensor traseiro.

Assim o terceiro comportamento criado foi o desvio de obstáculo frontal, com apenas o comportamento *OnAudio* definido. Assim, quando o sensor 2 for acionado, o movimento a ser executado seria: 300ms para trás, 200ms para trás à esquerda , 200 ms para a direita à frente e 100ms para a esquerda à frente. Este comportamento foi definido tendo prioridade 10 e maior do que qualquer outro comportamento. Assim, desviar de um obstáculo frontal é mais importante do que tentar atingir um objeto luminoso, vagar, ou desviar de obstáculos em posições diferentes.

O quarto comportamento criado foi o desvio de obstáculo lateral esquerdo, também apenas com o comportamento *OnAudio* definindo que, quando o sensor 2 for acionado, o movimento a ser executado seria 300ms para trás à esquerda. Da mesma forma o comportamento desvio de obstáculo lateral direito foi criado, porém definindo que, quando o sensor 3 for acionado, o movimento a ser executado seria 300ms para trás à direita. Os dois foram definidos com prioridade 8 no gerenciador de comportamentos, tendo menos importância apenas do que o desvio de obstáculos frontais.

O último comportamento criado foi o de rota de emergência, supondo que em algum movimento de desvio, o robô (frontal ou lateral) pudesse acabar fazendo com que o robô colidisse a parte traseira em um outro obstáculo. Assim, o movimento

determinado foi o de sempre que o sensor 4 for acionado, o movimento a ser executado deveria ser movimentar 200ms para a esquerda à frente. Sua prioridade foi definida como 6, sendo apenas menor que o desvio frontal ou lateral, porém mais importante que atingir um objeto luminoso ou vagar.

A seqüência da figura 6.4 mostra o movimento do robô em um ambiente controlado, no qual seu objetivo é atingir um objeto luminoso. O robô se move em direção ao objeto luminoso, utilizando o segundo comportamento criado (Fig 6.4 A-D), porém, colide frontalmente com um objeto (Fig 6.4 E) e, então, ativa o terceiro comportamento de desvio de obstáculo frontal (Fig 6.4 F-K). Depois de efetuado este comportamento, ele retorna ao segundo comportamento de seguir luminosidade, atingindo o objeto luminoso (Fig 6.4 L).

Na seqüência da figura 6.4, pode-se observar um exemplo perfeito de como o robô pode atingir um objeto luminoso, mesmo quando algum obstáculo encontra-se em seu caminho. Porém, na seqüência da figura 6.5, observa-se que, nem sempre isto acontece de forma tão simples.

Na seqüência da figuras 6.5, o robô tenta primeiramente atingir o objeto luminoso, já que nenhum sensor está acionado (Fig 6.5 A-B). Porém, o robô colide uma primeira vez frontalmente com um obstáculo (Fig 6.5 C) e efetua o comportamento de desvio de obstáculo frontal que possui maior prioridade. Ao tentar o desvio (Fig 6.5 D-G), ele volta a tentar seguir o objeto luminoso, porém, colide uma segunda vez (Fig 6.5 H-J), e ao tentar o desvio de obstáculo frontal, novamente, a fonte luminosa sai do foco da câmera do robô. Como o robô não encontra obstáculos e não encontra uma fonte luminosa para ser atingida, entra em ação o comportamento vagar de menor prioridade (Fig 6.5 K-L).

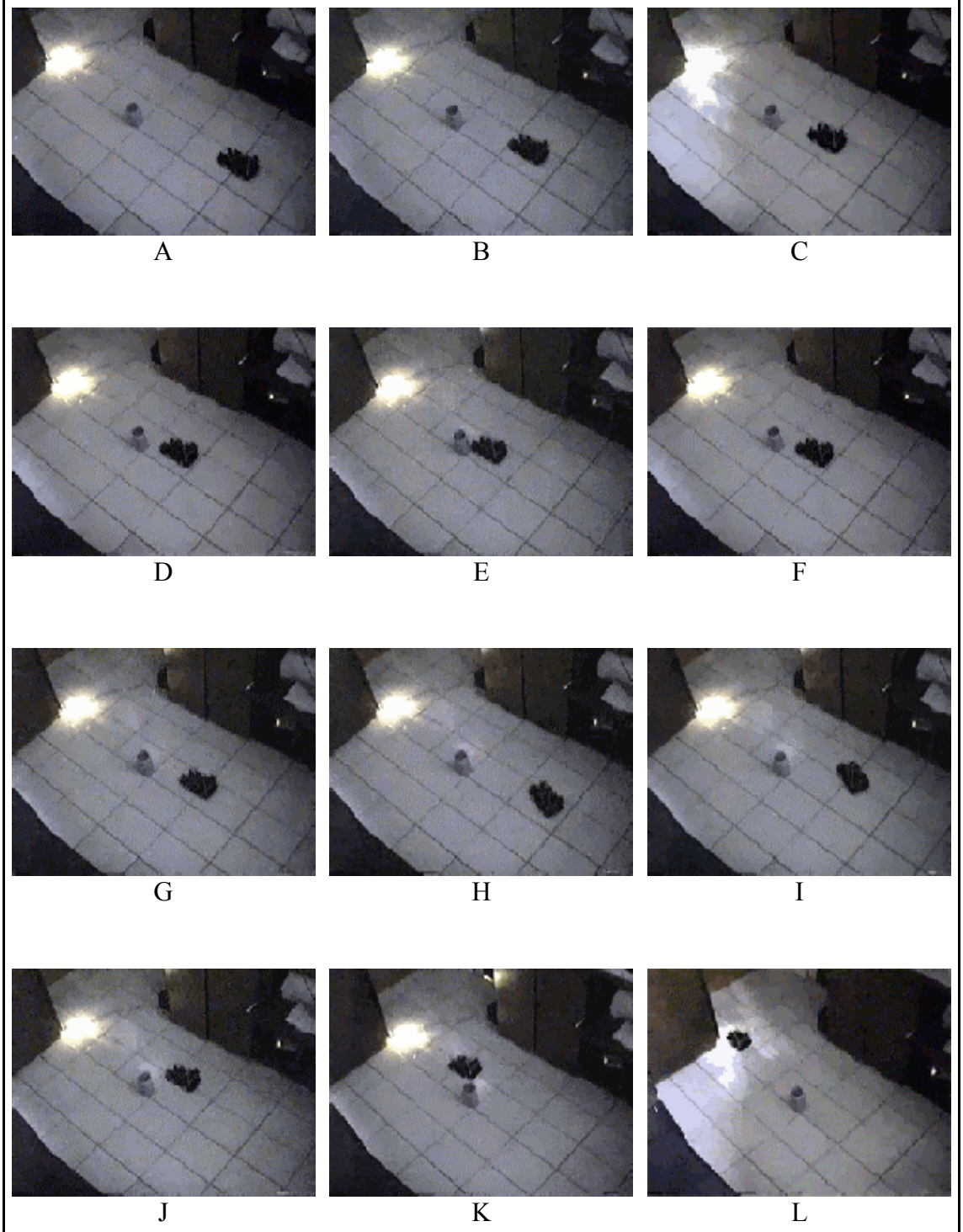

FIGURA 6.4 – Seqüência de imagens do robô efetuando desvio de obstáculo e atingindo um objeto luminoso

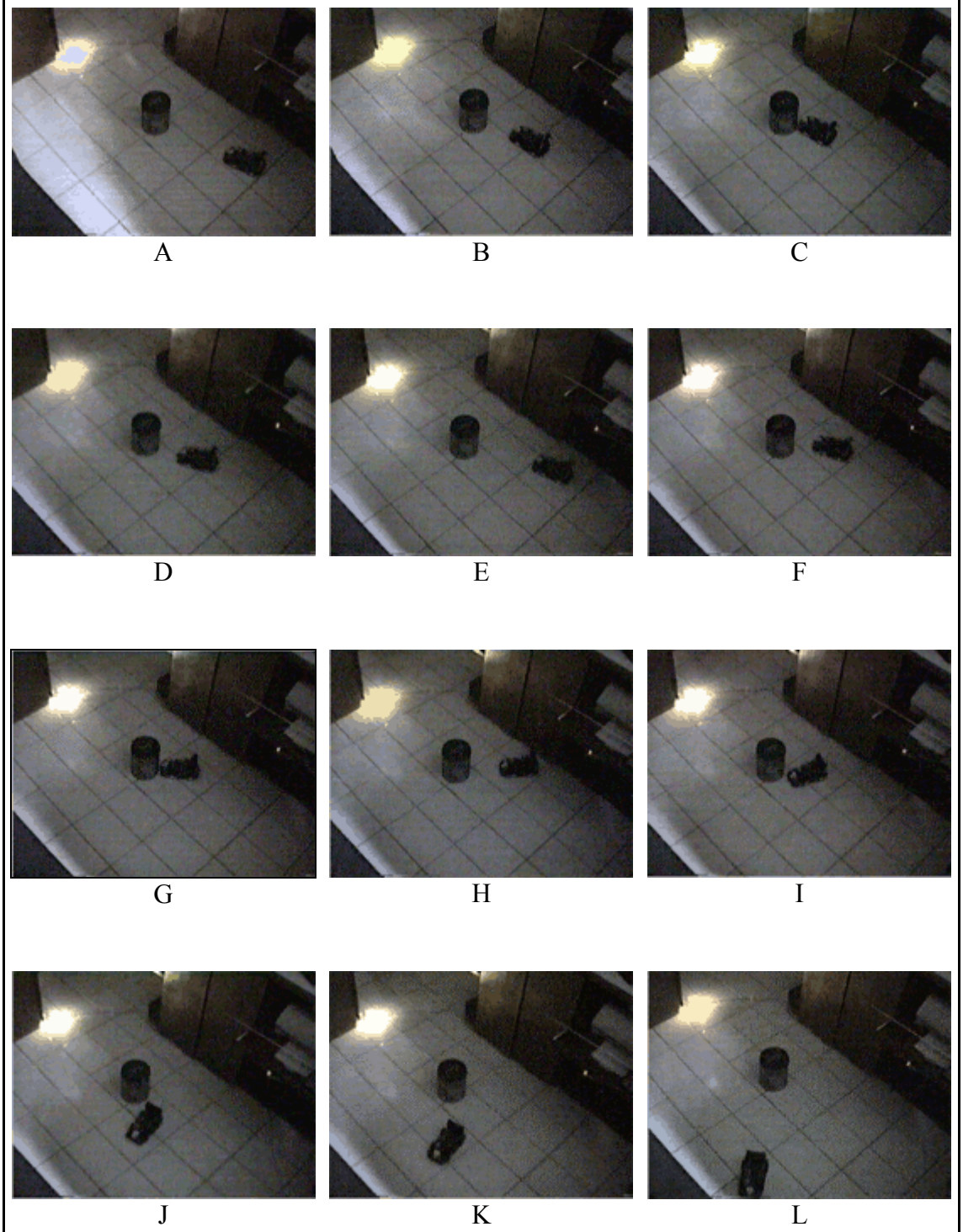

FIGURA 6.5 – Seqüência de imagens do robô executando comportamento de seguir luminosidade, desvio de obstáculo e vagar.

Na seqüência da figura 6.5, pode-se observar que o robô acabou se distanciando da fonte luminosa, visto que o comportamento desviar de obstáculo não funcionou de maneira esperada. Porém, isto não pode ser considerado uma falha no processo, pois mesmo sem encontrar a fonte luminosa após a segunda tentativa de desvio de obstáculo frontal, ele ativou seu comportamento vagar, que, em um ambiente controlado como este, levaria o robô a encontrar novamente a fonte luminosa.

É certo que o robô levaria mais tempo para atingir o objeto luminoso, porém, após vagar durante algum tempo e possivelmente encontrar mais obstáculos pela frente, ele acabaria por encontrar a fonte luminosa, conseguindo atingir seu objetivo mais cedo ou mais tarde.

7 Conclusões e trabalhos futuros

Atualmente, a construção e o desenvolvimento de robôs, especialmente os baseados em comportamentos, se mostram projetos caros e complicados para profissionais, estudantes e pesquisadores da área de Ciência da Computação. As metodologias de controle e de projeto utilizadas até então, em sua maioria, se mostram projetos inviáveis e complexos, exigindo inevitavelmente o auxílio de profissionais de engenharias Mecânica e Elétrica.

Este trabalho apresentou uma nova metodologia de controle, construção e desenvolvimento de robôs baseados em comportamentos dinâmicos de baixo custo e ainda dispensa conhecimentos profundos na área de engenharia Mecânica e/ou Elétrica. Isto foi possível graças à utilização, em sua maioria, de produtos prontos e de custo relativamente baixo em termos de construção e desenvolvimento de robôs e do estudo de robôs baseados em comportamentos, sistemas operacionais, processamento de imagens, reconhecimento de padrões, Transformada de Fourier, sensores e comunicação de dados sem fio.

Também se pode perceber que um sistema robótico de comportamentos dinâmicos, como o apresentado neste trabalho, é de grande utilidade já que pode ser

facilmente usado em diversas tarefas diferentes. Tudo isto, sem mencionar, a facilidade de gerar novos procedimentos ou simplesmente associar os já existentes e modificar suas variáveis, para formar novos comportamentos. Como exemplo, podemos citar a criação de duas funções novas, que não foram descritas neste trabalho, para que possibilitasse a criação das seqüências da figura 6.2 deste trabalho. A primeira função foi a de adicionar data e hora; e, a segunda, a de gravar imagens em disco.

Mostrou-se neste trabalho como as técnicas para gerenciamento de processos podem ser aplicadas para gerenciar comportamentos simulando a arquitetura de *Subsumption,* tornando simples e visível a gerência dos comportamentos criados e a materialização do conceito de biblioteca de comportamentos, através de uma biblioteca de procedimentos com funções, filtros e conversores, que podem ter suas variáveis modificadas para que atuem de forma diferente, ou mesmo, que possibilite a adição de novos procedimentos de forma simples.

A grande facilidade de adição de novos procedimentos e conseqüentemente o aumento de combinações e possibilidades de criação de novos comportamentos permite imaginar diversas utilidades para um sistema robótico como este. Por exemplo, em indústrias que procuram economizar gastos com robôs, que executam tarefas variadas em momentos e/ou situações diferentes dentre outras.

Pode-se imaginar ainda, um sistema robótico como este em residências com espaço limitado, que necessitam de um único robô para limpeza de dia e para segurança de noite; ou, empresas onde funcionários possam acessar dados pela internet, mas que às vezes precisam estar presentes na empresa para realizar uma determinada tarefa manualmente. Para isto, bastariam acessar um computador com o sistema operacional

do robô e criarem um comportamento para executar uma tarefa ou selecionar um previamente criado e colocar no gerenciador de comportamentos.

Assim, um robô na empresa poderia substituir a presença física de um funcionário, já que o robô poderia por meio de um determinado comportamento ou vários comportamentos, acionar um determinado botão. Pode-se pensar ainda em tarefas mais rotineiras, como levar uma ferramenta para o setor de mecânica seguindo uma faixa azul no chão, levar papéis para a administração seguindo uma faixa vermelha ou levar o lixo coletado por um comportamento anterior para fora da empresa seguindo uma faixa amarela.

A adição de novos sensores e procedimentos à biblioteca e controle do sistema robótico por meios diferenciados como a internet, são fatores que merecem ser considerados em trabalhos futuros. Pode-se ainda citar a melhoria neste mesmo trabalho, como a utilização de ultra-som ou infravermelho para detecção de obstáculos e a utilização de um meio mais eficiente para o controle dos dados enviados para o robô, a fim de garantir que um dado seja recebido e que ele seja executado com maior precisão.

Devido ao alto nível de processamento exigido pelo sistema operacional deste sistema robótico ao interagir com imagem, áudio, criar processos separados para serem enviados para o decodificador e ainda executar os comportamentos que podem ser criados de forma dinâmica, testes comprovaram instabilidade do sistema em um computador com processador Pentium 233MMX com 64 megas de memória RAM e apenas 10 megas livres no disco rígido. Porém este problema torna-se cada vez menor a cada dia, devido ao aumento do poder de processamento dos computadores e da evolução continua da tecnologia na computação.

Embora este sistema não possa ser considerado um sistema de tempo real e possua algumas imperfeições, pode-se concluir que ele atingiu os objetivos propostos de forma satisfatória, contribuindo para a evolução da robótica e adicionando uma visão diferente na construção de sistemas operacionais para sistemas robóticos.

REFERÊNCIAS BIBLIOGRÁFICAS

ARKIN, Ronald C. *Behavior-based robotics*. 2.ed. Cambridge: Massachusetts Institute of Technology, 1999. 491p.

FARIA, Elmo Batista de. *Controle de sistemas robóticos acoplados usando lógica nebulosa.* Uberlândia: Universidade Federal de Uberlândia, Faculdade de Engenharia Elétrica, 2003. 168p. (Tese, Doutorado em Engenharia Elétrica).

MAES, Pattie, BROOKS, Rodney A. *Learning to coordinate behaviors*. Cambridge: MIT Press, 1990.

MESSIAS, Antônio Rogério. *Comunicaão com a porta paralela.* Disponível na Internet: http://www.rogercom.com/pparalela/introducao.htm#inicio . 20 outubro 2002.

TANENBAUM, Andrew. *Sistemas operacionais modernos*. Trad. Nery Machado Filho. Rio de Janeiro: LTC, 1992. 493p. (Tradução de: *Modern operating system).*

ZAMBIASI, Saulo P., PAULA, Maurício B. de, SILVA, Juarez B. da. *Robôs baseados em comportamento*. Florianópolis: UFSC, 2001.

BIBLIOGRAFIA

BALLARD, C. Brown. *Computer vision.* Englewood-Cliffs: Prentice Hall, 1982.

CANTÙ, Marco. *Dominando o Delphi 3*. Trad. Makron Books; revisão técnica Álvaro Antunes e Marcos Jorge. São Paulo: Makron Books, 1997. 1090p. (Tradução de: *Mastering Delphi 3).*

CRAIZER, M., GRIVET, M.A., TAVARES, G. *Processamento de imagens e modelagem em biologia.* Petrópolis: II Escola de Verão, Laboratório Nacional de Computação Científica - LNCC, 2001. p

FRADEN, Jacob. *Handbook of modern sensors.* 2.ed. Woodbury: American Institute of Physics, 1996. 556p.

GONZALES, R. C.,WINTZ, P. *Digital image processing.* 3.ed. Massachussets: Addison Wesley, 1992.

GUIMARÃES, Alexandre Teodoro. *Utilização de sensores IR na transmissão digital de dados.* Uberlândia: UNIT, 2001. 49p. (Monografia, Curso de Ciência da Computação do Centro Universitário do Triângulo – UNIT).

HIRATA, Roberto Jr. *Segmentação de imagens por morfologia matemática*. São Paulo: USP, 1997.

MINI-TRANSMISSORES & RÁDIO-RECEPTORES: 13 montagens práticas. Rio de Janeiro: Seleções Eletrônicas, c1990. 64p.

MOURA Jr., José dos R.V. de, GUIMARÃES, Alexandre T., ESTEVES, Andreza M., RIBEIRO, Márcio J. *Utilização de técnicas de processamento digitais de imagens no desenvolvimento de ferramentas computacionais de auxílio à interpretação humana.*

In: ENCONTRO REGIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 6, 2001, Uberlândia. *Anais*... Uberlândia: Universidade Federal de Uberlândia, 2001. 90p. p.40-44.

OSCILADOR Astavél 555. Disponível na Internet: http://users.hotlink.com.br/rmenezes. 21 março 2003.

PARKER, J.R. *Practical Computer vision using C.* New York: John Wiley & Sons, 1994.

REVISTA ELETRÔNICA TOTAL: Fora de série. São Paulo: Saber, v.1, n.3, fevereiro 2002.

WHITE, Robert. Chromatography*: Fourier transform infrared spectroscopy and its applications.* New York: M. Dekker, 1990. 328p.

GLOSSÁRIO

VHF – ou Very High Frequency é a faixa de operação de transmissão dos canais 1 a 13 da televisão que ficam entre 30 até 300Mhz.

DIODO – um tipo simples de dispositivo semi-condutor; um tipo de material que é capaz de conduzir eletricidade de forma variada.

NTSC – ou National Television Standards Commitee é o formato padrão utilizado no Norte da América e Japão, criado em 1953, capaz de mostrar 525 linhas de resolução.

RGB – ou Red Green and Blue se refere a um sistema para representar as cores no computador. As cores vermelho, verde e azul são combinadas em varias proporções para obter qualquer cor visível.

PIXEL – é um ponto, que juntamente com outros é capaz de formar uma imagem.

TRANSFORMADA RÁPIDA DE FOURIER – algoritmo melhorado por matemáticos na década de 60 com base na Transformada de Fourier, porém, mais simples de computar.

WIRELESS – comunicação sem fio.