Desenvolvimento de um sistema de classificação
de cores em tempo real para aplicações robóticas

Éder Augusto Penharbel

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 3 de abril de 2008

Assinatura: _____

Desenvolvimento de um sistema de classificação de cores em tempo real para aplicações robóticas

Éder Augusto Penharbel

Orientador: *Prof^a. Dr^a. Roseli Aparecida Francelin Romero*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC/USP como parte dos requisitos para obtenção do título de Mestre em Ciência de Computação e Matemática Computacional.

USP - São Carlos
Abril/2008

Agradecimentos

A Deus, por conceder força, criatividade, coragem e por cada momento vivido durante a realização deste trabalho.

A minha Família, por todo apoio oferecido durante a realização deste trabalho.

A Prof.^a Dr.^a Roseli Aparecida Francelin Romero, por toda orientação, motivação, apoio, paciência, dedicação, por acreditar e investir no trabalho de seus alunos.

A Mariana, pela paciência, dedicação, coragem, carinho e pela ajuda na realização deste trabalho.

Aos meus amigos e amigas que me ajudaram neste trabalho: Marcelão, Denise, Thais, Quiles, Rodrigo, Pietro, André “Piu”, Ricardo “Homenzinho” e muitos outros.

Aos moradores e visitantes da “Rep”.

A todos meus amigos e amigas, cheios de qualidades, que compartilharam seus momentos comigo. Aos professores da FEI: Destro, Bianchi, Flávio e Hulot.

Aos diversos professores e funcionários do ICMC pelo excelente serviço prestado ao Instituto.

Às funcionárias da secretaria de pós graduação do ICMC por salvar os alunos que possuem problemas de esquecimento.

Ao pessoal do Aikido pelo ótimo humor e pelos treinos.

E, por fim, ao CNPq pelo apoio financeiro concedido.

Na visão computacional, a detecção de objetos é uma tarefa que tem significativa importância. Podemos verificar isto através da existência de inúmeros métodos propostos na literatura. Cada um destes métodos se apóia em algumas características presentes na imagem para alcançar um desempenho eficiente. Considerando ambientes que utilizam cores para determinação de objetos presentes em uma imagem, é possível utilizá-las como características que permitam detectar os objetos.

Neste trabalho, são investigados dois classificadores de cores. O primeiro é baseado em limiarização no espaço HSV e o segundo é constituído de um mapa auto-organizável para classificação dos pixels no espaço RGB. Objetivando a construção de um sistema classificador de cores eficiente, capaz de processar vídeo em tempo real, é proposta uma técnica que se baseia no conceito de quantização. Outro aspecto investigado foi a detecção de movimento para evitar o processamento de pontos indesejados.

O desempenho do sistema de classificação de cores é avaliado em um ambiente de futebol de robôs da categoria Mirosot, que é um ambiente dinâmico e que exige que todo o processamento da imagem seja rápido de modo a detectar corretamente todos objetos presentes em cada quadro.

Os resultados mostram que o classificador de cores é capaz de detectar todos objetos no ambiente de futebol de robôs, sendo cada quadro processado em menos de 30 milissegundos, tornando o sistema desenvolvido muito adequado ao processamento de vídeo.

Abstract

In computer vision, the detection of objects is a task of great importance. We can verify this by the existence of several methods proposed in the literature. Each one of these methods is based on some characteristics present in the image to reach an efficient performance. Considering environments that make use of colors for determining the objects present in a image, it is possible to utilize them as the characteristics that allow to detect the objects.

In this work, two color classifiers are investigated. The first one is based on the thresholding in the HSV space and the second is constituted by a self-organizing map for classifying of pixels in the RGB space. Aiming to construct an efficient color classifier able to process video in real time, it is proposed a technique that is based on the quantization concept. It is also investigated the detection of movement to avoid processing undesired points.

The performance of the color classifier system is validated in a MIROSOT robot soccer environment, which is a dynamic environment, requiring that all image processing be very fast in order to detect all the objects present in each frame.

The results show that the color classifier system is able to detect correctly all objects present in the robot soccer environment, processing each frame in less than 30 milliseconds, turning the developed system very appropriate for real time video processing.

Sumário

1	Introdução	1
1.1	Motivação	2
1.2	Trabalhos relacionados	2
1.3	Objetivo	4
1.4	Organização	4
2	Fundamentos teóricos	7
2.1	Imagem	7
2.1.1	Amostragem	8
2.1.2	Quantização	11
2.2	Espaços de cores	11
2.2.1	RGB	13
2.2.2	YUV	17
2.2.3	HSV	17
2.3	Classificação através do espaço de cores HSV	19
2.4	Quantização	22
2.4.1	Degradação da imagem	23
2.5	Deteção de movimento	23
2.5.1	Uma técnica adaptativa para detecção de movimento	25
2.6	Considerações finais	27
3	Redes Neurais e Mapas Auto Organizáveis	29
3.1	Introdução	29
3.2	Neurônio Artificial	30
3.3	Tipos de Aprendizagem	31
3.4	Mapa Auto Organizável	32
3.4.1	Algoritmo de treinamento	33
3.4.1.1	Competição	33
3.4.1.2	Cooperação	34
3.4.1.3	Adaptação	34
3.5	Exemplo de aplicação: agrupamento do Iris data-set	37
3.6	Considerações finais	39

4	Sistema de Visão Computacional Desenvolvido	41
4.1	Sistemas de visão computacional para futebol de robôs	41
4.1.1	Dortmond Droids	41
4.1.1.1	Processamento Inicial	42
4.1.1.2	Classificador de cores e segmentação	43
4.1.1.3	Correção da distorção da imagem	43
4.1.1.4	Rastreamento	43
4.1.1.5	Resultados da visão do Dortmund	45
4.1.2	Plymouth	45
4.1.2.1	Processamento Inicial	45
4.1.2.2	Classificação de cores	45
4.1.2.3	Busca de Círculos	46
4.1.2.4	Análise	47
4.1.3	Guaraná	47
4.1.3.1	Classificação de cores	47
4.1.3.2	Subtração da imagem de referência	48
4.1.3.3	Agrupamento e cálculo de centróides	49
4.1.3.4	Rastreamento	49
4.1.3.5	Orientação e posição dos objetos	50
4.1.3.6	Resultados	50
4.1.4	CMDragons - CMVision	50
4.1.4.1	Classificador de cores	50
4.1.4.2	Determinação das Regiões Conectadas	51
4.2	Sistema de visão computacional desenvolvido	51
4.3	Redução da Quantização	53
4.4	Classificador e tabela de otimização	54
4.5	Blob-coloring	55
4.6	Análise	58
4.7	Considerações Finais	58
5	O time de futebol de robôs do ICMC	61
5.1	A estrutura do time	61
5.2	Visão Computacional	63
5.3	Estratégia	63
5.4	Eletrônica	64
5.5	Mecânica	65
5.6	Controle	65
5.7	Considerações finais	66
6	Experimentos	67
6.1	Experimento utilizando o classificador HSV	67
6.2	Experimento: Redução utilizando quantização uniforme	68
6.3	Experimento: tempo de processamento para redução de quantização	70
6.3.1	Tempo de execução do monitor em aberto	71
6.3.2	Tempo de execução da redução usando aritmética de ponto flutuante	73
6.3.3	Tempo de execução da redução usando aritmética inteira	73
6.4	Experimento: Classificação dos pixels utilizando um Mapa Auto-organizável	76
6.5	Experimento: Redução da quantização através de Mapas Auto-organizáveis	78
6.6	Desempenho da classificação	81

6.7	Experimento: Detecção de movimento	82
6.8	Detecção de bola e robôs	84
6.9	Considerações finais	86
7	Conclusão	87
	Referências Bibliográficas	91

Lista de Figuras

2.1	Função contínua e suas versões amostradas com intervalos em x	9
2.2	Lena com resolução 256x256 reamostrada até 8x8	10
2.3	Redução da imagem de 256 para 2 níveis de cinza	11
2.4	Olho humano	12
2.5	Espectro de luz visível	12
2.6	Sensibilidade dos cones	13
2.7	Espaço de cores RGB	14
2.8	Diagrama de cromaticidade	14
2.9	Triângulo de cores - Gamut sRGB	15
2.10	Imagem RGB e suas componentes	16
2.11	Espaço de cores YUV	17
2.12	Imagem YUV e suas componentes	18
2.13	Espaço de cores HSV	19
2.14	Imagem HSV e suas componentes	20
2.15	Classificador HSV	21
2.16	Efeito da redução dos bits	23
2.17	“Fantasmas” na subtração das imagens (Yang et al., 2004)	24
3.1	Ligações entre os neurônios	33
3.2	inicialização e treinamento (0 a 1000 iterações) de um mapa auto-organizável	36
3.3	Agrupamentos do conjunto de dados <i>Iris data set</i>	38
4.1	Arquitetura do sistema de visão do Dortmund Droids	42
4.2	Classificação da relevância dos pixels	42
4.3	Pontos limitantes para aplicação das transformações bilineares	44
4.4	Etiquetas identificadoras do time Plymouth	45
4.5	(a) imagem sem filtro, (b) imagem com filtro.	45
4.6	Raio de busca do menor círculo ao redor do maior círculo.	47
4.7	Orientação de um robô.	47
4.8	Resultado da classificação e detecção das regiões conectadas	51
4.9	Fluxo do sistema	52
5.1	Robôs do time do ICMC	62
5.2	Estrutura geral	62
5.3	Sistema de visão	63
5.4	Eletrônica	64

5.5	Mecânica do time	66
6.1	Resultado da classificação no espaço HSV	68
6.2	Efeito da redução dos bits	69
6.3	Tempos de execução do monitor em aberto	72
6.4	Tempos de execução com aritmética de ponto flutuante	74
6.5	Tempos de execução com aritmética inteira	75
6.6	Comparativo entre a aritmética ponto flutuante e inteira	75
6.7	Resultado da classificação através de um mapa auto-organizável com 25 neurônios	77
6.8	Resultado da classificação através de um mapa auto-organizável com 25 neurônios	77
6.9	Efeito da redução da quantização utilizando mapa auto-organizável com diferentes números de neurônios	79
6.10	Efeito da redução da quantização utilizando quantização uniforme e mapa auto-organizável	80
6.11	Detecção de movimento em um trecho do filme Star Wars III	83
6.12	Detecção dos objetos de uma partida de futebol de robôs	85

Introdução

“Cientistas e escritores de ficção científica têm vislumbrado a possibilidade da criação de máquinas inteligentes, a capacidade de entender o mundo visual é um pré-requisito do que seria esperado de tais máquinas.” (Stockman e Shapiro, 2001).

A visão é um dos cinco sentidos presentes nos seres humanos e é utilizada em inúmeras atividades. Com ela é possível se movimentar evitando obstáculos, conduzir um veículo em segurança, escolher alimentos adequados, adquirir informações através de diversos tipos de mídias e outras tarefas do cotidiano.

O desenvolvimento de máquinas que permitiram uma automação flexível foi somente um começo e agora muitos esforços estão concentrados no desenvolvimento de máquinas que possam operar sem a intervenção humana (Pagello et al., 2002).

A percepção visual é, sem dúvida, de grande importância na construção de máquinas inteligentes capazes de operar de maneira autônoma, interagirem amigavelmente com seres humanos ou até mesmo simularem atividades humanas.

Aliado à modernização, ao aprimoramento e crescimento dos recursos computacionais, a criação destas máquinas pode estar mais perto, pois limitações como tempo de processamento, antes altamente proibitivas, já começam a ficar superáveis.

A visão é um tipo de percepção indispensável em nossa vida e, com o avanço da tecnologia atual, é possível estimular nossa imaginação na tentativa de criar novas possibilidades que permitam alcançar estas sonhadas máquinas. Hoje, já podemos ver máquinas realizando tarefas como: detecção de fogo em sequência de vídeo (Celik et al., 2007), rastreamento de faces (Gejguš e Šperka, 2003), Vigilância (Wang et al., 2005), detecção

de texto em sinais de trânsito (Wu et al., 2004), seleção de laranjas (Recce et al., 1998), análise de imagens de satélite (Stockman e Shapiro, 2001) e muitas outras.

1.1 Motivação

Uma grande parte da percepção visual humana é baseada no processamento da informação de cor e conseqüentemente os sistemas de visão não devem ignorar este importante aspecto (Etienne-Cummings et al., 2003).

Sistemas de visão para detecção de características de objetos podem ser classificados dentro de quatro categorias: 1) características extraídas de imagens em tons de cinza como bordas e formas, 2) características extraídas de cores, 3) características extraídas de texturas e 4) combinações das anteriores (Lee et al., 2007; Sonka et al., 2007).

Características de cores são geralmente empregadas como pré-processamento para acelerar processos mais complexos que têm como objetivo separar características e são principalmente úteis quando estas se diferem significativamente do restante da imagem.

Para aplicações que requerem alta velocidade de processamento, como por exemplo manuseio de alimentos ou inspeção visual, a variação de formas e outros empecilhos tornam métodos que utilizam cores como uma solução atrativa (Lee et al., 2007). Em (Browning e Veloso, 2005), é possível verificar que robôs operando em ambientes altamente dinâmicos necessitam reconhecer objetos coloridos rapidamente.

Objetos coloridos podem ser facilmente separados uns dos outros e até mesmo reconhecidos em aplicações que façam uso de restrições de cores específicas para determinação de objetos. Um exemplo de uma aplicação que usa de restrições de cores específicas para determinação de objetos é o futebol de robôs categoria MIROSOT. Nesta categoria, os objetos são determinados por cores diferentes, sendo a cor laranja reservada para bola e as cores amarela e azul reservada para determinação dos robôs em campo.

Observando que a detecção de objetos coloridos é uma aplicação útil, podemos explorar esta necessidade com a intenção de desenvolver técnicas para classificação de cores, fornecendo uma base para outras aplicações, como por exemplo de detecção de objetos.

A seguir são citados trabalhos que utilizam cores e estão relacionados com o presente trabalho.

1.2 Trabalhos relacionados

Existem diversos trabalhos presentes na literatura que utilizam cores para classificar cores ou detectar objetos. Thomas et al. descrevem um sistema de visão computacional para futebol de robôs capaz de classificar cores em tempo real. Neste sistema, é utilizado o

espaço de cores YUV e são aplicadas regras de classificação baseadas em limiarização. São utilizadas apenas as informações de crominância, ignorando a informação de iluminação. Os resultados deste trabalho demonstram a correta detecção da bola e também apresenta uma leve tolerância a variação de iluminação sutis.

(Borsato e Flores, 2004) implementam um sistema de visão para futebol de robôs baseado em classificação de cores, utilizando duas técnicas para eliminação dos pontos a serem classificados: a primeira é a subtração de fundo ou imagem de referência e a segunda é limiarização para remoção de pontos sombrios. A classificação de cores é realizada com base na construção de classes de cores obtidas através de um processo de calibração realizado com a ajuda de um usuário. Após definidas as classes de cores, é utilizada a distância Euclidiana no espaço de cores RGB para determinação da classe a qual o pixel pertence. Este sistema é capaz de classificar, no mínimo, trinta quadros por segundo, sendo capaz de funcionar em tempo real.

(Cao et al., 2007) desenvolvem um sistema de classificação de cores em tempo real para reconhecimento de robôs e bola em um ambiente de futebol de robôs RoboCup F-180. Neste sistema é utilizado o espaço de cores YUV para classificação de cores. É utilizado limiarização e redes neurais artificiais para classificar as cores dos robôs e bola. Este sistema assume a premissa que a bola e os robôs possuem formas circulares e coloridas. Realizada a classificação das cores é explorada a geometria do círculo para detectar os centros dos objetos.

Em (Simões, 2006), é apresentado um modelo de aprendizado não supervisionado para redes neurais pulsadas de base radial. Neste trabalho, redes neurais são utilizadas em um exemplo de caso de uso para classificação de cores. São realizados dois experimentos: o primeiro envolvendo uma imagem real selecionada de um banco de imagens de *benchmark* da Berkeley Computer Vision Group e o segundo envolvendo a imagem “lena”, uma imagem amplamente conhecida na literatura de processamento de imagens. Os resultados das redes neurais, segundo o autor, são satisfatórios.

Em (Browning e Veloso, 2005), é apresentada a implementação de um sistema de visão computacional para classificação de cores e detecção de objetos em ambientes abertos que apresentem ou não variação de iluminação. Para tais ambientes, é necessário que o sistema seja robusto o suficiente para que as possíveis variações não influenciem os resultados. Este sistema foi desenvolvido com uma técnica adaptativa para alcançar a robustez necessária e utiliza o espaço YUV para classificação de cores. São utilizadas técnicas de limiarização, adaptação dos limiares em tempo de execução e tabelas de lookup. O sistema desenvolvido é capaz de processar imagens de 320x240 e adaptar os limiares em aproximadamente 9 milissegundos, apresentando tempo de execução que o torna apropriado para funcionamento em tempo real.

Em (Quiles, 2004), são utilizadas redes neurais para classificação de cores para controle de um robô móvel. Cada pixel é classificado por uma rede neural perceptron de múltiplas

camadas. Para redução do tempo de treinamento da rede, é utilizado o algoritmo R-Prop, um algoritmo capaz de treinar a rede neural em tempo menor do que o algoritmo Back-propagation padrão. As imagens são processadas para detectar objetos coloridos com a finalidades de guiar o robô para estes objetos.

No presente trabalho, são utilizados dois classificadores, um baseado em limiarização no espaço HSV e o outro que utiliza um mapa auto-organizável para classificação de pixels no espaço RGB.

1.3 Objetivo

O principal objetivo deste trabalho é implementar um sistema para classificação de cores que funcione em tempo real, sendo tempo real entendido como processamento de 30 quadros por segundo ou, aproximadamente, 33 milissegundos de processamento por quadro.

Para isto, são estudados e implementados dois métodos: o primeiro, baseado na aplicação de limiarização no espaço HSV e o segundo, consiste na utilização de mapas auto-organizáveis treinados a partir de pixels RGB da imagem.

Visando acelerar o processamento, é explorada uma técnica de quantização uniforme da imagem de entrada para redução da variedade de pixels RGB.

Outro aspecto a ser explorado neste trabalho é detecção de movimento para evitar o processamento de pontos estáticos. Esses pontos podem ser interpretados como pontos de não interesse ou até mesmo ruídos em alguns tipos de aplicação. Desta maneira, a detecção de movimento se torna um pré-processamento interessante a ser estudado.

Pretende-se desenvolver um sistema de visão computacional para futebol de robôs da categoria MIROSOT para teste dos métodos estudados. Este sistema, que utiliza a classificação de cores para que os objetos sejam detectados, deverá ser capaz de fornecer as coordenadas dos objetos.

A próxima seção detalha a organização deste texto.

1.4 Organização

Este texto está organizado da seguinte forma:

- No capítulo 2, são discutidos os aspectos básicos de uma imagem, espaços de cores, quantização e detecção de movimento em vídeo.
- No capítulo 3, são apresentados o mapa auto-organizável e sua implementação para a classificação de cores através deste modelo de redes neurais.

-
- No capítulo 4, o sistema implementado é apresentado.
 - No capítulo 5, é apresentado o time de futebol de robôs do ICMC, construído durante o presente trabalho.
 - No capítulo 6, são apresentados os experimentos realizados neste trabalho e uma discussão dos resultados obtidos é realizada.
 - Por fim, no Capítulo 7, é apresentada a conclusão e possíveis trabalhos futuros.

Fundamentos teóricos

Neste capítulo são apresentados conceitos e técnicas relacionadas ao desenvolvimento do presente trabalho. Inicialmente, será estudado o conceito de imagens e suas propriedades. Em seguida, é introduzido o conceito de espaço de cores, possibilitando que imagens coloridas sejam exibidas ou armazenadas em dispositivos computacionais. Será introduzido o conceito de quantização e finalmente será introduzido o conceito de detecção de movimento em vídeo

2.1 Imagem

Uma imagem pode ser definida como um sinal (Sonka et al., 2007), onde um sinal é visto como uma função capaz de transmitir informações e é dependente de uma variável com sentido físico (Wolberg, 1994).

Para criar um modelo matemático de uma imagem, é possível defini-la como um sinal bi-dimensional representado por uma função dependente de duas variáveis x e y . Estas variáveis x e y , representam fisicamente cada coordenada (x, y) no domínio de $f(x, y)$. O valor de $f(x, y)$ representa o valor da intensidade luminosa naquela respectiva coordenada.

Segundo (Gonzalez e Woods, 2001), quando uma imagem é gerada a partir de um processo físico, cada um de seus pontos será proporcional a quantidade de energia irradiada pela fonte, fazendo com que os valores de $f(x, y)$ sejam não nulos e finitos, ou seja,

$$0 < f(x, y) < \infty.$$

O sinal bi-dimensional, ou função $f(x, y)$, pode ser caracterizada por dois componentes: *iluminância*, $i(x, y)$, a quantidade de luz incidente na cena vinda de uma fonte e *reflectância*, $r(x, y)$, a quantidade de luz refletida pelos objetos presentes na cena.

As produto das funções $i(x, y)$ e $r(x, y)$ formam a função $f(x, y)$, ou:

$$f(x, y) = i(x, y)r(x, y)$$

sendo

$$0 < i(x, y) < \infty$$

e

$$0 < r(x, y) < 1$$

Existem diversas maneiras para criação de imagens digitais, na qual podemos citar como exemplo a obtenção de imagens através de câmeras CCD, scanners, etc (Gonzalez e Woods, 2001),(Wolberg, 1994).

Uma imagem é então um sinal bi-dimensional que contém informações capazes de descrever uma cena do mundo real. Este sinal pode ser contínuo em relação ao sistema de coordenadas e a possíveis valores para cada ponto desse sistema de coordenadas, nos permitindo dizer que essa imagem é uma grandeza contínua. Entretanto, para o processamento ou armazenamento de uma imagem, devido a natureza discreta dos meios computacionais, é necessária a conversão da informação contida na imagem contínua para uma representação discreta, gerando uma imagem digital.

Essa conversão é feita através de um processo de amostragem, um processo que envolve as possíveis posições (x, y) como também os possíveis valores de amplitude para cada uma destas posições. O processo de amostragem envolve dois processos conhecidos como *amostragem* e *quantização*, que são descritos a seguir.

2.1.1 Amostragem

Amostragem é um processo no qual são definidos intervalos de tamanho fixo para que um sinal (ou função) seja analisado computacionalmente. Essa definição de intervalos permite que um sinal contínuo seja discretizado, fazendo com que a infinidade de pontos existentes entre dois pontos em um espaço contínuo desapareça.

A Figura 2.1 torna mais claro esse processo, onde podemos observar que os pontos deixam de existir entre um intervalo e outro.

(a) função contínua

(b) intervalo = 10

(c) intervalo = 30

(d) intervalo = 50

Figura 2.1: Função contínua e suas versões amostradas com intervalos em x

Ao falarmos de sensores, é comum o termo resolução, que é utilizado para descrever a menor mudança capaz de ser detectada, estando intimamente ligado com a precisão obtida. Em imagens digitais, a amostragem é o fator determinante da resolução espacial, que é o menor detalhe discernível em uma imagem (Gonzalez e Woods, 2001). Imagens com resoluções elevadas apresentam um alto nível de riqueza em detalhes e podemos observar o efeito da amostragem na Figura 2.2

Figura 2.2: Lena com resolução 256x256 reamostrada até 8x8

Uma imagem é, então, constituída de um número finito de elementos. Cada elemento de uma imagem, determinado por uma coordenada (x, y) recebe o nome de pixel ou pel, que são uma abreviatura para o termo “*picture element*”. Uma imagem com $M \times N$ pixels pode ser representada pela matriz dada pela equação (2.1).

$$f(x, y) = \begin{pmatrix} f(0, 0) & f(0, 1) & \dots & f(0, N - 1) \\ f(1, 0) & f(1, 1) & \dots & f(1, N - 1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M - 1, 0) & f(M - 1, 1) & \dots & f(M - 1, N - 1) \end{pmatrix} \quad (2.1)$$

2.1.2 Quantização

Quantização é o processo pelo qual a amplitude de um sinal é digitalizada (Gonzalez e Woods, 2001). O processo de quantização torna um sinal de amplitude infinita limitado a uma amplitude fixa e com determinados intervalos.

A quantização é facilmente exemplificada em imagens de cinza, onde a amplitude de um sinal está limitada a um valor de máximo e mínimo. Neste caso, o valor zero representa a cor preta e o valor máximo representa a cor branca. Uma imagem quantizada em 8 bits permite a existência de 2^8 ou 256 níveis de cinza. A redução dos níveis de cinza é menos perceptível do que a redução da resolução da imagens. Entretanto, essa redução introduz o efeito chamado falsos contornos (Gonzalez e Woods, 2001). A Figura 2.3 ilustra o efeito da redução dos níveis de cinza.

Figura 2.3: Redução da imagem de 256 para 2 níveis de cinza

Nesta seção, foi explicado que cada ponto (x, y) armazena a intensidade no ponto respectivo. Entretanto, para representarmos computacionalmente imagens coloridas, é necessário o estudo de modelos matemáticos que permitem descrever cores. Através destes modelos podemos exibir, gerar, armazenar, processar ou realizar qualquer outra operação sobre imagens coloridas. A próxima seção entra em detalhes sobre estes modelos.

2.2 Espaços de cores

No olho humano, uma imagem é formada pela luz focada sobre a retina pela córnea e cristalino (Sharma e Trussell, 1997). A retina é a parte responsável pela formação das imagens e funciona como uma tela de projeção. A Figura 2.4 ilustra um olho humano.

Na retina existem cerca de 100 milhões de células foto-receptoras. Estas células são denominadas cones e bastonetes. Os bastonetes são mais sensíveis a luz do que os cones, porém são os cones que possibilitam a percepção de cores.

Figura 2.4: Olho humano

Existem três tipos de cones, cada um possuindo pigmentos foto-sensíveis com diferentes capacidades de absorção espectral. Matematicamente, as repostas dadas pelos três tipos de cones podem ser modeladas como um vetor de três componentes dados por:

$$c_i = \int_{\lambda_{min}}^{\lambda_{max}} s_i(\lambda) f(\lambda) d\lambda \quad i = 1, 2, 3 \quad (2.2)$$

Nesta equação, $s_i(\lambda)$ representa a sensibilidade do i -ésimo tipo de cone e $f(\lambda)$ representa o comprimento de onda pela distribuição espectral de luz incidida na retina.

Tipicamente, o espectro de luz visível está entre $\lambda_{min} = 380$ nm e $\lambda_{max} = 830$ nm. A Figura 2.5 ilustra este intervalo e podemos perceber que a variação vai do violeta (menor comprimento) para o vermelho (maior comprimento).

Figura 2.5: Espectro de luz visível

A Figura 2.6 ilustra a sensibilidade dos cones, onde C, M e L representam, respectivamente, os comprimentos de onda curtos, médios e largos.

Figura 2.6: Sensibilidade dos cones

Desta maneira, podemos afirmar que o sistema visual humano é composto de três componentes que produzem a sensação de cores. Para produzir cores, os dispositivos de exibição de imagens usados em computadores utilizam modelos através da mistura das cores R, G e B. R, G e B são siglas vindas da língua inglesa para as palavras: *red* (vermelho), *green* (verde) e *blue* (azul).

Os modelos que governam as misturas de cores são chamados de espaço de cores e são modelos de volume tridimensional, onde a cor pode variar livremente dentro dos limites deste volume (Joblove e Greenberg, 1978). Existem diversos espaços de cores (Vandenbroucke et al., 2003) e podemos citar os mais comuns: RGB, YUV e HSV. As seções seguintes descrevem estes espaços de cores.

2.2.1 RGB

Como citado anteriormente, o olho humano é dotado de três tipos de cones (receptores) que respondem por diferentes comprimentos de onda. Devido a este fato biológico, um espaço de cores é um modelo de espaço tridimensional, onde podemos variar os valores de cada eixo conforme às respostas de cada tipo de receptor.

O espaço de cores RGB é um espaço de cores onde cada cor pode ser formada pela variação das quantidades de R (*red* ou vermelho), G (*green* ou verde) e B (*blue* ou azul) (Joblove e Greenberg, 1978). A Figura 2.7 ilustra um cubo onde podemos observar as cores dentro do espaço RGB.

A Figura 2.8, chamada de diagrama de cromaticidade, ilustra as possíveis cores visíveis. Neste diagrama, podemos observar algumas propriedades: (1) A curva limitante com a forma de um casco de cavalo é a faixa de todas as cores visíveis. (2) Se traçarmos um triângulo dentro do diagrama definimos todas as cores que podem ser reproduzidas. A Figura 2.9 demonstra, como exemplo, as cores que podem ser reproduzidas pelo espaço de cores s-RGB (um dos diversos RGB existentes). Estas faixas de cores que podem ser reproduzidas são denominadas *gamut*. (3) As cores nos pontos que residem nas linhas que

Figura 2.7: Espaço de cores RGB

formam os triângulos são as cores puras. (4) As cores fora do triângulo são as cores que não podem ser reproduzidas.

Figura 2.8: Diagrama de cromaticidade

Uma imagem descrita no espaço de cores RGB possui três componentes para descrever cada pixel. A Figura 2.10 ilustra uma imagem RGB e cada uma de suas componentes.

Outro espaço de cores bastante utilizado é o espaço de cores YUV, descrito na próxima seção.

Figura 2.9: Triângulo de cores - Gamut sRGB

(a) RGB

(b) Componente R

(c) Componente G

(d) Componente B

Figura 2.10: Imagem RGB e suas componentes

2.2.2 YUV

O espaço de cores YUV é parecido com o espaço YCbCr usado em televisores com sistema PAL. A componente Y é a componente de brilho, esta componente é a usada em televisores preto e brancos. As componentes U e V descrevem a tonalidade e a saturação da cor, sendo a componente U responsável pela tonalidade azul e a V pela tonalidade Vermelha (Reiter, 2004). A Figura 2.11 ilustra o cubo YUV.

Figura 2.11: Espaço de cores YUV

A transformação de RGB para YUV pode ser obtida através de uma transformação linear, conforme a equação (2.3) (Reiter, 2004).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ 0.439 & -0.368 & -0.071 \\ -0.148 & -0.291 & 0.439 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.3)$$

A Figura 2.12 ilustra uma imagem YUV e cada uma de suas componentes.

O próximo espaço de cores descrito é o HSV.

2.2.3 HSV

O espaço de cores HSV é um espaço que contem a tonalidade *hue* como uma de suas componentes. As outras componentes são saturação (*saturation*) e valor (*value*). O espaço HSV pode ser descrito como um cone, no qual H, a circunferência determina a tonalidade da cor, S, a distância do eixo central, é a saturação da cor e V, a altura dentro do cone, é correspondente ao brilho da cor (Reiter, 2004).

Figura 2.12: Imagem YUV e suas componentes

Se a componente saturação for igual a zero, então a tonalidade perde seu sentido e somente importa a componente V, descrevendo uma cor em escala de cinza.

A componente H pode variar de 0 a 359 graus, fixando todas as tonalidades possíveis dentro de uma circunferência. A Figura 2.13 ilustra o espaço de cores HSV.

Figura 2.13: Espaço de cores HSV

A Figura 2.12 ilustra uma imagem HSV e cada uma de suas componentes.

2.3 Classificação através do espaço de cores HSV

Um dos classificadores implementado neste trabalho utiliza o espaço de cores HSV. O espaço HSV foi escolhido por descrever cores de uma maneira simplificada e intuitiva. Esta classificação é feita através da aplicação de operação de limiarização, o qual pode ser entendido como um valor que estabelece os limites para classificação. Em (Gonzalez e Woods, 2001), limiarização é explicada como uma operação capaz de separar objetos brilhantes em uma imagem onde estes objetos são brilhantes e o fundo é sombrio.

Matematicamente falando, a limiarização pode ser modelada como na seguinte equação:

$$S(x, y) = \begin{cases} 1 & \text{se } I(x, y) \geq T \\ 0 & \text{caso contrário} \end{cases} \quad (2.4)$$

Assim, na equação (2.4), é possível observar que é atribuído o valor 1 para pixels que possuam valores maiores que T e 0 para os que possuam valores menores, ou seja, $S(x, y)$ somente conterá valores 0 ou 1. Como exemplo, podemos considerar que os objetos brilhantes em uma imagem $I(x, y)$ possuam valores maiores que T , se aplicarmos

Figura 2.14: Imagem HSV e suas componentes

a limiarização nesta imagem, a imagem resultante $S(x, y)$ conterá valores 1 nas posições (x, y) correspondentes a estes objetos.

Outras variações da equação também são possíveis, e a mais imediata é descrita na equação (2.5), onde a classificação é restrita entre um valor máximo e mínimo.

$$S(x, y) = \begin{cases} 1 & \text{se } I(x, y) \geq T_1 \text{ e } I(x, y) \leq T_2 \\ 0 & \text{caso contrário} \end{cases} \quad (2.5)$$

Para imagens HSV é possível determinar valores de limiares para cada canal H, S e V, permitindo que uma cor seja classificada de acordo com sua tonalidade (H), saturação (S) e valor (V).

No classificador desenvolvido neste trabalho foi utilizada a equação (2.6)

$$S(x, y) = \begin{cases} I(x, y)_h \geq hmin(X_n) \text{ e } I(x, y)_h \leq hmax(X_n) \text{ e} \\ X_n \text{ se } I(x, y)_s \geq smin(X_n) \text{ e } I(x, y)_s \leq smax(X_n) \text{ e} \\ I(x, y)_v \geq vmin(X_n) \text{ e } I(x, y)_v \leq vmax(X_n) \\ 0 \text{ caso contrário} \end{cases} \quad (2.6)$$

,

na qual é atribuída a classe X_n se os valores estiverem dentro dos limiares tonalidade mínima ($hmin(X_n)$), tonalidade máxima ($hmax(X_n)$), saturação mínima ($smin(X_n)$), saturação máxima ($smax(X_n)$), valor mínimo ($vmin(X_n)$), valor máximo ($vmax(X_n)$). Desta maneira é possível selecionar um conjunto de classes X_n para fazer a classificação dos pixels presentes na imagem $I(x, y)$. É interessante ressaltar que um pixel que não satisfaça nenhuma classe será rotulado como 0 e interpretado como sendo parte do fundo. A Figura 2.15 ilustra a tela de configuração do classificador desenvolvido.

Figura 2.15: Classificador HSV

Considerando que a tonalidade é representada por um círculo é interessante utilizar como limiar uma quantidade de graus de liberdade. Exemplificando, um limiar de $H = 0$ e graus de liberdade 10 classificaria cores no intervalo $-10^\circ(350^\circ), 10^\circ$.

A próxima seção entra em detalhes mais profundos sobre quantização, embora explicado anteriormente, este tópico é retomado por trabalhar com imagens coloridas.

2.4 Quantização

Como já dito anteriormente, quantização é o processo de atribuição de valores para determinadas faixas de valores de entrada (Heckbert, 1982). Desta maneira, toda imagem digital obtida por algum tipo de dispositivo já sofreu um processo de quantização.

Ao tratarmos a quantização de imagens coloridas, podemos dizer que o processo de quantização é realizado através de duas etapas: (1) seleção de um conjunto de cores para representar uma gama de cores e, (2) realizar o mapeamento entre as cores e o conjunto de cores representativas (Heckbert, 1982).

Segundo (Chung e Fung, 2003), é desnecessário e até inviável utilizar 24 bits para representar uma cor RGB. A redução da quantização possui primeiramente a vantagem da redução do tamanho do espaço necessário para armazenar uma imagem que pode implicar em fatores diretos como espaço em disco ou até mesmo tempo de transmissão de uma imagem.

Para visualizar o efeito da quantização, considere uma imagem que utiliza o espaço RGB e cada canal utiliza 8 bits, implicando com que cada pixel utilize 24 bits para definição de uma cor. Agora considere uma imagem com resolução de 1024 pontos verticais x 768 pontos horizontais, totalizando uma quantidade de 786432 pontos. Desta maneira, esta imagem utiliza 786432×24 bits, ou 18874368 bits, ou 2,25 MBytes. Porém, se utilizarmos uma quantização de 5 bits por canal, esta mesma imagem irá utilizar 1,40 MBytes, ou aproximadamente 60% da imagem original.

Segundo (Heckbert, 1982), o processo de quantização pode ser feito de duas formas: uniforme ou adaptativo. Existem diversos métodos de quantização de imagens (Heckbert, 1982) e (Chung e Fung, 2003), porém neste trabalho será somente utilizado o método uniforme, que consiste na simples divisão do espaço RGB em quantidades uniformes.

O processo de redução da quantização de uma imagem introduz uma degradação da imagem, conforme podemos ver na Figura 2.16 e é interessante estabelecer uma métrica para ver o quanto este processo deprecia a qualidade da imagem. A próxima seção introduz uma métrica para esta verificação.

Figura 2.16: Efeito da redução dos bits

2.4.1 Degradação da imagem

Como medida de degradação da qualidade entre a imagem original de entrada e a imagem quantizada de saída, é possível usar a métrica proposta em (Heckbert, 1982), que faz uso das Equações 2.7 e 2.8

$$D = \sum_{i,j} d(o_{i,j}, q(o_{i,j})) \quad (2.7)$$

$$d(x, y) = (x_r - y_r)^2 + (x_g - y_g)^2 + (x_b - y_b)^2 \quad (2.8)$$

onde o representa a imagem original, $q(o)$ a quantização da imagem original e $x = \{x_r, x_g, x_b\}$ e $y = \{y_r, y_g, y_b\}$ são os valores r, g, b do pixel na posição (i, j) .

No capítulo 6 é feito um experimento de redução de quantização com uma imagem real para verificar a degradação que esta operação introduz. Este experimento tem como objetivo demonstrar a possibilidade da utilização ou não da redução da quantização em aplicações reais.

A próxima seção discute a detecção de movimento em vídeo.

2.5 Deteção de movimento

A detecção de movimento é útil em diversas áreas da visão computacional (Wu e Huang, 2002). Ela permite que seja agregada a característica cinética as características espaciais e espectrais de um determinado objeto em movimento.

Pode ser utilizada para redução do processamento de vídeo, eliminando o processamento sobre pixels constantes, permitindo que o processamento seja aplicado sobre os pixels que mudaram seus valores.

Intuitivamente, esta técnica pode ser implementada através da subtração da imagem atual $I(t)$ de uma imagem de referência $I(t_0)$, ou:

$$S(t) = I(t) - I(t_0) \quad (2.9)$$

Na equação (2.9), a imagem $S(t)$ conterà em cada uma das suas posições (x, y) a diferença entre cada posição (x, y) da imagem $I(t)$ e sua respectiva (x, y) em $I(t_0)$. Entretanto, para criação de uma imagem binária, devemos aplicar uma operação de limiarização (Gonzalez e Woods, 2001). A equação (2.10) descreve a operação de limiarização.

$$S(t) = \begin{cases} 1 & \text{se } |I(t) - I(t_0)| \geq T \\ 0 & \text{caso contrário} \end{cases} \quad (2.10)$$

Fica claro que, se dentro do intervalo t e t_0 , não ocorrerem alterações, então as duas imagens serão equivalentes e a subtração sobre elas resultará em uma imagem $S(t)$ cujos pixels (x, y) possuem valor 0. Se ocorrer alguma alteração dentro deste intervalo, então $S(t)$ conterà o valor 1 nas respectivas posições onde ocorreram as alterações. Desta maneira, podemos fazer a suposição de que essas alterações correspondem a presença de movimento entre os instantes t e t_0 .

Este método, não levando em consideração a imperfeição do equipamento de aquisição de vídeo, se torna sensível ao ruído inserido por este. O ruído causa a detecção de movimento onde não ocorreu o movimento. Outras deficiências deste método são: (a) não modela as variações da iluminação, (b) posicionamento da câmera e (c) fantasmas, que são os objetos que estavam no fundo e começaram a se movimentar, criando dois movimentos na imagem $S(t)$. A Figura 2.17 exemplifica o que é um fantasma.

Figura 2.17: “Fantasmas” na subtração das imagens (Yang et al., 2004)

Desta maneira, para problemas que exigem uma solução mais robusta, se faz necessária a utilização de outro modelo. A próxima seção descreve uma técnica adaptativa proposta em (Yang et al., 2004).

2.5.1 Uma técnica adaptativa para detecção de movimento

A técnica proposta em (Yang et al., 2004) é uma técnica de detecção de movimento aliada a uma de subtração de fundo adaptativa. Esta técnica é baseada na diferenciação de duas imagens de uma mesma cena, porém obtidas com uma defasagem temporal δ . A diferenciação destas duas imagens resulta na imagem $F(t)$, utilizada para detectar o movimento.

A equação (2.11) descreve a criação de $F(t)$.

$$F(t) = \begin{cases} 1 & \text{se } |I(t) - I(t-\delta)| \geq T \\ 0 & \text{caso contrário} \end{cases} \quad (2.11)$$

Esta equação nos mostra a aplicação de uma operação de limiarização. Sendo assim, a imagem $I(t)$ é uma imagem binária, ou seja, todos os pixels só podem assumir dois valores: 0 e 1.

A presença de movimento (pixels com valor 1) é utilizada na atualização de uma matriz de movimento $D(t)$. Esta matriz de movimento $D(t)$ faz parte de um modelo que descreve uma espécie de efeito memória para os pixels que apresentaram alteração. Assim sendo, $D(t)$ é capaz de acusar a ocorrência de movimento no instante t e esquecê-lo com o decorrer da evolução de t .

A equação (2.12) é a descrição da manutenção da matriz $D(t)$.

$$D(t) = \begin{cases} D(t) - 1 & \text{se } F(t) = 0 \text{ e } D(t) \neq 0 \\ \lambda & \text{se } F(t) \neq 0 \end{cases} \quad (2.12)$$

Como podemos ver na equação (2.12), a cada evolução de t , a matriz é decrementada com a ausência de movimento (pixels com valor 0 em $F(t)$), representando assim o esquecimento. Entretanto, para cada pixel com valor 1 em $F(t)$, a matriz é atualizada com o valor λ , representando a ocorrência de movimento no instante t .

Além de detectarem movimento, estas equações são aplicadas em um modelo que descreve a adaptação do fundo, tornando esta técnica mais robusta que uma simples subtração de fundo.

A adaptação do fundo é feita de duas formas: (a) nível de quadro e (b) nível de pixel. A adaptação em nível de quadro é responsável pela absorção do quadro inteiro para o modelo do fundo, enquanto que a adaptação em nível de pixel é responsável pela absorção de um pixel para o modelo. Em uma maneira mais simples, podemos dizer, que quando ocorre muito movimento na cena, o fundo é atualizado mais rápido ou até mesmo substituído pela imagem $I(t)$ e também podemos afirmar que um pixel constante será atualizado com um caráter individual.

A equação (2.13) é responsável pelo modelo da adaptação do fundo (Wren et al., 1997).

$$B(t) = \alpha I(t) + (1 - \alpha)B(t - 1) \quad (2.13)$$

Na equação (2.13), o valor α representa o quanto a imagem atual $I(t)$ contribui para a adaptação do fundo. Para atualização em nível de quadro, essa equação pode usar um valor maior para α , acelerando a adaptação do fundo.

A detecção da necessidade da atualização a nível de quadro originalmente em (Yang et al., 2004) era feita utilizando a equação (2.14).

$$v = \frac{\sum_{j=1}^n \sum_{i=1}^m F(t)}{mn} \quad (2.14)$$

Porém, neste trabalho a equação (2.14) foi substituída pela equação (2.15), uma equação que realiza o mesmo trabalho, mas utiliza computação inteira ao invés de computação em ponto flutuante.

$$v = \sum_{j=1}^n \sum_{i=1}^m F(t) \quad (2.15)$$

A detecção da necessidade de atualização em nível de quadro é feita verificando se o valor v obtido na equação (2.15) está acima de um limiar. Uma outra modificação interessante da técnica original é a utilização dois limiares: (1) um com um valor muito elevado, o que representaria a mudança em praticamente todos os pixels. Desta forma, indica a necessidade da substituição do frame pela imagem $I(t)$, e (2) outro limiar moderado que somente detecta a necessidade da aceleração da adaptação, feita a partir de um α elevado em 2.13. A interpretação de muito elevado e moderado para o limiar depende da aplicação.

A detecção em nível de pixel ocorre quando o limiar aplicado em v não foi atingido. Desta maneira, é verificado para cada pixel, seu respectivo valor em $D(t)$, o qual indicará se deverá ou não ser atualizado. O indicador de atualização é a presença de um valor baixo, o que significa a não ocorrência de movimento e implica a necessidade de atualização deste pixel. A atualização deste pixel é feita pela aplicação da equação (2.13) sobre o mesmo.

Após a atualização do fundo, o método pode então realizar a detecção do movimento, feita através da subtração do fundo adaptado $B(t)$ e da imagem atual $I(t)$. Se utilizado $F(t)$, a diferenciação temporal entre $I(t)$ e $I(t_{t-\delta})$, iríamos descobrir a diferença instantânea, o movimento instantâneo, e não a diferença em relação ao fundo ou o movimento em relação ao fundo.

A equação (2.16) descreve a obtenção de $S(t)$, a imagem binária que corresponde aos pixels com movimento.

$$S(t) = \begin{cases} 1 & |I_r(t) - B_r(t)| \geq T_s \text{ ou} \\ & |I_g(t) - B_g(t)| \geq T_s \text{ ou} \\ & |I_b(t) - B_b(t)| \geq T_s \\ 0 & |I(t) - I(t - \delta)| < T_s \end{cases} \quad (2.16)$$

Uma outra mudança na implementação da técnica foi a adição da equação (2.17).

$$R(t) = F(t) \text{ AND } S(t) \quad (2.17)$$

Finalmente, para detectar os pixels com movimento, basta analisar a imagem $R(t)$ procurando por pixels com valores 1, que representam a presença de movimento e pixels com valores 0, que significam a ausência de movimento. A adição da 2.17 possibilitou a remoção instantânea de fantasmas, ou pelo menos mais rápida.

2.6 Considerações finais

Neste capítulo foi estabelecido o conceito de imagem, cores, limiarização no espaço HSV para classificação de cores, quantização com o intuito de formar uma base para o entendimento do trabalho realizado. No capítulo 6 serão realizados experimentos com redução da quantização e classificação de cores através de limiarização no espaço HSV.

Também foi investigado a detecção de movimento em vídeo, apresentando uma técnica para detectar os pontos onde ocorrem movimento. Esta técnica é interessante por ser adaptativa, ou seja, capaz de adaptar-se à mudanças de iluminação e até mesmo à mudanças no posicionamento da câmera. A técnica de detecção de movimento através da simples subtração de uma imagem de referência, ou fundo, deixam de funcionar adequadamente quando ocorre variação da iluminação. Por exemplo, em um ambiente com incidência de luz natural, na qual a quantidade de luz incidente é diferente durante o decorrer do dia. A utilização da detecção de movimento é interessante para evitar o processamento de pontos estáticos que já foram processados em algum instante t no passado, fazendo com que o processamento seja reduzido.

A técnica foi por nós implementada, modificada e gerou um artigo que foi publicado em (Penharbel et al., 2006). Nesta implementação, propusemos uma pequena modificação através da inserção de uma equação que possibilitou a remoção de fantasmas de forma mais eficiente.

No próximo capítulo, será discutido um modelo de rede neural chamado mapa auto-organizável, SOM, que foi utilizado para classificação de cores no espaço RGB.

Redes Neurais e Mapas Auto Organizáveis

Este capítulo introduz o conceito de redes neurais artificiais, neurônio, tipos de aprendizagem e mapas auto-organizáveis. É explicado o algoritmo de treinamento de um mapa auto-organizável e, em seguida, é feito um exemplo de aplicação utilizando o conjunto de dados *Iris data-set* (Fisher, 1936).

3.1 Introdução

Redes Neurais Artificiais são modelos computacionais inspirados no funcionamento biológico das Redes Neurais presentes em cérebros de animais (Haykin, 1989). Os fundamentos de Redes Neurais datam de trabalhos realizados no final do século 19 e começo do século 20 por cientistas como Hermann von Helmholtz e Ivan Pavlov. Entretanto a visão moderna de Redes Neurais só foi concebida no anos iniciais da década de 40 com o trabalho de Warren McCulloch e Walter Pitts (Hagan et al., 1996) que mostrava que uma rede neural seria capaz de realizar cálculos com funções lógicas e aritméticas. Posteriormente, Donald Hebb propôs uma lei segundo a qual o treinamento de neurônios artificiais poderia ser realizado.

A primeira aplicação prática utilizando Redes Neurais ocorre no final da década de 50 com a invenção da rede neural Perceptron. Essa aplicação ocorre alguns anos antes da publicação do livro *Perceptrons: An Introduction to Computational Geometry* (Minsky e Papert, 1969) que demonstrou a incapacidade desse modelo de rede para alguns problemas, causando a suspensão do estudo de Redes Neurais até os anos 80, quando aparecem: computadores mais poderosos; o computador pessoal; novas idéias e o algoritmo

backpropagation, um algoritmo de treinamento para redes perceptron de multi camadas, resolvendo algumas das questões que suspenderam o estudo das Redes Neurais e trazendo novo fôlego para essa área de pesquisa (Hagan et al., 1996)

Uma rede neural artificial é constituída de unidades de processamento chamadas neurônios artificiais, da mesma maneira que uma rede neural é constituída de neurônios. O termo artificial é utilizado para enfatizar que é utilizado um modelo e não células naturais na implementação de uma rede neural artificial. Como este é o elemento base de uma rede neural, a próxima seção o descreve mais detalhadamente.

3.2 Neurônio Artificial

Um neurônio artificial é constituído de uma ou mais entradas aliadas a uma grandeza escalar chamada peso sináptico (ou simplesmente peso), um somador, uma função de transferência e uma saída. Biologicamente, a entrada é equivalente a uma parte de um neurônio natural chamada dendrito; o peso é equivalente a força sináptica (sinapse é o ponto de contato da entrada com a saída do neurônio, ou biologicamente, o ponto de contato do dendrito com o axônio); o somador equivale ao corpo do neurônio e a função de transferência aliada à saída equivale ao sinal transmitido pelo axônio (Hagan et al., 1996). Em um neurônio artificial, também é possível encontrar diversas entradas (neurônio de múltiplas entradas) e a presença de um parâmetro adicional chamado *bias*, por vezes referido como deslocamento *offset*.

Considerando a entrada e os *bias* como grandezas vetoriais, e utilizando os pesos como uma matriz é possível dizer que um neurônio pode ser modelado segundo as equações (3.1) (Adaline) e (3.2) (Perceptron).

$$\mathbf{s} = \mathbf{W}\mathbf{e} + \mathbf{b} \quad (3.1)$$

$$\mathbf{s} = f(\mathbf{W}\mathbf{e}) + \mathbf{b} \quad (3.2)$$

Nessas equações $\mathbf{s}$ é a saída do neurônio, $\mathbf{e}$ o vetor da entrada, $\mathbf{W}$ a matriz de pesos, $\mathbf{b}$ os *bias* e a $f(\cdot)$ a função de transferência. A equação (3.1) difere da (3.2) por utilizar uma função aliada a soma das entradas multiplicadas pelos pesos.

A inicialização dos pesos dos neurônios presentes em uma Rede Neural é feita, geralmente, de forma aleatória (Haykin, 1989). Para que uma rede neural seja útil, ela deve ser capaz de processar as entradas na espera de fornecer as saídas desejadas para as entradas apresentadas. A saída desejada é obtida quando a matriz de pesos $\mathbf{W}$ dos neurônios possuem determinados valores, tais que, quando associados a uma entrada específica $\mathbf{e}$ e aplicados na equação (3.1) ou (3.2) geram uma saída específica $\mathbf{s}$.

Na tentativa de se obter essas saídas específicas são empregados algoritmos de treinamento capazes de ajustar os pesos seguindo regras de maneira sistemática. Um algoritmo de treinamento requer um conjunto de entradas e pode requerer, ou não, um conjunto de saídas. O fato do algoritmo requerer a saída específica, ou não, é determinado segundo a sua forma de aprendizado, que pode ser: supervisionada, reforço ou não supervisionada.

3.3 Tipos de Aprendizagem

“Aprendizagem é um processo onde os parâmetros livres de uma Rede Neural Artificial são adaptados por meio de estímulos do ambiente na qual a rede está inserida.” (Haykin, 1989).

O aprendizado é uma característica poderosa presente nas redes neurais. É neste processo que as informações do ambiente são armazenadas nos neurônios na forma de pesos sinápticos.

Existem diferentes formas de aprendizagem, sendo a primeira a supervisionada, a segunda não supervisionada e, segundo alguns autores existe uma terceira, que é a aprendizagem por reforço.

Na aprendizagem supervisionada é fornecida a rede neural as entradas, ou padrões, e suas respectivas saídas. Neste tipo de treinamento os a resposta para determinada entrada é comparada com a resposta ideal, determinando um erro. Através deste erro é possível fazer a correção dos parâmetros para que a resposta da rede se aproxime da resposta esperada.

No aprendizado por reforço o mapeamento entrada/saída é feito através de um processo de tentativa e erro projetado para maximizar a performance (Haykin, 1989).

No aprendizado não supervisionado, ou auto organização, não são fornecidas as respostas corretas junto aos padrões de entradas. Neste aprendizado, a rede extrairá conhecimento apenas das entradas. Este tipo de treinamento é utilizado geralmente para resolução de problemas de agrupamento onde os grupos não são conhecidos previamente. Em problema de agrupamento este tipo de aprendizado geralmente faz com que a rede agrupe os padrões similares.

Para treinar uma rede de uma maneira não supervisionada é possível empregar uma regra de aprendizado competitiva em uma rede de duas camadas, uma cada de entrada e uma camada competitiva, onde a camada de entrada receberá os padrões e a camada competitiva armazenará os neurônios que competirão pelos padrões. Na próxima seção é descrito um modelo de rede neural que é treinado de maneira não supervisionada.

3.4 Mapa Auto Organizável

Um Mapa Auto-Organizável (SOM - *Self Organizing Map*)(Kohonen, 2001) é um modelo de rede neural artificial competitiva não-supervisionada (Hagan et al., 1996) criado por Teuvo Kohonen (Kohonen, 1995). Este modelo de rede foi biologicamente inspirado na organização do córtex cerebral humano. O córtex apresenta regiões para atividades específicas, e que a dado estímulo, o grau de atividade dos neurônios decai conforme a distância da sua região (Kohonen, 2001).

O Mapa Auto-Organizável é um exemplo de como Redes Neurais podem aprender ou auto-organizar-se sem a apresentação da resposta correta para um dado estímulo ou padrão de entrada (Oliveira, 1997).

Este modelo é útil na redução da dimensionalidade de padrões através da auto-organização (Liu e Liu, 2004) e na classificação e agrupamento de padrões (Kohonen, 2001). Quando utilizado para classificar padrões, são empregadas duas fases: treinamento e classificação (Oliveira, 1997).

Durante a fase treinamento, os padrões são apresentados à rede para que os pesos dos neurônios sejam corretamente ajustados, para que sejam formadas as regiões dentro do mapa topológico conforme a semelhança dos padrões, ou seja, padrões semelhantes são agrupados em uma região durante a fase de treinamento.

Na fase classificação, quando uma entrada é apresentada a rede, é feita a procura da região, ou classe, que oferece a maior semelhança. A nova entrada é classificada conforme a classe de maior semelhança encontrada (Oliveira, 1997).

O Mapa Auto-Organizável apresenta uma arquitetura não hierárquica, ou seja, todos os neurônios são do mesmo tipo, desempenhando uma mesma função e pertencendo a uma única camada. Todos os neurônios funcionam como:

- unidade de entrada, recebendo os sinais da entrada de maneira paralela;
- unidade de treinamento, pois todos os neurônios, possuem pesos que são ajustados para armazenar o conhecimento;
- unidade de saída, pois todos os neurônios são responsáveis por armazenar o conhecimento, adquirido durante a fase de treino, para gerar um resultado.

A Figura 3.1 demonstra as ligações entre os neurônios em um Mapa Auto-Organizável.

Neste modelo, segundo seu criador (Kohonen, 2005), os neurônios são organizados de uma maneira bi-dimensional para que os padrões similares sejam aproximados e os padrões não similares sejam afastados.

A seguir é descrito o algoritmo para treinamento de um mapa auto-organizável.

Figura 3.1: Ligações entre os neurônios

3.4.1 Algoritmo de treinamento

A inicialização dos pesos é feita através da atribuição de valores aleatórios pequenos, evitando a imposição de qualquer pré-ordenação. O algoritmo prossegue em 3 etapas: competição, cooperação e adaptação. Cada uma destas etapas é descrita a seguir.

3.4.1.1 Competição

Para cada padrão de entrada apresentado é feita uma busca entre todos os neurônios para determinação de um neurônio vencedor. O neurônio vencedor é o neurônio que seja mais próximo ao padrão entre todos os neurônios. Para determinação da semelhança entre um neurônio e o padrão podemos aplicar a distância euclidiana, descrita na seguinte equação:

$$dist(\mathbf{n}, \mathbf{p}) = \sqrt{\sum_{k=1}^N (\mathbf{n}_k - \mathbf{p}_k)^2}, \quad (3.3)$$

onde n é o neurônio na coordenada (x, y) , N o número de descritores do padrão e p é o padrão apresentado.

A determinação do neurônio vencedor $\mathbf{v}$ pode ser feita através da comparação entre todas as distâncias entre o padrão e os neurônios da rede. A equação (3.4) descreve este processo.

$$\mathbf{v} = \arg \min_n dist(\mathbf{n}, \mathbf{p}) \quad (3.4)$$

Após determinado o neurônio vencedor $\mathbf{v}$, é feita a etapa de cooperação descrita a seguir.

3.4.1.2 Cooperação

O neurônio vencedor determina uma vizinhança de neurônios cooperantes. Um mapa auto-organizável apresenta geralmente uma implementação em uma, duas ou até três dimensões, dando-se preferência a implementação em uma ou duas dimensões. Sendo cada neurônio identificado por sua coordenada no mapa, a distância entre o neurônio e neurônio vencedor é a distância entre suas respectivas coordenadas. Esta informação é útil para determinação da vizinhança entre os neurônios cooperantes.

A intensidade entre a interação do neurônio com o neurônio vencedor utiliza geralmente uma função monotonicamente decrescente e simétrica em torno do ponto máximo. A função gaussiana é uma função que apresenta esta característica e é descrita na seguinte equação:

$$d(x) = e^{\left(-\frac{x - x_0}{2\sigma^2}\right)} \quad (3.5)$$

Interpretando geometricamente a utilização da função gaussiana, podemos ver que a interação entre os neurônios vai diminuindo com a sua distância $(x - x_0)$ em relação ao neurônio vencedor.

Além da diminuição da interação em relação a distância entre o neurônio e o neurônio vencedor, é interessante a aplicação da função gaussiana para diminuição da influência com a progressão do treinamento. A progressão do treinamento é evolução das épocas, na qual podemos definir época pela apresentação seqüencial de todo o conjunto de padrões. Após a execução de uma época é feita outra apresentação de padrões, dando início a uma nova época. São executadas várias épocas no treinamento de uma rede neural. A seguir é descrito o processo de adaptação, onde os pesos dos neurônios são ajustados.

3.4.1.3 Adaptação

A etapa de adaptação é onde os pesos sinápticos são atualizados para que o neurônio absorva uma fração do padrão apresentado. Esta absorção de informações pode ser interpretada como um aprendizado, no qual as informações representadas pelos padrões são organizadas e armazenadas no mapa auto-organizável.

A atualização dos pesos é feita pela equação (3.6)

$$\mathbf{n}_{x,y}(t + 1) = \mathbf{n}_{x,y}(t) + \alpha\mu(t)h_{c(\mathbf{x}),x,y}(\mathbf{x}(t) - \mathbf{n}_{x,t}(t)), \quad (3.6)$$

onde α é a taxa de aprendizado, $\mu(t)$ é uma função gaussiana que decresce conforme a época t , e $h_{c(\mathbf{x}),x,y}$ é uma outra função gaussiana que decresce conforme a distância. É comum a utilização de pequenos valores como taxa de aprendizado, o que faz com que a rede não apresente um comportamento oscilatório durante a adaptação, fazendo com

que ela somente absorva uma pequena fração do padrão apresentado. Se a rede absorver valores grandes dos padrões, ela apresentará muita variação não convergindo para uma solução. A função $h_{c(\mathbf{x}),x,y}$ descreve a interação entre os neurônios citada na fase de cooperação.

O processo adaptativo de um mapa auto-organizável pode ser decomposto em duas grandes etapas:

- Auto-organização: No estado inicial, o mapa auto-organizável apresenta um estado de total desordem (obtida na inicialização aleatória) e com o decorrer das épocas no treinamento a rede vai caminhando para um estado de ordem, através da diminuição das influências de adaptação e cooperação.
- Convergência: É quando as influências de adaptação e cooperação já apresentam valores baixos e os padrões apresentados geram uma atualização com menos impacto, porém com maior direção a obtenção de um estado ordenado.

Como critério de parada do treinamento, podemos utilizar uma das estratégias: 1) determinação de uma quantidade máxima de épocas, o que faz com que o treinamento tenha um finito número de épocas, e 2) determinação de uma atualização mínima das influências de adaptação e cooperação, ou seja, os valores de adaptação e cooperação são insignificantes para prosseguir o treinamento.

Na Figura 3.2 é exibido o resultado do treinamento de um mapa auto-organizável após t iterações. Nesta Figura podemos observar o comportamento de um Mapa Auto-Organizável, onde a organização (Figura 3.2(k)) é obtida a partir da desordem inicial (Figura 3.2(a)).

Figura 3.2: inicialização e treinamento (0 a 1000 iterações) de um mapa auto-organizável

3.5 Exemplo de aplicação: agrupamento do Iris data-set

O banco de dados *Iris data set* (Fisher, 1936) é um conjunto de dados multi variado utilizado em diversos experimentos envolvendo aprendizado de máquina. Este conjunto é um composto de 150 amostras de flores de Lírios e cada amostra é descrita através de 4 descritores, que são comprimento e largura da sépala e pétala. As 150 amostras estão divididas em 3 classes contendo 50 amostras de cada uma das seguintes espécies: Setosa, Virginica e Versicolor.

Neste exemplo de aplicação é utilizado um mapa auto-organizável para criar agrupamentos das 3 classes contidas no conjunto de dados. O mapa auto-organizável é treinado até que 10.000 épocas sejam executadas. Foram realizados 3 experimentos variando o tamanho do mapa, na qual foram utilizados 10x10 neurônios, 20x20 neurônios e 50x50 neurônios.

É interessante ressaltar o aprendizado não supervisionado, ou seja, a rede neural não considera nenhuma informação sobre as classes durante a fase de treinamento, considerando apenas os 4 descritores que formam o padrão.

As Figuras 3.3(a), 3.3(b) e 3.3(c) demonstram os agrupamentos obtidos com os mapas de 10x10, 20x20 e 50x50, respectivamente.

Neste exemplo de aplicação fica claro a efetividade de mapas auto-organizáveis na aplicação de agrupamentos e classificação de dados através de aprendizado não supervisionado.

(a) mapa 10x10

(b) mapa 20x20

(c) mapa 50x50

Figura 3.3: Agrupamentos do conjunto de dados *Iris data set*

3.6 Considerações finais

Neste capítulo, foi descrito um modelo de rede neural chamado mapa auto-organizável. Este modelo será testado em dois experimentos: o primeiro, utilizando uma cena real de um ambiente de futebol de robôs para classificação dos pixels RGB da imagem e o segundo, utilizando uma imagem real para demonstrar a redução de quantização não uniforme.

No próximo capítulo, é apresentado o sistema de visão computacional desenvolvido neste trabalho.

Sistema de Visão Computacional Desenvolvido

Este capítulo descreve a utilização das técnicas citadas anteriormente para a implementação de um sistema de visão computacional para futebol de robôs da categoria MI-ROSOT. Esta implementação tem como objetivos: 1) testar as técnicas em um ambiente real, ou seja, um ambiente que apresenta ruído e variação controlada da iluminação e 2) testar a performance das técnicas em relação ao tempo de processamento.

A descrição da implementação começa com uma revisão da literatura sobre os sistemas de visão computacionais existentes.

4.1 Sistemas de visão computacional para futebol de robôs

Nesta seção, alguns sistemas de visão, aplicados no ambiente de futebol de robôs, são apresentados, destacando-se como a análise de cores é realizada nestes sistemas.

4.1.1 Dortmund Droids

O sistema de visão implementado pelo time Dortmund Droids, da Universidade de Dortmund, Alemanha, utiliza um classificador de cores baseado no espaço de cores HSI acoplado à seleção de relevância dos pixels que passa por um algoritmo de segmentação aliado a um algoritmo de análise de formas, possibilitando a detecção dos robôs. Devido a utilização de etiquetas identificadoras idênticas para todos os robôs, é necessária a utilização de um algoritmo de rastreamento para determinação dos robôs.

A Figura 4.1 detalha a arquitetura deste sistema de visão.

Figura 4.1: Arquitetura do sistema de visão do Dortmund Droids

Nas seções a seguir, os passos envolvidos em todo o processo da arquitetura do Dortmund é descrito.

4.1.1.1 Processamento Inicial

O primeiro passo do sistema de visão do Dortmund é classificar os pixels segundo sua relevância. Para isso, é obtida uma imagem do campo vazio, que será chamada de imagem de referência. As imagens obtidas durante a partida são comparadas com a imagem de referência para que os pixels que possuam valores diferentes de seus respectivos na imagem de referência sejam classificados como relevantes. Esse método é similar ao método de subtração de fundo (Forsyth e Ponce, 2002).

A Figura 4.2 mostra o resultado da classificação dos pixels segundo a relevância.

Figura 4.2: Classificação da relevância dos pixels

4.1.1.2 Classificador de cores e segmentação

O classificador de cores utiliza o espaço HSI, um espaço de cores parecido com o HSV, aplicando o limiar nos canais para que um pixel seja classificado como relevante e como sendo de uma cor pertencente a algum objeto. Pixels com valores do canal I (intensidade) próximos de 1 (branco) ou 0 (preto) e pixels com valores do canal S (saturação) próximos de 0 (cores com pouca saturação) são classificados como irrelevantes. Esta estratégia é útil para redução do tempo de processamento.

A segmentação é realizada através de um algoritmo de crescimento de região. O resultado da segmentação é a obtenção das extremidades dos *blobs*, ou conjuntos de cores similares que podem ser considerados como sendo um único objeto. Essas extremidades são utilizadas nos cálculos de medidas geométricas chamadas circuncidade, quando objetos apresentam formas circulares, retangularidade, quando objetos apresentam formas retangulares e triangularidade, quando objetos apresentam formas triangulares. Isso torna o sistema capaz de diferenciar a forma de *blobs* que apresentem uma mesma cor. Assumindo que a bola seja laranja, esse sistema seria, por exemplo, capaz de diferenciar a bola mesmo que a parte superior dos robôs utilizassem retângulos laranjas. Essa é uma característica não explorada pelos outros sistemas de visão estudados.

4.1.1.3 Correção da distorção da imagem

A presença de imperfeições ópticas na lentes utilizadas pelo Dortmund, faz com que as imagens sejam obtidas com aberrações. Estas aberrações causam erros no cálculo da posição e orientação do robô, implicando na necessidade da aplicação de um método de correção.

A correção do campo é feita através da aplicação de transformações geométricas bilineares. Essas transformações são aplicadas na orientação e posição do robô ao invés de serem aplicadas na imagem, evitando que a imagem inteira seja corrigida. Esse método de tratamento é interessante por resolver o problema de maneira eficiente, já que os cálculos são aplicados a apenas algumas posições ao invés de todos os pixels da imagem.

A definição das transformações bilineares é feita através de uma calibração que utiliza o ponto central e oito pontos de extremidade para seccionar o campo em quatro partes conforme a Figura 4.3. Isto permite que qualquer posição e orientação de um objeto, em um determinado quadrante, sejam corretamente corrigidas.

4.1.1.4 Rastreamento

O time Dortmund Droids utiliza um esquema de etiquetas identificadoras idênticas em todos os robôs. Esta é uma estratégia que visa a diminuição da complexidade do

Figura 4.3: Pontos limitantes para aplicação das transformações bilineares

sistema de calibração. Porém, esta estratégia implica na implementação de um algoritmo de rastreamento para que os robôs sejam corretamente identificados.

Algoritmos de rastreamento são interessantes por simplificarem a calibração e a classificação, pois menos cores são necessárias para descrever cada robô. Entretanto, eles trazem problemas tais como: aumento do tempo de processamento e aumento da complexidade do desenvolvimento do software, já que o tratamento de colisões entre robôs se torna indispensável para manter a confiabilidade do rastreamento (Weiss e Hildebrand, 2004).

Segundo (Weiss e Hildebrand, 2004), o rastreamento é comumente implementado através de duas técnicas: enfoques gulosos ou rastreamento por otimização global. Enfoques gulosos são simples por serem baseados na proximidade da posição atual com a posição anterior, porém instáveis. A instabilidade surge quando existe a possibilidade de colisão entre os objetos ou um aglomeramento de robôs (duas situações comuns em um ambiente de futebol de robôs).

O rastreamento por otimização global utiliza todas as possíveis configurações entre o objeto na posição atual e sua posição anterior. O rastreamento por otimização global é mais estável, porém menos eficiente em termos de tempo de processamento em relação ao enfoque guloso. O Dortmund utiliza um método baseado no enfoque guloso com otimização local para o rastreamento de objetos.

4.1.1.5 Resultados da visão do Dortmund

Segundo (Weiss e Hildebrand, 2004), a precisão de posicionamento de um robô é aproximadamente ± 5 mm, a precisão da orientação é $\pm 5^\circ$, a precisão da orientação através da informação obtida pelo *tracking* é de $\pm 15^\circ$ e a precisão da velocidade é de ± 50 mm/s.

A seguir é descrito o time de futebol de robôs Plymouth.

4.1.2 Plymouth

O sistema de visão do time Plymouth utiliza o espaço HSI e, juntamente com todos os outros módulos (estratégia e comunicação), alcança o processamento de 25 quadros por segundo. Esse sistema de visão utiliza uma câmera JVC TK-C1381 conectada a uma placa de aquisição de vídeo Matrox Meteor II em um Pentium II de 350 MHz (Plymouth, 2006).

O time Plymouth utiliza etiquetas baseadas em círculos, conforme a Figura 4.4.

Figura 4.4: Etiquetas identificadoras do time Plymouth

4.1.2.1 Processamento Inicial

Como processamento inicial é utilizado um filtro passa baixas para diminuição de ruído e suavização da imagem. O resultado obtido pode ser visualizado na Figura 4.5:

(a)

(b)

Figura 4.5: (a) imagem sem filtro, (b) imagem com filtro.

4.1.2.2 Classificação de cores

A classificação de cores utilizado pela visão do Plymouth é baseado no espaço de cores HSI com a aplicação de limiarização no canal H. Por questões de eficiência, a imagem é

varrida de 4 em 4 pixels (ao invés de ser varrida de pixel em pixel) para que os pixels relevantes sejam detectados.

Para que o sistema seja robusto a mudança de iluminação, foi implementada uma técnica capaz de fazer com que os níveis do canal H dos pixels de cada objeto de interesse se mantenham o mais próximo possível de uma calibração padrão, independente da condição de iluminação do campo. Essa técnica é baseada no ajuste do parâmetro *white balance* presente na placa de captura para que os valores de picos dos canais R, G e B sejam alinhados. De acordo com (Plymouth, 2006), essa técnica possibilita que os valores não sejam recalibrados a cada mudança na condição de iluminação do campo.

A aplicação do classificador faz com que sejam criados *arrays* contendo as posições dos pixels pertencentes a bola ou aos robôs.

4.1.2.3 Busca de Círculos

Depois que as coordenadas dos pixels de bola ou robôs foram armazenadas nos arrays criados pelo classificador de cores, é utilizado um algoritmo chamado *Cross Algorithm*, detalhado no Algoritmo 1.

Algoritmo 1 Cross Algorithm

```

Processe o array com os pixels de cor da bola.
Conte o número de pixels próximos  $ph$  na linha  $l$ 
se  $ph \sim raiodabola$  então
    Calcule o centro do seguimento determinado por  $ph$ 
    Conte o número de pixels próximos  $pv$  na coluna  $c$ 
    se  $pv \sim raiodabola$  então
        Calcule o centro do seguimento determinado por  $pv$ 
        A coordenadas  $x$  é o centro do seguimento  $ph$ 
        A coordenadas  $y$  é o centro do seguimento  $pv$ 
    fim se
senão
    passe para a próxima linha  $l + 1$ 
fim se

```

O mesmo algoritmo é aplicado para que sejam achados os círculos maiores dos robôs. Após a detecção do círculo maior, é aplicado o Algoritmo 2.

Algoritmo 2 small circle detection

```

Analise cada pixel  $p$  ao redor de um raio ligeiramente maior do que o raio do círculo maior do robô. {conforme Figura 4.6}
para todos  $x,y$  pertencente ao raio ao redor do maior círculo faça
    execute o Cross Algorithm para encontrar o menor círculo e seu respectivo centro.
fim para

```

Figura 4.6: Raio de busca do menor círculo ao redor do maior círculo.

4.1.2.4 Análise

Para que a orientação (o ângulo de rotação com relação à horizontal) seja obtida a partir dos círculos contidos nas etiquetas identificadoras do time (veja Figura 4.4), é feito um cálculo geométrico simples, conforme ilustrado na Figura 4.7.

Figura 4.7: Orientação de um robô.

4.1.3 Guaraná

O sistema de visão computacional do time Guaraná é um sistema totalmente diferente dos demais sistemas de visão computacional aqui descritos. As principais diferenças são o modo como os pixels são classificados e a presença de um rastreamento de otimização exaustiva. Outro ponto que o torna interessante para um estudo de caso é a eficiência obtida em dois campeonatos: o primeiro em 1998 na França (Costa e Pegoraro, 2000), alcançando a segunda colocação e o segundo em 2005 durante a V Competição IEEE de Robótica realizada em São Luiz - MA, alcançando a primeira colocação, sendo utilizado pelo time da Unesp - Bauru.

4.1.3.1 Classificação de cores

Diferente de todos os outros times aqui descritos, nos quais se apóiam em espaços de cores que possuam um canal separando a iluminação da cromaticidade, o Guaraná utiliza a aplicação de limiares superiores e inferiores na relação $\frac{R}{G}$ e $\frac{G}{B}$ para classificação dos pixels.

Durante a fase de calibração, realizada 20 minutos antes da partida, um operador deve selecionar os *pixels* de interesse em diversos pontos do campo exibidos em uma tela de calibração. Este processo tem a finalidade da determinação dos limites superiores e inferiores das relações $\frac{R}{G}$ e $\frac{G}{B}$. Os pixels devem ser amostrados em diversas posições do campo para que as diferenças de iluminação sejam tratadas adequadamente na classificação dos pixels durante a partida.

4.1.3.2 Subtração da imagem de referência

A visão do time Guaraná utiliza uma técnica de subtração de fundo para eliminação de pixels similares ao fundo. A técnica de subtração de fundo aplicada pela visão do Guaraná se baseia na construção de uma imagem modelo, a qual é construída a partir de uma média de N imagens adquiridas com um espaçamento temporal Δt . A construção dessa imagem modelo busca um melhor tratamento para variações de iluminação e iluminação não uniforme no campo.

A imagem média, ou modelo, é calculada pela Equação (4.1):

$$\begin{aligned} \bar{p}(x, y) &= \frac{\sum_{i=1}^N p_i(x, y)}{N} \\ 0 &< x < largura \\ 0 &< y < altura \end{aligned} \quad (4.1)$$

Após calculada a imagem média, são realizados os seguintes cálculos, para determinação dos limiares dos canais vermelho (LR) e verde (LG).

$$\begin{aligned} LR &= \max(|r(\bar{p}(x, y)) - r(p_i(x, y))|) \\ LG &= \max(|g(\bar{p}(x, y)) - g(p_i(x, y))|) \\ i &= 1, \dots, N \end{aligned}$$

Durante o processamento da visão, é gerada uma imagem de diferença $p_d(x, y)$ da seguinte maneira:

$$\begin{aligned} r(p_d(x, y)) &= |r(\bar{p}(x, y)) - r(p_a(x, y))| \\ g(p_d(x, y)) &= |g(\bar{p}(x, y)) - g(p_a(x, y))|, \end{aligned} \quad (4.2)$$

onde p_a representa a imagem atual, $r(p)$ é o valor do componente vermelho e $g(p)$ é o valor do componente verde. Através da seguinte avaliação

$$\begin{aligned} r(p_d(x, y)) &= |r(\bar{p}(x, y)) - r(p_a(x, y))| \\ g(p_d(x, y)) &= |g(\bar{p}(x, y)) - g(p_a(x, y))|, \end{aligned} \quad (4.3)$$

um pixel é caracterizado como sendo de interesse ou não. Após um pixel ser caracterizado como de interesse, é feita uma análise de similaridade com os dados obtidos na fase de calibração para que seja finalizada a classificação.

4.1.3.3 Agrupamento e cálculo de centróides

Pixels similares em uma vizinhança são considerados como componentes conectados do mesmo objeto. Para que o ruído não seja considerado objeto, é feita, durante o estágio da calibração, a configuração do tamanho de um objeto que possua uma determinada cor. Para cada agrupamento de pixels é feito o cálculo dos centróides segundo a seguinte equação 4.4:

$$x = \frac{\sum_{i=1}^n x_i}{n} \quad y = \frac{\sum_{i=1}^n y_i}{n} \quad (4.4)$$

sendo x_i e y_i a coordenadas do objeto detectado e n o número de pixels do objeto detectado.

4.1.3.4 Rastreamento

O guaraná, assim como o time Dortmund Droids, utiliza um esquema de etiquetas idênticas para todos os robôs, sendo impossível a identificação de um robô específico. Isto torna necessário a implementação de um método de rastreamento. O método de rastreamento implementado pelo time Guaraná utiliza um método de otimização exaustiva.

No início da partida cada robô é identificado por um operador humano e depois o algoritmo de rastreamento é capaz de continuar a identificação correta dos robôs. Segundo (Costa e Pegoraro, 2000), o rastreamento implementado é eficaz, não perdendo ou fazendo a identificação incorreta dos robôs.

4.1.3.5 Orientação e posição dos objetos

As etiquetas do time Guaraná são constituídas de dois retângulos, sendo um da cor do time (azul ou amarela) e o outro da cor rosa. A cor rosa tem a finalidade de determinar a frente do robô, permitindo determinar a orientação do robô.

Após detectados os centróides das duas cores, é calculado o ângulo da orientação como mostra a Equação 4.5, onde x_t e y_t são as coordenadas do centróide da etiqueta do time e x_o e y_o são as coordenadas do centróide da etiqueta da orientação (etiqueta rosa).

$$\alpha = \tan^{-1} \frac{y_t - y_o}{x_t - x_o} \quad (4.5)$$

Para calcular a posição é feita a média ponderada das coordenadas dos centróides das etiquetas do time e orientação através da Equação (4.6).

$$x = \frac{c_1 x_t + c_2 x_p}{c_1 + c_2} \quad y = \frac{c_1 y_t + c_2 y_p}{c_1 + c_2} \quad (4.6)$$

4.1.3.6 Resultados

A software de visão do time Guaraná é executado em um Pentium II 233 MHz com uma placa de captura PixelView PV - BT848. O software foi desenvolvido para o sistema operacional Windows 95 e foi implementado em linguagem C e assembly. É uma visão robusta, porém utiliza a resolução de 320x240, o que faz com que os objetos de interesse: robôs e bolas sejam muito pequenos tornando-os facilmente confundidos com ruídos em condições de iluminação não ideais. A visão do time Guaraná consegue alcançar uma precisão máxima de $\pm 10^\circ$ na identificação da orientação do robô.

4.1.4 CMDragons - CMVision

O sistema de visão CMVision é um sistema de visão baseado em cores que utiliza o espaço cores YUV. Este sistema é capaz de processar 30 imagens por segundo com a resolução de 320 x 240 px em um segundo em uma máquina com o processador Pentium II de 200 MHz (Bruce et al., 2000).

O sistema CMVision foi utilizado pela equipe Eagle Knights do Instituto Tecnológico Autônomo do México - ITAM com algumas leves modificações para identificação dos robôs (Martinez-Gomez e Weitzenfeld, 2004).

4.1.4.1 Classificador de cores

O classificador aplica técnicas de limiarização sobre os pixels da imagem utilizando o espaço de cores YUV para representação de cores. A limiarização empregada pelo CM-

Vision é feita com uma otimização interessante que visa diminuir o número de instruções a serem executadas pelo processador e funciona substituindo:

```
if ((Y >= Ylowerthresh) AND (Y <= Yupperthresh) AND (U >= Ulowerthresh)
AND (U <= Uupperthresh) AND (V >= Vlowerthresh) AND (V <= Vupperthresh))
pixel_color = color_class;
```

por

```
pixel_in_class = YClass[Y] AND UClass[U] AND VClass[V].
```

Essa substituição utiliza *arrays* e operações *AND bitwise* para que sejam substituídas as comparações inteiras necessárias no código com *ifs*. Esta técnica explora o fato de que comparações *AND bitwise* são significativamente menos custosas do que comparações inteiras na maioria dos processadores modernos (Bruce et al., 2000).

4.1.4.2 Determinação das Regiões Conectadas

Após a aplicação do classificador, os pixels com valores idênticos devem ser agrupados para formar objetos. Para isso, é computado o *Run Lenght Code* (Mckerrow, 1991) da imagem discretizada.

A Figura 4.8 exibe o resultado da aplicação do classificador de cores junto ao *Run Lenght Code*.

Figura 4.8: Resultado da classificação e detecção das regiões conectadas

Na seção seguinte é descrito o sistema de visão computacional para o futebol de robôs do ICMC.

4.2 Sistema de visão computacional desenvolvido

O sistema de visão computacional proposto no presente trabalho para futebol de robôs do ICMC segue a seguinte estratégia: classificar todos os pixels, percorrer todos os

pixels classificados para unir os pixels adjacentes que foram classificados em uma mesma classe. A união destes pixels tem o intuito de segmentar a imagem para formar *blobs* ou macro-objetos, cujas coordenadas, pontos extremos, área e cor passarão por análise, onde finalmente, as coordenadas (x,y) dos objetos serão extraídas.

Na Figura 4.9, é possível visualizar a sequência dos processamentos, iniciando pela aquisição da imagem através da placa de captura.

Figura 4.9: Fluxo do sistema

Após a aquisição da imagem, é feita a redução da quantização para que a imagem seja representada por um espaço menor do que os 8 bits utilizados pela imagem original. A seguir, é aplicado um dos dois classificadores: limiar no HSV ou mapas auto-organizáveis. A aplicação do classificador tem a finalidade de processar todos os pixels da imagem, gerando uma imagem onde cada pixel conterá o identificador de sua respectiva classe.

O classificador através de limiarização no espaço HSV foi escolhido por possibilitar a descrição de cores de maneira simplificada e por ter demonstrado bom desempenho de classificação em testes empíricos. O mapa auto organizável é um tipo de rede neural que utiliza aprendizado não supervisionado. Esta característica elimina a necessidade de fornecer amostras de cores e suas respectivas classes. O treinamento não supervisionado, foi o critério decisivo para sua utilização, pois a configuração do classificador se torna mais simples.

Esta imagem classificada é processada por um algoritmo de segmentação que busca pixels adjacentes identificados por uma mesma classe. Esses pixels adjacentes com a mesma

classe são fundidos em *blobs*. Cada *blob* possui as seguintes propriedades: coordenada do centróide, área, pontos extremos na coordenada horizontal e pontos extremos na vertical.

Estes *blobs* passam por uma análise, na qual são identificados os robôs, adversários e bola, finalizando a detecção dos objetos.

Cada etapa do processo pode ser visualizada através de uma renderização utilizando OpenGL para que o processador gráfico da placa de vídeo faça a renderização sem aumentar o tempo de processamento de maneira significativa. A visualização de cada etapa é importante para que o operador verifique visualmente o resultado das configurações dos parâmetros de cada processamento empregado.

É importante observar que os classificadores necessitam ser configurados previamente para que possam funcionar adequadamente. Esta configuração é realizada antes do início da partida (etapa *off-line*), não implicando em nenhum processamento adicional durante o funcionamento.

Nas próximas seções, detalhes dos algoritmos empregados em cada processamento são apresentados.

4.3 Redução da Quantização

A redução da quantização é um ponto importante no sistema, pois determina o espaço necessário para criação da tabela, explicada posteriormente, que armazena os valores de cada possível cor e é chamada tabela de otimização (ou *lookup table*).

A tabela de otimização é utilizada para armazenar as classes correspondente de cada uma das cores de todas as possíveis combinações que um espaço pode oferecer. Levando em consideração que o espaço possua 8 bits por canal e sendo 3 canais, é então possível obter 16.777.216 combinações de cores, um fator que influencia dois pontos: a) tempo de criação da tabela e b) espaço necessário para o armazenamento da mesma.

É possível diminuir a influência dos dois fatores através da aplicação da redução da quantização, possibilitando que a criação da tabela seja mais rápida e que o espaço necessário para seu armazenamento seja reduzido. Reduzindo-se a quantização para 7 bits, as combinações de cores possíveis são 2.097.152 e utilizando-se 5 bits, a combinação é reduzida para 32.768, uma redução significativa e que, em testes empíricos, tem se tornado a quantização padrão do sistema.

Além dos dois fatores anteriormente citados, a redução da quantização ainda exerce um papel importante na redução da quantidade de amostras para o treino da rede. Esta redução faz com que a rede não tenha que ser treinada com amostras com valores muito semelhantes, o que aumenta o tempo de treinamento e não introduz nenhum efeito significativo na classificação. Esta operação de redução de quantização, implicitamente,

reduz a quantidade de amostras de treinamento, tornando-o mais rápido do que se fossem utilizadas imagens de 8 bits cada.

A seguir são descritos os classificadores e a tabela de otimização.

4.4 Classificador e tabela de otimização

O classificador empregado no sistema utiliza um dos dois métodos estudados no desenvolvimento do trabalho: limiarização no espaço HSV e mapas auto-organizáveis. Ambos classificadores requerem a intervenção do usuário para configuração, de maneira que estes gerem resultados corretos.

No classificador baseado em limiarização no HSV, o usuário deve ter conhecimento sobre a determinação de cores no espaço HSV. Este conhecimento permite que o usuário escolha corretamente a cor a ser classificada. Entretanto, é necessário um pouco de experiência para o ajuste dos valores de tolerância que determinam a limiarização.

No classificador utilizando mapas auto-organizáveis, é necessário que o usuário ajuste o número de neurônios do mapa e também é interessante a configuração dos parâmetros das equações empregadas neste modelo de rede neural.

A tabela de otimização desempenha um papel fundamental no sistema desenvolvido: armazenar os valores das classes para cada pixel RGB. Cada possível cor no espaço RGB possui uma entrada na tabela e cada entrada possui o valor de sua respectiva classe. Para determinação da classe de um pixel RGB basta fazer a consulta na tabela e valor será obtido em apenas uma operação. Entretanto, a tabela necessita ser corretamente preenchida e para cada classificador ela é preenchida de maneira diferente. É importante ressaltar que a tabela conterá todas as possíveis combinações de cores RGB.

No classificador baseado em limiarização no espaço HSV, o usuário determina os valores HSV e seus limiares, formando faixas de cores possíveis para uma determinada classe. Essas cores no espaço HSV que constituem a determinada classe são convertidas em cores RGB. Cada cor RGB e sua respectiva classe é armazenado na tabela, formando uma associação pixel para classe. Esse processo de conversão e associação pixel classe é a maneira de preenchimento da tabela quando é utilizado o filtro baseado em limiarização no espaço HSV.

Para preenchimento da tabela, quando estamos utilizando o classificador baseado em redes neurais, são percorridas todas as posições da tabela, apresentando cada uma destas posições (que descrevem cores RGB) ao mapa, previamente treinando. O mapa fornece a classe correta a qual a posição pertence. Considerando que a tabela tenha todas as cores possíveis, o processo de apresentação de cada cor possibilita a classificação total do espaço de entrada.

Esta idéia pode ser utilizada devido ao fato de que o classificador não é alterado durante a execução da partida. Este fato permite que uma pré-classificação seja realizada em uma etapa *off-line*, possibilitando a criação da tabela. A tabela é então utilizada durante a execução (etapa *on-line*), permitindo que um pixel seja classificado em apenas uma operação de acesso.

A redução de quantização é utilizada para diminuir o total de entradas da tabela, já que a quantização é o fator determinante da combinação de cores totais que a imagem poderá apresentar.

O classificador resulta em uma imagem contendo pixels com os valores de suas respectivas classes. O próximo passo é aplicação de um algoritmo de segmentação, descrito a seguir.

4.5 Blob-coloring

É empregado um algoritmo de segmentação para agrupar os pixels classificados com valores idênticos que sejam adjacentes. Estes agrupamentos formam um *blob* ou macro-objeto. O algoritmo classifica um pixel como sendo pertencente ao mesmo *blob* verificando os dois pixels imediatamente vizinhos: o superior e esquerdo. Caso este pixel não seja igual a estes, ou mesmo não havendo estes vizinhos previamente classificados é criado um novo objeto representado pela classe deste pixel.

O algoritmo utilizado é chamado *Blob Coloring* e é apresentado como um algoritmo de segmentação por crescimento de regiões, descrito em (Ballard e Brown, 1982). O *Blob Coloring* descrito em (Ballard e Brown, 1982) funciona sobre imagens binárias, que são imagens descrita por pixels assumindo somente dois valores.

O *Blob Coloring* é um algoritmo rápido, sendo capaz de detectar os “blobs” presentes na imagem em tempo linear $O(n)$. No sistema implementando, ele foi modificado para trabalhar com imagens criadas pelo classificador, não possuindo somente valores 0 e 1. Também foram adicionado o calcular da área, e as posições extremas esquerda, direita, topo e fundo de cada macro-objeto, bem como seu centróide.

Segue o algoritmo modificado:

Algoritmo 3 Modificação do *Blob Coloring* para o sistema de visão

Z = imagem criada pelo classificador

$blobs$ = array de blobs (objetos), onde serão guardadas as informações dos objetos detectados

etiqueta = 1

nblobs = 0

```

para  $i = y_{min}$  to  $y_{max}$  faça
  para  $j = x_{min}$  to  $x_{max}$  faça
    se  $Z(i, j) \neq 0$  então
      se  $Z(i, j) = Z(i - 1, j)$  e  $Z(i, j) \neq Z(i, j - 1)$  e  $\text{equiv}[\text{labels}(i - 1, j)] \neq 0$ 
        então
           $\text{labels}(i, j) = \text{labels}(i - 1, j)$ 
           $\text{tmp} = \text{equiv}[\text{labels}(i - 1, j)]$ 
           $\text{blobs}[\text{tmp}].\text{area} = \text{blobs}[\text{tmp}].\text{area} + 1$ 
           $\text{atualiza-limites}(\text{blobs}[\text{tmp}], i, j)$ 
        senão
          se  $Z(i, j) \neq Z(i - 1, j)$  e  $Z(i, j) = Z(i, j - 1)$  e  $\text{equiv}[\text{labels}(i, j - 1)] \neq 0$ 
            então
               $\text{labels}(i, j) = \text{labels}(i, j - 1)$ 
               $\text{tmp} = \text{equiv}[\text{labels}(i, j - 1)]$ 
               $\text{blobs}[\text{tmp}].\text{area} = \text{blobs}[\text{tmp}].\text{area} + 1$ 
               $\text{atualiza-limites}(\text{blobs}[\text{tmp}], i, j)$ 
            senão
              se  $Z(i, j) = Z(i - 1, j)$  e  $Z(i, j) = Z(i, j - 1)$  e
                 $(\text{equiv}[\text{labels}(i, j - 1)] \neq 0$  ou  $\text{equiv}[\text{labels}(i - 1, j)] \neq 0)$  então
                  se  $\text{equiv}[\text{labels}(i, j - 1)] = \text{equiv}[\text{labels}(i - 1, j)]$  então
                     $\text{labels}(i, j) = \text{labels}(i, j - 1)$ 
                     $\text{tmp} = \text{equiv}[\text{labels}(i, j - 1)]$ 
                     $\text{blobs}[\text{tmp}].\text{area} = \text{blobs}[\text{tmp}].\text{area} + 1$ 
                     $\text{atualiza-limites}(\text{blobs}[\text{tmp}], i, j)$ 
                  senão
                    se  $\text{equiv}[\text{labels}(i - 1, j)] < \text{equiv}[\text{labels}(i, j - 1)]$  então
                       $\text{tmp} = \text{equiv}[\text{labels}(i - 1, j)]$ 
                       $\text{labels}(i, j) = \text{tmp}$ 
                       $\text{tmp2} = \text{equiv}[\text{labels}(i, j - 1)]$ 
                       $\text{equiv}[\text{tmp2}] = \text{equiv}[\text{tmp}]$ 
                       $\text{blobs}[\text{tmp}].\text{area} = \text{blobs}[\text{tmp2}].\text{area} + \text{blobs}[\text{tmp}].\text{area} + 1$ 
                       $\text{blobs}[\text{tmp2}].\text{area} = 0$ 
                       $\text{atualiza-limites2}(\text{blobs}[\text{tmp}], \text{blobs}[\text{tmp2}])$ 
                    senão
                       $\text{tmp} = \text{equiv}[\text{labels}(i, j - 1)]$ 
                       $\text{labels}(i, j) = \text{tmp}$ 
                       $\text{tmp2} = \text{equiv}[\text{labels}(i - 1, j)]$ 
                       $\text{equiv}[\text{tmp2}] = \text{equiv}[\text{tmp}]$ 
                       $\text{blobs}[\text{tmp}].\text{area} = \text{blobs}[\text{tmp2}].\text{area} + \text{blobs}[\text{tmp}].\text{area} + 1$ 

```

```

        blobs[tmp2].area = 0
        atualiza-limites2( blobs[tmp], blobs[tmp2] )
    fim se
fim se
senão
    labels(i,j) = equiv[label] = label;
    blobs[label].label = label;
    blobs[label].area = 1;
    blobs[label].xmin = j;
    blobs[label].ymin = i;
    blobs[label].xmax = j;
    blobs[label].ymax = i; label++;
fim se
fim se
fim se
fim para
fim para

```

Algoritmo 4 Blob coloring - atualiza-limites(blob, x, y)

```

se blob.xmin > x então
    blob.xmin = x
fim se
se blob.ymin > y então
    blob.ymin = y
fim se
se blob.xmax < x então
    blob.xmax = x
fim se
se blob.ymax < y então
    blob.ymax = y
fim se

```

Algoritmo 5 blob coloring - atualiza-limites2(blob1, blob2)

```

blob1.xmin = min( blob1.xmin, blob2.xmin )
blob1.ymin = min( blob1.ymin, blob2.ymin )
blob1.xmax = max( blob1.xmax, blob2.xmax )
blob1.ymax = max( blob1.ymax, blob2.ymax )

```

Este algoritmo retorna um conjunto de objetos que serão processados por um modo de análise, explicado na seção seguinte.

4.6 Análise

A análise tem a finalidade de obter as posições (x, y) dos objetos em campo e as orientações θ dos robôs. É a etapa de mais alto nível de todo o processamento da visão.

Através do *Blob Coloring* são obtidos os centróides dos objetos que possuam as cores de interesse e possuam uma área superior ao limite determinado pelo operador durante a fase de calibração. Objetos com áreas superiores ou inferiores ao limite estipulado pelo operador são eliminados, evitando que sejam erroneamente detectados como objetos de interesse.

A posição da bola é obtida através centróide do maior objeto que possua a cor da bola e que área esteja dentro dos limites estipulados pelo operador. A posição dos adversários são obtidas pelos três maiores centróides que possuam a cor do adversário.

As posições dos três robôs são obtidas de uma maneira diferente, que é dado pelo algoritmo 6

Algoritmo 6 Posição dos robôs

```

objpri = três maiores objetos com as cores do time.
objsec = três maiores objetos com as cores secundárias.
ordene os objsec de acorde que minimize a distancia entre os objpri
para i = 1 to 3 faça
     $robo_{i,x} = \frac{objpri_x + objsec_x}{2}$ 
     $robo_{i,y} = \frac{objpri_y + objsec_y}{2}$ 
fim para

```

O algoritmo 7 descreve o cálculo da orientação dos robôs:

Através desta análise, são obtidos todas as informações de interesse, que são: coordenadas da bola, robôs e adversários e as orientações dos robôs.

4.7 Considerações Finais

Neste capítulo, o funcionamento do sistema de visão do time de futebol de robôs do ICMC foi apresentado. Este sistema utiliza os classificadores de cores estudados nos capítulos anteriores.

Observando que todo o processamento é feito em menos de 30 milissegundos, conclui-se que os classificadores podem operar em tempo real.

Algoritmo 7 Orientação dos Robôs

```

objpri = três maiores objetos com as cores do time.
objsec = três maiores objetos com as cores secundárias ordenadas com a proximidade
em relação ao objpri.
para i = 1 to 3 faça
  se objpri.x ≠ objsec.x então
     $\theta = \tan^{-1} \left( \frac{objsec_y - objpri_y}{objsec_x - objpri_x} \right)$ 
    se objprix > objsecx então
       $\theta = \theta + 180$ 
    senão
      se  $\theta < 0$  então
         $\theta = \theta + 360$ 
      fim se
    fim se
  senão
    se objpriy > objsecy então
       $\theta = 90$ 
    senão
       $\theta = 270$ 
    fim se
  fim se
  robo[i].theta =  $\theta$ 
fim para

```

O sistema demonstra que os classificadores podem ser utilizados para aplicações de robóticas similares. Um exemplo de possível aplicação seria a criação de um robô capaz de seguir uma linha colorida desenhada no chão.

O próximo capítulo descreve o time de futebol de robôs do ICMC, que foi por nós desenvolvido durante a realização do presente trabalho.

O time de futebol de robôs do ICMC

Este capítulo detalha o time de futebol de robôs do ICMC, um time composto de robôs autônomos capazes de jogar uma partida de futebol. Estes robôs seguem a especificação da MIROSOT e podem ser utilizados em competições que utilizem este tipo de robôs.

As seções seguintes descrevem o time de futebol de robôs do ICMC.

5.1 A estrutura do time

O futebol de robôs é composto pelos seguintes módulos: visão computacional, estratégia, controle, eletrônica e mecânica. Existem itens que fazem parte de mais de um módulo, como o rádio, que além de uma entidade eletrônica também precisa ser configurado no PC da estratégia e no PIC ou ainda os motores, que são elementos da mecânica e da eletrônica. Esses elementos são as pontes de conexão entre os vários módulos. A Figura 5.1 exibe imagens dos robôs do time do ICMC. A comunicação dos sistemas obedece o esquema ilustrado na Figura 5.2. O sistema de visão descrito neste trabalho foi testado no time aqui descrito e também na competição de robótica realizada junto ao Simpósio Brasileiro de Automação Inteligente - SBAI.

Nos próximos tópicos serão esclarecidos os métodos de trabalho de cada módulo, exceto o módulo de visão computacional já detalhado anteriormente. Mais detalhes podem ser encontrados na página do time: <http://uspds.sourceforge.net>.

Figura 5.1: Robôs do time do ICMC

Figura 5.2: Estrutura geral

5.2 Visão Computacional

A visão computacional captura imagens através de uma câmera posicionada sobre o campo de jogo. A câmera passa as imagens para um computador equipado com uma placa de captura de vídeo. Em seguida, esse mesmo computador calcula as posições dos robôs, assim como seus ângulos. O computador responsável pela visão passa, via rede, as coordenadas (x, y) e os ângulos α dos robôs para o computador da estratégia.

O módulo de visão circula em verde os robôs amigos, em vermelho os inimigos e em laranja a bola, que nesse caso está no canto direito superior. A imagem da Figura 5.3 ilustra o funcionamento do sistema de visão.

Figura 5.3: Sistema de visão

Foi neste módulo que participamos de forma mais efetiva, embora tivemos uma grande contribuição no desenvolvimento dos outros módulos.

5.3 Estratégia

O módulo de visão computacional envia para a estratégia, via LAN, as posições e os ângulos dos robôs, e é realizado um processamento baseado na teoria de campos potenciais. Através dessa teoria, o robô calcula a repulsão proveniente dos robôs adversários e a atração proveniente da bola. Com esses cálculos, é possível montar um trajeto que melhor se adapte à situação atual, e com isso é possível determinar a nova posição do robô.

Após uma sucessiva sequência de cálculos, a estratégia sabe para que ângulo o robô deve virar. Com este ângulo, as funções de controle, utilizando lógica nebulosa, deter-

minam a velocidade ideal de cada roda e conseqüentemente a proporção de fases que o PWM (Crowder, 1998) deve trabalhar.

O ciclo de visão é realizado sempre a cada 33 milisegundos, possibilitando que os cálculos empregados para o controle dos robôs sejam realizados com precisão necessária e com o tempo de resposta satisfatório.

Após realizado estes cálculos, a estratégia aciona a porta serial que está ligada diretamente a um circuito de rádio transmissor. Passa-se o vetor \$A,B,C,D,E*\$, onde o sinal \$ indica o início da transmissão, A o sentido da roda esquerda, B indica a velocidade desejada para a mesma, numa escala de 0 a 63, C o sentido da roda direita, D indica a velocidade desejada para a mesma, também numa escala de 0 a 63 e o caractere * indica o final da mensagem.

5.4 Eletrônica

Neste módulo estão presentes os seguintes componentes: rádio, microprocessador, duas baterias, motores, ponte-H, dentre outros componentes eletrônicos comuns. A Figura 5.4 detalha o esquema eletrônico dos robôs.

Figura 5.4: Eletrônica

Foram utilizadas baterias diferentes para o PIC e para o restante do circuito. Este modo de alimentação foi escolhido para evitar que o consumo do motor reinicie o PIC toda vez que tente sair do momento de inércia. Isto acontece porque um motor corrente contínua precisa de muita potência para sair do estado estático, e isso se resume a maiores correntes elétricas (Valentine, 1998).

Como os dois motores são corrente contínua, é necessária a utilização de uma ponte-H para que os motores possam funcionar nos dois sentidos de rotação. Foram utilizados

reguladores de tensão 7805 e 78L05 para que as tensões sejam ajustadas em 5 V, pois as baterias utilizadas fornecem tensões superiores a tensão desejada.

O módulo da eletrônica se comunica com dois outros: mecânica e controle. O controle passa através do transmissor do rádio o vetor \$A,B,C,D,E*\$ para o rádio da eletrônica. O rádio passa o comando para o PIC através da interface RS-232 de comunicação serial. O microprocessador começa a ler o sinal \$ e percebe que está chegando uma nova mensagem. A partir daí associa A ao sentido da roda esquerda, B à velocidade desejada para a mesma, C ao sentido da outra roda, D à velocidade desejada para a mesma. O caractere * indica o final da mensagem.

Os parâmetros do sentido das rodas passam por uma lógica e se dividem em 4 portas do PIC, acionando os pinos de sentido da ponte-H. Já as velocidades das rodas acionam o mecanismo PWM do microprocessador e partem dois pulsos de portas diferentes, também em direção à ponte-H. São esses dois pulsos, associados aos quatro pinos lógicos de sentido que realizam a comunicação com o módulo da mecânica. Toda a programação do microcontrolador foi feita na linguagem Assembly utilizando o GPASM, um software livre para programação de microcontroladores.

5.5 Mecânica

Um protótipo mecânico do robô foi desenhado na ferramenta Solid Edge, licenciada para USP - São Carlos e o modelo foi usinado em alumínio (carcaça) e náilon preto (rodas). As rodas foram usinadas no torno mecânico e o alumínio foi serrado e acabado em uma fresa de precisão. Os furos foram realizados através da furadeira de alta precisão. Os furos também foram escareados e ajustados para possuírem rosca para parafusos M3. As placas de alumínio possuíam profundidade de 5mm.

Foram utilizados motores PM-080 SS 05 da empresa nacional Action Motors. Estes motores trabalham com tensão nominal de 5V e possuem torque de 15 g/cm, atingindo 5450 rotações por minuto. A caixa de redução emprega uma redução de 1 para 4, a roda possui um diâmetro de 50mm, onde os valores de torque final e velocidade final são 204 e 1362, respectivamente.

A Figura 5.5 ilustra a mecânica desenvolvida.

5.6 Controle

O intuito do controle desenvolvido é ter os robôs constantemente, monitorados pela visão, em uma mesma trajetória. A visão informa a posição do robô e o ângulo para que o controle envie pulsos de PWM com maior ou menor duty-cycle, fazendo uma roda girar

Figura 5.5: Mecânica do time

mais que a outra, corrigindo a trajetória. Como o processamento da visão acontece em menos de 33 milissegundos, essas correções são imperceptíveis.

O controle está em desenvolvimento e estão sendo testados dois tipos de controle: o primeiro, utilizando um controle clássico PID e o segundo, utilizando lógica nebulosa. Detalhes do controle podem ser encontrado em (Gonçalves, 2008).

5.7 Considerações finais

Foi detalhado o time de futebol de robôs, um time de robôs autônomos que utilizam o sistema de visão computacional apresentado neste trabalho.

O sistema desenvolvido foi testado em uma competição real e foi verificado que tanto as variações de iluminação bem como o ruído presente nas imagens não influenciavam a performance do sistema.

O próximo capítulo demonstra os experimentos realizados neste trabalho.

Experimentos

Neste capítulo são apresentados os experimentos realizados durante o desenvolvimento deste trabalho. São apresentados experimentos de classificação de cores utilizando limiarização HSV, redução de quantização, do tempo necessário para redução da quantização, classificação de pixels utilizando um mapa auto-organizável, redução de quantização não uniforme através de um mapa auto-organizável e finalmente a aplicação do classificador no sistema de visão computacional para o time de futebol de robôs do ICMC.

6.1 Experimento utilizando o classificador HSV

Neste experimento foi utilizado o classificador HSV para classificar uma imagem de futebol de robôs MIROSOT. O objetivo é classificar: (1) bola, que possui tons de laranja e (2) robôs, que possuem tons de azul e rosa.

A imagem é obtida de uma cena real através de uma placa de captura. Esta imagem utiliza como espaço de cores o espaço RGB, sendo necessária a conversão de cada pixel para o espaço de cores HSV.

A conversão introduz o aumento de processamento necessário, porém utilizando a tabela de otimização, descrita no capítulo 4, é possível realizar esta operação de maneira eficiente.

Para classificar a imagem foi construído um filtro com as configurações presentes na Tabela 6.1. Esses valores foram obtidos de forma empírica.

A Figura 6.1 ilustra o resultado da classificação utilizando o classificador HSV configurado com os limiares presentes na Tabela 6.1.

Tabela 6.1: Configurações do classificador HSV

Cor	H	Graus de Liberdade	S min	S max	V min	V max
Laranja	30	10	30	100	40	100
Rosa	330	30	30	100	20	100
Azul	220	40	70	100	70	100

Figura 6.1: Resultado da classificação no espaço HSV

Podemos ver na Figura 6.1 a eficiência do classificador HSV, onde a bola e os robôs foram corretamente detectados.

Na próxima seção, é detalhado o experimento da redução da quantização de uma imagem utilizando a quantização uniforme.

6.2 Experimento: Redução utilizando quantização uniforme

Neste experimento foi feita a redução da quantização dos três canais RGB de uma imagem utilizando a redução de quantização uniforme. Foram realizadas reduções para 7, 6, 5, 4, 3, 2 e 1 bit.

A Tabela 6.2 demonstra o número de bits por canal, o número de pixels diferentes presentes na imagem para aquela quantização e a medida de degradação utilizando as equações (2.7) e (2.8). O efeito visual desta operação pode ser visualizado na Figura 6.2.

Neste experimento podemos concluir que a redução de quantização de 8 bits para 7, 6, ou até mesmo 5 bits, é útil na redução do número de cores possíveis e que esta operação não apresenta visualmente um caráter depreciativo para determinadas aplicações.

A vantagem da redução do número de combinações de cores possíveis foi explorada nos seguintes pontos do sistema implementado: na redução da quantidade de memória utilizada para armazenar a tabela de otimização, na diminuição do tempo de criação

Figura 6.2: Efeito da redução dos bits

Tabela 6.2: Configurações do classificador HSV

Numero de bits	Numero de pixels diferentes	Degradação
8	55959	0
7	27451	3.986
6	9374	27.977
5	2086	139.584
4	425	637.851
3	99	2604.384
2	28	11194.526
1	8	40457.734

de tabela e na redução do número de padrões de entrada para treinamento do mapa auto-organizável.

O próximo experimento faz uma análise da operação de redução de quantização, em termos de tempo de processamento, para verificar se sua utilização pode interferir no desempenho total do sistema.

6.3 Experimento: tempo de processamento para redução de quantização

Este experimento investiga o tempo usado para redução dos níveis da quantização dos canais de uma imagem. É um experimento que pretende investigar a influência externa de fatores externos como: sistema operacional GNU/Linux e parâmetros externos de otimização do compilador GNU G++. Também é analisada a influência de parâmetros internos, que é a utilização de aritmética inteira ou ponto flutuante.

É utilizada como imagem de entrada, uma imagem sintética RGB de 640x480 pixels, com quantização de oito bits por canal. A execução deste experimento foi realizada em um computador com as seguintes características:

- Processador Pentium 2.8 GHz com tecnologia Hyper-Threading
- 512 MB de memória RAM (dual channel)
- Placa Main-Board MSI Neo FISR2
- Sistema Operacional Debian/Linux
- Kernel 2.6.14 SMP PREEMPT

Para aquisição de tempo de execução dos experimentos foi criado um software chamado *monitor*, onde são implementadas duas funções: (a) *inicia* onde é capturado o tempo inicial

e (b) *agora* onde é capturado o tempo no instante da sua execução. Subtraindo-se o tempo obtido com a chamada de *inicia* do tempo obtido com a chamada de *agora* obtém-se o tempo entre uma chamada e outra, possibilitando medir o tempo de execução de um processo entre as chamadas.

O sistema de medição foi implementado na linguagem de programação C++, utilizando a função

```
int gettimeofday(struct timeval *tv, struct timezone *tz);
```

responsável pela captura do tempo e possui a precisão de micro segundos.

Devido ao sistema operacional GNU/Linux ter uma arquitetura multitarefa e multi-usuários, adotou-se realizar 1000 execuções para obtenção do tempo médio com intervalo de confiança de 99%.

As influências externas oriundas do sistema operacional avaliadas são as seguintes: (a) execução sem alteração de prioridade de execução no sistema operacional e (b) execução com alteração da prioridade execução para -20 utilizando o comando *nice*.

As influências de parâmetros de compilação do G++ testadas são sem otimização e com otimização nível 3 (parâmetro -O3). Os parâmetros de otimização presentes no G++ são capazes de instruir o compilador para criação de códigos de máquinas mais eficientes, podendo alterar o tempo de execução do programa gerado. São verificadas as influências internas, que é a execução dos implementação com aritmética inteira e ponto flutuante.

É importante observar que o experimento inicia com a medição da influência do monitor dado que o mesmo inclui chamadas próprias, aumentando o tempo de execução real do método. Para esta medição, adotou-se realizar 1000 execuções em aberto (chamada da função *inicia* imediatamente seguida da função *agora*).

6.3.1 Tempo de execução do monitor em aberto

Foi realizada a execução do monitor em aberto, ou seja, a chamada da função *agora* imediatamente seguida da chamada da função *inicia*. O objetivo é investigar a influência do software de monitoração do tempo de execução.

O mesmo processo foi executado em quatro maneiras diferentes: (a) com prioridade e com otimização, (b) com prioridade e sem otimização, (c) sem prioridade e com otimização e (d) sem prioridade e sem otimização.

Na Tabela 6.3, são apresentados o tempo médio, desvio padrão e intervalo de confiança das 1000 execuções do processo. A Figura 6.3 ilustra os gráficos dos tempos das 1000 execuções. Nesta figura, podem ser observados picos, os quais resultam da perda do privilégio da execução no processador.

Tabela 6.3: Tempos de execução do monitor em aberto

Execução	Média (μs)	Desvio Padrão	Intervalo de confiança 99%
a	3.076	0.264998	[3.06762,3.08438]
b	3.164	2.026600	[3.09991,3.22809]
c	4.308	1.010510	[4.27604,4.33996]
d	5.961	1.064650	[5.92733,5.99467]

Figura 6.3: Tempos de execução do monitor em aberto

Através destes experimentos, com o monitor em aberto, podemos concluir que a influência do monitor é muito pequena. Sua ordem de magnitude está em 10^{-6} e esta influência é desprezível para os processos seguintes, que possuirão ordem de grandeza de 10^{-3} .

6.3.2 Tempo de execução da redução usando aritmética de ponto flutuante

Para redução dos nível de quantização de cada canal, foi utilizada a seguinte equação:

$$qn_c = \frac{qo_c}{2^{bo-bn}} \quad (6.1)$$

onde qn_c é o novo valor quantizado para o canal c , qo_c é a quantização original, bo o número de bits usados originalmente e bn o novo número de bits desejados.

Esta equação pode ser computada utilizando aritmética de ponto flutuante ou inteira. Utilizando a aritmética inteira temos o problema da precisão, porém utilizando a aritmética de ponto flutuante temos uma redução de desempenho significativa.

O experimento detalhado aqui será utilizado para comparar, em termos de tempo, a execução do processo de redução da quantização utilizando aritmética de ponto flutuante com aritmética inteira. O processo, assim como no experimento anterior, foi executado em quatro maneiras diferentes: (a) com prioridade e com otimização, (b) com prioridade e sem otimização, (c) sem prioridade e com otimização e (d) sem prioridade e sem otimização.

Na Tabela 6.4, são exibidos o tempo médio, desvio padrão e intervalo de confiança das 1000 execuções do processo. A Figura 6.4 ilustra os gráficos dos tempos das 1000 execuções.

Tabela 6.4: Tempos de execução da redução utilizando aritmética de ponto flutuante

Execução	Média (μs)	Desvio Padrão	Intervalo de confiança 99%
a	9605.12	1229.56	[9566.24,9644]
b	15435.5	1863.62	[15376.6,15494.4]
c	11667.7	4141.56	[11536.7,11798.7]
d	16123.6	1989.86	[16060.6,16186.5]

Com a obtenção destes resultados, pode ser realizada a execução do mesmo processo, porém com aritmética inteira.

6.3.3 Tempo de execução da redução usando aritmética inteira

Na medição do tempo de execução utilizando aritmética inteira, assim como as duas anteriores, foram feitas quatro execuções de maneiras diferentes: (a) com prioridade e com

Figura 6.4: Tempos de execução com aritmética de ponto flutuante

otimização, (b) com prioridade e sem otimização, (c) sem prioridade e com otimização e (d) sem prioridade e sem otimização.

Na Tabela 6.5, são apresentados o tempo médio, desvio padrão e intervalo de confiança das 1000 execuções do processo. A Figura 6.5 ilustra os gráficos dos tempos das 1000 execuções.

Tabela 6.5: Tempos de execução do monitor em aberto

Execução	Média (μs)	Desvio Padrão	Intervalo de confiança 99%
a	2397.41	488.086	[2381.98,2412.85]
b	17913.7	1667.05	[17860.9,17966.4]
c	2479.83	577.818	[2461.56,2498.1]
d	18033.2	1564.27	[17983.8,18082.7]

Com este experimento podemos concluir que o parâmetro de otimização `-O3` do compilador G++ consegue melhoras expressivas na computação da redução da quantização utilizando aritmética inteira, onde neste caso o tempo médio baixou de 17,9 milisegundos para 2,4 milisegundos. O parâmetro externo de prioridade do processo não trouxe melhoras significativas. Entretanto, tendo essa possibilidade a disposição, é interessante a utilização do mesmo, pois ele ganhará prioridade de execução.

Figura 6.5: Tempos de execução com aritmética inteira

A redução é considerada significativa quanto a utilização de aritmética de ponto flutuante e inteira pois o processo reduz de 9,6 (ponto flutuante) para 2,4 milisegundos (inteira).

Na Figura 6.6, é demonstrado o gráfico comparativo entre os modos execuções a, b, c e d com a variação da utilização aritmética de ponto flutuante e inteira.

Figura 6.6: Comparativo entre a aritmética ponto flutuante e inteira

Outra conclusão obtida é que a aplicação da redução de quantização introduz um pequeno aumento no processamento, porém traz as vantagens já citadas anteriormente. A próxima seção traz o experimento utilizando a classificação dos pixels de uma imagem utilizando um mapa auto-organizável.

6.4 Experimento: Classificação dos pixels utilizando um Mapa Auto-organizável

Neste experimento foi utilizado um mapa auto-organizável unidimensional com 25 neurônios para classificar uma imagem de futebol de robôs MIROSOT. O objetivo é classificar: (1) bola, que possui tons de laranja e (2) robôs, que possuem tons de azul e rosa.

Foi escolhida a implementação unidimensional por possibilitar qualquer número de neurônios sem a restrição de potências de 2 imposta pela implementação bidimensional.

A imagem é obtida de uma cena real através de uma placa de captura. Esta imagem utiliza como espaço de cores o espaço RGB, e é diretamente utilizada no treinamento da rede neural. Para reduzir os exemplos de treinamento é utilizada uma redução de quantização para 5 bits.

Foram utilizadas diversas variações dos parâmetros das equações de aprendizado e distância, porém empiricamente foram encontradas as equações (6.2) e (6.3) para as equações de aprendizado e distância, respectivamente.

$$\mu(t) = 0.4e^{\left(\frac{t}{2(10)^2}\right)} \quad (6.2)$$

$$h(\mathbf{x}), i = 0.1e^{\left(\frac{|\mathbf{n}_c - \mathbf{n}_i|}{2^2}\right)} \quad (6.3)$$

Na Figura 6.7 pode ser visto o resultado da classificação utilizando o mapa auto-organizável. Foram testados mapas auto-organizáveis contendo 5, 10, 15, 20, 25 e 30 neurônios. Observou-se que com 25 neurônios, o mapa auto-organizável produz resultados satisfatórios.

Durante a execução deste experimento foi levantada a dúvida se a rede neural seria capaz de classificar corretamente tons amarelos e laranjas, por serem próximos, então foi realizado um experimento com outra imagem. Este experimento é ilustrado na Figura 6.8 e mostra a correta separação entre laranja e amarelo.

Pela imagem fica claro que o mapa auto-organizável foi capaz de separar os rótulos dos robôs e bolas, satisfazendo a necessidade da aplicação.

Figura 6.7: Resultado da classificação através de um mapa auto-organizável com 25 neurônios

Figura 6.8: Resultado da classificação através de um mapa auto-organizável com 25 neurônios

Além de classificação, foi feito um experimento para redução de quantização adaptativa utilizando este mesmo esquema de rede neural. Na próxima seção, é descrito este experimento.

6.5 Experimento: Redução da quantização através de Mapas Auto-organizáveis

Neste experimento, é realizada a redução da quantização de maneira adaptativa, de uma maneira que os neurônios se adaptem aos pixels de entrada.

São testados os seguintes números de neurônios: 32, 64, 128 e 256. Também são utilizadas as equações (2.7) e (2.8) para obtenção de uma medida de degradação da imagem.

Numericamente, podemos ver que os efeitos da quantização na Tabela 6.6, onde é demonstrado o número de neurônios, a degradação e o tempo de treinamento.

Tabela 6.6: Degradação e tempo na redução da quantização

Número de neurônio	Degradação	Tempo de treinamento
32	878.351	34 minutos
64	433.928	57 minutos
128	227.238	1 hora e 34 minutos
256	142.844	3 horas e 16 minutos

O efeito visual é ilustrado na Figura 6.9.

Como podemos ver a métrica de degradação demonstra que a redução na qualidade não é muito grande, sendo possível utilizar um mapa auto-organizável para reduzir a quantização da imagem. Entretanto, o tempo necessário de processamento é inviável para aplicações reais.

Na tentativa de melhorar o tempo de processamento, foi aplicada a quantização uniforme reduzindo para 5 bits para reduzir os padrões de entrada de 55.959 pixels para 9.374 pixels. Utilizando este artifício, o tempo de processamento caiu consideravelmente, porém a qualidade visual das imagens piorou e a medida de degradação apresentou aumento, como era de se esperar. Os resultados numéricos podem ser verificados na Tabela 6.7 e as imagens pode ser verificadas na Figura 6.10.

Tabela 6.7: Degradação e tempo na redução da quantização

Número de neurônio	Degradação	Tempo de treinamento
32	1938.804	3 minutos
64	787.265	5 minutos
128	428.185	8 minutos
256	265.54	17 minutos

Figura 6.9: Efeito da redução da quantização utilizando mapa auto-organizável com diferentes números de neurônios

Figura 6.10: Efeito da redução da quantização utilizando quantização uniforme e mapa auto-organizável

O próximo experimento faz uma avaliação do desempenho dos classificadores implementados.

6.6 Desempenho da classificação

Foi desenvolvida uma interface gráfica que permite um usuário classificar imagens manualmente para medir o desempenho da classificação. Nesta interface, é possível carregar uma imagem e atribuir a classe correta para cada pixel da imagem. Após classificar todos os pixels, o usuário gera um mapa de coordenadas e suas respectivas classes. A imagem que foi utilizada para classificação manual é submetida ao classificador para gerar um mapa de coordenadas com suas respectivas classes. Através da comparação dos mapas gerados pelo processo manual e pelo classificador, é possível determinar o total de classificações corretas e incorretas.

Os dois classificadores implementados neste trabalho foram submetidos a este teste para determinação de seus desempenhos. A imagem 6.8(a) foi utilizada neste experimento. A Tabela 6.8 demonstra os valores de classificações corretas e incorretas (falso positivo e falso negativo) para a classificação através do mapa auto organizável.

Tabela 6.8: Classificador por mapa auto organizável

Classe	Correto	Incorreto	Falso +	Falso -
0	305956	251	66	185
1	107	20	2	18
2	416	89	35	54
3	154	56	54	2
4	141	26	24	2
5	155	100	90	10
Total	306929	542	271	271

Na tabela 6.9 é possível verificar valores de classificações corretas e incorretas (falso positivo e falso negativo) para a classificação através de limiarização no espaço HSV.

Tabela 6.9: Classificador por limiarização no HSV

Classe	Correto	Incorreto	Falso +	Falso -
0	305843	420	122	298
1	97	58	30	28
2	416	100	46	54
3	142	173	172	1
4	132	57	24	33
5	115	102	61	41
Total	306745	910	455	455

Através das Tabelas 6.8 e 6.9, é possível verificar que o classificador baseado em mapa auto organizável possui um desempenho levemente superior. Entretanto, este desempenho nem sempre é superior, pois o usuário pode configurar o classificador HSV com melhor precisão, dependendo da imagem de entrada.

O próximo experimento demonstra a aplicação do método de detecção de movimento.

6.7 Experimento: Detecção de movimento

Neste experimento é avaliada a técnica descrita no capítulo 2. Além de avaliar o aspecto qualitativo, pretende-se medir o tempo necessário para a execução do método.

Para execução deste experimento, foi implementada a técnica utilizando a linguagem de programação C++, o compilador GNU G++ 4.1 com flags de otimização nível 3 e flag de arquitetura Pentium 4 (g++ -O3 -march=pentium4).

O computador no qual foi executado o experimento possuía um processador Pentium 4 de 2.8 GHz com 1 Gb de memória RAM, rodando o sistema operacional Debian GNU/Linux com kernel 2.6.17-1.

A resolução das imagens de vídeo utilizada para os testes foi de 480x360 pixels. Foi escolhido um trecho do filme Star Wars III, onde o fundo pode ser considerado menos dinâmico.

O tempo médio de processamento de um frame quando não existe movimento (atualização de quadro) é de $8065 \pm 7\mu s$, considerando intervalo de confiança de 99%. Já o tempo médio de processamento de um frame quando existe movimento é de $15782 \pm 78\mu s$ com intervalo de confiança de 99%.

Na Figura 6.11, é demonstrado o resultado sobre a aplicação do trecho do filme.

Figura 6.11: Detecção de movimento em um trecho do filme Star Wars III

6.8 Detecção de bola e robôs

Este experimento demonstra o resultado da detecção dos objetos de uma partida de futebol de robôs. A Figura 6.12(a) exibe a imagem de entrada e o resultado da detecção da imagem de entrada é mostrado na Figura 6.12(b).

A imagem de entrada passa por todos os processamentos descritos no capítulo 4 resultando na detecção dos objetos. Observando a Figura 6.12(a), podemos ver o campo com fundo próxima a cor preta, a bola com a cor vermelha e robôs. Os robôs do time adversário possuem um rótulo colorido em sua tampa superior. Os rótulos dos robôs adversários são coloridos com a cor azul e três outras cores. Os robôs do nosso time são coloridos com a cor amarela e três outras cores: verde, rosa e ciano. As cores verde, rosa e ciano são utilizadas como um identificador único para cada robô.

Na Figura 6.12(b), é demonstrado o resultado da detecção, onde os objetos são sobrepostos por um triângulo colocado no centro do objeto detectado. Os triângulos possuem cores diferentes para identificar a detecção correta. A bola é identificada pelo triângulo laranja, os robôs adversários por um triângulo vermelho e os robôs do nosso time por um triângulo verde. É interessante destacar que os triângulos que identificam os robôs possuem uma linha que aponta para a frente do robô, demonstrando a correta detecção do ângulo de cada robô.

Este experimento deixa claro o pleno funcionamento do sistema desenvolvido, no qual foi observado que, mesmo em diferentes tipos de iluminação e sobre variações não bruscas da iluminação, o sistema funciona adequadamente.

(a) Imagem de entrada

(b) Objetos detectados

Figura 6.12: Detecção dos objetos de uma partida de futebol de robôs

6.9 Considerações finais

Neste trabalho, foram realizados diversos experimentos para testar as técnicas que foram implementadas no desenvolvimento do sistema de visão.

O sistema desenvolvido foi útil para teste do funcionamento do classificador em aplicações de tempo real e gerou um software para uma aplicação real que pode ser utilizado como base para futuras pesquisas envolvendo robótica.

No próximo capítulo, a conclusão do trabalho realizado e propostas de trabalhos futuros são apresentados.

Conclusão

Neste trabalho foi desenvolvido um sistema de visão computacional para futebol de robôs da categoria MIROSOT. Dois métodos de classificação de cores foram estudados e implementados. O primeiro é baseado em aplicação de limiarização no espaço HSV e o segundo, consiste na utilização de um mapa auto-organizável, SOM, treinado a partir de pixels RGB.

Uma técnica de quantização uniforme da imagem de entrada foi proposta para redução da variedade de pixels RGB. Tal técnica possibilitou a aceleração de alguns passos envolvidos na classificação das cores. Esta técnica permite a criação de uma tabela de otimização que facilita o processo.

Em (Penharbel et al., 2004) foi desenvolvido um sistema de futebol de robôs que emprega as mesmas características do classificador HSV. Entretanto, são utilizadas imagens com 8 bits de quantização, implicando em uma tabela de 16Mb de memória, impossibilitando a aplicação em dispositivos com menos capacidade de armazenamento.

No presente trabalho, foi explorado o conceito de quantização para que a tabela de otimização ocupe menos espaço, e para que a sua criação fosse feita de forma mais rápida.

Foram realizados testes e foi utilizada a quantização de 5 bits como a quantização padrão de funcionamento do sistema de visão computacional proposto. Adotando-se esta, foram necessários somente 32K de memória para o armazenamento da tabela.

Foi provado que a redução de quantização não afeta significativamente a qualidade de imagem para prejudicar o funcionamento adequado do sistema, porém são introduzidas três divisões por ponto processado, uma operação que pode ser custosa em alguns tipos de processadores. Para a verificação da influência da inserção das divisões na família de

processadores Pentium IV, foi realizado um experimento que provou que todas as divisões de um total de 921.600 (uma imagem RGB com resolução de 640x480) podem ser efetuadas em menos de 3 milissegundos.

Foi realizado também neste trabalho o estudo dos mapas auto-organizáveis para construir um classificador capaz de aprender as cores com a(s) imagem(s) de entrada, uma novidade que pode tornar sistemas que utilizam classificação de cores mais simples de serem ajustados. Foi realizado um experimento para verificação da capacidade de classificação deste modelo de rede neural no futebol de robôs. Este experimento demonstrou que é possível classificar corretamente as cores utilizadas e também utilizar mapas auto-organizáveis para separar corretamente a cor laranja da cor amarelo.

Vale ressaltar que a tabela de otimização se tornou fundamental na utilização dos mapas auto-organizáveis para eliminar as equações que fazem a classificação de cada ponto. A redução de quantização foi essencial em dois aspectos: redução do espaço de entrada e diminuição do espaço necessário para armazenamento da tabela de otimização.

Outro estudo realizado foi um método de detecção de objetos em movimento, um método capaz de adaptar-se ao meio em que está operando, uma característica atrativa em ambientes onde a imagem vai alterando com o decorrer do tempo, ou até mesmo onde a câmera sofre algum impacto, fazendo com que um simples método de subtração não funcione adequadamente.

Uma outra contribuição importante do trabalho foi a construção de um time de futebol de robôs, constituído de 1 protótipo e 4 robôs, o qual poderá ser utilizado em competições nacionais de futebol de robôs. Este ambiente fornece um ambiente muito rico de temas de pesquisas a serem investigadas.

O software de visão computacional implementado neste trabalho é baseado em uma idéia que possibilita que novos classificadores sejam adicionados. Essa característica torna possível que outros alunos que desejam trabalhar com classificação de cores para futebol de robôs possam utilizar seus métodos sem ter que implementar um sistema inteiro de visão computacional. É interessante ressaltar que o software desenvolvido pode ser utilizado em outras aplicações similares envolvendo visão e robótica.

O desenvolvimento deste trabalho possibilitou a publicação de dois artigos (Penharbel et al., 2006) e (Romero et al., 2006). Também foi possível participar da competição de futebol de robôs, realizada em Florianópolis durante o Simpósio Brasileiro de Automação Inteligente, SBAI, em outubro de 2007.

Como direções futuras deste trabalho é interessante realizar experimentos com a utilização de mapas auto-organizáveis em diversos espaços de cores e até mesmo utilizando múltiplos espaços de cores para classificação de imagens.

Também é interessante fazer a integração do software de detecção de movimento ao sistema de visão computacional de futebol de robôs para eliminar o processamento dos

pontos estáticos. Um outro ponto a ser investigado seria a aplicação de alguma técnica de filtragem nas coordenadas e ângulos dos objetos detectados para tornar os resultados menos suscetíveis a ruídos.

Uma grande contribuição futura, na classificação de cores, seria a aplicação de outros modelos de redes neurais auto-organizáveis como *Growing Neural Gas* (Fritzke, 1995) e outros modelos construtivos de redes neurais.

Referências Bibliográficas

- BALLARD, D. H.; BROWN, C. M. *Computer vision*. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc., 1982.
- BORSATO, F. H.; FLORES, F. C. A real time method to object detection and tracking applied to robot-soccer. *IEEE Conference on Cybernetics and Intelligent Systems, 2004*, p. 174–178, 2004.
- BROWNING, B.; VELOSO, M. Real-time, adaptive color-based robot vision. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, p. 3871–3876, 2005.
Disponível em <http://ieeexplore.ieee.org/iel5/10375/32977/01545424.pdf?tp=&isnumber=32977&arnumber=1545424&punumber=10375>
- BRUCE, J.; BALCH, T.; VELOSO, M. Fast and inexpensive color image segmentation for interactive robots. In: *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, 2000, p. 2061 – 2066.
- CAO, A. W.; JUN, Z.; LI, J. B. Recognizing robot-soccer with half-chord method. *IEEE International Conference on Control and Automation, 2007. ICCA 2007*, p. 2097–2100, 2007.
Disponível em http://www.ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=4376307&arnumber=4376730&count=660&index=422
- CELIK, T.; DEMIREL, H.; OZKARAMANLI, H.; UYGUROGLU, M. Fire detection using statistical color model in video sequences. *J. Vis. Comun. Image Represent.*, v. 18, n. 2, p. 176–185, 2007.
- CHUNG, F.; FUNG, BENNY, Y. M. Fuzzy color quantization and its application to scene change detection. In: *MIR '03: Proceedings of the 5th ACM SIGMM international*

- workshop on Multimedia information retrieval*, New York, NY, USA: ACM, 2003, p. 157–162.
- COSTA, ANNA, H. R.; PEGORARO, R. Construindo robôs autônomos para partidas de futebol: o time guaraná. In: *SBA Controle & Automação*, 2000.
- CROWDER, R. M. *Electric drives and their controls*. Oxford, UK: Oxford University Press, 1998.
- ETIENNE-CUMMINGS, R.; POULIQUEN, P.; LEWIS, M. A. A vision chip for color segmentation and pattern matching. *EURASIP J. Appl. Signal Process.*, v. 2003, n. 1, p. 703–712, 2003.
- FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, v. 7, p. 179–188, 1936.
- FORSYTH, D. A.; PONCE, J. *Computer vision: A modern approach*. Prentice Hall, 2002.
- FRITZKE, B. A growing neural gas network learns topologies. In: TESAURO, G.; TOURETZKY, D. S.; LEEN, T. K., eds. *Advances in Neural Information Processing Systems 7*, Cambridge MA: MIT Press, p. 625–632, 1995.
- GEJGUŠ, P.; ŠPERKA, M. Face tracking in color video sequences. In: *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, New York, NY, USA: ACM Press, 2003, p. 245–249.
- GONZALEZ, R. C.; WOODS, R. E. *Digital image processing*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.
- GONÇAVELS, B. H. *Controle de sistemas mecânicos por lógica fuzzy*. Relatório Técnico, submetido para ser publicado como RT/ICMC/USP, 2008.
- HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. *Neural network design*. Boston, MA, USA: PWS Publishing Co., 1996.
- HAYKIN, S. *Neural networks: a comprehensive foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.
- HECKBERT, P. Color image quantization for frame buffer display. In: *SIGGRAPH '82: Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, New York, NY, USA: ACM, 1982, p. 297–307.
- JOBLOVE, G. H.; GREENBERG, D. Color spaces for computer graphics. *SIGGRAPH Comput. Graph.*, v. 12, n. 3, p. 20–25, 1978.

- KOHONEN, T. *Self-organizing maps*. Berlin, Heidelberg: Springer, 1995.
- KOHONEN, T. *Self-organizing maps*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.
- KOHONEN, T. The self-organizing map (som). 2005.
Disponível em <http://www.cis.hut.fi/projects/somtoolbox/theory/somalgorithm.shtml> (Acessado em 15 de janeiro de 2008)
- LEE, K.-M.; LI, Q.; DALEY, W. Effects of classification methods on color-based feature detection with food processing applications. In: *IEEE Transactions on Automation and Engineering*, 2007, p. 40–51.
- LIU, K.; LIU, P. Color model based 3-d self-organizing map. In: *IV '04: Proceedings of the Information Visualisation, Eighth International Conference on (IV'04)*, Washington, DC, USA: IEEE Computer Society, 2004, p. 403–408.
- MARTINEZ-GOMEZ, L. A.; WEITZENFELD, A. Real time vision system for a small size league team. In: *1st Latin American Robotics Symposium*, 2004.
- MCKERROW, P. *Introduction to robotics*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1991.
- MINSKY, M.; PAPERT, S. *Perceptrons: An introduction to computational geometry*. Cambridge, Mass.: MIT Press, 1969.
- OLIVEIRA, P. R. *Redes neurais para extração de características*. Dissertação de Mestrado, Universidade de São Paulo, 1997.
- PAGELLO, E.; ARAI, T.; DILLMANN, R.; STENTZ, A. Towards the intelligent autonomous systems of the third millennium. *Robotics and Autonomous Systems*, v. 40, n. 2-3, p. 63–68, 2002.
- PENHARBEL, ÉDER, A.; DESTRO, RICARDO, C.; TONIDANDEL, F.; BIANCHI, REINALDO, A.-C. Filtro de image baseado em matriz rgb de cores-padrão para futebol de robôs. In: *Anais do XXIV Congresso da Sociedade Brasileira de Computação*, 2004.
- PENHARBEL, ÉDER, A.; MELO, DIOGO, O.; ROMERO, ROSELI, A.-F. Detecção de objetos em movimento sobre um fundo constante em tempo-real. *II Workshop de Visão Computacional*, 2006.
- PLYMOUTH Robot soccer team of university plymouth team. 2006.
Disponível em <http://www.tech.plym.ac.uk/robofoot/> (Acessado em 15 de janeiro de 2006)

- QUILES, M. G. *Sistema de visão computacional baseado em redes neurais para o controle de robôs móveis*. Dissertação de Mestrado, Universidade de São Paulo, 2004.
- RECCE, M.; PLEBE, A.; TROPANO, G.; TAYLOR, J. Video grading of oranges in real-time. *Artif. Intell. Rev.*, v. 12, n. 1-3, p. 117–136, 1998.
- REITER, C. With j: image processing 2: color spaces. *SIGAPL APL Quote Quad*, v. 34, n. 3, p. 3–12, 2004.
- ROMERO, ROSELI, A.-F.; GEDSON, F.; PENHARBEL, ÉDER, A.; SANTOS, DAVI, P. Um sistema integrado para controle de um time de futebol de robôs. *ENRI 2006 / EnRI Painel: Competições de Robótica*, 2006.
- SHARMA, G.; TRUSSELL, H. J. Digital color imaging. *IEEE Transactions on Image Processing*, v. 6, n. 7, p. 901–932, 1997.
Disponível em citeseer.ist.psu.edu/sharma97digital.html
- SIMÕES, A. S. *Aprendizado não-supervisionado em redes neurais pulsadas de base radial*. Tese de Doutorado, Universidade de São Paulo, 2006.
- SONKA, M.; HLAVAC, V.; BOYLE, R. *Image processing, analysis, and machine vision*. Thomson-Engineering, 2007.
- STOCKMAN, G.; SHAPIRO, L. G. *Computer vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.
- VALENTINE, R. *Motor control electronics handbook*. McGraw-Hill Professional, 1998.
- VANDENBROUCKE, N.; MACAIRE, L.; POSTAIRE, J.-G. Color image segmentation by pixel classification in an adapted hybrid color space: application to soccer image analysis. *Comput. Vis. Image Underst.*, v. 90, n. 2, p. 190–216, 2003.
- WANG, Y.-F.; CHANG, E. Y.; CHENG, K. P. A video analysis framework for soft biometry security surveillance. In: *VSSN '05: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, New York, NY, USA: ACM Press, 2005, p. 71–78.
- WEISS, N.; HILDEBRAND, L. An exemplary robot soccer vision system. In: *CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby*, 2004.
- WOLBERG, G. *Digital image warping*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994.
- WREN, C. R.; AZARBAYEJANI, A.; DARRELL, T.; PENTLAND, A. P. Pfunder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 19, n. 7, p. 780–785, 1997.

- WU, W.; CHEN, X.; YANG, J. Incremental detection of text on road signs from video with application to a driving assistant system. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, New York, NY, USA: ACM Press, 2004, p. 852–859.
- WU, Y.; HUANG, T. Nonstationary color tracking for vision-based human-computer. In: *IEEE Transactions on Neural Networks*, 2002, p. 948– 960.
- YANG, T.; LI, S. Z.; PAN, Q.; LI, J. Real-time and accurate segmentation of moving objects in dynamic scene. In: *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, New York, NY, USA: ACM Press, 2004, p. 136–143.