

ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL

TESE DEFENDIDA POR Arthur de Miranda

Neto

E APROVADO

PELA COMISSÃO JULGADORA EM 26.10.2007

Douglas E. Zampieri
ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Navegação de Robôs Autônomos baseada em Monovisão

Autor: Arthur de Miranda Neto

Orientador: Douglas Eduardo Zampieri

Co-orientador: André Mendeleck

26/2007

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

Navegação de Robôs Autônomos baseada em Monovisão

Autor: Arthur de Miranda Neto
Orientador: Douglas Eduardo Zampieri
Co-orientador: André Mendeleck

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Dissertação de mestrado acadêmico apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2007
S.P. – Brasil

UNIDADE BC
CHAMADA: T/UNICAMP M672n
EX. 75159
OMBO BCCL 16.145-07
ROC D X
REÇO 400
ATA 05/12/07
IB-ID 418211

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

✓ M672n Miranda Neto, Arthur de
Navegação de robôs autônomos baseada em
monovisão. / Arthur de Miranda Neto. --Campinas, SP:
[s.n.], 2007.

Orientadores: Douglas Eduardo Zampieri, André
Mendeleck

Dissertação (mestrado) - Universidade Estadual de
Campinas, Faculdade de Engenharia Mecânica.

1. Robôs móveis. 2. Veículos autônomos. 3.
Navegação de robôs móveis. 4. Visão por computador.
5. Inteligência artificial. 6. Algoritmos genéticos. I.
Zampieri, Douglas Eduardo. II. Mendeleck, André. III.
Universidade Estadual de Campinas. Faculdade de
Engenharia Mecânica. IV. Título.

Título em Inglês: Autonomous robot's navigation based on monocular vision

Palavras-chave em Inglês: Mobile robots, Autonomous vehicles, Navigation,
Computer vision, Artificial intelligence, Classifying
systems, Genetic algorithms

Área de concentração: Mecânica dos Sólidos e Projeto Mecânico

Titulação: Mestre em Engenharia Mecânica

Banca examinadora: Clésio Luis Tozzi e Eurípedes Guilherme de Oliveira
Nóbrega

Data da defesa: 26/02/2007

Programa de Pós-Graduação: Engenharia Mecânica

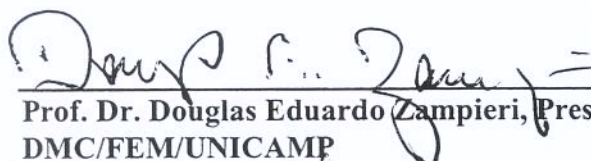
UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO ACADEMICO

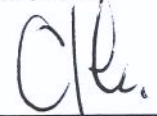
**Navegação de Robôs Autônomos baseada em
Monovisão**

Autor: Arthur de Miranda Neto
Orientador: Douglas Eduardo Zampieri
Co-orientador: André Mendeleck

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:



Prof. Dr. Douglas Eduardo Zampieri, Presidente
DMC/FEM/UNICAMP



Prof. Dr. Clésio Luis Tozzi
DCA/FEEC/UNICAMP



Prof. Dr. Eurípedes Guilherme de Oliveira Nóbrega
DMC/FEM/UNICAMP

Campinas, 26 de fevereiro de 2007.

2007 54 256

Dedicatória

É com muito orgulho que dedico meu mestrado a Deus, à minha esposa e filhos, aos meus pais, meus irmãos, familiares e amigos.

Agradecimentos

Agradeço primeiramente a Deus pela incomparável presença, e pela capacidade para a realização deste trabalho.

Aos meus professores orientadores Douglas Eduardo Zampieri e André Mendeleck pela incansável dedicação, motivação e confiança.

Em especial para a minha querida esposa Débora Gouvêa Miranda pelo amor, dedicação, companheirismo e compreensão, e também aos meus filhos Davi Gouvêa Miranda e Yasmin Gouvêa Parente.

Aos meus pais Maurício Miranda e Denise Pereira Miranda, ao meu irmão Fábio Miranda e irmãs Raquel Miranda Ribeiro, Talita Pereira Miranda e Alcilia Libalde Gonçalves, à minha avó Cecília Julieta Zanioli Pereira, aos meus sogros Décio Gouvêa e Albertina Maria Cardoso Gouvêa, e demais familiares, pelo amor, apoio e incentivo.

Aos professores Clésio Luis Tozzi e Neucimar J. Leite pelos ensinamentos que impactaram diretamente nos resultados deste trabalho, e também pela dedicação dos professores Eurípedes Guilherme de Oliveira Nóbrega, Pablo Siqueira Meirelles, Janito Vaqueiro Ferreira, Luiz Otávio Saraiva Ferreira, Gilberto de Martino Jannuzzi, André Felipe Simões, Alice M. Tokarnia, Hans Liesenberg e Wu Shin – Ting.

Aos meus colegas do Departamento de Mecânica Computacional da FEM e do Laboratório de Computação e Automação da FEEC, em especial para a minha fiel amiga e colega de publicações Leticia Rittner, e também para os amigos André Luiz Delai, Romaric Audigier, Patricia Rocha de Toro e Greyce Schroeder.

Aos chefes, pares e subordinados do 3ºCTA - Exército Brasileiro, em especial aos amigos José Antonio Cardoso Marum, Paulo Ricardo Corrêa Bonifácio, Geraldo Monteiro, Wagner Luís Barbosa de Abreu Lopes, Roberto de Oliveira Soares, Judson Benévolo Xavier Junior, Roberto Paulo Moreira Nunes, Emerson Alves da Silva, Humberto de Menezes Filho, Fabrício Abreu

Índice

Lista de Figuras.....	XI
Lista de Tabelas.....	XV
Capítulo 1: Introdução.....	1
Capítulo 2: Revisão Bibliográfica.....	6
2.1 Definição de Robô.....	8
2.2 Ambiente Cliente-Servidor.....	10
2.3 Visão Computacional.....	10
2.3.1 Segmentação de Imagens.....	15
2.3.2 Correlação de Imagens.....	18
2.4. Sistemas Classificadores e Algoritmo Genético.....	19
2.4.1 Sistemas Classificadores.....	21
2.4.1.1 Classificadores.....	22
2.4.1.2 Competição entre Classificadores.....	23
2.4.1.3 Atribuição de Créditos.....	23
2.4.2 Algoritmo Genético.....	25
Capítulo 3: Visão Computacional.....	29
3.1 Pré-processamento de Imagem.....	30
3.2 Segmentação de Imagem.....	31
3.3 Ângulo de Correção de Direção.....	39
3.3.1 Determinação do Ponto de Origem do Sistema.....	40
3.3.2 Centro de Massa da Área de Navegação.....	41
3.3.3 Correção de Direção.....	41
3.4. Velocidade.....	42
3.5. Identificação de Obstáculos.....	50
3.6. Descarte de Informações Redundantes.....	51

3.6.1 Limiar de Descarte de Informações não Determinístico.....	56
Capítulo 4: Estratégias de Navegação.....	58
4.1 Sistema Gerador de Classificadores.....	59
4.1.1 Parte-Conseqüente a partir do Vetor Maximização.....	61
4.1.2 Parte-Conseqüente a partir do Ângulo de Correção de Direção.....	63
4.1.3 Consolidação da Parte-Conseqüente.....	64
4.2 Sistema de Aprendizado Evolutivo.....	65
4.2.1 Fase de Competição.....	68
4.2.2 Fase de Seleção.....	69
4.2.3 Fase de Aprendizado e Reprodução.....	71
4.2.4 Migração.....	73
Capítulo 5: Experimentos Realizados.....	76
5.1 Protótipo.....	76
5.1 Servidor.....	77
5.1.2 Comunicação sem Fio.....	77
5.1.3 Eletrônica Embarcada.....	77
5.1.4 Projeto Físico.....	77
5.2 Camada Comandos e Transmissão.....	78
5.3 Experimento.....	79
5.3.1 Simulações.....	80
Capítulo 6: Conclusão e Trabalhos Futuros.....	84
Referências Bibliográficas.....	86
Apêndice A: A simple and efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision.....	92
Apêndice B: Pearson's Correlation Coefficient for Discarding Redundant Information in Real Time Autonomous Navigation Systems.....	100
Apêndice C: Sistemas Classificadores Aplicados à Navegação de Robôs Autônomos.....	108
Apêndice D: Resultados complementares do algoritmo TH Finder.....	113
Apêndice E: Protótipo.....	116

Lista de Figuras

Figura 1.1 – (a) <i>Wheelchair Robots</i> (SGOUROS N. M, 2002); (b) <i>Volkswagen Touareg R5</i> (STANFORD RACING TEAM's, 2005).....	2
Figura 1.2 – Sistema de Navegação em camadas (MIRANDA NETO & RITTNER, 2006).....	4
Figura 1.3 – Fluxo de informações entre as camadas do sistema de navegação.....	5
Figura 2.1 – Modalidade do Futebol de Robôs com visão de cima do ambiente (SCHWARTZ et al., 2003).....	13
Figura 2.2 – Visão embarcada: <i>Volkswagen Touareg R5</i> (STANFORD RACING TEAM's, 2005).....	13
Figura 2.3 – Histograma (SIMÕES, 2000).....	17
Figura 2.4 – Exemplo de classificadores e suas especificidades, parte antecedente. A parte conseqüente é apresentada na figura como mais escura (MOUSSI, 2002).....	22
Figura 2.5 – Exemplo da fase de competição entre classificadores apresentada por CAZANGI (2004).....	23
Figura 2.6 – Exemplo de Mutação (FONSECA, 2006).....	27
Figura 2.7 – Exemplo de Cruzamento (FONSECA, 2006).....	28
Figura 3.1 – Sistema de Navegação ênfase Visão Computacional.....	29
Figura 3.2 – Imagem Original, Segmentação Otsu e Histograma da Imagem Original Pré-Processada.....	33

Figura 3.3 – Corte da Imagem em Up e Down.....	34
Figura 3.4 – Resumo do processo de Segmentação: (a) Resultado final: definição do corte e segmentação; (b) Representação das sub-imagens resultantes do fatiamento; (c), (d) e (e) Respectivamente, sub-imagens: 10%, 60% e 100%.....	36
Figura 3.5 – Imagem Original, Imagem Negada e Resultado da Segmentação <i>TH Finder</i>	37
Figura 3.6 – Estágios do processamento do sistema de visão computacional de <i>Stanford</i> : (a) Imagem Original; (b) Imagem Processada com laser quadrilateral e classificação de pixel. (c) A classificação de pixel antes do <i>thresholding</i> ; (d) Detecção do horizonte para remoção do céu.....	38
Figura 3.7 – Resultados da segmentação <i>TH Finder</i> : (a) Imagem Original utilizada por <i>Stanford</i> ; (b) Imagem em Níveis de Cinza; (c) Imagem Filtrada; (d) Identificação da Área de Navegação após Segmentação.....	39
Figura 3.8 – Resultados do algoritmo <i>TH Finder</i> : (a) Imagem Original 1; (b) Imagem com Oclusão; (c) Imagem Original 2; (d) Imagem com Textura da Área de Navegação Irregular.....	39
Figura 3.9 – Ângulo de Correção de Direção.....	40
Figura 3.10 – Centro da Área de Navegação.....	42
Figura 3.11 – Imagem com 100% de Área de Navegação.....	44
Figura 3.12 – Par de imagens 1: Imagem Original e Resultado <i>TH Finder</i>	46
Figura 3.13 – Par de imagens 2: Imagem Original e Resultado <i>TH Finde</i>	47
Figura 3.14 – Par de imagens 3: Imagem Original e Resultado <i>TH Finder</i>	48
Figura 3.15 – Imagem com 100% de Obstáculo.....	49
Figura3.16 – Representação da informação horizontal na imagem através do Vetor Maximização.....	51
Figura 3.17 – Fluxo dos dados da camada Visão Computacional.....	52
Figura 3.18 – Coeficiente de Correlação de Pearson na camada Visão Computacional.....	53

Figura 3.19 – Sequência de Imagens 1.....	55
Figura 3.20 – Sequência de Imagens 2.....	55
Figura 4.1 – Sistema de Navegação ênfase Estratégias de Navegação.....	58
Figura 4.2 – Possibilidades mapeadas como movimentos para execução.....	61
Figura 4.3 – Divisão da imagem em três fatias. Valores do vetor $\vec{VM}$ não normalizados....	62
Figura 4.4 – Valores do vetor $\vec{PC}_{vm}$ transformados para valores inteiros.....	63
Figura 4.5 – Ângulos possibilidades de movimentos para um raio de 180° de visão.....	64
Figura 4.6 – Diagrama do Sistema de Navegação. A linha tracejada indica os componentes da FAR.....	68
Figura 4.7 – Sub-populações de classificadores.....	71
Figura 4.8 – Novo Classificador Filho a partir do cruzamento de dois Classificadores Pais.....	73
Figura 4.9 – Novo Classificador a partir da operação de mutação.....	73
Figura 4.10 – Operação de Migração entre sub-populações.....	74
Figura 5.1 – Componentes do Protótipo.....	76
Figura 5.2 – Sistema de Navegação ênfase Comandos e Transmissão.....	78
Figura 5.3 – Resultado 1: Deslocamento em Ambiente Real: (a) Visão da tarefa, (b) Cumprimento da Tarefa, (c) Mapeamento Virtual do deslocamento do robô móvel, (d) Imagens vista do robô móvel e resultado da segmentação <i>TH Finder</i>	81
Figura 5.4 – Resultado 2: Deslocamento em Ambiente Real: (a) Visão da tarefa, (b) Cumprimento da Tarefa, (c) Mapeamento Virtual deslocamento do robô móvel, (d) Imagens vista do robô móvel e resultado de segmentação <i>TH Finder</i>	81
Figura 5.5 – Resultado 3: (a) Deslocamento em Ambiente Real; (b) Mapeamento Virtual do deslocamento do robô móvel para travessia de um corredor com aproximadamente 8 metros.....	82

Fig. 5.6 – (a): Desvio de Obstáculo; (b): Ponto de partida.....	83
Figura 5.7 – Sequência de movimentos: (a) Ponto de partida; (b), (c) e (d): Deslocamentos; (e): Deslocamento em direção ao Centróide da Área de Navegação; (f): Centróide da Área de Navegação; (g) e (h): Cumprimento da Tarefa.....	83
Figura 6.6 – Resultado 2: Deslocamento em Ambiente Real: (a) Visão da tarefa, (b) Cumprimento da Tarefa, (c) Mapeamento Virtual deslocamento do robô móvel, (d) Imagens vista do robô móvel e resultado de segmentação <i>TH Finder</i>	107
Figura 6.7 – Resultado 3: Deslocamento em Ambiente Real: Mapeamento Virtual do deslocamento do robô móvel para travessia de um corredor com aproximadamente 8 metros.....	108

Lista de Tabelas

Tabela 3.1 – Valores para a Fig. 3.11.....44

Tabela 3.2 – Valores para a Fig. 3.12.....46

Tabela 3.3 – Valores para a Fig. 3.13.....47

Tabela 3.4 – Valores para a Fig. 3.14.....48

Tabela 3.5 – Valores para a Fig. 3.15.....49

Tabela 3.6 – Fatias da Imagem e limiares CCP.....57

Tabela 5.1 – Relação de Movimentos.....78

Capítulo 1

Introdução

A pesquisa na área de navegação de robôs autônomos tem motivado muitos pesquisadores de diferentes grupos devido ao desafio que representa. Várias propostas tem sido estudadas, baseadas principalmente na multidisciplinaridade e no desenvolvimento de sistemas com inspiração biológica. Digno de nota é o número de publicações nos últimos anos, incorporando os desenvolvimentos da área de telemática.

Dentre as diversas aplicações para navegação autônoma destacamos duas delas:

- A primeira é o projeto apresentado na Fig. 1.1 (a), cujo objetivo principal é o desenvolvimento de aplicações autônomas para auxílio a pessoas com restrições de locomoção e de direcionamento de uma cadeira de rodas. Para este caso, SGOUROS N. M (2002) apresenta o seguinte método qualitativo: Para cumprir a tarefa de navegação, o ambiente do robô é mapeado virtualmente num diagrama com as linhas do ambiente, e representações qualitativas das variações dos sensores entre as regiões adjacentes do espaço, que são armazenadas. Posteriormente, um planejador *off-line* usa estas representações para armazenar em cada célula de grade os comandos de movimentos apropriados. Os dados obtidos pelo planejador em conjunto com um controlador *on-line* permitem o deslocamento e posicionamento da cadeira de rodas entre os ambientes da casa.

A partir da definição de um objetivo a ser alcançado no ambiente, comparações são realizadas para calcular a posição atual do robô, e o módulo planejador faz a combinação de instruções para calcular um risco apropriado para aquele deslocamento, objetivando evitar colisões. O autor destaca que com os contínuos avanços da área de robótica, pessoas serão beneficiadas, e a partir do uso deste tipo de tecnologia, será possível aumentar sua independência e qualidade de vida.

Porém, devido às rígidas exigências de confiança, segurança e ergonomia, o desenvolvimento de sistemas robóticos automatizados ainda é classificado em uma fase inicial.

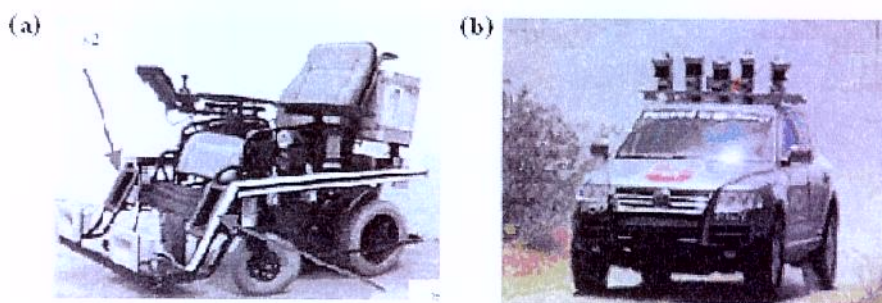


Fig. 1.1 - (a) *Wheelchair Robots* (SGOUROS N. M, 2002); (b) *Volkswagen Touareg R5* (STANFORD RACING TEAM's, 2005).

- Na Fig. 1.1 (b) apresentamos a próxima aplicação. Neste caso, temos o *Volkswagen Touareg R5* utilizado pela universidade de *Stanford* no desafio conhecido como *Grand Challenge*, que é organizado pelo *DARPA*¹, uma agência que visa melhorar a defesa do *E.U.A.* a médio e longo prazo (STANFORD RACING TEAM's, 2005).

A principal intenção na organização do desafio foi estimular o desenvolvimento de veículos *off-road* autônomos, já que o objetivo do governo dos *E.U.A.* é transformar um terço de sua frota de veículos militares em autônomos até 2015. Neste intuito, a competição foi aberta para universidades, colégios e empresas de dentro e fora dos *E.U.A.*, além de serem permitidos veículos de todos os tipos para a competição.

Na ocasião do campeonato, as equipes participantes seguiram algumas regras:

- O veículo deve ser completamente autônomo, sendo que o único sinal externo permitido é do GPS²;
- O veículo deve obrigatoriamente completar a prova passando entre os pontos de fiscalização especificados;
- O veículo deve continuar navegando mesmo com o sinal do GPS bloqueado, isto é, sob chuva e/ou nevoeiro;

¹ *Defense Advanced Research Projects Agency.*

² *Global Position System.*

- O veículo deve evitar colisões com objetos, por exemplo: veículos, motocicletas, barris de tráfico e demais objetos do ambiente.

Em 2004 foram 106 competidores inscritos. Nesta ocasião, apesar de nenhuma equipe ter completado a prova, a universidade *Carnegie Mellon* foi a que chegou mais longe. Já em 2005, cinco equipes completaram a prova, entretanto, a equipe vencedora foi a universidade de *Stanford*, que realizou a travessia do deserto de *Mojave* em 6:53:08h, a uma velocidade média de 30.7 km/h.

O 2007 *Grand Challenge*, denominado *DARPA Urban Challenge*, está previsto para o dia 03 novembro de 2007, e ocorrerá em um ambiente urbano.

A partir da análise dos exemplos acima citados, destacamos que, independente do tipo de aplicação, a geração de trajetórias para um robô móvel pode ser macro classificada em duas grandes categorias, tendo o ambiente como elemento principal: conhecida e determinística; ou desconhecida e dinâmica (MENDELECK et al., 2005).

Para obter as informações sobre o ambiente, um robô móvel deve possuir sensores e atuadores, que em muitas vezes, podem ser de alcance limitado e sujeitos a ruídos. No entanto, ao incorporar vários tipos de sensores aos robôs móveis, podemos aumentar seu grau de autonomia e “inteligência”, principalmente em relação à navegação em ambientes desconhecidos. Em contrapartida, o tipo e a quantidade de sensores determinam o volume de dados para o processamento e composição da “imagem” do ambiente, impondo na maioria dos casos, um alto custo computacional. Este custo, por muitas vezes, pode inviabilizar projetos que utilizem equipamentos de pouca capacidade e/ou de aplicações de tempo real.

Neste projeto, desenvolvemos um sistema de navegação para controle de robôs autônomos em ambientes desconhecidos, porém, considerando apenas informações oriundas de um sistema de visão computacional monocular.

Visto que não estamos utilizando sensores de posicionamento global, ou mecanismos para mapeamento prévio do ambiente, além do nosso protótipo não possuir atributos para controle da dinâmica do robô, para a elaboração de movimentos durante a navegação estamos desconsiderando informações globais do ambiente.

Desta forma, considerando o nível de autonomia do sistema de navegação, nosso objetivo principal foi criar um controle de robôs autônomos baseado em:

- Monovisão;
- Aprendizado evolutivo através de Sistemas Classificadores e Algoritmos Genéticos;
- De tempo real;
- Multiprocessado;
- Arquitetura Cliente-Servidor e de Sistema Distribuído;
- Multicamadas independentes entre si;
- Baixo custo;

Nosso primeiro passo na direção dos objetivos foi a abstração de um sistema em camadas, conforme apresenta a Fig. 1.2. Tal abstração, além de permitir a implementação de uma arquitetura cliente-servidor e de sistema distribuído, está mais próxima às características da nova geração de computadores multiprocessados, e da estrutura dos sistemas operacionais (MIRANDA NETO & RITTNER, 2006).

Na Fig. 1.3 apresentamos o fluxo de informações entre as camadas do sistema de navegação, cujo detalhamento será apresentado nos capítulos subseqüentes.



Fig. 1.2 – Sistema de Navegação em camadas (MIRANDA NETO & RITTNER, 2006).

Esta dissertação está organizada da seguinte forma. No Capítulo 2, apresentamos a revisão da literatura que fundamenta este trabalho, tendo como principais temas: Visão Computacional, Sistemas Classificadores e Algoritmos Genéticos. A partir do Capítulo 3 apresentamos nossos resultados, onde abordamos a estratégia de reconhecimento baseada na Visão Computacional. No Capítulo 4 apresentamos a camada Estratégias de Navegação, onde apresentamos uma proposta de adoção de Sistemas Classificadores e Algoritmos Genéticos para as decisões do sistema de navegação. No Capítulo 5, temos a camada Comandos e Transmissão, onde também

apresentamos nosso protótipo e os experimentos realizados. Finalmente, as conclusões e propostas futuras estão reunidas no Capítulo 6.

Complementarmente são anexados ao texto, apêndices onde apresentamos os artigos resultantes de nosso trabalho e uma descrição do protótipo.

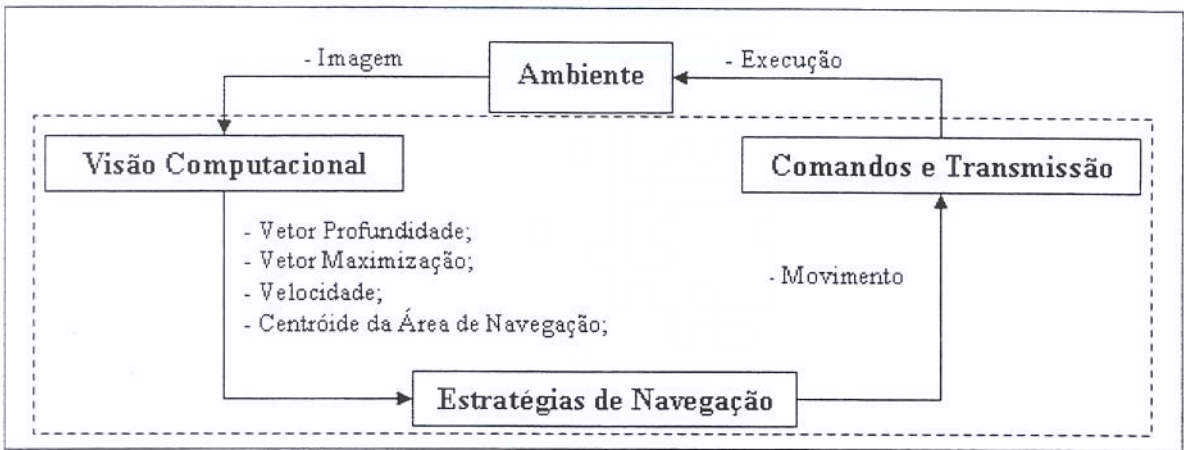


Fig. 1.3 – Fluxo de informações entre as camadas do sistema de navegação.

Capítulo 2

Revisão Bibliográfica

A partir da abstração do sistema de navegação em camadas, conforme apresentado na Fig. 1.2 do Capítulo de Introdução, nossa revisão bibliográfica objetivou buscar amparo em literaturas pertinentes às áreas de navegação autônoma, visão computacional e sistemas de aprendizado evolutivo.

Nesta revisão, a observação ao comportamento biológico humano não implicou diretamente na coleta de informações para o trabalho, no entanto, serviu de apoio à seleção de arquiteturas de sistemas, em vista dos requisitos do projeto. As características biológicas observadas foram: a organização do sistema nervoso, o sistema de visão, e o compartilhamento de conhecimento entre pessoas. Neste sentido, iniciamos este capítulo apresentando algumas características biológicas básicas.

Independente do tipo de ambiente, na navegação de humanos, o sistema nervoso em conjunto com o sistema endócrino desempenha funções de controle do organismo. Este sistema recebe milhares de informações dos diferentes órgãos sensoriais, e seu papel é integrá-las a fim de determinar respostas a serem executadas. Sua composição tem origem nas experiências sensoriais adquiridas através dos receptores sensoriais, podendo ser provenientes das experiências auditivas, visuais e vinculadas ao tato (GUYTON, 1973).

Uma das funções principais do sistema nervoso é atingir respostas ideais. Muitas das informações adquiridas pelos sensores são desprezadas após passarem pelo processamento de informações do cérebro, e isso pode ser justificado se considerarmos o fato de que humanos podem desempenhar diversas atividades simultaneamente. Como exemplo, durante a condução de um veículo, uma pessoa pode desenvolver outras atividades, como conversar, pensar em

outras coisas diferentes daquela primordial que é a navegação, ou mesmo ignorar o fato que sua pele está em contato com suas vestimentas. Especialmente aplicado à visão humana, em determinado momento, o foco da visão de uma pessoa pode estar centrado em uma área específica do ambiente, por exemplo, na estrada por onde navega, no entanto, as outras áreas deste ambiente não estão sendo ignoradas, isto é, para estas sub-imagens estará sendo utilizado um segundo plano.

Para realizar o armazenamento das informações, seres humanos possuem o córtex cerebral e a medula espinhal. Este processo de armazenamento pode ser chamado de memória. As informações a serem armazenadas trafegam através das sinapses, que possuem propriedades muito diversas, tais como, transmitir um sinal de um neurônio para outro. A partir do momento que as recordações tenham sido armazenadas no sistema nervoso, se tornarão parte do mecanismo de elaboração, ou seja, poderão ser utilizadas para atividades motoras posteriores. Tais recordações serão comparadas futuramente a novas experiências, a fim de auxiliar na seleção de novas informações (GUYTON, 1973).

Estabelecendo uma relação inicial com o discutido acima e o problema específico do aprendizado de robôs, em um sistema de navegação baseado em visão computacional, quando o processo de visão é iniciado, isto é, quando uma primeira imagem é adquirida, o sistema poderá não ter informação sobre o ambiente, e neste caso, o processo de recordação não contribuirá para o processo de decisão, semelhante a humanos.

Neste sentido, para um sistema computacional, dois tipos de armazenamento podem ser utilizados para suporte ao aprendizado: temporário e permanente, respectivamente, dependente e não dependente de fonte de alimentação. O último caso, armazenamento permanente, se comparado a humanos, pode representar o papel realizado pela medula espinhal e o córtex cerebral.

Outro problema comum aos robôs móveis, além de capacidade de processamento e armazenamento de informações, está ligado a sua capacidade de carregamento de carga, lembrando que, tanto a fonte de energia como os sensores estão embarcados.

Para minimizar a relação desempenho-peso, alguns projetos podem optar por uma arquitetura cliente-servidor, e algumas das tarefas como armazenamento e processamento de informações, possam ser atribuídas a uma máquina servidora mais potente. Esse tipo de solução também permite o compartilhamento da base de dados entre robôs, permitindo a cooperação de robôs.

Assim, informações adquiridas a partir da exploração de ambientes por um robô podem ser aproveitadas por outro. Tal argumento também tem inspiração humana, pois geralmente, humanos compartilham suas experiências anteriores, por exemplo, ao explicar um caminho que leve de um lugar a outro.

Caso a tarefa de navegação seja em ambiente conhecido e estático, os problemas podem ser primeiramente reduzidos à modelagem do ambiente e à busca por caminhos seguindo algum critério de otimização, por exemplo: distância, energia, processamento, número de movimentos, qualidade dos deslocamentos, etc. Já para ambientes desestruturados, o cenário para estudo é dinâmico, com vários elementos em movimentos. Assim, conduzir um robô de um ponto inicial até o seu destino, envolve a execução de operações complexas e não determinísticas, como exemplo: interação com o ambiente, identificação dos elementos ambientais e tomadas de decisões. Neste caso, a geração de trajetória exige o tratamento de uma série de fatores desconhecidos pelo software de planejamento, tais como, o volume da área de trabalho e os elementos móveis e fixos. Sem estas informações, o custo computacional para a geração de trajetória torna-se bastante elevado, principalmente se considerarmos as formas convencionais de programação de robôs (MENDELECK et al., 2005).

2.1 Definição de Robô

Segundo SHIROMA (2004), atualmente robôs podem ser classificados em dois grupos:

- **Robô Manipulador Industrial** - definição ISO³: “Robôs Manipuladores Industriais são máquinas controladas automaticamente, reprogramáveis, multi-propósitos com diversos graus de liberdade, que podem ser tanto fixas no local ou móveis para uso em aplicações de automação industrial”;
- **Robô Móvel**: “Um robô capaz de se locomover sobre uma superfície através da atuação de rodas montadas no robô em contato com a superfície (MUIR, 1988).”;

³ “Manipulating industrial robot is an automatically controlled, reprogrammable, multi-purpose, manipulative machine with several degrees of freedom, which may be either fixed in place or mobile for use in industrial automation application.”

Ainda segundo SHIROMA (2004), temos as seguintes definições:

- **Sistema Mecânico:** “A parte física do robô. A plataforma móvel do veículo. Encontra-se imerso em um ambiente global ou mundo real e é capaz de interagir com uma parcela dele – o ambiente local, por exemplo: uma sala, um corredor ou uma estrada, através de seus atuadores.”;
- **Sensores:** “Provêem os dados a partir do qual o estado do robô é determinado.” Por exemplo, temos: sensores de distância, GPS, laser e baseados em análise de imagens;
- **Inteligência:** “O cérebro do robô; responsável pela inteligência do sistema”.

Em literaturas que abordam o tema navegação robôs autônomos, por vezes, encontramos o termo robô móvel em substituição a veículo ou robô autônomo. Devido a isso, utilizaremos robô móvel.

Entretanto, dizer que um agente é autônomo implica em afirmar que ele, além de agir só, deve conseguir se auto-regular gerando as próprias regras que regem sua atuação. Desta forma, a definição distingue autônomo de automático, pois ser automático é ter a capacidade de operar em um ambiente percebendo-o e impactando-o, visando o cumprimento de tarefas definidas. Já um agente autônomo é antes de tudo automático, mas deve se autodirigir com base na capacidade própria de aprender e adaptar seus comportamentos (CAZANGI, 2004).

Então, um robô móvel é definido como aquele que possui a capacidade de realizar tarefas sem a necessidade da intervenção humana, devendo ser capaz de perceber o ambiente ao seu redor e tomar decisões a fim de cumprir seus objetivos. Porém, a existência de níveis de autonomia deve ser considerada, pois não existe um ser totalmente autônomo. Por exemplo, os animais, em geral, e o ser humano, em particular, dependem de fatores externos para sobreviverem. Estes organismos dependem do ambiente em que vivem para obter alimento e oxigênio. Logo, quando se afirma que um agente é autônomo, deve-se ter em mente que ele detém certo nível de autonomia e não que ele é completamente autônomo, sendo que quanto mais autônomo for o agente, menos auxílio externo ele necessita (CAZANGI, 2004).

2.2 Ambiente Cliente-Servidor

Considerando as características de grande parte dos sistemas robóticos e particularmente do robô utilizado neste projeto, também devido à exigência de recursos de processamento para algoritmos de visão computacional, a arquitetura cliente servidor foi selecionada. Em consequência disso, o software de navegação foi atribuído a um computador remoto mais potente, o que gerou a necessidade de um sistema de comunicação entre o servidor e o robô. Também ficou possível a implementação de um banco de dados para representar a memória permanente de informações no servidor. Acreditamos que o armazenamento permanente poderá contribuir para a geração de estratégias de navegação futuras, isto é, para o sistema de aprendizado evolutivo.

Embora não utilizado explicitamente neste trabalho, o banco de dados, certamente, pode ser útil para sistemas onde a cooperação entre robôs é um fator importante, a exemplo do que propõem SCHMITT et al. (2002).

É importante lembrar, que a utilização de um sistema de navegação na arquitetura cliente-servidor, não obriga que todas as informações sejam armazenadas e tratadas no servidor, mas pelo contrário, caso seja viável ao projeto, devem ser realizados nas duas extremidades do sistema, pois para casos de descontinuidade do sistema de comunicação por longos períodos, o cenário operacional deve ser preservado.

2.3 Visão Computacional

Como citado anteriormente, diversos tipos de sensores podem ser utilizados para proveito a navegação de um robô móvel. Contudo, destacamos que ao incorporar vários tipos de sensores aos robôs móveis, podemos aumentar seu grau de autonomia, de processamento e “inteligência”, principalmente em relação à navegação em ambientes dinâmicos.

Em contrapartida, o tipo e a quantidade de sensores determinam o volume de dados para o processamento e composição da “imagem” do ambiente, impondo na maioria dos casos, um alto custo computacional.

Particularmente estamos interessados em aplicações que além de operar em tempo real, tenham um bom desempenho em ambientes dinâmicos.

Atualmente, muitas das aplicações para controle de veículos autônomos que apresentaram bons resultados, tanto para ambientes externos quanto para ambientes internos, utilizavam a visão computacional como parte integrante do conjunto de sensores (BERTOZZI et al., 2000).

Por exemplo, os veículos participantes do DARPA, já citado anteriormente, possuem diversos tipos de sensores para permitir a interface entre o sistema de navegação e o ambiente a ser explorado. Comumente, o sistema de visão utilizado nestes veículos é composto por duas ou mais câmeras (e.g., DAHLKAMP, 2006; THRUN, 2005). No entanto, em nosso trabalho consideramos apenas um sensor, a monovisão, isto é, um sensor câmera irá adquirir imagens do ambiente e estas alimentarão o sistema computacional. Embora a monovisão seja uma solução bastante econômica, possui por desvantagem a obtenção de informações de profundidade, limitando a percepção do sistema.

O que é trivial para o sistema humano, construir cenas tridimensionais a partir de imagens bidimensionais captadas pelo sistema de visão, e a partir destas tomar decisões precisas para a navegação, pode não ser trivial para os sistemas computacionais.

Diferentemente do sistema humano, sistemas de processamento e análise de visão computacional mais complexos podem desencadear alguns prejuízos devido ao tempo de processamento. Como exemplo, temos a alta demanda de informações que são submetidas ao sistema de visão, em função da relação existente entre um sistema de decisão em tempo real, com um sistema de leitura de imagens que opera em uma determinada taxa de captação/leitura, isto é, quantidade de imagens geradas por segundo.

Durante a navegação de humanos, facilmente percebemos a importância do sistema de visão, no entanto, em virtude da sua extrema complexidade e alta demanda de informações, um grande número de informações pode ser perdido. Estima-se que apenas 90% da percepção visual humana são utilizadas por pessoas enquanto dirigem seus veículos. Desta forma, a visão computacional pode ser uma importante ferramenta para recuperar ou obter informações sobre ambientes (BERTOZZI et al., 2000).

Neste sentido, a decisão por um sistema de visão computacional mais complexo pode levar a um sistema lento demais para uma aplicação autônoma de tempo real. Ou ainda a escolha automática por um processo randômico de descarte de informações pelo sistema, para torná-lo suficientemente rápido, pode resultar em perda de informações importantes.

Para evitar o descarte de informações, muitas aplicações utilizam bases de armazenamento. No entanto, a submissão de imagens adquiridas, por exemplo, a uma rede neural para decisão, este grande número de informações não necessariamente se traduz em melhores decisões e podem inclusive prejudicar o desempenho do sistema, sobrecarregando-o. As aplicações mais comuns baseiam-se no processamento de duas ou mais imagens, a partir da análise do campo de fluxo óptico ou através de imagens não monoculares (HORN & SCHUNCK, 1981).

Num primeiro caso, temos sistemas de navegação em que mais de uma imagem é adquirida por uma câmera em momentos diferentes, enquanto que em um segundo caso, múltiplas câmeras são usadas para adquirir imagens simultaneamente, porém, de pontos de “vistas” diferentes.

Além da alta complexidade computacional intrínseca nesses modelos, causando um incremento significativo na quantidade de dados a serem processados, estas técnicas também devem ser robustas o bastante para tolerar o ruído causado pelos movimentos do veículo e provenientes da calibração de múltiplas câmeras (BERTOZZI et al., 2000).

A técnica baseada no fluxo-óptico requer a análise de uma sucessão de duas ou mais imagens: um vetor bidimensional é computado no domínio da imagem, calculando-se as componentes horizontais e verticais de velocidade de cada pixel. Os resultados podem ser usados para detecção de movimentos, detecção de obstáculos por análise das diferenças entre o esperado e a velocidade real dos campos, entre outros.

Sabendo que na maioria dos sistemas de navegação autônomos, o sistema de visão computacional vem acompanhado de outros sensores, com o objetivo de diminuir o número de informações submetidas ao sistema, optamos pela monovisão para permitir a interface entre o robô e o ambiente, mesmo que com isso, estejamos abrindo mão das informações de profundidade nas imagens. Acreditamos que utilizando técnicas de processamento de imagens mais simples e consequentemente algoritmos mais rápidos, o sistema será capaz de apresentar um bom desempenho em tempo real, onde o compromisso entre tempo de processamento e aquisição das imagens é fundamental.

Outro assunto que não poderíamos deixar de lado em visão computacional é a calibração de câmera, pois para muitos dos projetos de visão computacional, a obtenção de informações prévias sobre o ambiente a ser explorado é de fundamental importância. Para obter tais informações, em conjunto ou não com a calibração de câmera, encontramos o uso de *landmarks* para auxílio na localização do robô no ambiente (NAGAONKAR et al., 2005).

A calibração da câmera permite que um sensor câmera de um robô seja usado como um dispositivo de medidas. No entanto, para efetuar a calibração são necessários seis ou mais pontos conhecidos do ambiente, e a partir destes, relacioná-los a pontos conhecidos na imagem, considerando uma determinada posição da câmera (e.g., GONZALEZ & WOODS, 2000; TSAI, 1987).

Para nosso projeto, a adoção da calibração de câmera e/ou de *landmarks* se torna inviável, pois consideramos que o robô não possui informações prévias do ambiente a ser conquistado.

Para o controle do robô na área de navegação, dois tipos de sistemas de visão podem ser utilizados: vista de cima do ambiente, como na Fig. 2.1, que apresenta o futebol de robôs (e.g., SCHWARTZ et al., 2003; GUPTA et al., 2005) e visão embarcada, como na Fig. 2.2, que apresenta o *Volkswagen Touareg R5* utilizado por Stanford no *Grand Challenge* (STANFORD RACING TEAM's, 2005).

Independente do tipo de sistema de visão que possa ser utilizado, o objetivo da visão computacional é, além da informação visual, processar dados de imagens para percepção de máquinas (GONZALEZ & WOODS, 2000).

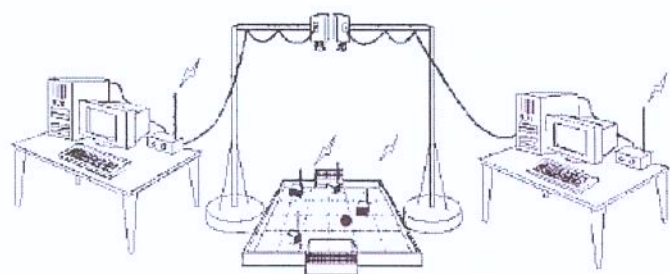


Fig 2.1 – Modalidade do Futebol de Robôs com visão de cima do ambiente (SCHWARTZ et al., 2003).

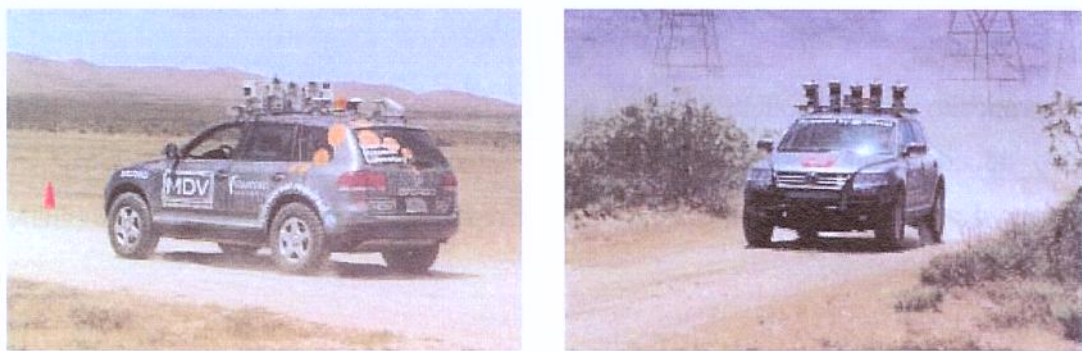


Fig. 2.2 – Visão embarcada: *Volkswagen Touareg R5* (STANFORD RACING TEAM's, 2005).

Com este objetivo, os algoritmos da visão computacional podem realizar diversas operações sobre as imagens, e permitir, por exemplo: a redução de ruídos, realce de detalhes em uma imagem, processamento de cores, segmentação, extração de informações, classificação, etc.

Segundo GONZALEZ & WOODS (2000), as principais atividades de um sistema de processamento são: aquisição, pré-processamento, segmentação e classificação.

Na aquisição de imagens digitais, dois elementos são fundamentais: um dispositivo físico, que produza um sinal elétrico de saída proporcional a um nível de energia percebida, e um digitalizador, que converta a saída elétrica de um dispositivo de sensoriamento físico para a forma digital, como por exemplo: nas câmeras fotográficas digitais.

As atividades do pré-processamento compreendem: redução de ruídos, realce de detalhes em uma imagem, entre outras. Em nosso trabalho esta atividade será realizada em uma máquina servidora remota, entretanto, é importante notar que o pré-processamento de imagens pode ser implementado em hardware reconfigurável, dentro do conceito de sistemas SoC⁴, o que tem demonstrado um sensível ganho no custo computacional (NAGAONKAR et al., 2005).

Para a execução da atividade de pré-processamento, diversos filtros podem ser utilizados, nos quais podemos destacar os filtros passa-baixa, filtros passa-alta e aqueles que suavizam uma imagem. A partir destes filtros podemos permitir que baixas ou altas frequências possam ser mantidas em uma imagem, ou até mesmo suavizar detalhes. Estes filtros, ou máscaras, podem ser de diversas dimensões, sendo que os filtros passa-baixa e os filtros passa-alta são geralmente utilizados para o realce das imagens, enfatizando as mudanças de níveis de cinza que estão associadas às altas frequências.

Antes de apresentar a atividade de segmentação de imagens, terceira atividade de um sistema de processamento de imagens, dois modelos de cores serão apresentados, visto que, consideramos a possibilidade que nosso sistema de aquisição de imagens adquira imagens digitais coloridas.

O propósito de um modelo de cor é a especificação das cores em uma forma padrão. Um modelo de cor é uma especificação de um sistema de coordenadas tridimensionais e um subespaço dentro deste sistema, onde cada cor é representada por um único ponto (GONZALEZ & WOODS, 2000).

⁴ *System-on-a-Chip*.

Para a obtenção de qualquer cor, é condição necessária e suficiente, a adoção de três ondas eletromagnéticas, tal que duas delas não produzam a terceira (NOVAKOVSKY, 1975). Esse processo é similar ao sistema sensório humano, amostrados pelos cones vermelho, verde e azul (SIMÕES & COSTA, 2000). O modelo que mais se aproxima desse processo é o *RGB*⁵, que é formado por três bandas ou que variam de 0 a 255 valores para cada cor.

O modelo *HSI*⁶ representa respectivamente, matiz, saturação e intensidade. O matiz descreve quanto uma cor é pura, a saturação dá a medida grau de diluição de uma cor pura com o branco e a intensidade é o nível de luminância da imagem (RAMESH et al., 1995). A grande importância deste modelo é devido à intensidade poder ser desacoplada da informação de cor na imagem e os componentes de matiz e saturação são intimamente relacionados à percepção humana de cores (GONZALEZ & WOODS, 2000).

Devido a sua extensão, diferentemente das atividades de aquisição e pré-processamento de imagens, apresentadas anteriormente, a segmentação de imagens será apresentada no próximo item.

2.3.1 Segmentação de Imagens

O processo de segmentação em uma imagem digital tem como objetivo subdividi-la em suas partes ou em objetos constituintes. Esse processo de segmentação deve seguir até que os objetos de interesse da imagem tenham sido isolados (GONZALEZ & WOODS, 2000). Tal tarefa pode ser extremamente sofisticada, e a partir de filtros bem elaborados, pode-se chegar a resultados muito eficientes (LOPES, 2003). Porém, estes resultados podem exigir um alto preço, isto é, normalmente algoritmos de segmentação robustos apresentam grande complexidade.

A segmentação pode ser considerada como partição de imagens digitais em conjuntos de pixels, considerando-se a necessidade geral e critérios previamente definidos.

Uma das técnicas para se segmentar uma imagem é a partir da utilização de limiares (e.g., GONZALEZ & WOODS, 2000; GÓMEZ et. al., 2003; GUPTA et al., 2005), no entanto, para imagens coloridas que utilizam o modelo *RGB*, deve se considerar mais de um limiar, isto é, um para cada banda de cor.

⁵ *Red, Green, Blue.*

⁶ *Hue, Saturation, Intensity.*

Um exemplo deste tipo de técnica foi apresentado por GÓMEZ et. al. (2003). Os autores destacaram que o espaço de cores *RGB* é muito sensível as condições de luminosidade, e em oposição a isso realizaram testes no modelo *HSI*. Com esse teste foi constatado que a robustez do novo modelo adotado não trouxe resultados significantes para o processamento das imagens, porém o ponto negativo foi a queda de desempenho por causa da conversão do modelo *RGB* para o *HSI*, considerando a CPU utilizada. Desta forma, adotaram o modelo *RGB*.

Como uma opção aos métodos convencionais de limiarização, no trabalho realizado por SIMÕES & COSTA (2000), uma rede neural artificial do tipo *perceptron* multicamada foi utilizada para a classificação dos pixels da imagem. Para isso, a trinca de valores de um dos modelos *RGB*, *HSV*⁷ e *YUV*⁸ de cada pixel da imagem foram submetidos à rede e classificadas em um dos padrões: laranja, rosa, verde, amarelo, branco e preto. Os autores ressaltaram, ao observar os resultados obtidos, que a melhoria na distribuição de pixels esperada teoricamente não se confirmou, pois as regiões encontraram-se mais agrupadas nos modelos *HSV* e *YUV*, e mais disjuntas, portanto melhor separáveis no sistema *RGB*.

Devido à sensibilidade as condições de luminosidade dos modelos de segmentação baseados em *RGB* e *HSI* e o alto custo computacional imposto por algoritmos de segmentação baseados em redes neurais, neste trabalho adotamos a técnica de limiarização para classificação de pixels, porém aplicada somente para imagens em níveis de cinza.

Os métodos de limiarização são divididos em dois grupos: globais e locais. Os definidos como globais dividem a imagem usando apenas um limiar e os definidos como locais são aqueles que dividem a imagem em sub-imagens, definindo um limiar para cada uma delas. Em SAHOO et al. (1988) os limiares locais são definidos como limiares multiníveis.

Segundo SIMÕES & COSTA (2000), tradicionalmente, a aplicação de limiares está ligada a um estudo do histograma da imagem, e a determinação de um nível capaz de separar duas regiões, conforme apresentado na Fig. 2.3.

Em resumo, a partir da definição de limiares globais e/ou multiníveis, o uso de um limiar global ou local em uma imagem bidimensional $I(x, y)$ com 256 níveis de intensidade consiste em determinar um único limiar T que separe os pixels em duas classes distintas: objeto e fundo. O

⁷ Hue, Saturation, Value.

⁸ Y stands for the luma component (the brightness) and U and V are the chrominance (color) components.

limiar T , normalmente é aplicado ao histograma da imagem $\vec{h}(N)$, o qual pode ser visto como uma descrição da distribuição das intensidades pela imagem.

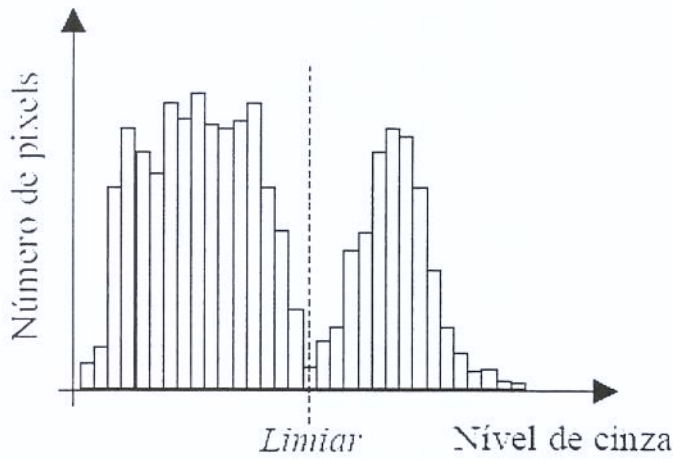


Fig. 2.3 - Histograma (SIMÕES, 2000).

A limiarização multiníveis, é geralmente mais trabalhosa que o limiar global, isso porque existe uma dificuldade em encontrar múltiplos limiares que determinam efetivamente as regiões de interesse, especialmente quando houver vários grupos de objetos na imagem.

Para GONZALEZ & WOODS (2000), a limiarização global alcança bons resultados apenas quando a iluminação da imagem é relativamente uniforme e as regiões de interesse, representadas pelos objetos, possuem uma diferença significativa de intensidade do seu fundo, o que provavelmente não se encontra na maioria dos ambientes naturais.

Um dos métodos de limiarização considerados ótimos foi apresentado por OTSU (1978). A principal característica deste método é a maximização da variância entre as classes da imagem, isto é, a maximização da separação entre o objeto e o fundo da imagem. O processo de limiarização representado pela Eq. (2.1) é visto como o particionamento dos pixels de uma imagem em duas classes: $C1=\{0,1,...,T\}$ e $C2=\{T+1,T+2,...,N-1\}$, sendo que T é o limiar escolhido e N é o número de níveis de intensidade da imagem.

$$\sigma_w^2(T) = q_1(T)\sigma_1^2(T) + q_2(T)\sigma_2^2(T) \quad (2.1)$$

onde $\sigma_w^2(T)$ é a variância intra-grupo, $q_1(T)$ a proporção dos pixels do fundo, $\sigma_1^2(T)$ a variância das intensidades dos pixels do fundo, $q_2(T)$ a proporção dos pixels do objeto e $\sigma_2^2(T)$ a variância das intensidades dos pixels do objeto.

Uma das grandes vantagens deste método é que ele não se restringe ao tipo de histograma da imagem, isto é, pode ser aplicado em histogramas unimodais, bimodais ou multimodais, mas possui melhor desempenho em imagens com maior variância de intensidade. Sua principal desvantagem está no desempenho sensível à presença de ruídos na imagem, o que pode ser reduzido com a aplicação do filtro de suavização. Em contrapartida, SAHOO et al. (1988) e LEE et al. (1990) consideram este método como uma das melhores escolhas para aplicações em tempo-real em visão computacional.

Apesar do método proposto por Otsu ser um ótimo método para a escolha do limiar ideal, para todos os casos ele considera a informação da imagem como um todo, o que por muitas vezes pode prejudicar o processo de segmentação.

2.3.2 Correlação de Imagens

Uma das alternativas para desafogar o processo de segmentação de imagens é, em uma etapa anterior ao pré-processamento, efetuar o cálculo da correlação entre uma imagem atual submetida ao sistema e a sua anterior imediata, isto é, uma imagem que já tenha sido submetida ao processo de segmentação. Este tipo de procedimento pode trazer ganhos ao sistema computacional, pois caso a correlação entre as imagens seja alta, os comandos gerados para o robô utilizados para uma imagem anterior podem ser reutilizados para uma imagem atual, dispensando que a imagem atual seja submetida ao processo de segmentação.

De acordo com EUGENE & JOHNSTON (2005), o Coeficiente de Correlação de Pearson - CCP, r , é extensamente usado em reconhecimento de padrões, processamento de imagens, entre outros. Aplicações destas finalidades incluem comparação entre duas imagens com propósitos de reconhecimento de objetos e medida de disparidade. Para imagens em níveis de cinza, o CCP é descrito na Eq.(2.2).

$$r = \frac{\sum_i (x_i - x_m)(y_i - y_m)}{\sqrt{\sum_i (x_i - x_m)^2} \sqrt{\sum_i (y_i - y_m)^2}} \quad (2.2)$$

onde x_i é a intensidade de um pixel na imagem 1, y_i é a intensidade de um pixel na imagem 2, x_m é a média das intensidades da imagem 1, e y_m é a média das intensidades da imagem 2.

O CCP tem valor $r = 1$ se as duas imagens são absolutamente idênticas, $r = 0$ se as duas imagens são completamente descorrelatas, e $r = -1$ se as duas imagens forem completamente anti-correlatas, isto é, se uma imagem for o negativo da outra.

Para EUGENE & JOHNSTON (2005), o CCP obtém um valor de $r = 1$ se os objetos estão intactos no par de imagens, e um valor menor que 1 se aconteceu alguma alteração ou movimento. Na prática, distorções no sistema de imagem, ruídos, variações leves na posição dos objetos relativos ao sistema de aquisição, e outros fatores produzem um valor de r menor que 1, até mesmo se o objeto não foi fisicamente movido ou foi alterado de alguma maneira. Em aplicações de segurança, os valores típicos de r entre duas imagens digitais da mesma cena, obtidas uma imediatamente após a outra, usando o mesmo sistema de imageamento e iluminação, variou de 0.95 a 0.98.

2.4 Sistemas Classificadores e Algoritmos Genéticos

O processo de aprendizagem dos robôs móveis é baseado na idéia que um sistema pode se tornar apto para atuar de forma eficiente diante de novas situações, sustentando-se na sua capacidade de generalizar conhecimentos adquiridos anteriormente, mesmo que sejam diversos e incompletos. Entretanto, a generalização não é a única propriedade que um sistema com aprendizagem deve possuir (CAZANGI, 2004). Alguns atributos básicos relacionados a abordagens que contemplam aprendizagem são descritos a seguir:

- O sistema deve ser robusto em relação a ruídos;
- Os mecanismos devem convergir rapidamente e têm que permitir a aprendizagem durante a operação do sistema;
- A aprendizagem deve ser incremental e continuada;
- O processo de aprendizagem precisa ser computacionalmente tratável, isto é, deve possibilitar sua execução em tempo real;
- O aprendizado deve depender apenas de informações obtidas pelo próprio robô, por meio de sua capacidade sensorial.

Segundo GONZALEZ & WOODS (2000), a aquisição de valores amostrados por um sensor câmera pode ser comparada ao processamento de imagem da visão humana. Resumidamente, o sistema de visão humano permite que a luz de um objeto fora do olho seja tornada imagem na retina, sendo que a percepção desta acontece devido à relativa excitação dos receptores de luz, que transformam a energia radiante em pulsos elétricos que são posteriormente decodificados pelo cérebro.

Em outras palavras, podemos dizer que o sistema de visão humano sem o auxílio do cérebro não seria capaz de conduzir pessoas durante seus deslocamentos. E isso pode ser levado para os sistemas computacionais, pois para permitir que robôs móveis naveguem somente com base na análise de imagens, estas aplicações devem ser compostas por softwares suficientemente “inteligentes” para gerenciar a estrutura mecânica durante o deslocamento.

Ao selecionar um sistema de visão computacional como único sensor de entrada de dados para o sistema robótico, e considerando a possibilidade da influência pontual do fator iluminação nas imagens em ambientes não controlados, além de fatores desconhecidos, tais como, os elementos móveis e fixos, a seleção de movimentos ideais pode se tornar uma tarefa não trivial, tornando o sistema instável. Para isso, a utilização de um sistema de aprendizado evolutivo para se obter movimentos ideais se torna imprescindível (e.g., KOSA et al., 1999; MOUSSI et al., 2001; MICHALEWICZ, 1996; MICHALEWICZ, 2000; MENDELECK et al., 2005).

2.4.1 *Sistemas Classificadores*

Durante a navegação em ambientes desconhecidos, um robô móvel deve se adaptar a cada situação nova apresentada pelo ambiente, e esta dificuldade aumenta significativamente caso o ambiente seja dinâmico.

Segundo BOOKER et al. (1989), um sistema com capacidade de aprendizagem deve tratar quatro atividades principais:

- Ocorrência continuada de novidades no ambiente, freqüentemente misturada a ruídos e dados irrelevantes;
- Tomada de decisão em tempo real;
- Objetivos do sistema que são implícitos ou imprecisos, além de serem múltiplos e conflitantes;
- Dependência de longas seqüências de ações e eventos para obtenção de realimentação;

Com este intuito, segundo CAZANGI (2004), contemplando o processo de aprendizagem por meio de classificação das regularidades do ambiente, John Holland criou os Sistemas Classificadores - SC em meados da década de 70. Este sistema surgiu a partir da combinação de sistemas especialistas com algoritmos genéticos, sendo este vinculado à computação evolutiva. Basicamente, um SC é um mecanismo para criação e atualização evolutiva de conhecimento representado por regras, denominadas classificadores, responsáveis pela atuação do sistema em um ambiente arbitrário.

Existem duas variantes para formação dos classificadores:

- *Pittsburg*: um indivíduo representa solução para o problema;
- *Michigan*: a solução é dada por uma população de indivíduos;

Adotamos *Michigan*, cujos componentes principais são:

- Tratamento de Regras e Mensagens;
- Atribuição de Créditos;
- Descoberta de Novas Regras;

2.4.1.1 Classificadores

Um SC é composto por um conjunto de regras de inferência, isto é classificadores, do tipo antecedente-conseqüente, sendo a parte-antecedente composta por uma condição, e a parte-conseqüente por uma ação.

Para entender o fundamento da composição de um SC, podemos fazer uma analogia com Sistemas Especialistas. Segundo MOUSSI (2002), um SC pode ser entendido como se constituindo de um conjunto de regras proposicionais, isto é, dada uma condição tem-se, por exemplo, uma conclusão ou ação.

Na proposta original de Holland, os termos parte-antecedente ou condição são codificados como vetores de tamanho fixo, do alfabeto $\{0,1,\#\}$ e o termo parte-conseqüente ou ação é formado pelo alfabeto $\{0,1\}$. O símbolo “#” (*don't care*) pode valer tanto 0 quanto 1, o que permite a existência de regras genéricas, ou seja, quanto mais símbolos deste tipo estiverem presentes em um classificador, mais genérico ele será.

Conforme apresentado em CAZANGI (2004), o conceito usado para quantificar esta propriedade é chamado de especificidade. Desta forma, caso um classificador possua todos os caracteres do seu antecedente iguais a “#”, sua especificidade será 0.

A Fig. 2.4 de MOUSSI (2002) apresenta um exemplo de classificadores e suas especificidades.

Cada classificador ou regra também possui uma energia específica, que está diretamente ligada a sua utilidade dentro de sua população. Quanto maior for a energia de um classificador maior a probabilidade de sua atuação no ambiente.

#	1	#	0	1	#	0	#	#	0	1
#	#	0	1	1	1	0	#	1	0	1
0	1	1	1	#	0	1	1	1	0	0
0	1	1	0	1	0	0	1	0	0	0
1	0	0	#	#	1	#	#	1	1	0

Fig 2.4 – Exemplo de classificadores e suas especificidades, parte-antecedente. A parte-conseqüente é apresentada na figura como mais escura (MOUSSI, 2002).

2.4.1.2 Competição entre Classificadores

A cada vez que um sensor captura uma mensagem do ambiente, esta é codificada na forma de um classificador e um processo de competição é iniciado. É neste processo que as partes antecedentes de todos os classificadores da população serão comparadas.

Na Fig. 2.5, CAZANGI (2004) apresenta a Fase de Competição. Conforme a figura, em uma fase de ativação de classificadores, o sistema pode encontrar mais do que um classificador para seleção. Neste caso, a competição e a seleção podem ser realizadas de diversas formas, utilizando-se métricas e processos de seleção específicos.

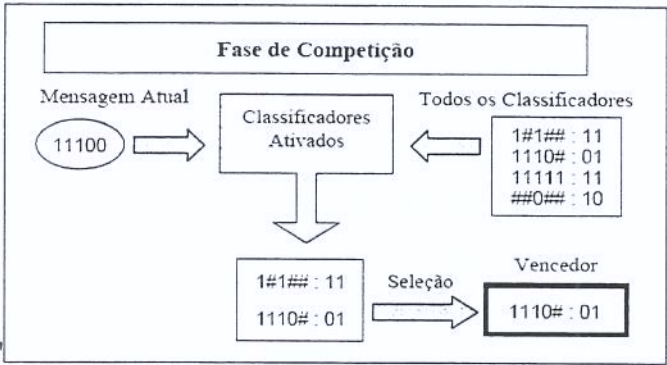


Fig 2.5 – Exemplo da fase de competição entre classificadores apresentada por CAZANGI (2004).

2.4.1.3 Atribuição de Créditos

Diversos tipos de algoritmos de atribuição de créditos podem ser utilizados em Sistemas Classificadores. O algoritmo mais disseminado é conhecido como *Bucket Brigade*. Este algoritmo foi especialmente proposto para problemas em que a realimentação é espaçada e que envolvam classificadores postadores de mensagem, isto é, que ativam outros classificadores. Através de mecanismos inter-relacionados: leilão, recompensa e taxação, este algoritmo é capaz de identificar e recompensar ou punir todos os classificadores que levaram a uma certa modificação no ambiente (e.g., MOUSSI, 2002; CAZANGI, 2004).

Para o processo de leilão todos os classificadores são selecionados, e competem a partir de uma espécie de leilão. Cada lance de um classificador é calculado em função da sua energia e

especificidade. O classificador vencedor paga seu lance, Eq. (2.3), descontando de sua energia a quantia ofertada (CAZANGI, 2004).

$$lance_t = k_0 * (k_1 + k_2 * Spec^{SPow}) * S_t \quad (2.3)$$

na equação acima, $lance_t$ é o lance de um classificador na iteração t , k_0 é uma constante positiva menor que 1 que age como um fator de risco global, influenciando na fração de energia do classificador, que será ofertada e possivelmente subtraída, k_1 representa um coeficiente constante, positivo e menor que 1 para a porção não específica do lance, k_2 representa um coeficiente constante, positivo e menor que 1 para a porção específica do lance, S_t representa a energia do classificador na iteração t , $Spec$ é a especificidade do classificador, $SPow$ é um parâmetro de controle da influência da especificidade no valor do lance.

A especificidade do classificador é definida pela Eq. (2.4) (CAZANGI, 2004).

$$Spec = \frac{N - total_ \#}{N} \quad (0 \leq Spec \leq 1) \quad (2.4)$$

onde N representa o tamanho da parte-antecedente do classificador, $total_ \#$ é a quantidade de símbolos “#” - *dont'care* - presentes na parte-antecedente do classificador.

Dependendo do efeito da ação do classificador, obtido pelo processo de realimentação, que neste caso, pode ser favorável ou prejudicial ao ambiente, aplica-se o mecanismo de recompensa (CAZANGI, 2004), Eq. (2.5).

$$S_{t+1} = S_t + R_t - lance_t \quad (2.5)$$

onde S_t representa a energia do classificador na iteração t , $lance_t$ representa o lance do classificador na iteração t , R_t é um valor baseado na realimentação dada pelo ambiente, caso o classificador tenha sido vencedor da competição na iteração $t-1$, sendo que:

- $R_t = 0$, caso o classificador não tenha sido vencedor;
- $R_t > 0$, em caso de recompensa - ação positiva;
- $R_t < 0$, em caso de punição - ação negativa;

2.4.2 Algoritmo Genético

Um breve histórico sobre a origem dos AG é apresentado por SILVA (2005). Neste histórico, o autor apresenta que até meados do século 19, naturalistas acreditavam que cada espécie havia sido criada separadamente por um ser supremo ou através de geração espontânea.

Após de mais de 20 anos de observações e experimentos, Charles Darwin apresentou em 1858 sua teoria da evolução através da seleção natural, simultaneamente com outro naturalista inglês Alfred Russel Wallace. No ano seguinte, Darwin publica o seu *On the Origin of Species by Means of Natural Selection* com a sua teoria completa, sustentada por muitas evidências colhidas durante suas viagens a bordo do *Beagle*.

Tal trabalho influenciou muito no futuro não apenas da biologia, botânica e zoologia, mas também teve grande influência sobre o pensamento religioso, filosófico, político e econômico da época. A teoria da evolução e a computação nasceram praticamente na mesma época: Charles Babbage, um dos fundadores da computação moderna e amigo de Darwin, desenvolveu sua máquina analítica em 1833. Ambos provavelmente estariam surpresos e orgulhosos com a ligação entre estas duas áreas.

Em sequência a isso, por volta de 1900, o trabalho de Gregor Mendel, desenvolvido em 1865, sobre os princípios básicos de herança genética, foi redescoberto pelos cientistas e teve grande influência sobre os futuros trabalhos relacionados à evolução. A moderna teoria da evolução combina a genética e as idéias de Darwin e Wallace sobre a seleção natural, criando o princípio básico de Genética Populacional: a variabilidade entre indivíduos em uma população de organismos que se reproduzem sexualmente é produzida pela mutação e pela recombinação genética.

Por fim, os AG foi desenvolvido durante os anos 30 e 40, por biólogos e matemáticos de importantes centros de pesquisa. Já nos anos 50 e 60, muitos biólogos começaram a desenvolver simulações computacionais de sistemas genéticos. Entretanto, foi John Holland quem começou,

seriamente, a desenvolver as primeiras pesquisas no tema. Holland foi gradualmente refinando suas idéias e em 1975 publicou o seu livro *Adaptation in Natural and Artificial Systems*, o que atualmente pode ser considerado a Bíblia de Algoritmos Genéticos. Desde então, estes algoritmos vêm sendo aplicados com sucesso nos mais diversos problemas de otimização e aprendizado de máquina.

Segundo o conceito de sistemas classificadores apresentado por CAZANGI (2004), os classificadores, isto é, informações armazenadas, precisam ser capazes de substituir os classificadores de pouca utilidade presentes na população atual por novos e tentar aprimorar os demais. Neste contexto, o autor destaca que o principal mecanismo empregado para descoberta de classificadores são os Algoritmos Genéticos - AG. Assim, ao final de uma época de iterações do SC, o AG e seus procedimentos evolutivos entram em ação em busca de uma população de classificadores mais adaptada ao ambiente.

Os AG baseiam-se em três aspectos principais:

- Trabalham com uma população de classificadores e não com um único classificador;
- Utilizam informações de custo ou recompensa;
- Utilizam regras de transição probabilísticas e não determinísticas;

Neste sentido, por serem baseados na evolução biológica, são capazes de identificar e explorar fatores ambientais, podendo convergir, na maioria das vezes, para soluções ótimas ou aproximadas, considerando níveis globais.

Sabendo que os AG trabalham com uma população de classificadores, e que cada classificador pode ser visto como uma possível solução ao problema, durante o processo evolutivo toda sua população é avaliada.

Para a avaliação da população, CAZANGI (2004) apresenta um processo anterior chamado de seleção. O autor destaca que o propósito da seleção é eleger classificadores genitores. Descreve ainda, que diversos esquemas de seleção já foram criados, como exemplo, elitista, bi-classista ou por torneio.

No modelo original, após a seleção, na avaliação somente os classificadores mais adaptados são mantidos. Em seguida, os classificadores mantidos podem sofrer modificações de suas características fundamentais, através de operações genéticas de mutação e cruzamento. Estes processos visam encontrar soluções satisfatórias para o problema.

Para preservar, isto é, não permitir que melhores classificadores desapareçam durante operações genéticas, existe a reprodução elitista, onde indivíduos podem ser colocados automaticamente na próxima geração.

CAZANGI (2004) apresenta que tradicionalmente classificadores são representados genotipicamente por vetores binários. Esta representação é independente do problema, pois encontrada uma representação, por exemplo, em vetores binários, as operações genéticas podem ser utilizadas, independente de classes de problemas.

Segundo SILVA (2005), a operação de mutação, representado na Fig.2.6, é necessária para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, o que permite a introdução de novos elementos na população. Ainda, assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois com este mecanismo, altera-se levemente a direção da busca.

Antes da Mutação:	1 1 1 0 0
Depois da Mutação:	1 1 0 0 0

Fig 2.6 – Exemplo de Mutação.

Já o cruzamento é responsável pela recombinação, isto é, reprodução de um novo classificador a partir da base genética dos classificadores pais. Este operador pode permitir operações de mais de um tipo (SILVA, 2005), sendo as mais utilizadas:

- um-ponto: um ponto de cruzamento é escolhido e a partir deste ponto as informações genéticas dos pais serão trocadas, conforme apresenta a Fig. 2.7.
- multi-pontos: é uma generalização desta idéia de troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados.
- uniforme: não utiliza pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

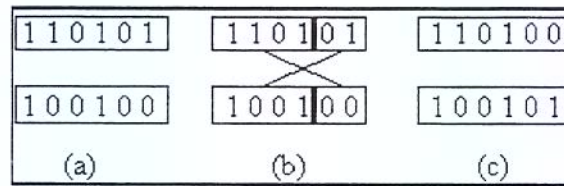


Fig 2.7 – Exemplo de Cruzamento.

SILVA (2005) conclui apresentando ainda quatro importantes pontos:

- **Tamanho da População:** afeta o desempenho global e a eficiência dos AGs; população pequena o desempenho pode cair, pois a população fornece uma pequena cobertura do espaço de busca do problema; uma grande população fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais, no entanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais, ou que o algoritmo trabalhe por um período de tempo muito maior;
- **Taxa de Cruzamento:** quanto maior for a taxa mais rapidamente novas estruturas serão introduzidas na população, porém se esta for muito alta, estruturas com boas aptidões poderão ser retiradas mais rápido;
- **Taxa de Mutação:** quando a taxa é baixa previne que uma dada posição fique estagnada em um valor, no entanto, com uma taxa muito alta a busca se torna essencialmente aleatória;
- **Intervalo de Geração:** controla a porcentagem da população que será substituída durante a próxima geração: com um valor alto, a maior parte da população será substituída, mas com valores muito altos pode ocorrer perda de estruturas de alta aptidão. Com um valor baixo, o algoritmo pode tornar-se muito lento.

Capítulo 3

Visão Computacional

Na Fig. 3.1 rerepresentamos a estrutura do Sistema de Navegação em camadas da Fig. 1.2, porém, agora com ênfase à camada de Visão Computacional.

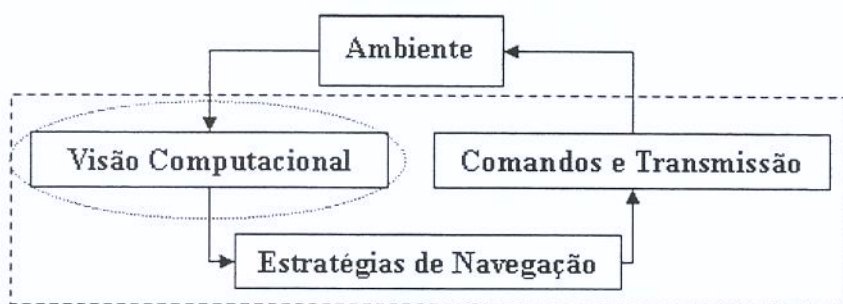


Fig. 3.1- Sistema de Navegação ênfase Visão Computacional.

Antes de iniciar a descrição dos processos desta camada, vale a pena chamar a atenção para as suas quatro variáveis de saída, isto é, as variáveis que representam o resultado do processo de visão computacional aplicado sobre cada imagem. São elas:

- Vetor de Profundidade $\vec{VP}$ - variável para análise da profundidade da área de navegação, Eq. (3.9) na Seção 3.2;
- Vetor Maximização $\vec{VM}$ - representação da distribuição de pontos de navegação versus pontos não navegáveis na imagem, Eq. (3.21) na Seção 3.5;
- Velocidade v - velocidade calculada a partir do vetor, Eq. (3.19) na Seção 3.4;

- Ângulo de Alinhamento ao Centro da Área de Navegação (α) - ângulo de correção de direção ao centro da área de navegação, Eq. (3.17) na Seção 3.3.3;

Conforme apresentado no capítulo Revisão Bibliográfica, os processos desempenhados pela camada de Visão Computacional têm como objetivo processar os dados das imagens submetidas ao sistema. Seus algoritmos realizam operações com o objetivo de reduzir ruídos e segmentá-las. Particularmente, deve alimentar a camada Estratégias de Navegação.

Diversos métodos podem ser aplicados para permitir o reconhecimento de padrões em imagens. Esses métodos geralmente são conhecidos como métodos de segmentação de imagens e, segundo GONZALEZ & WOODS (2000), podem ser considerados como a partição de imagens digitais em conjuntos de pixels considerando-se a necessidade geral e critérios previamente definidos.

A seguir apresentamos as técnicas selecionadas para o processo de visão computacional, iniciando com o pré-processamento da imagem.

3.1 Pré-Processamento de Imagem

Muitos dos grupos de pesquisa em visão computacional utilizam algoritmos de filtragem altamente sofisticados. Na maior parte, imagens em nível de cinza são usadas, porém em outros, imagens coloridas: este é o caso do MOSFET⁹ robô móvel, que utiliza um algoritmo de segmentação para maximizar o contraste entre as linhas da pista e a estrada (BERTOZZI et al., 2000).

Em nosso sistema, através da Eq. (3.1), imagens coloridas no formato RGB são convertidas para níveis de cinza antes de serem submetidas ao processo de segmentação.

$$C(x, y) = \frac{R(x, y) + G(x, y) + B(x, y)}{3} \quad (3.1)$$

⁹ *Michigan Off-road Sensor Fusing Experimental Testbed.*

Em um passo seguinte, na Eq. (3.2) suavizamos a nova imagem obtida na Eq. (3.1).

$$I_n = I \bullet S \quad (3.2)$$

onde I_n é a nova imagem após a convolução, I é a imagem original e o filtro S é apresentado a seguir:

$$S = \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$

3.2 Segmentação de Imagem

O objetivo da segmentação é a identificação da área de navegação nas imagens, ou seja, a classificação dos pixels da imagem em duas classes: Área de Navegação e Obstáculos. Logo após o pré-processamento da imagem, a busca por um limiar ideal é iniciada utilizando-se a técnica de segmentação proposta por OTSU (1978), na qual sua principal característica é a maximização da variância entre as classes da imagem, isto é, a maximização da separação entre o objeto e o fundo da imagem. Na Eq. (3.3), temos que este processo de limiarização é o particionamento dos pixels de uma imagem em duas classes: $C1=\{0,1,...,T\}$ e $C2=\{T+1,T+2,...,N-1\}$, sendo que T é o limiar escolhido e N é o número de níveis de intensidade da imagem. As Eq. (3.4), (3.5), (3.6), (3.7) e (3.8) encontram o limiar ótimo.

$$\sigma_w^2(T) = q_1(T)\sigma_1^2(T) + q_2(T)\sigma_2^2(T) \quad (3.3)$$

onde $\sigma_w^2(T)$ é a variância intra-grupo, $q_1(T)$ a proporção dos pixels do fundo, $\sigma_1^2(T)$ a variância das intensidades dos pixels do fundo, $q_2(T)$ a proporção dos pixels do objeto e $\sigma_2^2(T)$ a variância das intensidades dos pixels do objeto.

$$O = \frac{[m_1(K)P_2(K) - m_2(K)P_1(K)]^2}{P_1(K)P_2(K)} \quad (3.4)$$

$$m_1(K) = \sum_{i=0}^K ih(G_i) \quad (3.5)$$

$$m_2(K) = \sum_{i=K+1}^{G_{\max}-1} ih(G_i) \quad (3.6)$$

$$P_1(K) = \sum_{i=0}^K h(G_i) \quad (3.7)$$

$$P_2(K) = \sum_{i=K+1}^{G_{\max}-1} h(G_i) \quad (3.8)$$

onde m_1 é o somatório das intensidades de cores dos pixels de 0 até o limiar, m_2 é somatório das intensidades de cores dos pixels do limiar até o nível máximo, P_1 é o somatório dos pixels de intensidade de 0 até o limiar, P_2 é o somatório dos pixels de intensidade limiar até o nível máximo e $h(G_i)$ o histograma da imagem.



Fig. 3.2 – Imagem Original, Segmentação Otsu e Histograma da ImagemOriginal Pré-Processada.

Uma das grandes vantagens deste método é que não se restringe ao tipo de histograma da imagem. Sua principal desvantagem está no desempenho sensível à presença de ruídos, o que pode ser reduzido com a aplicação do filtro de suavização. No entanto, SAHOO et al. (1988) e LEE et al. (1990) consideram este método como uma das melhores escolhas para aplicações em tempo real em visão computacional.

Apesar do método proposto por Otsu ser um ótimo método para a escolha de um limiar ideal, para todos os casos ele considera a informação da imagem como um todo. A Fig. 3.2 mostra que para imagens que tenham em sua composição a imagem do horizonte, é provável que o algoritmo falhe na identificação da área de navegação.

Para encontrar um limiar ideal, alguns sistemas foram projetados para investigar apenas uma porção da estrada à frente do veículo, onde a ausência de outros veículos pode ser assumida. Como exemplo, o *AKand SAVE* veículos autônomos (BERTOZZI et al., 2000), confiam no processo da porção de imagem que corresponde a 12 metros de estrada à frente do veículo, e foi demonstrado que esta aproximação permite que o veículo possa manobrar seguramente em rodovias e em outras regiões em curvas de até 50 metros.

O sistema RALPH¹⁰, ao invés disso, reduz a porção da imagem a ser processada de acordo com o resultado do módulo de detecção de obstáculo baseado em radar (BERTOZZI et al., 2000).

¹⁰ *Rapidly Adapting Lateral Position Handler.*

Em outros sistemas a área com as linhas da pista são localizadas primeiro. O grupo de pesquisa do *Laboratoire Régional des Ponts-et-Chaussées de Strasbourg* explora a suposição que sempre deveria haver um contraste cromático entre estrada e não-estrada, pelo menos em uma componente de cor; para separar as componentes, o conceito de saturação cromática é usado (BERTOZZI et al., 2000).

No método *TH Finder*¹¹ proposto por MIRANDA NETO & RITTNER (2006), o algoritmo proposto por Otsu é utilizado para encontrar o ponto de corte ideal na imagem. O resultado necessariamente divide a imagem em partes iguais, mas em duas sub-imagens complementares.

A Fig. 3.3 mostra um exemplo desta divisão em *Up* e *Down*, respectivamente, contribuem para decisões futuras e para decisões imediatas.

Embora tenhamos adotado a divisão da imagem como uma boa opção, ainda permanecia a dúvida de qual seria a porcentagem ideal para atribuir para cada parte da imagem.

Alguns trabalhos previamente citados administram este problema fixando uma distância específica para cortar a imagem. Esta abordagem os faz dependentes de parâmetros de calibração de câmera e/ou de diferentes sensores para adquirir estas informações, principalmente às de profundidade.



Fig. 3.3 – Corte da Imagem em *Up* e *Down*.

Conforme apresenta a Fig. 3.4, para encontrar o ponto de corte de forma automática pelo sistema, o *TH Finder* divide a imagem em dez fatias de alturas iguais. Após isso, o algoritmo inicia analisando a fatia mais próxima ao robô móvel, isto é, a fatia vai da margem inferior da imagem até o primeiro ponto de corte imaginário. Desta forma, a primeira sub-imagem terá 10%

¹¹ *Threshold and Horizon Finder*.

da altura total da imagem original, a segunda sub-imagem 20%, e assim sucessivamente até completar os 100%.

A divisão da imagem em fatias nos permitiu analisar o quanto que a inclusão de uma fatia superior da imagem ao processo de segmentação contribuía positivamente para o aumento de pontos de navegação. Em outras palavras, por estarmos utilizando um método de segmentação global, nem sempre a análise de uma porção maior da imagem pode contribuir para um resultado de segmentação melhor na região mais crítica, isto é, na região mais próxima ao robô móvel.

Todas as dez sub-imagens obtidas através do processo apresentado na Fig. 3.4 são submetidas ao algoritmo de segmentação *TH Finder*, e como resultado final deste processo, temos na Eq (3.9) um vetor de profundidade ($\vec{VP}$), que em cada índice possui o valor da porcentagem de pontos de navegação para cada sub-imagem analisada.

$$\vec{VP}_i = \frac{NP_i * 100}{(H * W)_i} \quad (3.9)$$

onde i representa o índice da fatia analisada, NP a quantidade de Pontos de Navegação, H a altura da imagem e W a largura da imagem.

Uma vez criado o vetor com valores representativos de cada uma das sub-imagens, no próximo passo, o cálculo do desvio padrão amostral de seus valores é realizado na Eq. (3.10).

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (VP_i - \bar{VP})^2} \quad (3.10)$$

Para encontrar o ponto de corte na imagem, subtraímos o resultado do desvio padrão amostral do valor da porcentagem de pontos de navegação contido na primeira posição do vetor $\vec{VP}$. Em seguida, iniciando a partir da segunda posição do vetor, buscamos o índice que contenha o último

valor maior do que o resultado do último cálculo. Este índice representará o ponto de corte na imagem original, obtendo-se então as duas imagens complementares: *Down* e *Up*.

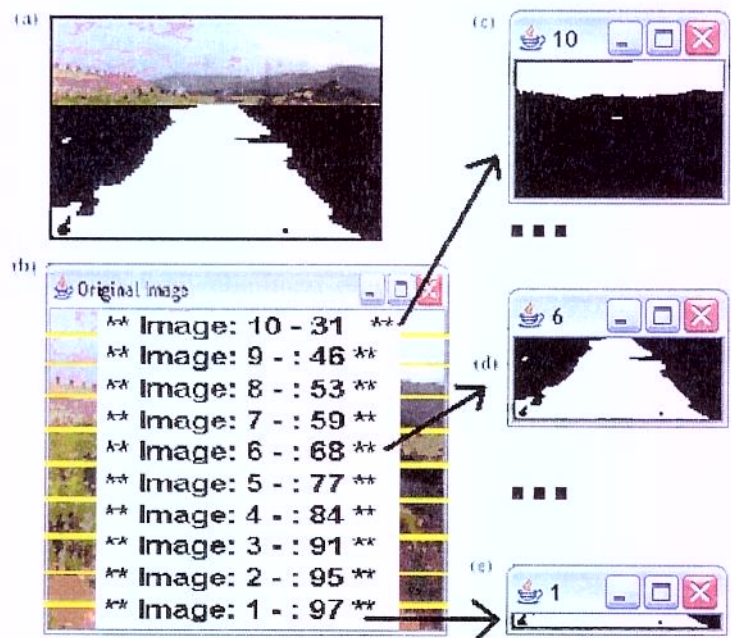


Fig. 3.4 – Resumo do processo de Segmentação *TH Index* : (a) Resultado final: definição do ponto de corte e segmentação; (b) Representação das sub-imagens resultantes do fatiamento; (c), (d) e (e) Respectivamente, sub-imagens: 10%, 60% e 100%.

Como exemplo, na Fig. 3.4 (a) temos como resultado o ponto de corte igual a 6. Desta forma, a imagem *Down* conterá 60% da altura da imagem original e a imagem *Up* conterá 40% da altura da imagem original.

Diferente do exemplo acima, para uma imagem com 100% de área de navegação em todas as suas fatias, o valor do desvio padrão amostral será igual a 0. Neste caso, a imagem toda será aproveitada.

Apresentado o processo de partição da imagem em fatias, para concluir com êxito a etapa de segmentação ainda é preciso fazer uma última análise dos resultados obtidos, isto é, ao analisar o resultado da segmentação da primeira fatia da imagem, o algoritmo verificará a porcentagem de pontos de navegação desta região, e caso o valor de porcentagem de pontos de navegação seja inferior a 30%, o que significa 70% de obstáculos, o processo de segmentação é interrompido e a imagem original pré-processada é invertida, e nova imagem negada é submetida ao processo de

segmentação. O resultado do processo de negação de uma imagem é apresentado na Fig. 3.5 e na Eq.(3.11), onde r e s denotam os níveis de cinza de entrada e saída, respectivamente.

$$s = T(r) \quad (3.11)$$

No entanto, em virtude do processo de negação, para uma imagem com 100% de obstáculos na primeira fatia, o sistema a validaria como área de navegação. Para isso, após a negação da imagem, caso a nova imagem pós-negação contenha em sua primeira fatia 100% de pontos de navegação, o processo de negação é desfeito e a imagem original antes da negação é utilizada. Como exemplo, tendo uma imagem 100% de pontos não navegáveis, o que representa uma possível colisão, em todos os índices do vetor $\vec{VP}$ teremos valores iguais a 0.



Fig. 3.5 – Imagem Original, Imagem Negada e Resultado da Segmentação *TH Finder*.

Uma vez definida a imagem *Down*, para todos os índices do vetor $\vec{VP}$ que representam a imagem *Up* será atribuído o valor 0. Este procedimento permite encontrar um valor da velocidade do robô. Isto porque, o vetor $\vec{VP}$ é uma fonte de informações sobre a profundidade da área de navegação nas imagens, principalmente considerando um sistema de monovisão e o fato do sistema não possuir informações complementares de sensores de profundidade.

Na Fig. 3.6 apresentamos o resultado do trabalho realizado por Thurn da Universidade de Stanford em parceria com empresas (THURN et al., 2005). Trata-se do processo de visão computacional utilizado na ocasião do *2005 Grand Challenge*, porém com auxílio de laser quadrilateral, excelente ferramenta para detecção de fendas na área de navegação.

Na Fig. 3.7 apresentamos o resultado da segmentação *TH Finder* utilizando a imagem da Fig. 3.6 (a). Na Fig. 3.7 (b) apresentamos a conversão da imagem original para uma imagem em níveis de cinza, e na Fig. 3.7 (c) o resultado da operação de filtragem. Por fim, na Fig. 3.7 (d) o resultado da segmentação *TH Finder*.

É importante notar que algoritmo *TH Finder* não é baseado em conhecimento prévio do ambiente, por exemplo: forma da pista, inferência geométrica; e não exige conhecer informações do sistema de aquisição de imagens. Também, não depende de informações de marcas ou sinais da área de navegação e obtém bons resultados em estradas estruturadas e não estruturadas. E por ser baseado no método de Otsu, busca limiares dinamicamente e não sofre alteração de resultado em virtude dos efeitos das mudanças de iluminação, além de não exigir ajustes de contraste. Por último, apresentamos na Fig. 3.8 os resultados do algoritmo para imagens que possuem atributos de oclusão e/ou textura da área de navegação irregular.

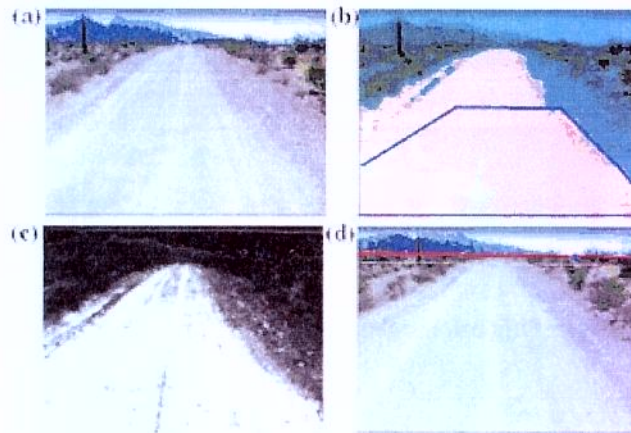


Fig. 3.6 – Estágios do processamento do sistema de visão computacional de *Stanford*: (a) Imagem Original; (b) Imagem Processada com laser quadrilateral e classificação de pixel. (c) A classificação de pixel antes do *thresholding*. (d) Detecção do horizonte para remoção do céu.

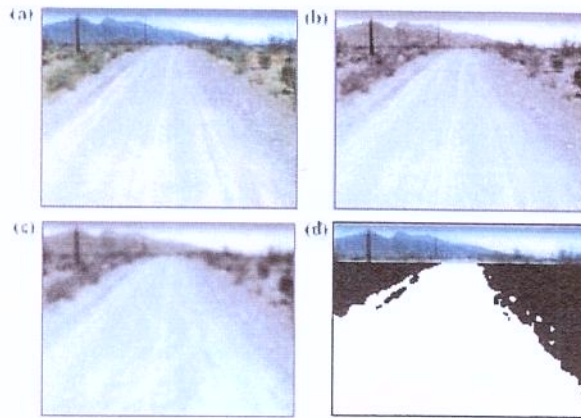


Fig. 3.7 – Resultados da segmentação *TH Finder*: (a) Imagem Original utilizada por Stanford; (b) Imagem em Níveis de Cinza; (c) Imagem Filtrada; (d) Identificação da Área de Navegação após Segmentação.

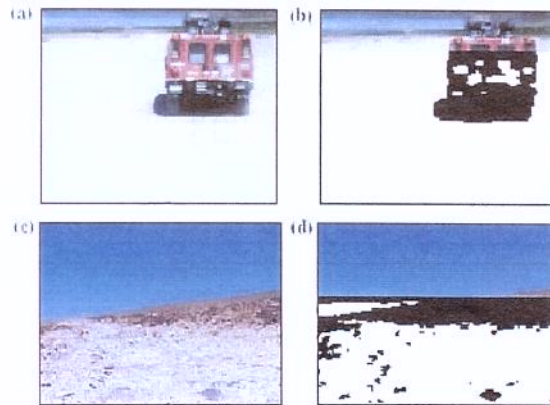


Fig. 3.8 – Resultados do algoritmo *TH Finder*: (a) Imagem Original 1; (b) Imagem com Oclusão; (c) Imagem Original 2; (d) Imagem com Textura da Área de Navegação Irregular.

3.3 Ângulo de Correção de Direção

Conforme apresenta a Fig. 3.9, para efetuar o cálculo do ângulo (α) de correção de direção três passos são necessários:

- Determinação do ponto de origem na imagem;
- Cálculo do centro de massa na imagem segmentada: o centróide representa o ponto de destino;
- Cálculo do ângulo (α) entre ponto de origem e ponto de destino;

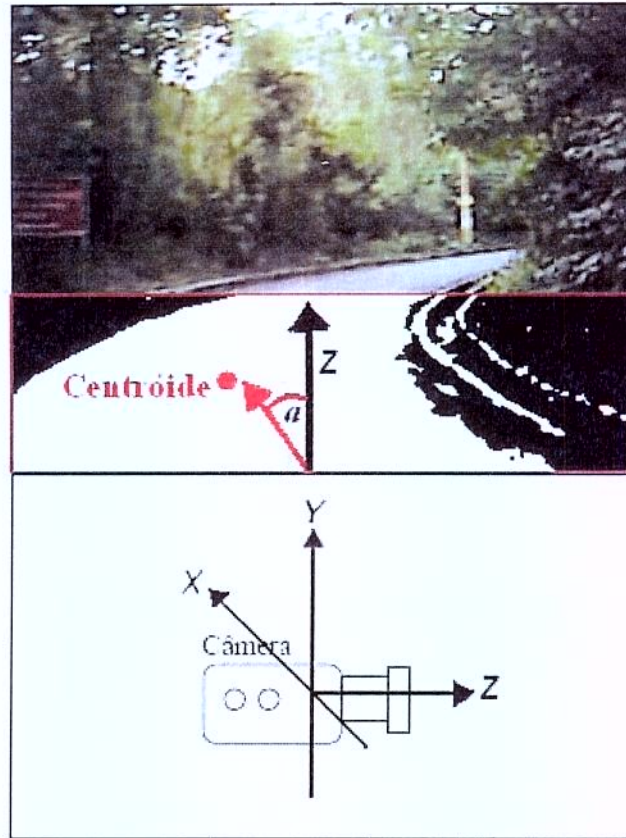


Fig. 3.9 – Ângulo de Correção de Direção

3.3.1 Determinação do Ponto de Origem do Sistema

A determinação do ponto de origem na imagem será de fundamental importância para a identificação da direção do robô móvel, isto é, eixo Z da câmera, considerando que o plano imagem XY está alinhado com a frente do robô móvel.

A Eq. (3.12) e Eq. (3.13) calculam o ponto de origem do robô:

$$x = \frac{\text{imagem.Largura}}{2} \quad (3.12)$$

$$y = \text{imagem.Altura} \quad (3.13)$$

3.3.2 Centro de Massa da Área de Navegação

Após a segmentação do algoritmo *TH Finder*, teremos uma imagem com duas classes, isto é, c_1 área de navegação e c_2 obstáculos.

Segundo BALAN et al. (2004), para encontrar o centro de massa da classe que representa a área de navegação, podemos utilizar a Eq. (3.14).

$$m_c = \sum_{x,y} h_c(x,y) \quad (3.14)$$

$$\text{onde } h_c(x,y) = \begin{cases} 1, & \text{se } S_{x,y} = c_1 \\ 0, & \text{se } S_{x,y} \neq c_1 \end{cases}$$

O centro de massa da região de classe c_1 denotado o_{c1} é dado pelas coordenadas xo_{c1} e yo_{c1} , calculadas nas Eq. (3.15) e Eq. (3.16).

$$xo_{c1} = \frac{\sum_{x,y} x \cdot h_c(x,y)}{m_c} \quad (3.15)$$

$$yo_{c1} = \frac{\sum_{x,y} y \cdot h_c(x,y)}{m_c} \quad (3.16)$$

3.3.3 Correção de Direção

Calculado o ponto de origem pelas Eq. (3.12) e Eq. (3.13) e destino pelas Eq. (3.15) e Eq. (3.16), na Eq. (3.17) temos o ângulo a de correção de direção, sendo que x e y são os pontos de origem, e xo_{c1} e yo_{c1} os pontos de destino. O objetivo principal deste cálculo é permitir que o

robô móvel permaneça alinhado ao centro da área de navegação. Na Fig. 3.10 temos um exemplo que representa esta operação.

$$a = ((\text{arc} . \cos(\frac{|(yo_{cl} - y)|}{\sqrt{(xo_{cl} - x)^2 + (yo_{cl} - y)^2}})) * 180) / \pi \quad (3.17)$$

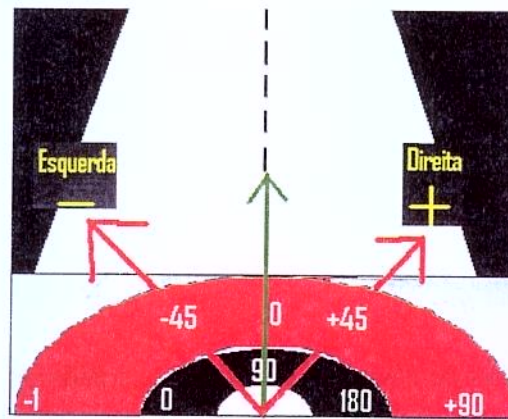


Fig. 3.10 – Centro da Área de Navegação.

3.4 Velocidade

A partir dos valores do vetor $\vec{VP}$ calculamos a velocidade do robô móvel. É importante notar que os valores resultantes das operações a seguir não indicam a velocidade real do robô móvel, no entanto, encontramos um valor velocidade classificado num intervalo de 0 a 10.

Para o primeiro caso apresentado na Fig. 3.11 e a Tabela 3.1, temos uma imagem com 100% de área de navegação, visto que somente para uma imagem deste tipo, o valor da velocidade pode ter o seu valor máximo. A justificativa, baseia-se no fato que, para a normalização de valores de um vetor, devemos encontrar primeiro o seu maior valor, para depois realizar as operações seguintes. Dois valores chamam a atenção na Tabela 3.1:

- **292,90, VMI :** o Valor Máximo possível obtido por uma Imagem (vetor $\vec{VP}$) submetida ao sistema;
- **10:** o valor máximo possível atribuído à velocidade;

É importante notar que o VMI , isto é, o valor **292,90**, será uma constante do sistema. Seu valor foi obtido na Eq. (3.18), considerando os valores da Tabela 3.1.

$$VMI = \sum \vec{VP}_i / i \quad (3.18)$$

Para um melhor entendimento, a seguir detalhamos cada uma das informações da Tabela 3.1:

- **Coluna Vetor (i):** índices (i) do vetor $\vec{VP}$, sendo o valor 1 sua primeira posição e 10 a última;
- **Coluna Porcentagens (P):** valores de cada posição $\vec{VP}_i$ do vetor $\vec{VP}$, sendo que todas as posições estão preenchidas com o valor 100, pois se trata de uma imagem com 100% de área de navegação;
- **Coluna P/i :** considerando os valores das duas colunas apresentadas anteriormente, os valores desta coluna são obtidos a partir do seguinte cálculo: $\vec{VP}_i / i$, onde o divisor i representa o peso que recai sobre o valor de $\vec{VP}_i$, que representa a distribuição vertical de pontos de navegação na imagem;
- **Coluna Cálculo da Velocidade [0..10]:** considerando os valores das três colunas apresentadas anteriormente, os valores desta coluna são obtidos a partir do seguinte cálculo: $\frac{(\vec{VP}_i / i) * 10}{VMI}$, sendo que a constante 10 representa o maior valor possível para a velocidade;

O valor da velocidade (v), representado na Tabela 3.1, foi calculado na Eq. (3.19). Nesta equação, a constante 10 representa o maior valor possível para a velocidade:

$$v = \sum_{i=1}^{10} \frac{(\vec{VP_i} / i) * 10}{VMI} \tag{3.19}$$

Nota-se que uma imagem com 100% de área de navegação será totalmente branca, conforme Fig. 3.11. Já um obstáculo cobrindo 100% da imagem será representado preta, conforme a Fig. 3.15.

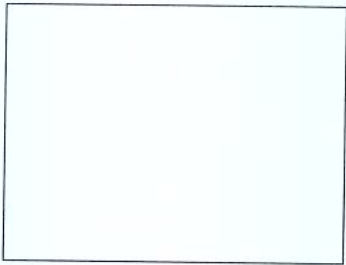


Fig. 3.11 – Imagem com 100% de Área de Navegação.

Tabela 3.1 – Valores para a Fig. 3.11.

Vetor (i)	Porcentagens (P)	P/i	Cálculo da Velocidade [0..10]
10	100	10,00	0,34
9	100	11,11	0,38
8	100	12,50	0,43
7	100	14,29	0,49
6	100	16,67	0,57
5	100	20,00	0,68
4	100	25,00	0,85
3	100	33,33	1,14
2	100	50,00	1,71
1	100	100,00	3,41
Total:	1000	292,90	10,00
Constante VMI :		292,90	

Para os casos a seguir, a Eq. (3.19) foi utilizada para obter o valor da velocidade (v).

- Os resultados para a Fig. 3.12 são apresentados na Tabela 3.2;
 - Imagem com obstáculo móvel ao longe e grande espaço para navegação;
- Os resultados para a Fig. 3.13 são apresentados na Tabela 3.3;
 - Imagem com grande influência de iluminação, isto é, ruído elevado;
- Os resultados para a Fig. 3.14 são apresentados na Tabela 3.4;
 - Imagem com curva acentuada e espaço reduzido de área de navegação;
- Os resultados para a Fig. 3.15 são apresentados na Tabela 3.5;
 - Imagem com 100% de obstáculo, possível colisão;

Nas tabelas a seguir, os valores contidos nos círculos representam valores atribuídos ao vetor $\vec{VP}$. A região com valores diferentes de 0 representam a imagem *Down*.



Fig. 3.12 – Par de imagens 1: Imagem Original e Resultado *TH Finder*.

Tabela 3.2 – Valores para a Fig. 3.12.

Vetor (<i>i</i>)	Porcentagens (<i>P</i>)	<i>P</i> / <i>i</i>	Cálculo da Velocidade [0..10]
10	0	0,00	0,00
9	0	0,00	0,00
8	99	12,38	0,42
7	100	14,29	0,49
6	100	16,67	0,57
5	100	20,00	0,68
4	100	25,00	0,85
3	100	33,33	1,14
2	100	50,00	1,71
1	100	100,00	3,41
Total:	799	271,66	9,27
Constante <i>VMI</i> : 292,90			



Fig. 3.13 – Par de imagens 2: Imagem Original e Resultado *TH Finder*.

Tabela 3.3 – Valores para a Fig. 3.13.

Vetor (<i>i</i>)	Porcentagens (<i>P</i>)	<i>P</i> / <i>i</i>	Cálculo da Velocidade [0..10]
10	0	0,00	0,00
9	0	0,00	0,00
8	0	0,00	0,00
7	0	0,00	0,00
6	0	0,00	0,00
5	46	9,20	0,31
4	47	11,75	0,40
3	46	15,33	0,52
2	46	23,00	0,79
1	42	42,00	1,43
Total:	227	101,28	3,46
Constante <i>VMI</i> :		292,90	



Fig. 3.14 – Par de imagens 3: Imagem Original e Resultado *TH Finder*.

Tabela 3.4 – Valores para a Fig. 3.14.

Vetor (<i>i</i>)	Porcentagens (<i>P</i>)	<i>P</i> / <i>i</i>	Cálculo da Velocidade [0..10]
10	0	0,00	0,00
9	0	0,00	0,00
8	0	0,00	0,00
7	0	0,00	0,00
6	0	0,00	0,00
5	0	0,00	0,00
4	58	14,50	0,50
3	63	21,00	0,72
2	68	34,00	1,16
1	68	68,00	2,32
Total:	257	137,50	4,69
Constante <i>VMi</i> : 292,90			



Fig. 3.15 – Imagem com 100% de Obstáculo.

Tabela 3.5 – Valores para a Fig. 3.15.

Vetor (i)	Porcentagens (P)	P/i	Cálculo da Velocidade [0..10]
10	0	0,00	0,00
9	0	0,00	0,00
8	0	0,00	0,00
7	0	0,00	0,00
6	0	0,00	0,00
5	0	0,00	0,00
4	0	0,00	0,00
3	0	0,00	0,00
2	0	0,00	0,00
1	0	0,00	0,00
Total:	0	0,00	0,00
Constante VMI : 292,90			

3.5 Identificação de Obstáculos

A idéia inicial do processo de segmentação e identificação da área de navegação não é obter resultados perfeitos a ponto de deixar toda a área de navegação com pontos brancos, mas sim encontrar o maior número deles, para que o sistema possa indicar as correções de rota necessárias. Isso porque, ao analisar os resultados do algoritmo de segmentação selecionado, percebe-se claramente que para uma navegação segura, a visão computacional não pode ser a única fonte de informações do sistema de navegação, a não ser que o ambiente onde se deseja navegar seja controlado.

Desta forma, para desviar dos obstáculos e evitar colisões, é preciso encontrar o maior número de pontos brancos na imagem, que representam a área de navegação.

Isolando apenas a imagem *Down*, conforme apresentado na Fig. 3.16, uma matriz maximização é criada com mesmas dimensões desta sub-imagem. Para cada ponto da sub-imagem, o valor atribuído para a matriz maximização será calculado com base na função de maximização, Eq. (3.20), onde x representa as colunas e y as linhas da imagem. O objetivo desta função é definir a influência dos pontos de obstáculos na imagem analisada.

$$f (x , y) = y * \left\{ \begin{matrix} 0 \text{ White} \\ 1 \text{ Black} \end{matrix} \right\} \quad (3.20)$$

Após isso, um vetor maximização ($\vec{VM}$) é criado, e em cada posição deste temos a soma de todos os pontos referentes a cada coluna da matriz de maximização M , conforme a Eq. (3.21).

$$\vec{VM}_i = \sum_{j=1}^n M (x_i, y_j) \quad (3.21)$$

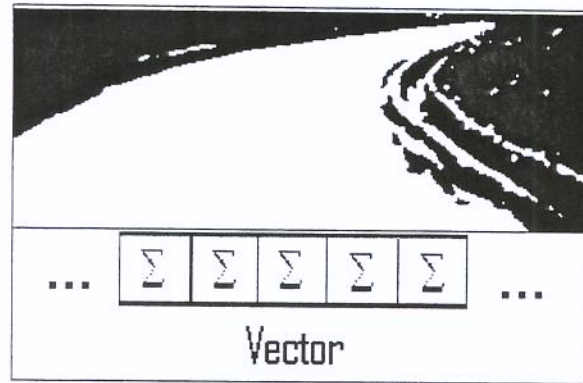


Fig.3.16 – Representação da informação horizontal na imagem através do Vetor Maximização.

Com esta operação, em cada posição do vetor $\vec{VM}$, temos valores que correspondem à influência de pontos de obstáculos versus pontos de navegação para cada região da imagem por coluna.

O vetor de maximização será um dos parâmetros de entrada para a camada Estratégias de Navegação, que decidirá por determinada rota, dependendo da disposição dos valores do vetor $\vec{VM}$.

3.6 Descarte de Informações Redundantes

Geralmente o emprego contínuo do processo de segmentação de imagens demanda um alto custo computacional, contribuindo negativamente no quesito tempo de resposta para sistemas de navegação autônomos que operam em tempo real. Em particular, o Coeficiente de Correlação de Pearson - CCP foi utilizado como uma alternativa, conforme apresentado por MIRANDA NETO et al. (2007).

Diferente do sistema humano, para sistemas computacionais, a simples tarefa de segmentação pode desencadear alguns prejuízos devido ao tempo de processamento. Desta forma, a decisão por um sistema de visão computacional mais complexo leva possivelmente a um sistema lento demais para uma aplicação autônoma de tempo real. Ou ainda, a escolha automática por um

processo randômico de descarte de imagens pelo sistema, para torná-lo suficientemente rápido, pode resultar em perda de informações importantes.

Mesmo que o sistema possua uma base de armazenamento de imagens adquiridas e as submeta, por exemplo, a uma rede neural para decisão, o grande número de informações não necessariamente se traduz em melhores decisões e podem inclusive prejudicar o desempenho do sistema, sobrecarregando-o.

Porém, definir o critério de descarte automático de informações redundantes com o menor prejuízo possível, isto é, permitir que apenas algumas das imagens sejam submetidas ao processo de segmentação e garantir que informações importantes não tenham sido perdidas, pode não ser uma tarefa trivial para sistemas computacionais, principalmente quando estes operam em tempo real.

Nesse intuito, por possuir baixa complexidade e ser de fácil implementação, utilizamos o CCP, apresentado anteriormente na Eq. (2.2).

Conforme apresentado na Fig. 3.17, complementado na Fig. 3.18, o CCP é um processo da camada Visão Computacional e precede o processo de pré-processamento e segmentação.

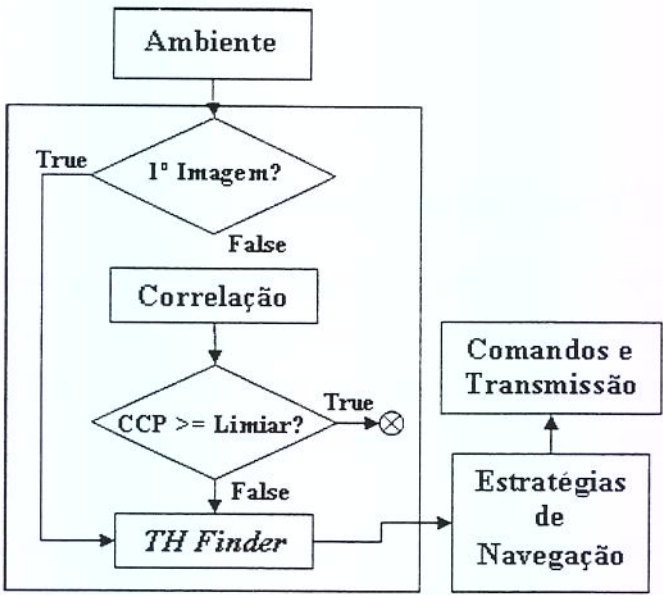


Fig. 3.17 – Fluxo dos dados da camada Visão Computacional.

Antes de submeter uma imagem ao processo de segmentação, o CCP é utilizado para obter o fator de correlação entre a imagem que está sendo submetida ao sistema e a imagem anterior, denominada de imagem parâmetro.

Caso o valor de correlação CCP obtido esteja dentro da margem previamente definida como adequada, a imagem atual não será submetida ao processo de segmentação, mas sim descartada pelo sistema. Neste caso, o sistema executará os movimentos definidos para a imagem parâmetro, isto é, repetirá o último movimento realizado.

Já para o caso em que o valor resultante da correlação CCP esteja abaixo do limiar definido, a imagem atual será submetida ao processo de segmentação e se tornará a nova imagem parâmetro do sistema.

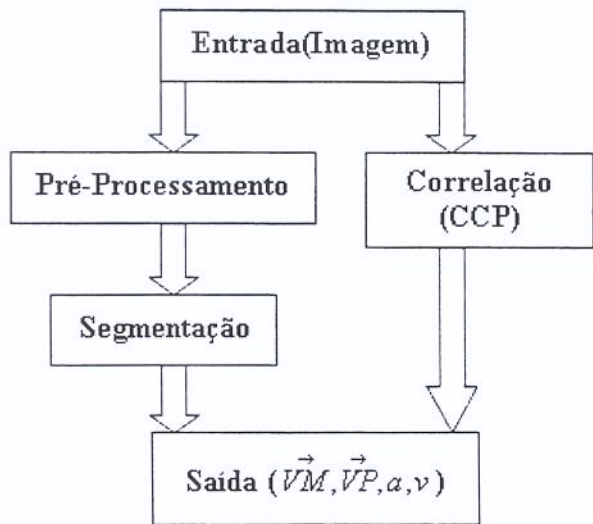


Fig. 3.18 – Coeficiente de Correlação de Pearson na camada Visão Computacional.

Na Fig.3.19 apresentamos quatro imagens extraídas de um trecho do vídeo *RaceDay* disponível no DARPA (2004). Na Fig.3.19 (a) temos o *Frame* 4440, na Fig. 3.19 (b) temos o *Frame* 4441, na Fig. 3.19 (c) temos o *Frame* 4450 e na Fig.3.19 (d) temos o *Frame* 4460. Os Coeficientes de Correlação de Pearson - CCP obtidos foram:

- Entre o *Frame* 4440 e *Frame* 4441: **0.9949**;
- Entre o *Frame* 4440 e *Frame* 4450: **0.8673**;
- Entre o *Frame* 4440 e *Frame* 4460: **0.7464**;

Analisando os valores resultantes do CCP entre as imagens da Fig. 3.19, é possível perceber que quanto maior for a influência de um objeto entre cenas subsequentes, menor será a correlação, portanto, o processo de segmentação será necessário. Isto é justificável, pois a cada vez que o um objeto fica maior na cena analisada, mais precisos deverão ser os movimentos definidos pelo sistema robótico.

Na Fig. 3.20 apresentamos outras três imagens do mesmo vídeo. Na Fig.3.20 (a) temos o *Frame* 4020, na Fig. 3.20 (b) temos o *Frame* 4120, na Fig.3.20 (c) temos o *Frame* 4220 e na Fig.3.20 (d) temos o *Frame* 4320. Os coeficientes obtidos foram:

- Entre o *Frame* 4020 e *Frame* 4120: **0.9591**;
- Entre o *Frame* 4020 e *Frame* 4220: **0.9407**;
- Entre o *Frame* 4020 e *Frame* 4320: **0.8861**;

Neste segundo caso, verificamos que o processo de segmentação quase não foi requisitado, pois para uma sequência de 300 *frames*, dependendo do limiar adotado para corte, os comandos de movimentos para o robô móvel, provenientes da análise da primeira imagem, podem ser aproveitados para todas as imagens do intervalo, isto é, os comandos de movimentos definidos a partir do processo de segmentação para o *Frame* 4020, imagem parâmetro, podem ser aproveitados até o *Frame* 4320. Graças a alta correlação entre as imagens desta sequência, o processo de segmentação pôde ser ignorado trezentas vezes.

É importante notar que o tempo computacional médio do nosso processo de correlação foi aproximadamente 85% menor que o tempo computacional médio do processo de segmentação somado do pré-processamento. Destacamos também, que o pré-processamento é um processo após a correlação, isto é, caso uma imagem seja descartada pelo processo de correlação, os processos de pré-processamento e segmentação serão dispensados.

Os tempos de processamento citados acima referem-se aos algoritmos implementados em Java e executados em um Laptop Computer – Intel Processor Core Duo T2300E - 2M L2 667Mhz – 512 MB - Windows XP Professional.

Em outra simulação submetemos ao nosso sistema um vídeo com 10 mil *frames*, equivalente a um vídeo de 6.6 minutos captados a uma taxa de 25 *frames* por segundo.

Dos 10 mil *frames* submetidos à camada de Visão Computacional, apenas 639 foram submetidos ao processo pré-processamento e segmentação *TH Finder*, isto é, ao selecionarmos um limiar de correlação de 0.8861, 9361 *frames* foram descartados automaticamente pelo sistema, mostrando que apenas 6.39% dos *frames* do vídeo foram utilizados para a geração de comandos para o robô móvel.

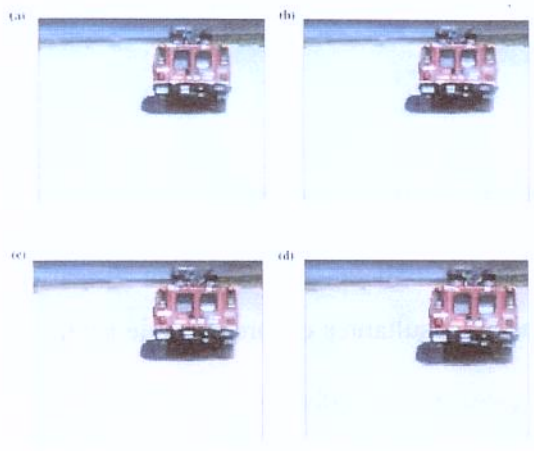


Fig. 3.19 – Sequência de Imagens 1.

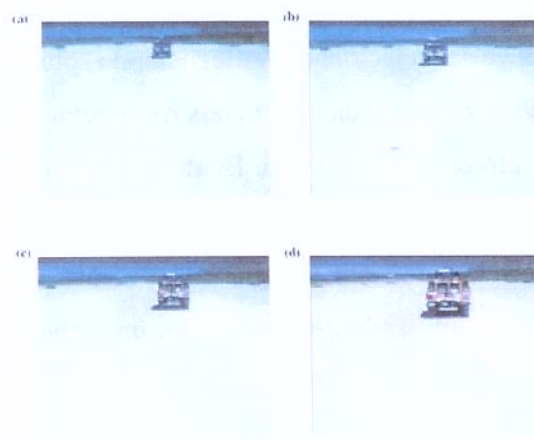


Fig. 3.20 – Sequência de Imagens 2.

Em outras simulações utilizando seqüências menores este percentual variou entre 8% e 10%, o que não está ligado ao tamanho do vídeo ou à quantidade de *frames*, mas sim à influência dos objetos nas imagens.

Finalmente, para assegurar que o processo de segmentação não fique ocioso num período grande de tempo, introduzimos um procedimento de interrupção no sistema de correlação, forçando a segmentação. Este processo de interrupção pode ser inicializado de tempos em tempos ou quando um número determinado de imagens já tiver sido descartada pelo sistema. Particularmente, decidimos que a cada 30 imagens descartadas uma interrupção é instanciada. Tal procedimento reforça a segurança no processo de descarte de imagens.

3.6.1 Limiar de Descarte de Informações não Determinístico

Para a adoção de um limiar não determinístico no descarte de informações redundantes, aproveitaremos as informações resultantes do processo de segmentação. Para isso, utilizamos as informações dispostas no vetor $\vec{VP}$. Conforme já exposto anteriormente, em cada índice deste vetor temos representadas informações de profundidade da área de navegação. Também é a partir deste vetor que obtemos a divisão da imagem em *Up* e *Down*.

Considerando que determinado índice do vetor $\vec{VP}$ representa um ponto de corte na imagem, para cada ponto selecionado como corte na imagem, um limiar de segurança no descarte de informações redundantes deve ser selecionado, influenciando sobre o valor de um limiar global.

Esta relação entre os valores dos índices e níveis de segurança do descarte pode ser melhor entendida observando os valores da Tabela 3.6. Evidenciamos, que para cada um dos possíveis índices de corte em uma imagem, dispostos nas linhas da coluna Índices da Tabela 3.6, um limiar de correlação deve ser selecionado, isto é, para as próximas comparações entre as imagens submetidas ao sistema, o limiar utilizado deve ser maior ou menor, ou seja, o nível de segurança do sistema no descarte de imagens deverá variar para mais ou para menos, dependendo do ponto de corte obtido no processo de segmentação para cada imagem.

Considerando a influência do fator iluminação e as características diferenciadas passíveis de serem encontradas a cada ambiente navegável, atribuir um valor arbitrário para o limiar de descarte de informações redundantes seria uma tarefa difícil. Para isso, convencionamos que o valor do limiar global para o descarte de imagens irá variar de acordo com a média dos limiares obtidos pelo processo de segmentação de cada imagem submetida ao sistema.

A partir da relação entre os valores apresentados na Tabela 3.6, temos o seguinte exemplo para uma sequência de 5 imagens:

- 1. Imagem 1: **Ponto de Corte 3**: Valor equivalente na coluna **CCP: 0.7**;
- 2. Imagem 2: **Ponto de Corte 4**: Valor equivalente na coluna **CCP: 0.6**;
- 3. Imagem 3: **Ponto de Corte 7**: Valor equivalente na coluna **CCP: 0.3**;
- 4. Imagem 4: **Ponto de Corte 5**: Valor equivalente na coluna **CCP: 0.5**;
- 5. Imagem 5: **Ponto de Corte 3**: Valor equivalente na coluna **CCP: 0.7**;

Somando-se todos os valores, temos: $0.7+0.6+0.3+0.5+0.7 = 2.8$, e valor médio $2.8/5 = 0.56$. Neste exemplo o valor do limiar global utilizado para o descarte de informações redundantes nas próximas interações com o ambiente é de 0.56, até que um novo valor contribua para a nova média.

Tabela 3.6 – Fatias da Imagem e limiares CCP.

Índices do VP (Fatias da Imagem)	Níveis de Segurança do Descarte (CCP)
10	0
9	0,1
8	0,2
7	0,3
6	0,4
5	0,5
4	0,6
3	0,7
2	0,8
1	0,9

No processo para obter o limiar para o descarte de informações redundantes, ainda é importante a determinação de uma faixa mínima e máxima para a variação, isto é, quando o valor encontrado for maior que o valor máximo determinado previamente, permanece o valor máximo. Já para o caso inverso, permanece o valor mínimo.

Particularmente, adotamos 1 como valor máximo e 0.7 como valor mínimo. Para o início da interação do robô móvel com o ambiente desconhecido, atribuímos ao limiar CCP o valor 1.

Capítulo 4

Estratégias de Navegação

Neste capítulo temos como objetivo apresentar a geração de movimentos para o robô móvel a partir das variáveis de saída da camada Visão Computacional, além da forma de tratamento para o fator iluminação, já que este não é uma constante em nosso sistema.

Destacamos que seu conteúdo não será detalhado em nossos experimentos, uma vez que o sistema não possui mecanismos de retro-alimentação, inviabilizando a validação de movimentos executados pelo robô. Desta forma, temos que o sistema de navegação estrutura as estratégias de movimentos a partir dos dados fornecidos pelo sistema de visão, alimentando o robô com novos movimentos. Enquanto o robô móvel interage com o ambiente, um sistema de aprendizado evolutivo atua como um ponderador de movimentos, ferramenta capaz de atenuar a influência pontual do fator iluminação sobre as imagens de uma cena. Particularmente, como estamos trabalhando com um único sensor de visão, o único fator a ser levado em conta será a iluminação.

Na Fig. 4.1 reapresentamos a estrutura do Sistema de Navegação em camadas da Fig. 1.2, porém com ênfase à camada Estratégias de Navegação.

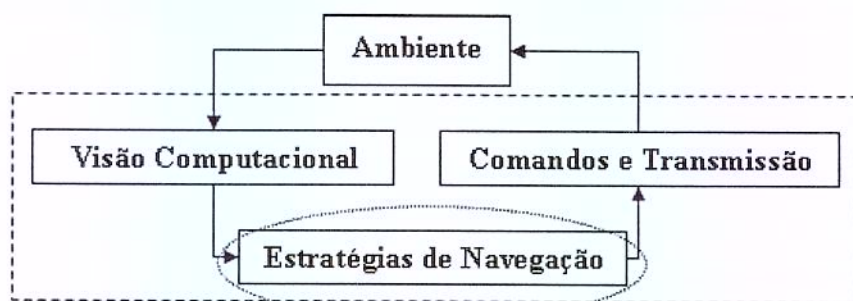


Fig. 4.1 – Sistema de Navegação ênfase Estratégias de Navegação.

Como entrada para a camada Estratégias de Navegação temos as seguintes variáveis: Vetor Maximização $\vec{VM}$, Vetor Profundidade $\vec{VP}$, velocidade v e ângulo α de alinhamento ao centro da área de navegação.

A camada Estratégias de Navegação esta dividida em duas camadas:

- Sistema Gerador de Classificadores - SGC;
- Sistema de Aprendizado Evolutivo - SAE;

O SGC é o responsável pela geração de classificadores. Cada classificador gerado é enviado para o SAE, que realiza as operações de seleção e evolução da população de classificadores. No SAE, o SC seleciona classificadores que possuem maior energia¹² para envio à próxima camada Comandos e Transmissão, e através de um Algoritmo Genético - AG, que atua como um ponderador secundário de movimentos, evoluções a partir de sub-populações¹³ são realizadas.

4.1 Sistema Gerador de Classificadores - SGC

Como foi discutido no capítulo Revisão Bibliográfica, para todo classificador temos uma parte-conseqüente. Para entender a trinca de valores que a representa, faz-se necessário conhecer as possibilidades de movimentos mapeados para o protótipo utilizado neste trabalho. Apresentadas a seguir, relacionando-as a vetores de movimentos, destacamos que em cada trinca de valor, 1 representa regiões de obstáculos e 0 regiões de navegação em uma imagem. A Fig. 4.2 apresenta os movimentos mapeados e suas respectivas trincas de valor.

1. 0 0 0: Frente $\vec{F}$;
2. 1 0 1: Frente-Estreta $\vec{FE}$;
3. 0 0 1: Frente-Esquerda $\vec{E}$;

¹² *Fitness.*

¹³ *Clusters.*

4. 0 1 1: Frente-Esquerda-Plus $\vec{EP}$;
5. 1 0 0: Frente-Direita $\vec{D}$;
6. 1 1 0: Frente-Direita-Plus $\vec{DP}$;
7. 0 1 0: Frente-Indefinida $\vec{FI}$;
8. 1 1 1: Frente-Reduzida $\vec{F_R}$, velocidade v próxima a 0;
9. 1 1 1: Trás $\vec{T}$, velocidade v igual a 0;

Os movimentos Trás-Direita, Trás-Direita-Plus, Trás-Esquerda e Trás-Esquerda-Plus, possíveis de serem executados pelo robô móvel, não foram inseridos nos itens acima, mas podem atuar em substituição a comandos semelhantes. Não estamos enfatizando seu uso devido à utilização de uma única câmera frontal como sensor de entrada de dados do sistema, isto é, com campo de visão limitado a 180° graus. Também, devido não considerarmos atributos de dinâmica do robô, o posicionamento das setas na Fig. 4.2 podem não coincidir com movimentos reais executados pelo robô móvel.

Para o processo de geração de um classificador, as variáveis analisadas são: vetores $\vec{VM}$ e $\vec{VP}$; ângulo a e velocidade v . Notamos, que para a formação do vetor $\vec{PC}$, os valores dos atributos $\vec{VM}$ e a serão utilizados. Para encontrar o valor do atributo energia e , serão utilizados os valores de $\vec{PA}$ e v . Desta forma, um classificador possui os seguintes atributos:

- a. Parte-Antecedente: $\vec{PA} = \vec{VM}$;
- b. Parte-Conseqüente: $\vec{PC}$;
- c. Ângulo – Centróide da Área de Navegação: a ;
- d. Velocidade: v ;
- e. Energia: e ;

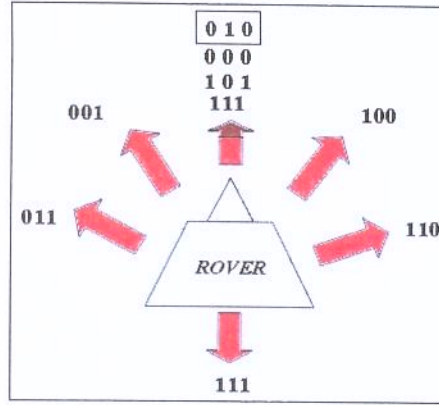


Fig. 4.2 – Possibilidades mapeadas como movimentos para execução.

4.1.1 Parte-Conseqüente $\vec{P}_{C_{vm}}$ a partir do Vetor Maximização $\vec{VM}$

Para gerar a parte-conseqüente a partir do vetor maximização, seus valores são normalizados na Eq. (4.1):

$$\vec{VM}_{i..n} = \frac{\vec{VM}_{i..n} * 1}{\vec{VM}_{max}} \quad (4.1)$$

onde i representa os índices do vetor $\vec{VM}$, n o tamanho do vetor $\vec{VM}$ e max o índice onde se encontra o maior valor contido no vetor $\vec{VM}$.

Sendo o vetor $\vec{VM}$ a variável que representa a informação horizontal de uma imagem, na Eq. (4.2) temos sua divisão em três fatias complementares¹⁴: *left*, *front* e *right*, objetivando encontrar uma trinca de valor com valores próximos aos movimentos mapeados. A Fig. 4.3 apresenta a divisão da imagem em três fatias.

¹⁴ Como o algoritmo *TH Finder* foi desenvolvido na língua inglesa, com o intuito de universalizar o conceito proposto, utilizaremos aqui *left* para a fatia esquerda, *front* para a fatia do meio e *right* para a fatia da direita.

$$\vec{PC}_{vm \ i=1...3} = \sum_{j=1}^n \vec{VM}_j \left\{ \begin{array}{l} 1: j \leq n/3 \\ 2: (j > n/3) e ((j \leq n/3) * 2) \\ 3: j > (n/3 * 2) \end{array} \right\} \quad (4.2)$$

onde i representa os índices do vetor $\vec{PC}_{vm}$, j representa os índices do vetor $\vec{VM}$ e n o tamanho do vetor $\vec{VM}$.

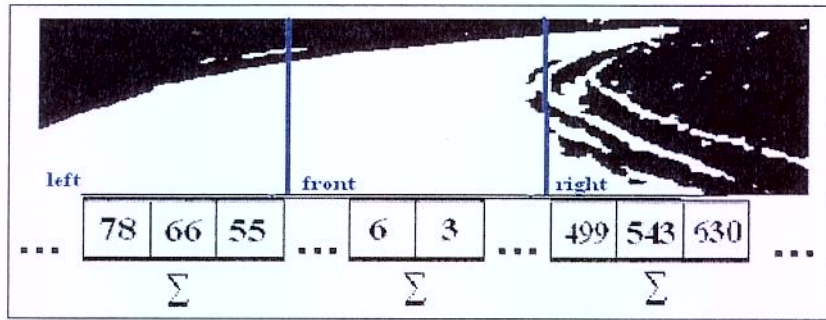


Fig. 4.3 – Divisão da imagem em três fatias. Valores do vetor $\vec{VM}$ não normalizados.

Na Eq. (4.3) os valores do vetor $\vec{PC}_{vm}$ são normalizados e armazenados como flutuantes.

$$\vec{PC}_{vm \ i..n} = \frac{\vec{PC}_{vm \ i..n} * 1}{\vec{PC}_{vm \ max}} \quad (4.3)$$

onde i representa os índices do vetor $\vec{PC}_{vm}$, n o tamanho do vetor $\vec{PC}_{vm}$ e max o índice onde se encontra o maior valor contido no vetor $\vec{PC}_{vm}$.

É importante notar, que quanto maior for o valor de cada posição do vetor $\vec{PC}_{vm}$ maior será a possibilidade de obstáculos naquela fatia da imagem. Já para o caso inverso, quanto menor for o

valor de cada posição do vetor $\vec{PC}_{vm}$ maior será a possibilidade de pontos de navegação naquela fatia da imagem.

Com os valores do vetor $\vec{PC}_{vm}$ variando no intervalo 0 e 1, para encontrar o movimento que melhor representa a parte-conseqüente do classificador, é realizado o produto entre todos os vetores de movimentos $[\vec{F}, \vec{FE}, \vec{E}, \vec{EP}, \vec{D}, \vec{DP}, \vec{FI}, \vec{FR}$ e $\vec{T}]$ e o vetor $\vec{PC}_{vm}$. O movimento selecionado como parte-conseqüente será aquele que possuir como resultado o valor mais próximo ou igual a 1. A Fig. 4.4 resume os passos deste processo.

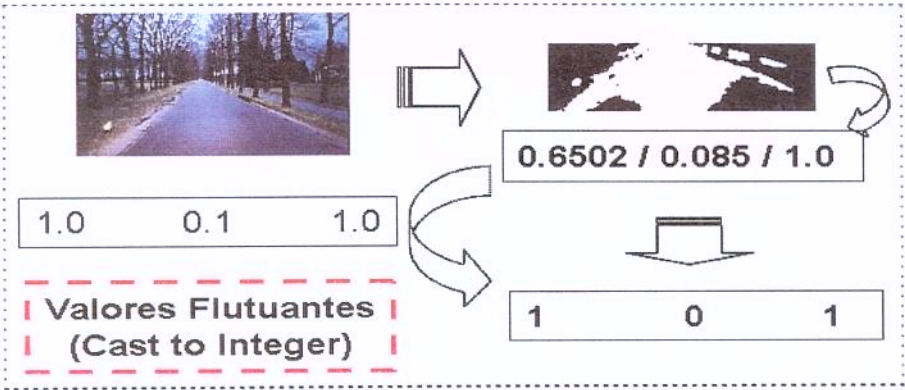


Fig. 4.4 – Valores do vetor $\vec{PC}_{vm}$ transformados para valores inteiros.

4.1.2 Parte-Conseqüente $\vec{PC}_a$ a partir do Ângulo de Correção de Direção a

Considerando os movimentos já mapeados, para formar a parte-conseqüente a partir do ângulo a de correção de direção, conforme apresenta a Fig. 4.5, temos:

- $\vec{PC}_a \leftarrow \vec{F} : \text{se } ((a > 0).e.(a \leq 5)).ou.((a < 0).e.(a \leq (-85)))$;
- $\vec{PC}_a \leftarrow \vec{D} : \text{se } (a > 5).e.(a \leq 45)$;

- $\vec{PC}_a \leftarrow \vec{DP} : \text{se } (a > 45);$
- $\vec{PC}_a \leftarrow \vec{E} : \text{se } (a > (-85)).e.(a \leq (-45));$
- $\vec{PC}_a \leftarrow \vec{EP} : \text{se } (a > (-45));$

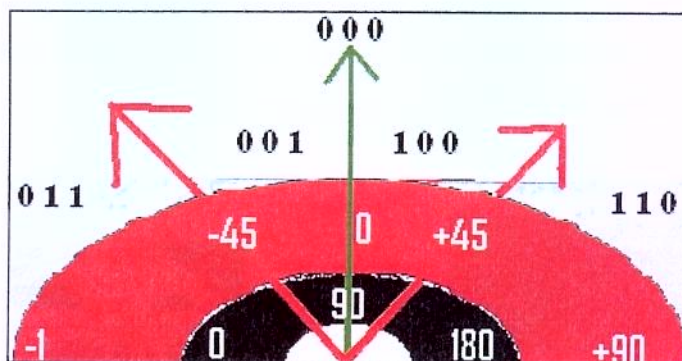


Fig. 4.5 – Ângulos e possibilidade de movimentos para um raio de 180° de visão.

4.1.3 Consolidação da Parte-Conseqüente $\vec{PC}$

Para decidir por um valor ideal a atribuir ao vetor $\vec{PC}$, os valores dos vetores $\vec{PC}_{vm}$ e $\vec{PC}_a$ são analisados. Como resultado deste processo, temos:

- $\vec{PC} \leftarrow \vec{PC}_a : \mathbf{se} (\vec{PC}_{vm} = \vec{FI})$;
- $\vec{PC} \leftarrow \vec{F} : \mathbf{se} (\vec{PC}_{vm} = \vec{F})$;
- $\vec{PC} \leftarrow \vec{PC}_{vm} : \mathbf{se} (\vec{PC}_{vm} = \vec{PC}_a)$;
- $\vec{PC} \leftarrow \vec{PC}_a : \mathbf{se} ((\vec{PC}_{vm}[0] = \vec{PC}_a[0]).\text{ou.}(\vec{PC}_{vm}[2] = \vec{PC}_a[2]))$;
- $\vec{PC} \leftarrow \vec{T} : \mathbf{se} (\vec{PC}_{vm} \neq \vec{PC}_a)$;

4.2 Sistema de Aprendizado Evolutivo - SAE

É certo que alguns fatores influenciaram para o comportamento determinístico do sistema, uma vez que este possui um único sensor para entrada de dados, a monovisão, que além de ser uma ferramenta de difícil aplicação para garantir uma navegação segura, dificulta a obtenção ou construção de mecanismos para retro-alimentação do SC. Outros fatores a considerar, são as restrições de controle do protótipo, que não possui atributos para controle da dinâmica do robô.

Porém, devido à quantidade de situações que podem ser apresentadas ao sistema de navegação, à dificuldade de mapeamento entre todas as situações possíveis de serem encontradas nas imagens captadas pelo sistema de visão, e a seleção de movimentos ideais para cada situação, criamos um sistema de aprendizado evolutivo.

Além disso, caso haja a inclusão de novos sensores junto à aplicação, para a geração de trajetórias para robôs móveis que possuem um conjunto de sensores e atuam em ambientes desconhecidos, nossa experiência aponta para a utilização de sistemas com aprendizado evolutivo (MENDELECK et al., 2005).

Semelhante ao modelo original de Holland (1975), a estrutura de nossos classificadores é constituída por uma parte-antecedente e uma parte-consequente.

Ao contrário do SC original, em nosso SC, tanto o valor “*don't care*” (#) quanto a variável de energia na sua forma original não são utilizadas, visto que, os atributos de nossos classificadores possuem codificação flutuante, fazendo com que o grau de semelhança excursionem em um intervalo mais amplo, reduzindo muito as chances de empate. Também, se levarmos em consideração somente a influência da iluminação sobre imagens adquiridas, dificilmente teremos um resultado determinístico.

Segundo CAZANGI (2004), sem a energia, os mecanismos de manipulação da energia, tais como, leilão, recompensa e taxaço, também ficam desativados. Outra justificativa para que a energia não seja utilizada é a falta de implementação do algoritmo de *bucket brigade*.

Excluindo-se do SC algoritmos do tipo *bucket brigade*, seu papel principal se resume à preparação e agrupamento de dados para a execução do AG (CAZANGI, 2004), tornando-se o módulo de descoberta de novos classificadores o principal integrante do SC.

Semelhante ao SC apresentado por Holland, em nosso modelo, para a descoberta de novos classificadores, o algoritmo genético é disparado cada vez que um número fixo de iterações

acontece, isto é, quando uma época de iterações é encerrada. No entanto, é importante destacar que, exceto para primeira iteração, quem determina toda ação a ser executada pelo robô sempre é o SC, isto é, a cada vez que um novo classificador é submetido ao sistema, a Fase de Competição - FC é disparada, e um classificador vencedor é selecionado. O novo classificador também será incluído na base de classificadores, porém este processo de inclusão só será realizado após a FC. Tal procedimento visa garantir que o novo classificador não seja selecionado no mesmo momento que foi inserido no SC.

Outra semelhança ao SC original é o número de descendentes produzidos pelo algoritmo evolutivo ser uma parcela fixa. No entanto, no modelo original a quantidade de novos classificadores, em geral, não é muito grande (CAZANGI, 2004).

Em nosso modelo, a quantidade de novos classificadores equivale a aproximadamente 50% do número de classificadores existentes no momento da execução do algoritmo evolutivo, isto é, o novo número de classificadores será de aproximadamente 50% menor em cada sub-população. Isso porque, na Fase de Seleção - FS, que antecede a Fase de Aprendizado e Reprodução - FAR, os classificadores são agrupados em 10 grupos menores por similaridade. Tal procedimento protege informações específicas de ambientes, por exemplo: locais específicos que não estão ocorrendo com frequência, permitindo que estas informações não sejam eliminadas pelo processo de procriação num curto prazo de tempo. Por fim, os novos classificadores substituirão os classificadores antigos.

O Sistema de Aprendizado Evolutivo é responsável por todo processo de decisão do sistema de navegação, compreendido nas ações imediatas e no processo de aprendizagem. É importante notar, que o sistema de navegação não utiliza informações prévias sobre a topologia do ambiente a ser conquistado, tomando suas decisões apenas com base nas imagens oriundas da camada Visão Computacional, e quando há, também com base no conhecimento adquirido.

Na Fig. 4.6 apresentamos a estrutura do SAE. Este sistema é composto pelas seguintes subcamadas:

- Fase de Competição - FC:
 - ocorre a cada iteração;
- Fase de Seleção - FS:

- classificadores são sublocados, por similaridade, em populações menores;
- seus dados passam pela FAR ao final de cada época;
- para efeitos de desempenho do algoritmo, a alocação em sub-populações acontece em conjunto com a FC;
- Fase de Aprendizado e Reprodução - FAR:
 - cruzamento;
 - mutação;

O principal objetivo da execução de todas as fases do SAE é a evolução da base de conhecimento durante o processo de navegação do robô móvel, além de atuar como um ponderador secundário de movimentos, evitando que a influência pontual do fator iluminação sobre imagens adquiridas provoque a execução de um movimento inadequado. Destacamos que esse resultado só tem efeito a partir das operações de evolução do AG.

Para responder ao estímulo do ambiente, isto é, encontrar uma resposta para uma imagem captada pelo sistema de aquisição, em um momento pré-Fase de Competição, uma verificação se fim de época é realizada, a fim de decidir o fluxo do sistema. Este fluxo pode ser somente via Fase de Competição ou via Fase de Aprendizado e Reprodução e Fase de Competição, considerando no segundo caso, que a Fase de Seleção já foi realizada. A Fig. 4.6 apresenta este processo.

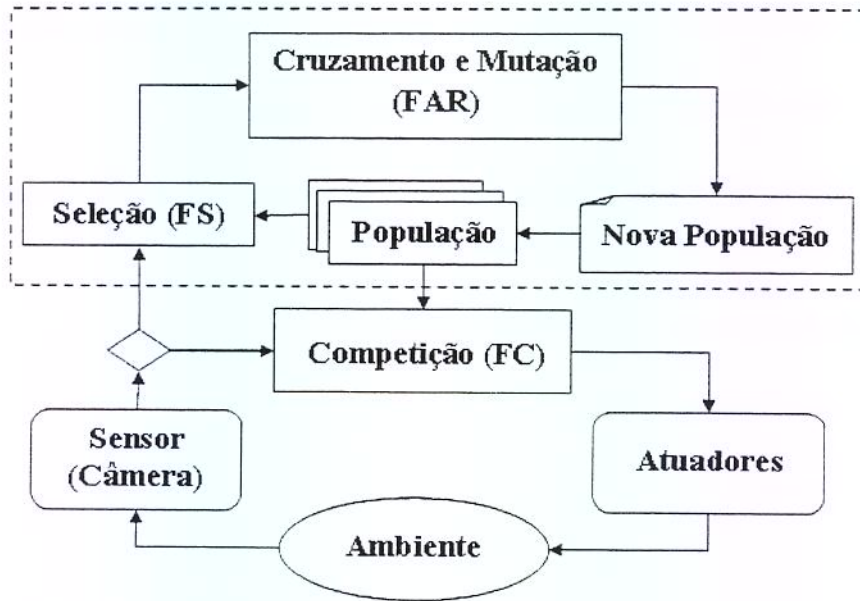


Fig. 4.6 – Diagrama do Sistema de Navegação. A linha tracejada indica os componentes da FAR.

4.2.1 Fase de Competição - FC

A principal tarefa da FC é determinar qual classificador dentre toda a população que melhor corresponde ao estímulo do ambiente, que por sua vez, no Sistema Gerador de Classificadores, foi transformado em um classificador. É nesta fase onde todos os classificadores competem para definir qual será o vencedor. Portanto, o cálculo da energia para todos os classificadores é necessário.

Após o cálculo da energia para todos os classificadores, o classificador da base de dados vencedor será aquele que possuir o valor da energia mais próximo ou igual a 1, tomando-se como base para o cálculo os atributos do novo classificador estímulo, isto é, nova imagem captada pelo sistema de aquisição, que será inserido na população após a Fase de Competição.

Para se definir por um classificador vencedor o atributo que interessa é a energia e , sendo que para atribuir um valor adequado para o novo classificador estímulo e todos da base de dados, utilizamos a Eq. (4.4). Destacamos que a energia de todos os classificadores da base de dados será recalculada a cada interação com o ambiente, isto é, a cada novo estímulo.

$$e = \left(\frac{\frac{\vec{PA}_{C_E} * \vec{PA}_{C_{BD}}}{\sqrt{(\vec{PA}_{C_E})^2} \sqrt{(\vec{PA}_{C_{BD}})^2}} + \frac{(v_{C_E} + v_{C_{BD}})}{20}}{2} \right)_i \quad (4.4)$$

onde $\vec{PA}_{C_E}$ e v_{C_E} representam, respectivamente, os atributos: vetor $\vec{PA}$ e velocidade v de um novo Classificador Estímulo, $\vec{PA}_{C_{BD}}$ e $v_{C_{BD}}$ representam, respectivamente, os atributos: vetor $\vec{PA}$ e velocidade v de um Classificador da Base de Dados. A constante 20 representa duas vezes o valor máximo da velocidade v e a constante 2 permite que o resultado final da equação fique no intervalo 0 e 1. i representa uma interação do sistema com o ambiente, e e representa o resultado da energia local do classificador da base de dados na interação i .

4.2.2 Fase de Seleção - FS

O papel desta fase é agrupar classificadores por similaridade em 10 sub-populações, utilizando para isso, os valores do atributo energia e de cada classificador. Vale lembrar que a Fase de Seleção agrupa classificadores a cada interação do Sistema Classificador, isto é, a cada vez que temos um novo estímulo. Em resumo, a cada FC uma nova organização em sub-populações é realizada devido aos novos valores do atributo energia calculados para todos os classificadores da base de dados, em virtude de uma nova imagem captada. A justificativa para o número de sub-populações tem vínculo com o intervalo em que o resultado da equação Eq. (4.4) pode variar.

A cada interação com o ambiente, quando um novo classificador estímulo C_E é apresentado ao SC, todos os classificadores disponíveis na base de conhecimento em execução passam por uma re-classificação. A Fig. 4.7 apresenta este processo. Este procedimento se assemelha ao uso de *clusters* em redes neurais, quando dados são agrupados por similaridades. O objetivo principal deste procedimento é a preservação de espécies durante a fase de evolução da população. Desta

forma, a evolução dos classificadores será intra-grupo, de modo semelhante à evolução natural da espécie animal.

As sub-populações ($\vec{S}_{i=0..9}$) serão abastecidas com classificadores (C) obedecendo-se o seguinte critério:

- $\vec{S}_{i=0} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0 do que 0.1});$
- $\vec{S}_{i=1} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.1 do que 0 e 0.2});$
- $\vec{S}_{i=2} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.2 do que 0.1 e 0.3});$
- $\vec{S}_{i=3} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.3 do que 0.2 e 0.4});$
- $\vec{S}_{i=4} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.4 do que 0.3 e 0.5});$
- $\vec{S}_{i=5} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.5 do que 0.4 e 0.6});$
- $\vec{S}_{i=6} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.6 do que 0.5 e 0.7});$
- $\vec{S}_{i=7} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.7 do que 0.6 e 0.8});$
- $\vec{S}_{i=8} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.8 do que 0.7 e 0.9});$
- $\vec{S}_{i=9} \leftarrow C : \text{se } (e_i \text{ estiver mais próximo de 0.9 do que 0.8});$

Vale ressaltar que, na Fase de Seleção as sub-populações geradas não substituem a base de conhecimento corrente, mas são formados a partir dela. O procedimento de substituição de classificadores acontece somente ao final da Fase de Aprendizado e Reprodução.

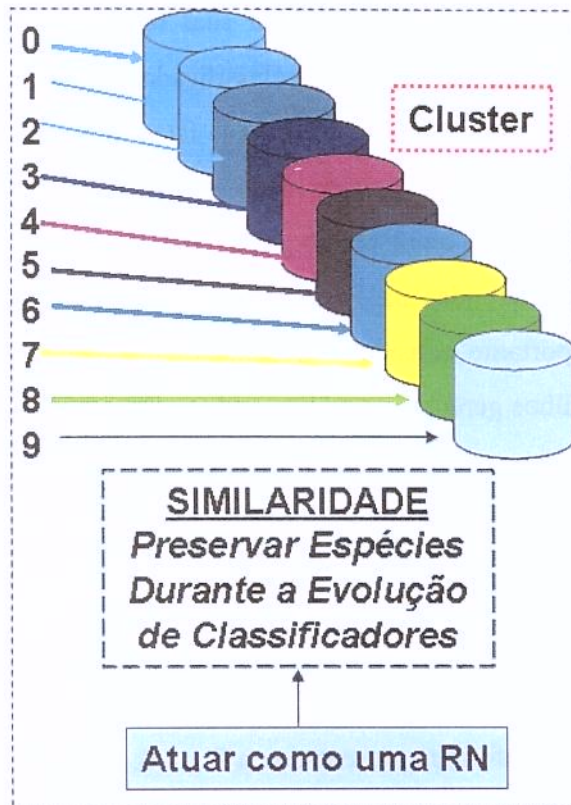


Fig. 4.7 – Sub-populações de classificadores.

4.2.3 Fase de Aprendizado e Reprodução - FAR

A FAR é responsável pelo processo de descoberta de novos classificadores, isto é, evolução da população, e atua sempre sobre sub-populações. No entanto, vale lembrar que esta fase só tem início quando uma época de iterações é encerrada.

Para entender sua execução é importante remeter-se ao processo de geração de classificadores, pois seu processo evolutivo transformará valores, porém não as estruturas.

Destacamos que a FAR não é um processo executado em paralelo a camada Estratégias de Navegação. Isso significa que a cada término de época, enquanto o processo de aprendizado é executado, as fases Fase de Competição e Fase de Seleção ficam congeladas.

Em virtude da arquitetura distribuída do sistema de navegação, e sendo a camada Visão Computacional serviços que correm em paralelo, independente do tempo computacional que o

processo de aprendizado possa levar para concluir suas operações, as operações de captura e leitura de imagens, e o processo de geração de classificadores, estarão sendo realizados e seus resultados disponíveis para tratamento posterior. Tal fato, não causa atrasos ou paralisações significativas ao sistema de navegação, pois de modo geral, o tempo consumido pela evolução de uma geração da população é menor que o tempo gasto em cada ciclo do sistema.

As operações genéticas incluídas na Fase de Aprendizado e Reprodução são: mutação e cruzamento. Assim, é importante considerar o conceito de taxa de procriação, que define a quantidade de indivíduos filhos gerados.

No processo de procriação ou cruzamento, os classificadores genitores de cada sub-população serão tomados dois a dois e formarão um descendente, sendo seus pais descartados do sistema, exceto para os casos em que a sub-população tenha apenas um indivíduo. Neste caso, o indivíduo único ou que ficou sem par não será descartado.

A operação de cruzamento é iniciada por um processo randômico, na qual um ponto de corte no vetor $\vec{PA}$ é selecionado, conforme apresentado na Fig 4.8. Com esta operação, o atributo $\vec{PA}$ do novo classificador local será formado com os dados do classificador pai1 até o ponto de corte, e a partir deste ponto de corte, por dados do classificador pai2. Já os atributos ângulo de correção α e velocidade v são formados pela média dos valores dos classificadores pais.

Cada novo classificador gerado na operação de cruzamento passa pela operação genética de mutação, onde dois valores do vetor $\vec{PA}$ são trocados de posição por um processo randômico, conforme apresentado na Fig 4.9.

Para ambos os casos apresentados acima, isto é, operações de cruzamento e mutação, considerando que um novo classificador foi gerado, e que durante as operações genéticas os valores dos atributos foram alterados, para efeito de padronização da geração de classificadores, todos os atributos devem passar novamente pelas regras de geração de classificadores.

Finalmente, os procedimentos evolutivos disparados ao fim de cada época são concluídos substituindo-se a população antiga. É importante dizer, que não há perda de qualidade ou divergência de conhecimento relevante pelo fato do reacoplamento da parte-antecedente ser feito de forma randômica, pois se considerarmos o fato que a operação de cruzamento é realizada dentro de um grupo de similaridade, logo os novos classificadores têm vários aspectos em

comum. E, eventualmente, os classificadores que divergirem do grupo serão naturalmente suprimidos ao longo do processo evolutivo (CAZANGI, 2004).

Como resultado da FAR espera-se que o robô móvel aprenda sobre o ambiente navegado, mantendo-se na área de navegação, não colidindo com obstáculos, principalmente quando não for a primeira vez em que o robô móvel execute tal percurso.

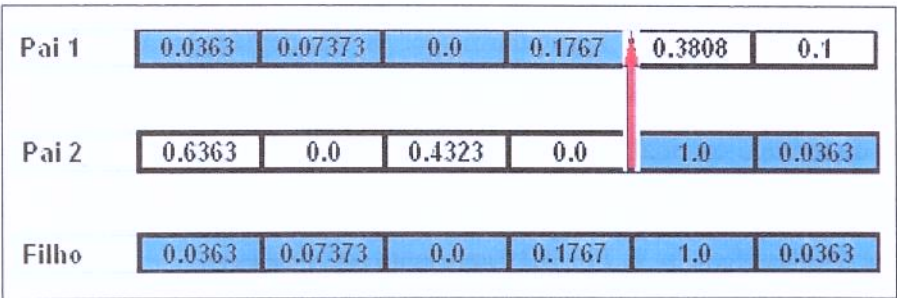


Fig. 4.8 – Novo Classificador Filho a partir do cruzamento de dois Classificadores Pais.

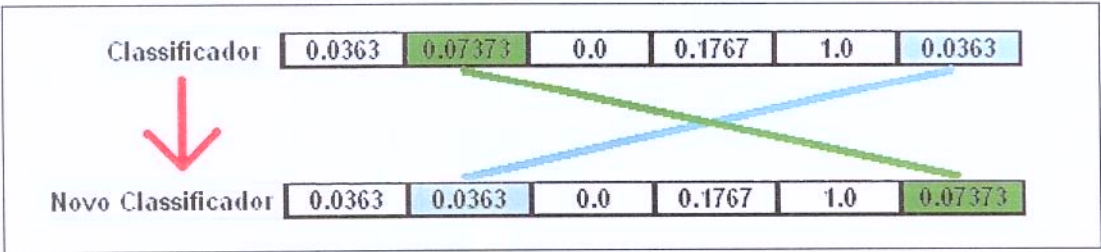


Fig. 4.9 – Novo Classificador gerado a partir da operação de mutação.

4.2.4 Migração

Comum para animais e insetos, a migração de indivíduos externos a uma população contribui à sua miscigenação. Dentro deste conceito, temos o Modelo das Ilhas¹⁵, apresentado em SILVA (2005) e BERKENBROCK & ROJAS (2006). Neste, o procedimento sequencial do AG é

¹⁵ *Island Model.*

executado em sub-populações que compartilham os melhores indivíduos de cada geração entre si, semelhante ao elitismo apresentado por CAZANGI (2004).

A comunicação entre as sub-populações é dada pela operação de migração, responsável pelas trocas entre indivíduos entre as ilhas. Neste modelo, devido ao custo de comunicação inter-processos, a taxa de migração não deve ser muito elevada, conforme avaliado por BERKENBROCK & ROJAS (2006).

Conforme apresenta o modelo original, em nosso sistema, apenas o melhor classificador migra para outra sub-população. Particularmente, as sub-populações de origem e destino são selecionadas de forma randômica.

Diferente do modelo original, nosso processo de migração não excluirá o classificador da sua sub-população de origem, porém a operação de migração será representada por uma operação de cópia ou clone do classificador. A Fig. 4.10 apresenta este processo.

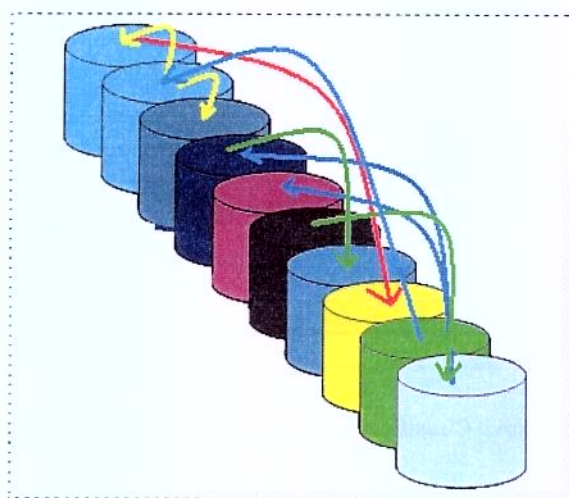


Fig. 4.10 – Operação de Migração entre sub-populações.

Ressaltamos que, para a operação de migração, o sorteio entre sub-populações é decidido por um processo randômico, e executado no número de sub-populações existentes. Em virtude do processo de sorteio entre sub-populações ser aleatório, não é possível garantir que todas as populações sejam requisitadas e/ou contempladas em todas as iterações do sistema.

As operações de migração apresentadas acima terão efeito somente a partir do processo de execução do AG, isto é, produzirá efeito para nos resultados do sistema a partir da operação cruzamento. Este classificador externo a população contribuirá para a miscigenação da mesma.

Capítulo 5

Experimentos Realizados

Neste capítulo apresentamos os experimentos realizados que validam a metodologia proposta.

5.1 Protótipo

A exemplo da concepção de processamento que foi realizada em multicamadas independentes, o protótipo representado na Fig. 5.1 foi concebido com três camadas:

- Servidor;
- Comunicação sem fio;
- Eletrônica Embarcada;

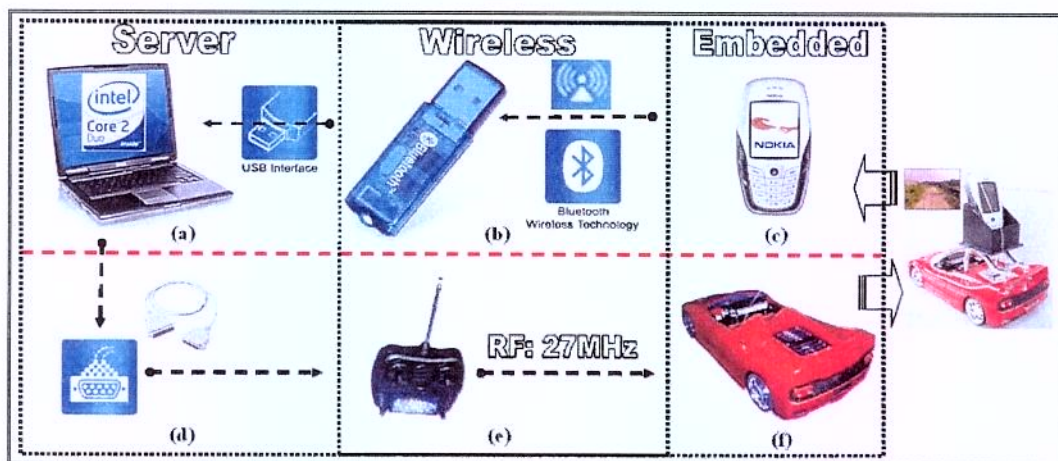


Fig. 5.1 – Componentes do Protótipo.

5.1.1 Servidor

Por servidor entende-se um microcomputador multiprocessado responsável pelo software de navegação, que é baseado no conceito de sistema distribuído, já que todas as camadas e subcamadas são serviços. Para um melhor desempenho da máquina servidora representada na Fig. 5.1 (a), desenvolveu-se a arquitetura lógica do projeto, de tal modo que a mesma esteja próxima à do sistema operacional.

5.1.2 Comunicação sem Fio

Para permitir a comunicação entre a máquina servidora e robô móvel, aproveitando as características do carro RC¹⁶ e do aparelho de celular, respectivamente representados nas Fig. 5.1 (c) e Fig. 5.1 (f), utilizamos duas faixas de frequência para tráfego de dados, pois adaptadores *wireless bluetooth* operam em uma faixa de frequência de 2.4 GHz, diferente do carro RC, que opera na frequência de 27 MHz.

5.1.3 Eletrônica Embarcada

Por possuir câmera de vídeo integrada, além do serviço de transmissão de dados via *bluetooth*, o aparelho de celular representado na Fig. 5.1 (c) foi selecionado para o projeto.

5.1.4 Projeto Físico

As três camadas constituem a lógica de funcionamento do protótipo físico, que é um carro RC com visão computacional embarcada e transmissão de dados sem fio.

Embora o sistema físico comporte-se como um sistema de segunda ordem, neste trabalho não foi levado em conta a dinâmica própria do sistema. Além dessa simplificação deve-se salientar que o controle do motor do carro RC foi mantido original, isto é, sem controle de aceleração.

¹⁶ *Remote Control.*

5.2 Camada Comandos e Transmissão

Na Fig. 5.2 reapresentamos a estrutura do Sistema de Navegação em camadas, já descrita pela Fig. 1.2, porém agora com ênfase à camada Comandos e Transmissão. Dentre os componentes apresentados na Fig. 5.1, a camada Comandos e Transmissão está representada no caminho lógico compreendido entre as Fig. 5.1 (a) e Fig. 5.1 (d).

A partir de uma ação de movimento originada na camada Estratégias de Navegação, representada pela parte-consequente de um classificador vencedor, complementada com informações de ângulo α de correção de direção e velocidade v , o objetivo principal desta camada é transmissão de comandos para execução do robô móvel, seguindo as relações apresentadas na Tabela 5.1. Basicamente, levando-se em consideração as limitações de controle apresentadas pelo protótipo, temos os seguintes comandos:

- 2 - Trás;
- 3 - Frente;
- 4 - Esquerda;
- 5 - Direita;

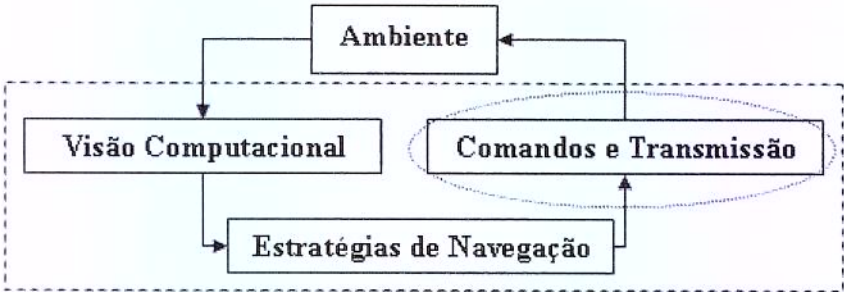


Fig. 5.2 - Sistema de Navegação ênfase Comandos e Transmissão.

Tabela 5.1 – Relação Movimentos.

Movimentos do Robô Móvel	Ação de Movimento	Movimento enviado para o Robô
FRENTE	000 .e. velocidade > 0 Passagem livre à frente	3

	101 .e. velocidade > 0 Passagem estreita à frente	3
	111 .e. velocidade > 0 Área de navegação com ruído excessivo	3
TRÁS	111 .e. velocidade = 0 Possível colisão	2
FRENTE/DIREITA	100 .e. velocidade > 0 Obstáculo à esquerda	5 e 3
	110 .e. velocidade > 0 Grande quantidade de obstáculos à esquerda	5 e 3
FRENTE/ESQUERDA	001 .e. velocidade > 0 Obstáculo à direita	4 e 3
	011 .e. velocidade > 0 Grande quantidade de obstáculos à direita	4 e 3
TRÁS DIREITA	011 .e. velocidade > 0 Ângulo α de Correção de Direção > 68°	4 e 2
TRÁS ESQUERDA	110 .e. velocidade > 0 Ângulo α de Correção de Direção > -22°	5 e 2

5.3 Experimento

Com o intuito de avaliar o desempenho do sistema de navegação aplicado ao controle de robôs autônomos, bem como validar suas potencialidades e detectar limitações, este item apresenta nossos resultados.

Nos experimentos em ambiente real ressaltamos que os deslocamentos do robô sempre foram por superfície plana.

É importante salientar que neste estágio de desenvolvimento inexistiu um mecanismo de confirmação de execução de comandos, similar ao controle de pacotes nas redes de

computadores. Isto porque, durante a tarefa de navegação, verificamos que comandos eram gerados e enviados ao robô móvel, porém o mesmo não os executava. Nestes casos, houve uma correlação entre as falhas no sistema de comunicação e a situação momentânea do robô, como por exemplo, escorregamento de rodas e curvas muito fechadas.

5.3.1 Simulações

Os casos apresentados a seguir representam os resultados de simulações realizadas em ambiente real. Certamente, a simulação virtual de sistemas de navegação autônomos traz facilidades, no entanto, é praticamente impossível reproduzir todas as características disponíveis nos ambientes reais, principalmente quanto tratamos de um sistema de visão computacional como sensor de entrada de dados, onde a influência da iluminação é um fator determinante.

Nas Fig. 5.3, Fig. 5.4, Fig. 5.5, Fig. 5.6 e Fig. 5.7, apresentamos cinco diferentes deslocamentos em ambiente real executados pelo robô móvel. Acrescentamos que a tarefa principal a ser cumprida foi o deslocamento desviando-se de obstáculos. Para isso, realizamos nossos experimentos em ambientes fechado e aberto.

No primeiro caso apresentado na Fig. 5.3 temos em (a) a visão global da tarefa a ser cumprida, já em (b) temos a representação do cumprimento da tarefa. Na Fig. 5.3 (c) temos a representação dos comandos enviados para o robô móvel mapeados por pontos brancos em uma imagem toda preta. Vale lembrar que os comandos enviados não necessariamente foram executados pelo robô móvel, visto que nosso protótipo não possui mecanismos para confirmação de execução. Como um fato importante, na Fig. 5.3 (c), além da influência da iluminação, temos os efeitos de inércia versus a taxa de captação de imagens do sistema de aquisição. Por último, na Fig. 5.3 (d) temos dois pares de imagens que representam cenas “observadas” pelo robô e resultado da segmentação do *TH Finder*.

Em contrapartida ao caso anterior, onde o experimento foi realizado em um corredor relativamente estreito, apresentamos na Fig. 5.4 um ambiente interno mais amplo. Em (a) temos a visão global da tarefa a ser cumprida, já em (b) temos a representação do cumprimento da tarefa. Na Fig. 5.4 (c) temos a representação dos comandos enviados para o robô móvel mapeados por pontos brancos em uma imagem toda preta. Nota-se neste caso, a inexistência da

característica de zigue-zague da situação anterior. Por último, na Fig. 5.4 (d) temos dois pares de imagens que representam cenas “observadas” pelo robô e resultado da segmentação do *TH Finder*.

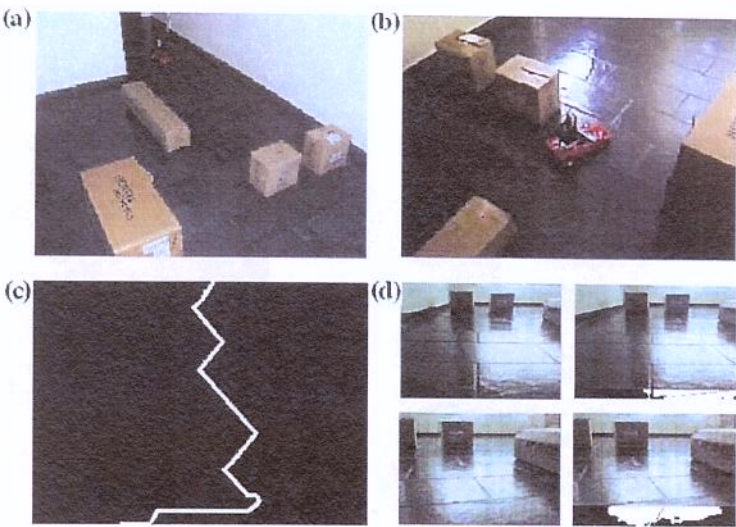


Fig. 5.3 - Resultado 1: Deslocamento em Ambiente Real: (a) Visão da tarefa, (b) Cumprimento da Tarefa, (c) Mapeamento Virtual do deslocamento do robô móvel, (d) Imagens vista do robô móvel e resultado da segmentação *TH Finder*.

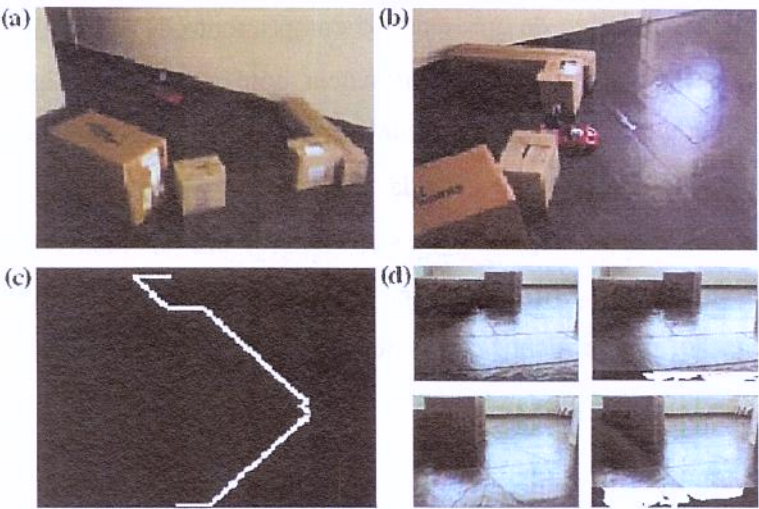


Fig. 5.4 - Resultado 2: Deslocamento em Ambiente Real: (a) Visão da tarefa, (b) Cumprimento da Tarefa, (c) Mapeamento Virtual deslocamento do robô móvel, (d) Imagens vista do robô móvel e resultado de segmentação *TH Finder*.

Com o objetivo de apresentar a série de comandos gerados para a travessia de um corredor com aproximadamente 8 metros, temos na Fig. 5.5 a representação dos comandos enviados para o robô móvel. Como era de se esperar, o protótipo mantém-se no centro do corredor com pequenos desvios em relação a linha média do mesmo.

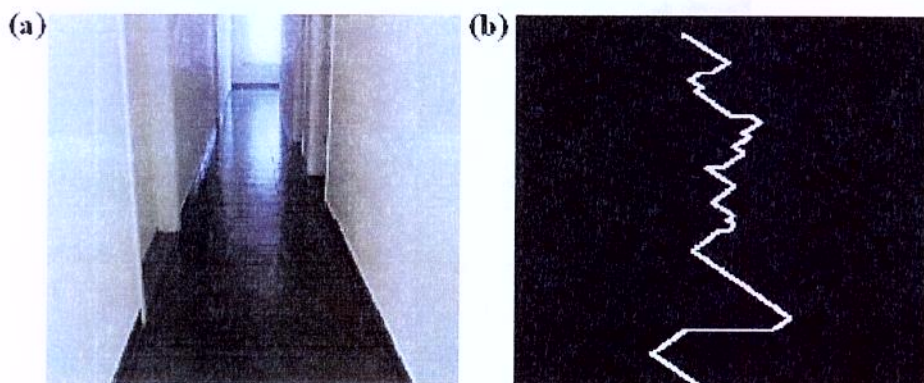


Fig. 5.5 - Resultado 3: (a) Deslocamento em Ambiente Real; (b) Mapeamento Virtual do deslocamento do robô móvel para travessia de um corredor com aproximadamente 8 metros.

Finalmente, com o objetivo de apresentar o cumprimento de um deslocamento em ambiente externo, onde a iluminação pode exercer influência, apresentamos na Fig. 5.6 duas situações da tarefa executada. A Fig. 5.6 (a) mostra o protótipo se aproximando do obstáculo, enquanto que, a Fig. 5.6 (b) representa o ponto de partida quando o protótipo desconhece a existência do obstáculo que virá. A Fig. 5.7 representa a sequência de movimentos desde o início até o final. Nota-se que a partir da Fig. 5.7 (d) o protótipo mantém-se em uma trajetória baseada no centróide da área de navegação, pois a identificação do único obstáculo existente é feita na parte (c) da figura citada.

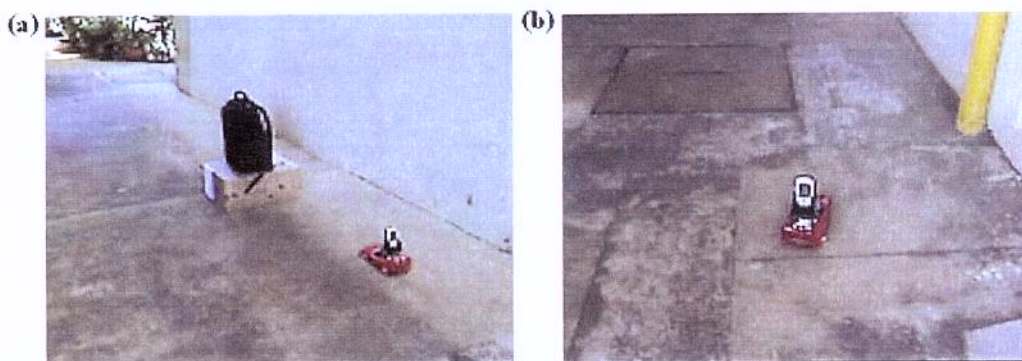


Fig. 5.6 – (a): Desvio de Obstáculo; (b): Ponto de partida.

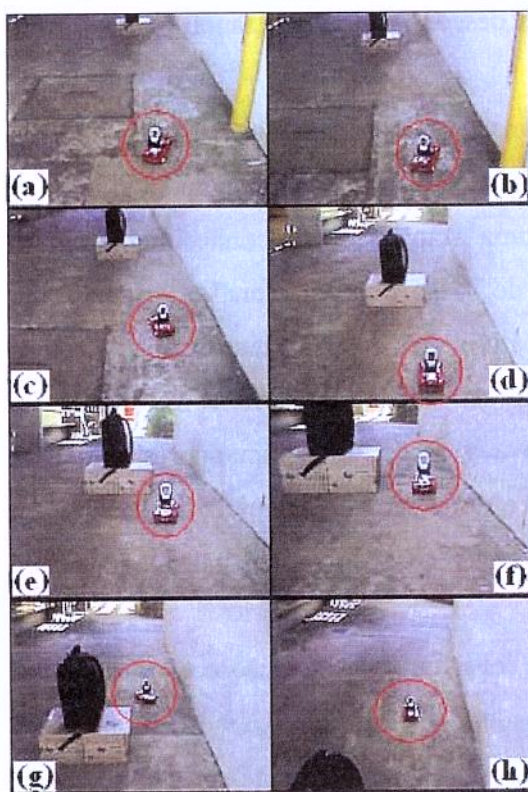


Fig. 5.7 – Sequência de movimentos: (a) Ponto de partida; (b), (c) e (d): Deslocamentos; (e): Deslocamento em direção ao Centróide da Área de Navegação; (f): Centróide da Área de Navegação; (g) e (h): Cumprimento da Tarefa.

Capítulo 6

Conclusão e Trabalhos Futuros

Este trabalho objetivou o desenvolvimento de um software para controle de robôs móveis no deslocamento em ambientes desconhecidos, utilizando para isso um único sensor de entradas de dados, a monovisão.

Considerando que sistemas robóticos interagem com o mundo real, que são inerentemente dinâmicos, demandando requisitos de resposta em tempo real, a pesquisa na área de navegação de robôs se torna muito oportuna à aplicação de conhecimentos multidisciplinares. Neste intuito, além do estudo de técnicas de Visão Computacional e Sistemas Classificadores com Algoritmos Genéticos, no decorrer das atividades deste projeto tivemos a oportunidade de incluir requisitos adicionais ao projeto, na qual seu estudo objetivou: compatibilidade com sistemas de tempo real; compatibilidade com a nova geração de computadores multiprocessados; baseado em uma arquitetura cliente-servidor e de sistema distribuído; multiplataforma e a partir de tecnologias livres; mobilidade durante a navegação; máxima independência do projeto físico mecânico; e baixo custo.

Não podemos deixar de levar em consideração o nível de autonomia de sistemas robóticos, principalmente dos que possuem seu controle baseado em análise de imagens, haja visto que a interpretação da informação visual ainda representa um grande desafio e, embora muito se tenha avançado nos últimos anos, os protótipos mesmo com estério-visão estão muito aquém da capacidade visual humana.

Devido a isso, neste trabalho introduzimos a aplicação de sistemas com aprendizado evolutivo, que além da capacidade de aprendizagem e adaptação, possibilitam que um sistema robótico possa inferir sobre um ambiente desconhecido, mesmo com ausência de conhecimento

inicial. Estas habilidades são originadas na teoria que as fundamenta, tendo sido propostas por HOLLAND (1975).

As principais contribuições de nosso trabalho estão associadas aos métodos aplicados à visão computacional, destacando o método de segmentação *TH Finder* e o processo de Descarte de Informações Redundantes.

Como perspectivas futuras, vinculados a camada Visão Computacional, vislumbramos a necessidade de aprimoramento do método segmentação *TH Finder*, mais especificamente, a substituição do desvio padrão amostral como ferramenta para encontrar o ponto de corte horizontal nas imagens, além da utilização das imagens *Up* para contribuições nas decisões futuras do sistema de navegação. Outro aprimoramento se refere à busca por mecanismos para obtenção de informações de profundidade em imagens monoculares.

Para a camada Estratégias de Navegação, visamos buscar mecanismos de retro-alimentação para sistemas baseados em análise de imagens nonoculares, além da implementação da base de dados permanente para armazenamento de classificadores. Acreditamos que a utilização deste tipo de recurso contribuirá para a execução de tarefas já realizadas e para a cooperação entre robôs.

Para o projeto físico, sentimos a necessidade de seu aperfeiçoamento, principalmente dos recursos de controle do motor, além da possibilidade de inclusão de mais sensores, por exemplo, de posicionamento global, que permitiriam realizar uma dada tarefa específica.

Por último, acreditamos que a busca por parcerias junto a empresas de tecnologias permitirá a integração de dispositivos móveis junto a aplicações robóticas, possibilitando o armazenamento e processamento de informações no hardware embarcado, além do uso de recursos atualmente disponíveis nas redes *wireless*.

Referências Bibliográficas

- ABUTALEB, H.A., (1989), "Automatic Thresholding of Gray-Level Pictures Using Two-Dimensional Entropy", Computer Vision, Graphics, and Image Processing.
- BALAN, A. G. R., TRAINA, A. J. M., TRAINA Jr., C., & MARQUES, P. M. A., (2004), "Integrando textura e forma para a recuperação de imagem por conteúdo". IX Congresso Brasileiro de Informática em Saúde - CBIS, pp.66.
- BERKENBROCK, G. R. & ROJAS, M. A. T., (2006), "Projeto de Caixa-S Utilizando Algoritmo Genético Paralelo", <http://inf.unisul.br/~ines/workcomp/cd/pdfs/2855.pdf>.
- BERTOZZI, M., BROGGI, A. & FASCIOLI, A., (2000), "Vision-based intelligent vehicles: state of the art and perspectives". Robotics and Autonomous systems 32, pp. 1-16.
- BLUETOOTH TECHNOLOGY OVERVIEW, Version 1.0, (2003), FORUM NOKIA, <http://www.nokia.com/>.
- BOOKER, L. B., GOLDBERG, D. E. & HOLLAND, J. H., (1989), "Classifier Systems and Genetic Algorithms" - Artificial Intelligence 40, pp. 235-282.
- CAZANGI, R. R., (2004), "Uma Proposta Evolutiva para Controle Inteligente em Navegação Autônoma de Robôs", Dissertação de Mestrado, FEEC, Universidade Estadual de Campinas (Unicamp).
- DAHLKAMP, H., KAEHLER, A., STAVENS, D., THRUN, S., & BRADSKI, G., (2006), "Self-supervised monocular road detection in desert terrain". In G. Sukhatme, S. Schaal, W. Burgard, and D. Fox, editors, Proceedings of the Robotics Science and Systems Conference, Philadelphia, PA.

- DARPA, (2004), "DARPA Grand Challenge Rulebook", <http://www.darpa.mil/grandchallenge05/>
- DEITEL, M. H. & DEITEL J. P., (2003), "Java Como Programar", São Paulo: Bookman.
- EUGENE, Y. K. & JOHNSTON, G. R., (2005), "The Ineffectiveness of the Correlation Coefficient for Image Comparisons", <http://lib-www.lanl.gov/la-pubs/00418797.pdf>.
- GÓMEZ, V.M., CAÑAS, J.M., SAN MARTÍN, F. & MATELLÁN, V., (2003), "Vision based Schemas for an Autonomous Robotic Soccer Player", Proceedings of IV Workshop de Agentes FísicosWAF- 2003, pp 109-120, Universidad de Alicante, Spain.
- GONZALEZ, C. R. & WOODS, E. R., (2000), "Processamento de Imagens Digitais", São Paulo, Editora Edgard Blucher Ltda.
- GUPTA, G.S., MESSOM, C.H. & DEMIDENKO, S., (2005), "Real-time identification and predictive control of fast mobile robots using global vision sensing", Instrumentation and Measurement, IEEE Transactions on, vol.54, pp. 200- 214.
- GUYTON, C. A., (1973), "Tratado de Fisiologia Médica", Rio de Janeiro, Guanabara Koogan.
- HEISS, J. J., (2005), "Embedded Systems Go Real Time With Java™ Technology", <http://java.sun.com/features/2001/04/embed.html?frontpage-headlinesfeatures>
- HOLLAND, J. H., (1975), "Adaptation in Natural and Artificial Systems : an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", The MIT Press, Ann Arbor, MI.91.

- HORN, P. K. B. & SCHUNCK, G. B., (1981), "Determining optical flow", *Artificial Intelligence*, vol. 17, pp. 185-203.
- KOZA, J.,R., BENNETT III, F.H., ANDRÉ, D., KEANE, M.A., (1999), "Genetic Programming III: Darwinian Invention and Problem Solving. Morgan", Kaufmann Publishers.
- LEE, S.U, CHUNG, S.Y & PARK, R.H., (1990), "A Comparative Performance Study of Several Global Thresholding Techniques for Segmentation", *Computer Vision, Graphics, and Image Processing*, (52), pp.171-190.
- LOPES, R. R., (2003), "Um Modelo Perceptivo de Limiarização de Imagens Digitais", *Dissertação de Mestrado, Universidade Federal do Paraná (UFPR)*.
- MENDELECK, A., ZAMPIERI, D. E. & MIRANDA NETO, A., (2005), "Sistemas Classificadores Aplicados à Navegação de Robôs Autônomos", *Congresso Brasileiro de Redes Neurais (CBRN 2005)*, Natal, RN, Brasil.
- MICHALEWICZ, Z., (1996), "Genetic Algorithms + Data Structures = Evolution Programs", 3rd edition, Springer.
- MICHALEWICZ, Z., (2000), "Evolutionary Algorithms for Constrained Optimization Problems Tutorial Program", *Genetic and Evolutionary Computation Conference - GECCO 2000*, Las Vegas, Nevada.
- MIRANDA NETO, A. & RITTNER, L., (2006), "A Simple and Efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision", *Proceedings of the 2006 IEEE 3rd Latin American Robotics Symposium (LARS 2006)*.
- MIRANDA NETO, A., RITTNER, L., LEITE, N, ZAMPIERI, D. E., LOTUFO, R., MENDELECK, A., (2007), "Pearson's Correlation Coefficient for Discarding

- Redundant Information in Real Time Autonomous Navigation Systems”, Proceedings of the 2007 IEEE Multi-conference on Systems and Control (2007 MSC), in Press.
- MOUSSI, L.N., GUDWIN, R. R., VON ZUBEN, F. J. & MADRID, M. K., (2001), “Sistemas Classificadores com Redes Neurais (NNCS): Aplicação ao controle de um veículo autônomo simulado computacionalmente”, V Simpósio Brasileiro de Automação Inteligente, Canela, RS, BRASIL - V SBAI, Vol. 1, pp.1-6.
- MOUSSI, L.N., VON ZUBEN, F. J., GUDWIN, R. R. & MADRID, M. K., (2002), “A Simulator using Classifier Systems with Neural Networks for Autonomous Robot Navigation”, Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'2002), in the 2002 IEEE World Congress on Computational Intelligence (WCCI'2002), Honolulu, Hawaii, vol. 1, pp. 501-506.
- MOUIR, P. F., (1988), “Modelling an control of wheeled mobile robots”, PhD thesis, DECE - Carnegie Mellon University, USA.
- MULCHANDANI, D., (1998), “Java for Embedded Systems”, Internet Computing, vol. 31, pp. 30–39.
- NAGAONKAR, Y., CALL, B., CLUFF, S., ARCHIBALD, J. & LEE, D.J., (2005), “Autonomous mobile robotic system with onboard vision using configurable logic,” Industrial Electronics Society, 2005. IECON 2005. 32nd Annual Conference of IEEE, pp.6.
- NOKIA, (2005), “Bluetooth Technology Overview, Version 1.0”, <http://www.nokia.com/>.
- NOVAKOVSKY, S. V., (1975), “Colour tele vision: A theory of colour reproduction”, MIR Publishers.

- OTSU, N., (1978), "A threshold selection method from gray-level histogram", IEEE Transactions on Systems, Man, and Cybernetics, pp.62-66.
- PRIMDAHL, K., KATZ, I., FEINSTEIN, O., DAHLKAMP, MOK, Y., STAVENS, H., D., MONTEMERLO, M. & THRUN, S., (2005), "Change Detection from Multiple Camera Images Extended to Non-Stationary Cameras", In Proceedings of Field and Service Robotics 2005 (FSR05), Port Douglas, Australia [Oral Presentation].
- RAMESH J., RANGACHAR K. & BRIAN G. S., (1995), "Machine Vision", McGraw-Hill, Inc.
- SAHOO, P.K., SOLTANI, S. & WONG, A.K.C., (1988), "A Survey of Thresholding Techniques", Computer Vision, Graphics, and Image Processing, (41), pp.233-260.
- SCHMITT, T., HANEK, R., BEETZ, M., BUCK, S. & RADIG, B., (2002), "Cooperative probabilistic state estimation for vision-based autonomous mobile robots," Robotics and Automation, IEEE Transactions on , vol.18, no.5.pp.670- 684.
- SCHWATZ, R. T., SILVA, G. V. M., KAICK, M. O., PEDRINI, H., (2003), "Reconhecimento em Tempo Real de Agentes Autônomos em Futebol de Robôs," VI Simpósio Brasileiro de Automação Inteligente, Bauru.
- SGOUROS N. M., (2002), "Qualitative Navigation for Autonomous Wheelchair Robots", 2002 Kluwer Academic Publishers - Autonomous Robots 12.pp.257-266.
- SHIROMA, P. M., (2004), "Controle por Visão de Veículos Robóticos", Dissertação de Mestrado, Instituto de Computação, Universidade Estadual de Campinas (Unicamp).
- SILVA, A. J. M., (2005), "Implementação de um Algoritmo Genético Utilizando o Modelo de Ilhas", Dissertação de Mestrado, Universidade Federal do Rio de Janeiro (UFRJ).

SIMÕES, A. S. & COSTA, A. H., (2000), "Using Neural Color Classification in Robotic Soccer Domain", International Joint Conference IBERAMIA'2000 and SBIA'2000, Workshop Proceedings, Meeting on Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning. L. N. Barros, R. M. César Jr., F. G. Cozman, A. H. Reali Costa (eds.), São Paulo, ISBN 85-901664-1-4, November, Atibaia, SP, Brazil, pp.208-213.

STANFORD RACING TEAM'S Entry In The 2005 DARPA Grand Challenge, (2006), <http://www.stanfordracing.org>

THRUN, S., MONTEMERLO, M., STAVENS, D., DAHLKAMP, H., et. al., (2005), "Stanford Racing Team's Entry in the 2005 DARPA Grand Challenge.", Technical Report: DARPA Grand Challenge.

TSAI, R., (1987), "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses", IEEE Journal of Robotics and Automation, vol. RA-3, n. 4, pp.323-344.

Apêndice A

A simple and efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision

A. Miranda, Neto, and L. Rittner

Abstract— Navigation of a Mobile Robot is based on its interaction with the environment, through information acquired by sensors. By incorporating several kinds of sensors in autonomous vehicles, it is possible to improve their autonomy and “intelligence”, specially regarding Mobile Robot Navigation in unknown environment. The type and number of sensors determines the data volume necessary for the processing and composition of the image from the environment. Nevertheless, the excess of information imposes a great computational cost in data processing. Currently many applications for control of autonomous vehicles are being developed, for example, the Grand Challenge, world-wide championship organized by DARPA (Defense Advanced Research Projects Agency). Machine Vision is, together with a set of sensors, one of the tools used in the championship. In our work, we propose a Mobile Robot Navigation Software based in monocular vision. Our system is organized in hierarchic and independent layers, with distributed processing. After reading an image, the system processes, segments and identifies the navigation area (road detection), that will be used as input by a motion generator system. The considered system is based on images acquired by a single camera and uses simple filtering and thresholding techniques, which make it fast and efficient for real time navigation.

Index Terms— Mobile Robots, Autonomous Vehicles, Navigation, Machine Vision, Image Segmentation.

I. INTRODUCTION

Lately, several applications for control of autonomous vehicles are being developed, and in most cases, the machine vision is an important part of the set of sensors used for the navigation. Perhaps this happens because we want to reproduce human vision in machines.

According to Gonzalez and Woods in [1], the acquisition of values sampled by a camera (vision) can be compared with image processing performed by human vision (role played by

the eye jointly with the brain). In a few words, the human vision allows the light of an object outside the eye to become image in the retina (more internal membrane of the eye). The perception of this image happens due to the excitement of light receivers, that transform the radiating energy into electric pulses that later are decoded by the brain.

In the other hand, it is known that the human vision without assistance from the brain is not capable to allow people displacement (navigation) in an efficient way. This is also truth for computer systems, which in order to navigate based on images, should contain software intelligent enough to manage mechanical structure through navigation.

Although extremely complex and highly demanding, thanks to the great deal of information it can deliver (it has been estimated that humans perceive visually about 90% of the environment information required for driving), machine vision is a powerful means for sensing the environment and has been widely employed to deal with a large number of tasks in the automotive field [2].

With the purpose to study machine vision techniques applied to the navigation of autonomous vehicles, the world-wide championship organized by the DARPA (Defense Advanced Research Projects Agency) [3], [4], known as Grand Challenge, was one of our motivations for the accomplishment of this work.

In general, the participant vehicles of this competition are equipped with all kinds of sensors to allow the interface between the navigation system and the environment to be explored. The vision system usually used in these vehicles is composed by two or more cameras, as we see in [5] and [6]. In our work, a single sensor (monocular vision) will be considered, or either, a camera will acquire images (samples) of the environment and these will feed the computer system.

What is trivial for the human system, that is, to construct three-dimensional scenes from two-dimensional images captured by the vision system and to use it on the decision process for navigation, is not necessary trivial for the computer systems.

Different from the human system, complex computational vision systems can lead to some damages due to the processing time. Thinking about the existing relation between

Manuscript received August 7, 2006. This work was supported in part by CNPq. Arthur de Miranda Neto is with UNICAMP, Campinas - SP, 13083-852 Brazil, phone: 55-19-3521-3823; (e-mail: arthur@fem.unicamp.br).

Leticia Rittner is with UNICAMP, Campinas - SP, 13083-852 Brazil (e-mail: lrittner@dca.fee.unicamp.br).

- [9] H.A. Abutaleb, "Automatic Thresholding of Gray-Level Pictures Using Two-Dimensional Entropy", Computer Vision, Graphics, and Image Processing, 1989.
- [10] P. K. Sahoo, S. Soltani, and A. K. C. Wong, "A survey of thresholding techniques," Comput. Vision Graphics Image Processing, vol. 41, pp. 233-260, 1988.
- [11] N. Otsu, "A threshold selection method from gray-level histogram". IEEE Transactions on Systems, Man, and Cybernetics, 1978.
- [12] U.S. Lee, Y.S. Chung and H.R. Park, "A Comparative Performance Study of Several Global Thresholding Techniques for Segmentation". Computer Vision, Graphics, and Image Processing, 1990.
- [13] JAI (2006, May 15), "Java Advanced Imaging", <http://java.sun.com/products/java-media/jai/>
- [14] IMAGEJ (2006, May 10), "Image Processing and Analysis in Java", <http://rsb.info.nih.gov/ij/>

Apêndice B

Pearson's Correlation Coefficient for Discarding Redundant Information in Real Time Autonomous Navigation Systems

A. Miranda Neto, *Member, IEEE*, L. Rittner, *Member, IEEE*, N. Leite, D. E. Zampieri, R. Lotufo and A. Mendeleck

Abstract—Lately, many applications for control of autonomous vehicles are being developed and one important aspect of them is the excess of information, frequently redundant, that imposes a great computational cost in data processing. Based on the fact that real-time navigation systems could have their performance compromised by the need of processing all this redundant information (all images acquired by a vision system), this work proposes an automatic image discarding method using the Pearson's Correlation Coefficient (PCC). The proposed algorithm uses the PCC as the criteria to decide if the current image is similar to the reference image and could be ignored or if it contains new information and should be considered in the next step of the process (identification of the navigation area by an image segmentation method). If the PCC indicates that there is a high correlation, the image is discarded without being segmented. Otherwise, the image is segmented and is set as the new reference frame for the subsequent frames. This technique was tested in video sequences and showed that more than 90% of the images can be discarded without loss of information, leading to a significant reduction of computational time necessary to identify the navigation area.

I. INTRODUCTION

Lately, several applications for control of autonomous vehicles are being developed, and in most cases, the machine vision is an important part of the set of sensors used for navigation. Although extremely complex and highly demanding, thanks to the great deal of information it can deliver, machine vision is a powerful means for sensing the

environment and has been widely employed to deal with large number of tasks in the automotive field [1].

But complex computer vision systems can lead to so damages due to the processing time. Thinking about the existing relation between a real time decision system and an image reading system that operates in a specific acquiring/reading rate, that is, amount of images read per second, one can question: how many images acquired must be discarded by the image processing system to guarantee an acceptable real time navigation of an autonomous vehicle?

Therefore, the decision for a more complex machine vision system possibly leads to an excessively slow system for an independent real time application. Although the system could maintain a database of acquired images and submit it, for example, to a neural network for decision, the great number of information not necessarily would lead to better decisions and could also harm the performance of the system, overloading it.

Taking in account that it has been estimated that humans perceive visually about 90% of the environment information required for driving [1], it is not a bad idea to reduce the information acquired by a navigation system, in order to reduce processing time. But the definition of an automatic image discharge criteria, that leads to a minimum loss of information, may not be a trivial task for computational systems, specially for real time ones.

In this work we propose an automatic image discarding method based on Pearson's Correlation Coefficient (PCC) as a low complexity and easy implemented solution.

With the purpose to study machine vision techniques applied to the navigation of autonomous vehicles, the worldwide championship organized by the DARPA (Defense Advanced Research Projects Agency) [2], [3], known as Grand Challenge, was one of our motivations for the accomplishment of this work. Nevertheless it is important to point out that the purpose of this work is to propose a solution for a specific task: identifying the navigation area.

In general, the participant vehicles of this competition are equipped with all kinds of sensors to allow the interface between the navigation system and the environment to be explored. The vision system usually used in these vehicles

Manuscript received January 10, 2007. This work was supported in part by CNPq.

Arthur de Miranda Neto is with the Mechanical Engineering Faculty, University of Campinas, SP, 13083-970, Brazil, phone: 55-19-3521-3386; (e-mail: arthur@fem.unicamp.br).

Leticia Rittner is with the Electrical Engineering Faculty, University of Campinas, SP, 13083-970 Brazil, (e-mail: lrirtner@dca.fee.unicamp.br).

Neucimar J. Leite is with the Institute of Computing, University of Campinas, SP, 13084-971 Brazil, (e-mail: neucimar@ic.unicamp.br).

Roberto Lotufo is with the Electrical Engineering Faculty, University of Campinas, SP, 13083-970 Brazil, (e-mail: lotufo@unicamp.br).

Douglas Eduardo Zampieri and André Mendeleck are with the Mechanical Engineering Faculty, University of Campinas, SP, 13083-852 Brazil, (e-mails: douglas@fem.unicamp.br, andre.mendeleck@uol.com.br)

- Vision, Graphics, and Image Processing, 1989.
- [10] A.Miranda Neto and L.Rittner, "A Simple and Efficient Road Detection Algorithm for Real Time Autonomous Navigation based on Monocular Vision", Proceedings of the 2006 IEEE 3rd Latin American Robotics Symposium (LARS 2006), in Press.
 - [11] N. Otsu, "A threshold selection method from gray-level histogram". IEEE Transactions on Systems, Man, and Cybernetics, 1978.
 - [12] Y.K. Eugene and R.G. Johnston, "The Ineffectiveness of the Correlation Coefficient for Image Comparisons", <http://lib-www.lanl.gov/la-pubs/00418797.pdf>.
 - [13] JAI (2006, May 15), "Java Advanced Imaging", <http://java.sun.com/products/java-media/jai/>
 - [14] IMAGEJ (2006, May 10), "Image Processing and Analysis in Java", <http://rsb.info.nih.gov/ij/>

Apêndice C

SISTEMAS CLASSIFICADORES APLICADOS À NAVEGAÇÃO DE ROBÔS AUTÔNOMOS

André Mendeleck^{1,2}, Douglas Eduardo Zampieri¹, Arthur Miranda Neto^{1,2},

¹UNICAMP - Universidade Estadual de Campinas

²FIPEP-Faculdades Integradas do Instituto Paulista de Ensino e Pesquisa

andre.mendeleck@uol.com.br

douglas@fem.unicamp.br

arthur@fem.unicamp.br

RESUMO: A geração de trajetórias para robôs móveis pode ser macro classificada em duas grandes categorias, tendo o ambiente (área de trabalho) como elemento principal: conhecida e determinística; ou desconhecida e dinâmica. Enquanto a navegação de robôs por ambiente conhecido e determinístico pode-se resumir à aplicação de algoritmos de busca com critérios de otimização, a navegação por ambientes desconhecidos e dinâmicos exige a utilização de algoritmos adaptativos. Neste artigo analisamos a navegação de robôs em ambiente desconhecidos utilizando-se aprendizado evolutivo com sistemas classificadores.

Palavras-Chave: Inteligência Artificial, Sistemas Classificadores, Algoritmos Genéticos, Wireless, Robótica.

I. INTRODUÇÃO

A geração de movimentos para robôs móveis em ambientes desestruturados é um dos tópicos mais estudados na área de robótica e automação. Se o ambiente for conhecido e estático, os problemas podem ser primeiramente reduzidos à modelagem do ambiente e à busca por caminhos seguindo algum critério de otimização (distância, energia, processamento, número de movimentos, qualidade dos deslocamentos, ...). Para ambientes desestruturados, o cenário para estudo é dinâmico, com vários elementos em movimentos. Assim, conduzir um robô de um ponto inicial até o seu destino, envolve a execução de operações complexas e não determinísticas, como por exemplo, a interação com o ambiente, identificação dos elementos ambientais e tomadas de decisões. Neste caso, a geração de trajetória exige o tratamento de uma série de fatores desconhecidos

pelo software de planejamento, tais como, o volume da área de trabalho e os elementos móveis e fixos. Sem estas informações, o custo computacional para a geração de trajetória torna-se bastante elevado, principalmente se considerarmos as formas convencionais de programação de robôs. No entanto, em algumas situações não é possível caracterizar, com antecedência o ambiente de trabalho. Uma situação mais complexa é a de um robô móvel que não conhece a trajetória a seguir, devendo desenvolver ("determinar"), em tempo real de execução, a trajetória a ser seguida, além de deslocar-se desviando de obstáculos.

Por outro lado, o ser humano é capaz de realizar a navegação por ambientes desestruturados com alto grau de adaptação, uma vez que sua estrutura neuronal propicia um certo grau de plasticidade e adaptabilidade nas ações, capacitando o indivíduo a identificar, analisar, propor soluções, atuar e a aprender com os problemas que lhe são apresentados. As aleatoriedades da navegação ativam mecanismos de compensação, que permitem a adaptação às novas necessidades. Por exemplo, um ser humano cujo sistema neural não recebe informações visuais pode suprir essa falta de informação com o auxílio de sinais de outros sensores (audição, tato, ...), adaptando o seu processamento neural. Situação semelhante ocorre com um robô móvel que perde os seus valores de referência de trajetória. O mecanismo executará os movimentos sem ter a certeza da sua validade. A orientação e o posicionamento absoluto do robô são facilmente resolvidos com um sistema de GPS interconectado ao sistema de navegação. Os sistemas robóticos também podem incorporar estas características, tornando-os flexíveis e adaptáveis à dinâmica do deslocamento.

Contudo, existem outros aspectos que necessitam ser considerados. A capacidade de deslocamento do robô, de reconhecimento do ambiente e de desvio de obstáculos fornece ao sistema de navegação um volume grande de

informações, permitindo gerar mapas físicos do ambiente. Estes sistemas são, geralmente, dedicados e dificilmente podem ser transferidos para outros sistemas.

Associado ao deslocamento livre de fios e problemas de processamento e armazenamento de informações, podemos utilizar um ambiente cliente/servidor com um sistema de comunicação wireless para atualização do status dos sensores do robô.

Neste artigo analisamos o problema da navegação e desvio de obstáculos para robôs móveis deslocando-se em ambientes não conhecidos. Abordaremos uma solução computacional que permita a auto-evolução e a auto-escolha de regras baseada nos princípios de computação evolutiva e sistemas classificadores [5][6][7][9].

Propomos a arquitetura do sistema em um modelo cliente/servidor, na qual concentraremos todo processamento pesado no servidor, aliviando o sistema embarcado, utilizando recursos de conectividade sem fio para propiciar melhor mobilidade.

II. SISTEMAS CLASSIFICADORES

Ações de um robô podem ser elaboradas e implementadas a partir de um mecanismo do tipo ação-reação-poderação. Os movimentos do robô são planejados a partir de macro informações (origem e destino, por exemplo) e por informações fornecidas por sensores. Pode-se estabelecer uma macro trajetória, ou uma direção preferencial de movimento, e durante a realização das ações, o sistema de navegação, interagindo com o ambiente, efetua as devidas correções de trajetória e "aprende" as características ambientais. Uma forte ferramenta computacional para abordar este tipo de problema são os sistemas computacionais baseados em redes neurais com supervisão. Para o desvio de obstáculos, estamos estudando a utilização de sistemas classificadores (SC). Estes sistemas permitem a definição, através do treinamento evolutivo e a geração de um conjunto de regras [11][12][13][14][15], a seleção e classificação de ações que envolve certo grau de aleatoriedade.

Para o sistema biológico, um dos principais elementos para o controle das atividades celulares é o código genético estruturado no DNA. Esta estrutura é formada por um conjunto de bases protéicas devidamente organizadas. Utilizaremos uma estrutura semelhante para gerar o conjunto de regras para o sistema classificador.

As regras são formadas por duas partes: identificador (antecedente) e ação (consequente) [16][20]:

- **IDENTIFICADOR** - é um vetor de tamanho fixo cujos elementos pertencem ao alfabeto

$\{0,1,\#\}$, onde " #" (don't care) significa que aquela posição poderá assumir tanto o valor 1 quanto o valor 0.

- **AÇÃO** - é um outro vetor de tamanho fixo, os elementos deste vetor pertencem ao alfabeto $\{0,1\}$.

A representação de uma regra é a seguinte: **001##00 : 01**, sendo a parte antecedente separada da parte consequente por " : ". Cada elemento (ou uma sequência) do identificador deve ser relacionado a algum elemento do robô, como por exemplo, às informações de sensores. A parte consequente deve ser associada a uma ação (movimento, mudança de status) do robô. A interpretação da regra é semelhante à utilizada em estruturas condicionais do tipo SE/ENTÃO/SENÃO. Se o código IDENTIFICADOR for ativado, então se executa a AÇÃO.

O elemento "0" indica ausência do sinal, o elemento "1" indica presença de sinal e o elemento "#" indica qualquer valor (0 ou 1).

O sistema classificador tem três subsistemas:

- Subsistema de Tratamento de Regras e Mensagens;
- Subsistema de Apropriação de Crédito;
- Subsistema de Descoberta de Novas Regras;

O subsistema de Tratamento de Regras e Mensagens recebe informações (mensagens) dos sensores e efetua a sua codificação, gerando uma parte antecedente específica para um determinado instante de amostragem. Essa codificação é comparada com a parte antecedente de todos os classificadores previamente cadastrados (gerados), os indivíduos compatíveis são escolhidos e são enviados ao Subsistema de Apropriação de Crédito. A comparação ocorre elemento a elemento e se todos os elementos de uma regra existem (ou são coincidentes - "matching" com os dados de entrada), então a regra é selecionada.

No subsistema de Apropriação de Crédito todos os classificadores compatíveis participarão de uma competição. Cada competidor (regra) deverá fazer uma aposta "**Bid**". O classificador que vencer a competição atuará sobre o ambiente.

Todos os classificadores escolhidos pagam uma taxa para participar da competição. O classificador vencedor enviará ao robô uma "ação" (parte consequente) que deverá ser executada. Dependendo dos resultados conseguidos pela execução desta ação, o sistema será realimentado e a regra vencedora será punida ou recompensada, alterando o seu nível de energia (strength).

A energia de uma regra indica o seu nível de aceitabilidade para atuação no meio ambiente.

Os classificadores (regras) são criados inicialmente de forma aleatória e para que possam ter um comportamento adequado, será necessário que o Sistema Classificador descubra quais classificadores geram comportamentos apropriados, criam-se novos classificadores e descartam-se aqueles que sejam ou tornem-se inapropriados, considerando a variabilidade do ambiente. Este mecanismo de aprendizagem de regras é implementado pelo algoritmo de *Atribuição de Créditos* e por um *Algoritmo Genético (AG)* [1][2][3].

O Sistema Classificador [10][11] interage com o ambiente em que atua através da troca de mensagens entre os dados dos sensores e a ação desenvolvida pelo robô. O ambiente manda mensagens sobre seu estado atual para o sistema, e o sistema envia para o ambiente a ação que deve ser executada. Por exemplo: um robô está navegando por um ambiente, encontra um obstáculo, manda esta mensagem para o Sistema Classificador, e este dirá ao robô que direção tomar para evitar o obstáculo. Dependendo do efeito que a ação escolhida teve sobre o ambiente (boa ou má), o conjunto de classificadores responsáveis por aquela ação será premiado ou punido. Desta forma os Sistemas Classificadores conseguem determinar novas regras através de regularidades verificadas no ambiente, e com isto refinar o seu modelo de ambiente.

Após um processo de atribuição de créditos seguido por etapas evolutivas, estes classificadores podem apresentar (incorporar) soluções para o problema proposto. Estes sistemas se guiam por metas implícitas, e promovem a evolução baseada em recompensas ou punições.

O subsistema de Descoberta de Novas Regras gera aleatoriamente novas regras. Adotamos a técnica clássica de algoritmos genéticos. O Algoritmo Genético selecionará os classificadores com os maiores strengths (*fitness*), gerando novos indivíduos por meio da recombinação e mutação destes. Os novos classificadores gerados tomam o lugar dos mais fracos, modificando o conjunto de classificadores do sistema [5].

Neste tipo de abordagem, o ajuste dos parâmetros, a geração de novas regras e sua seleção para a formação de uma lista de mensagens apropriadas para as tarefas em análise são os pontos críticos do processo, exigindo uma etapa de treinamento e simulação.

III. SIMULAÇÕES

Para as simulações do sistema classificador definimos um robô retangular (o robô que utilizaremos para os testes é retangular) capaz de deslocar-se para frente, para trás, girar sobre o próprio eixo e possui 7 sensores de

proximidade na parte frontal. O ambiente de navegação é desconhecido para o sistema classificador, contudo para facilitar a visualização dos movimentos criamos um “macro ambiente” utilizando-se primitivas gráficas como retas e polígonos, que pode ser observado somente pelo usuário. A atividade operacional básica é deslocar-se de um ponto inicial a um ponto final definidos pelo usuário, no simulador. Este procedimento é realizado clicando-se com o mouse sobre a área do “macro ambiente”, em tempo de execução. O sistema classificador deve conhecer o ponto objetivo da trajetória, contudo, ele não tem qualquer informação sobre o ambiente, a não ser as fornecidas em tempo de execução, pelos sensores do robô. A cada deslocamento do robô efetua-se uma amostragem dos valores dos sensores, formando-se uma mensagem que é enviada ao sistema classificador. Este retorna a ação a ser executada.

O classificador é representado por um vetor de 13 posições, sendo que as 11 primeiras posições representam a parte antecedente formada pelas informações dos sensores (7 sensores) e pela direção preferencial de movimento (4 bits). A parte consequente é formada pelos dois últimos bits restantes, correspondendo às seguintes decisões a serem tomadas: mover para frente, girar para esquerda, girar para a direita e girar 180 graus. O ângulo de giro é definido durante as simulações. Realizamos simulações com valores variando de 30° a 110°, sendo que os melhores resultados foram obtidos com giros de 80°, 90° e 110°.

Para as simulações definimos um ambiente com obstáculos fixos e móveis, desconhecidos para o sistema classificador. Utilizamos inicialmente 1000 regras geradas aleatoriamente, posicionamos o robô em um ponto de origem, definimos um ponto final de trajetória e estabelecemos uma direção preferencial de movimento.

A cada deslocamento do robô, os procedimentos para a formação de novas regras e execução dos movimentos devem seguir os seguintes procedimentos: formação da mensagem dos sensores, seleção de regras, identificação das que melhor adaptam-se ao movimento, geração do movimento, interação com o meio ambiente. Para cada ciclo, preservamos 40% das melhores regras, dentre as de maior nível de energia, e então se aplica “crossover” (taxa de 5%) e mutação (taxa de 2%).

Na figura 1 mostramos a área de trabalho, o robô, sua trajetória e, as áreas de amostragem dos sensores.