UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

SISTEMA ÓPTICO PARA IDENTIFICAÇÃO DE VEÍCULOS EM ESTRADAS

DANIEL DOS SANTOS

BLUMENAU
2008

2008/2-06

DANIEL DOS SANTOS

SISTEMA ÓPTICO PARA IDENTIFICAÇÃO DE VEÍCULOS EM ESTRADAS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, M. Sc. - Orientador

BLUMENAU
2008

2008/2-06

SISTEMA ÓPTICO PARA IDENTIFICAÇÃO DE VEÍCULOS EM ESTRADAS

Por

DANIEL DOS SANTOS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, Titulação – Orientador, FURB
Membro:	Prof. Paulo César Rodacki Gomes – FURB
Membro:	Prof. Mauro Marcelo Mattos – FURB

Blumenau, 17 de dezembro de 2008

Dedico este trabalho a minha família, pelo apoio recebido durante os anos de graduação.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

Aos meus amigos, pela compreensão quanto às horas ausente.

Ao Alexandre Heberle, pelos equipamentos fornecidos e pela ajuda na coleta de materiais necessários para o desenvolvimento deste trabalho.

Ao professor Dalton Solano dos Reis, pela orientação e apoio no desenvolvimento deste trabalho.

Ao Maurício Edgar Stivanello, Michael Bove Jr e Simon Denman, pelas dúvidas esclarecidas e a ajuda oferecida para o desenvolvimento deste trabalho.

Por mais que o preguiçoso deseje alguma coisa, ele não conseguirá, mas a pessoa esforçada consegue o seu desejo.

Provérbios 13:4

RESUMO

Este trabalho apresenta um sistema de monitoramento para contagem de veículos, utilizando visão computacional. Estudos mostram que pessoas que trabalham em sistemas de monitoramento têm dificuldades de atenção, o que contribui na utilização de visão computacional nesta área. Para verificar a viabilidade da utilização de algoritmos de segmentação adaptativa de background/foreground, o sistema permite comparar o algoritmo NHD com o algoritmo de subtração de fundo. Além de outras técnicas de processamento de imagens, foram utilizados os descritores de Fourier, importante abordagem para a descrição de formas a partir da borda. Para interpretação das representações de veículos foi utilizada uma rede neural artificial do tipo Perceptron Multicamandas, comum em sistemas de reconhecimento de padrões. A combinação dessas técnicas possibilitou o desenvolvimento de um sistema para contagem de veículos, independente de cenário e luminosidade.

Palavras-chave: Segmentação por subtração de fundo. Algoritmo NHD. Descritores de Fourier. Redes neurais artificiais. Perceptron multicamadas. Visão computacional. Contagem de veículos.

ABSTRACT

This work presents a monitoring system for counting of vehicles, using computer vision. Studies show that people working on monitoring systems have difficulties with attention, which helps in the use of computational vision in this area. To verify the feasibility of using algorithms adaptive Foreground/Background segmentation, the system compares the algorithm NHD with the algorithm background subtraction. The software uses Fourier descriptors in order to describe the edge of products shapes, amonget other image processing techniques such as segmentation. For the interpretation of the representations of the vehicles, a neural network of the type Multilayer Perceptron was implement. The combination of these techniques made possible the development of a software for counting of vehicles, regardless of background and brightness.

Key-words: Background subtraction. NHD algorithm. Fourier descriptors. Artificial neural networks. Multilayer perceptron. Computational vision. Counting of vehicles.

LISTA DE ILUSTRAÇÕES

Figura 1 – Seqüência de imagens com identificação de objetos no trânsito 14
Figura 2 – Exemplo de aplicação de sistema de análise de vídeo inteligente 18
Figura 4 – Modelo de cor proposto no espaço tridimensional RGB 23
Figura 5 – Exemplo de resultado de subtração de fundo 24
Figura 6 – Segmentação utilizando algoritmo GMM 25
Figura 7 – Segmentação utilizando algoritmo NHD 27
Figura 8 – Segmentação de imagem através de limiar 28
Figura 9 – Aplicação do operador Sobel sobre a imagem 29
Figura 11 – Código de cadeia 30
Figura 12 – Forma original e reconstruções a partir de *M* coeficientes 32
Figura 13 – Camadas da Rede Neural 33
Figura 14 – Resultado do algoritmo de segmentação 35
Figura 15 – Exemplo de produto válido e inválido 35
Figura 16 – Diagrama de caso de uso 37
Figura 17 – Diagrama de classe 38
Figura 18 – Diagrama de seqüência configurar ambiente 44
Quadro 1 – Parte do algoritmo de Subtração de fundo 47
Quadro 2 – Método de operações de morfologia matemática 48
Quadro 3 – Método de desenho de área de contorno 48
Figura 19 – Fluxograma do Algoritmo NHD 49
Quadro 4 – Rotina de busca pelo Matching Cluster 50
Figura 20 – Tela de configuração de ambiente, através do algoritmo NHD 51
Figura 21 – Tela de comparação de algoritmos de segmentação 52
Figura 22 – Teste adaptação de fundo NHD 55
Figura 23 – Objeto entrando em movimento 56
Figura 24 – Adaptação de alteração de luminosidade 56
Figura 25 – Exemplos de objetos reprovados pela subtração de fundo 57
Figura 26 – Exemplos de objetos aprovados pela subtração de fundo 58
Figura 27 – Exemplos de objetos reprovados pelo algoritmo NHD 59
Figura 28 – Exemplos de objetos aprovados pelo algoritmo NHD 59

LISTA DE TABELAS

Tabela 1 – Resultados dos testes com algoritmos de Segmentação 55
Tabela 2 – Resultados dos casos de teste com rede neural 60

LISTA DE SIGLAS

CFTV – Circuito Fechado de Televisão

CRT – Tubo de Raios Catódicos

DVR – *Digital Video Rercorder*

GMM – *Gaussian Mixture Models*

OpenCV – *Open Source Computer Vision Library*

RF – Requisito Funcional

RNF – Requisito Não Funcional

RGB – Acrônimo de vermelho (*Red*), verde (*Green*) e azul (*Blue*).

SMS – *Short Message Service*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS DO TRABALHO 14
1.2 ESTRUTURA DO TRABALHO 14
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 SISTEMAS DE MONITORAMENTO 16
2.1.1 SISTEMAS DE ANÁLISE DE VÍDEO INTELIGENTE 17
2.2 VISÃO COMPUTACIONAL 18
2.2.1 Processo Informativo 19
2.2.2 Detecção 20
2.2.3 Realce e Restauração 20
2.3 PROCESSAMENTO DE IMAGENS DIGITAIS 21
2.3.1 Detecção de Elementos em Movimento 22
2.3.1.1 Remoção de Fundo 22
2.3.1.2 Algoritmo de Segmentação Adaptativa 24
2.3.2 Segmentação de Imagens 27
2.3.2.1 Limiarização 27
2.3.2.2 Detecção de Bordas 28
2.3.3 Descritores de Imagens 29
2.3.3.1 Descritores de Fourier 30
2.3.4 Interpretação de Imagens 32
2.4 REDES NEURAIS ARTIFICIAIS 33
2.5 TRABALHOS CORRELATOS 34
3 DESENVOLVIMENTO 36
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 36
3.2 ESPECIFICAÇÃO 36
3.2.1 Diagrama de Caso de Uso 37
3.2.2 Diagrama de Classe 38
3.2.2.1 Classe `Ambient` 39
3.2.2.2 Classe `Processor` 39
3.2.2.3 Classe `Codec` 40
3.2.2.4 Classe `NHDCodec` 40

3.2.2.5 Classe `SubtractionCodec` 41
3.2.2.6 Classe `MorphologyCodec` 41
3.2.2.7 Classe `NeuronCodec` 41
3.2.2.8 Classe `ClassificationCodec` 42
3.2.2.9 Classe `Segment` 42
3.2.2.10 Classe `Functions` 43
3.2.3 Diagrama de Seqüência 44
3.3 IMPLEMENTAÇÃO 46
3.3.1 Técnicas e ferramentas utilizadas 46
3.3.1.1 OpenCV 46
3.3.1.2 Implementação do Algoritmo NHD 49
3.3.2 Operacionalidade da implementação 50
3.4 RESULTADOS E DISCUSSÃO 52
3.4.1 Condições da Implementação 53
3.4.2 Testes 54
3.4.2.1 Testes com Algoritmos de Segmentação 54
3.4.2.2 Teste com a Rede Neural 56
4 CONCLUSÕES 61
4.1 EXTENSÕES 62
REFERÊNCIAS BIBLIOGRÁFICAS 63

1 INTRODUÇÃO

Com o crescimento da frota de veículos nas cidades, houve um aumento no número de acidentes, causados pela falta de controle de tráfego nas estradas e pela imprudência dos motoristas. *Programa institucional* (2008) afirma que sistemas de monitoramento dão oportunidade de observar grandes áreas de forma organizada, aumentando significativamente a segurança da região.

Portanto, problemas como estes incorrem ações de inspeção, que requerem o deslocamento de equipes, bem como o uso de equipamentos específicos. Em ambientes onde a quantidade de veículos muda conforme o tempo, a automatização no controle de tráfego é necessária. Ribeiro e Lima (1999, p. 96) explicam que se for determinado um único tempo de atuação (como por exemplo a utilização de semáforos para controle de tráfego) a partir da necessidade média, ou por qualquer outro método, este fica claramente bem aquém da solução ideal.

A garantia de uma melhor e mais rápida solução para um determinado problema – a ocorrência de ações de inspeção – está nas mãos do operador de segurança que controla e monitora uma determinada via. Para manter-se sempre alerta, é necessário que o mesmo faça algumas pausas para descanso. Segundo Garcia (2001, p. 113), seres humanos não são bons detectores de eventos, principalmente durante longos períodos. Testes realizados com profissionais desta área mostraram que os mesmos não conseguem manter uma efetiva atenção após 30 minutos de observação.

Decorrente destes fatos, torna-se necessária uma ferramenta que possa identificar objetos ou interações com o ambiente. Onde seja possível utilizar apenas imagens de câmeras, *Central Processing Unit* (CPU) e software, em vez de sensores específicos fixados no ambiente. Permitindo assim o monitoramento de vários fenômenos em um único equipamento, sendo tarefa do software identificar a ocorrência ou não de eventos (COMPANHIA HIDRO ELÉTRICA DO SÃO FRANCISCO, 2004).

Uma solução seria a adoção de um programa de monitoramento com análise inteligente de imagens, contando com premissas específicas para o problema. Segundo Tafner (1996, p. 24), os sistemas especialistas, que procuram representar o conhecimento de especialistas, têm como objetivo deduzir alguns conhecimentos e regras no ambiente.

Fonte: OBJECT VIDEO (2008).

Figura 1 – Seqüência de imagens com identificação de objetos no trânsito

A figura 1 mostra um sistema especialista para monitoramento de trânsito, onde pessoas são contornadas por um retângulo vermelho e os carros são contornados por um retângulo azul. A imagem mostra um possível acidente, com uma pessoa que atravessou a rua fora da faixa e com um carro que para após o ocorrido.

Diante do exposto, propõe-se o desenvolvido de um sistema capaz de reconhecer veículos do tipo automóvel em vídeos capturados em estradas, disponibilizando uma ferramenta de contagem de veículos para cálculos estatísticos de tráfego.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um sistema capaz de reconhecer veículos do tipo automóvel, utilizando técnicas de visão computacional, para que seja possível programar um contador de veículos para controle de tráfego.

Os objetivos específicos do trabalho são:

a) utilizar técnicas de processamento de imagem com tratamento de luminosidade e foco, para remoção de ruído;
b) definir técnica para captura de contorno de borda para ser aplicada sobre os objetos encontrados no cenário, possibilitando a identificação dos veículos;
c) implementar um identificador de veículos utilizando uma rede neural artificial.

1.2 ESTRUTURA DO TRABALHO

Primeiramente realiza-se uma revisão bibliográfica dos diversos temas que fundamentam o desenvolvimento do sistema de software de identificação de veículos. Após

isso é demonstrada a especificação e são abordados detalhes da implementação do sistema. Finalmente são apresentadas as conclusões do presente trabalho, assim como sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os assuntos e técnicas utilizadas para o desenvolvimento do sistema óptico para identificação de veículos em estradas. Na seção 2.1 são apresentadas definições de sistemas de monitoramento, bem como a evolução desta área e a utilização de sistemas de análise de vídeos inteligente. Na seção 2.2 são apresentados conceitos e técnicas da visão computacional. Na seção 2.3 são apresentados os principais conceitos de processamento de imagens digitais, bem como técnicas de detecção de borda, descrição de borda e captura de objetos em movimento. Na seção 2.4 são apresentados os principais conceitos de redes neurais artificiais com sua estrutura e funcionamento básico.

2.1 SISTEMAS DE MONITORAMENTO

Os sistemas de monitoramento ganharam uma grande importância nos últimos 50 anos, pois ajudam na vigilância de áreas que dependem de acompanhamento, tais como bancos, cassinos, aeroportos, instalações militares e lojas de conveniência. Em sua versão mais simples estes sistemas são constituídos por câmera(s), meio de transmissão e monitor.

Tais sistemas são conhecidos como CFTV, que têm a função de distribuir sinais provenientes de câmeras localizadas em locais específicos, para um ou mais pontos de visualização como centrais de controle. Inicialmente utilizavam sinais analógicos e transmitiam as imagens das câmeras por meio de cabo coaxial para monitores CRT. Estes

Uma forma mais avançada de circuitos fechados de televisão utiliza gravadores de vídeo digital (DVR), permitindo gravar informações com uma variedade de opções de qualidade e desempenho. Tais sistemas oferecem funcionalidades adicionais, como resolução de detecção, e envio de alertas por e-mail e SMS.

Os sistemas de circuitos internos não são aplicados apenas com propósitos de segurança e vigilância, mas também em outras áreas sem utilizar a designação CFTV. Processos industriais que apresentam condições perigosas para os seres humanos são hoje muitas vezes supervisionados. Estes processos são principalmente na indústria química, o interior dos reatores ou de instalações de fabrico de combustível nuclear. Também é possível

encontrá-los em áreas de pesquisa e monitoramento de fauna e flora, monitoramento de condições climáticas e de relevo, bem como medicina.

Devido à sua larga possibilidade de utilização, o circuito interno acaba se tornando em um sistema promissor, com um amplo mercado. Conforme Portal da Automação (2004), as empresas que desenvolvem sistemas de segurança já vinham crescendo em média 15% ao ano e observa-se que este crescimento tende cada vez mais aumentar. A gestão de segurança por vídeo implica em gestão da exceção, dado o crescente número de câmeras e a redução de pessoal dedicado a monitorá-las.

2.1.1 SISTEMAS DE ANÁLISE DE VÍDEO INTELIGENTE

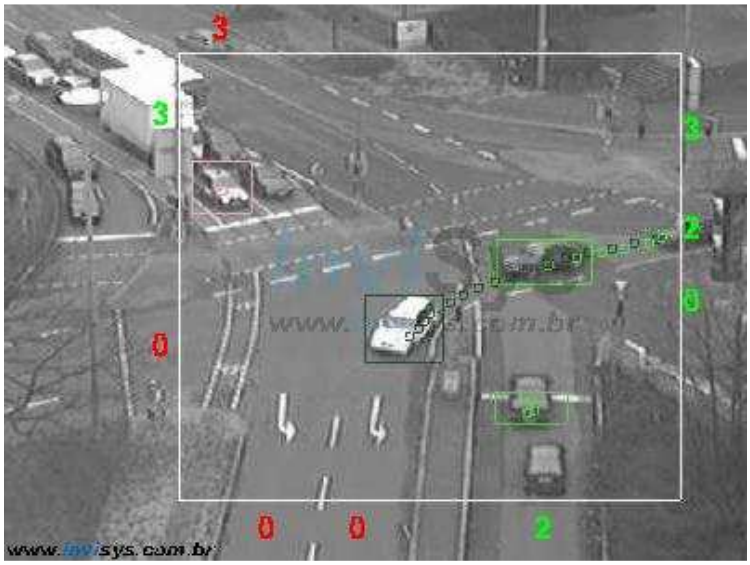
Sistemas de análise de vídeo inteligente estão sendo utilizados aos poucos nas organizações e cidades, com o objetivo de resolver problemas de segurança. Um dos fatores para tal preocupação com segurança deve-se aos roubos, infrações e ao terrorismo, principalmente em grandes cidades de países de primeiro mundo.

O principal problema dos sistemas de monitoramento inteligente é quanto ao desempenho. De acordo com Reis Junior e Soares Filho (2002, p. 17), "a entrada de dados não é interrompida em momento algum e nem existe muito tempo para que uma resposta seja retornada." Mesmo com a evolução dos computadores o problema de desempenho não se resolve, uma vez que as resoluções das imagens também aumentam, permitindo utilizar cenários maiores e mais complexos.

Em vários desses sistemas, como os que inspecionam o tráfego de veículos, a remoção de dados irrelevantes na imagem é fundamental. Estas aplicações de inspeção utilizam freqüentemente a tecnologia de visão computacional, para seguir padrões de tráfego de consumidores e quantificar o comportamento do consumidor de maneira automática. Através de algoritmos poderosos de processamento de vídeo digital, permitem detectar veículos em movimento no campo de visão da câmera, para verificar a direção tomada, velocidade e até mesmo a contagem dos mesmos.

Um exemplo de sistema de inspeção de tráfego de veículos é apresentado na figura 2, onde o retângulo no centro da imagem mostra o cenário inspecionado destacando os

automóveis em movimento, bem como sua trajetória. Os números na cor verde representam a contagem dos veículos que entraram no cenário inspecionado, já os vermelhos indicam os automóveis que saíram do mesmo.



Fonte: INVISYS (2008).

Figura 2 – Exemplo de aplicação de sistema de análise de vídeo inteligente

Outro exemplo de monitoramente inteligente são os de ambientes e de pessoas, que permitem fazer contagens, estimar o caminho completo percorrido por um ser humano ou objeto. Sistemas de controle de ambientes do tipo garagem, por exemplo, utilizam identificação de placas para automatização de estacionamentos e autenticação de usuários.

Portanto, a visão computacional atualmente procura não se limitar somente a armazenar imagens ou identificar pessoas, mas ela basicamente converte vídeo em dados úteis sem a necessidade da intervenção humana. Com o avanço das técnicas de processamento de imagem e o barateamento dos equipamentos, muitas tecnologias estão surgindo para ajudar as instituições à melhor zelarem pelos seus bens.

2.2 VISÃO COMPUTACIONAL

A visão é considerada pela Ciência um dos cinco sentidos que permitem aos seres vivos, dentre eles os seres humanos, a aprimorarem suas percepções do mundo. Segundo Azevedo, Conci e Leta (2008, p. 11), o aparelho sensorial da visão é responsável por, além de detectar luz e imagens, a função de interpretação ou percepção visual. Através desta

percepção é possível analisar e sintetizar informações recolhidas em termos de forma, cor, textura e relevo das imagens.

As imagens, por sua vez, são formadas a partir da quantidade de luz refletida ou emitida pelo objeto observado. Já a cor pode ser considerada a propriedade de um objeto ou fonte de luz relacionada não apenas às propriedades físicas do objeto ou fonte de luz, mas também às características do sistema visual do observador.

Para a extração de informações das imagens, bem como a identificação e classificação de objetos presentes nestas imagens, utiliza-se técnicas de Visão Computacional. Esta área vem se desenvolvendo muito nos últimos anos, com o objetivo de simular a visão humana, dando aos computadores a capacidade de interpretar o conteúdo de uma imagem.

Os sistemas de Visão Computacional envolvem Análise de Imagens e Inteligência Artificial (tomada de decisão). Além destas duas grandes áreas da computação, também se utiliza a mineração de dados ou imagens. Nas próximas seções serão apresentados conceitos de percepção e cognição, bem como técnicas de restauração e realce.

2.2.1 Processo Informativo

A cognição está em algum lugar entre áreas tradicionalmente chamadas de percepção e aprendizado, incorporando elementos de outras áreas. Segundo Azevedo, Conci e Leta (2008, p. 46), estes termos ou abordagens não podem ser considerados como áreas distintas de estudo. Portanto, a cognição é o ponto de ligação entre as técnicas de processamento de imagens e as de visão computacional.

A fase cognitiva ou de memorização é conhecida como processo informativo, que descreve o comportamento, assumindo que a maneira como o observador processa a informação inclui uma fase de registro ou sensorial. Exige o conhecimento do ambiente ou cenário, para monitorar objetos e suas iterações com os mesmos.

2.2.2 Detecção

A grande maioria dos problemas dos sistemas de visão computacional estão associados a detecção, decorrentes de mudanças de energia presente no ambiente, como estímulos de luz, sombras, imagens tremidas e oclusão. Como solução para este problema, comumente utilizar-se a média dos resultados encontrados, formando um ambiente de aprendizagem estatística.

Conforme ZHANG et al. (2006), em ambientes de aprendizagem estatística, imagens são classificadas por um conjunto de características e, em seguida, são utilizados métodos de aprendizagem para identificar objetos de uma classe de interesse. Em geral, estes métodos pretencem a duas abordagens: abordagens baseadas em aparência global e abordagens baseadas em componentes.

Abordagens baseadas em aparência global consideram um objeto como uma única unidade, para então executar a classificação sobre as características extraídas de todo o objeto. Muitos destes mecanismos de aprendizagem estatísticos são explorados para caracterizar e identificar padrões, a fim de encontrar particularidades entre os objetos. Muitas vezes torna-se necessário utilizar funções de pré-processamento, para corrigir variações de iluminação, contraste e orientações.

Outras abordagens, estas baseadas em componentes, tratam um objeto como uma coleção de partes, extraindo primeiramente componentes dos objetos e, em seguida, os detectam utilizando informações geométricas. Tais algoritmos permitem, por exemplo, classificar uma pessoa constituída por um conjunto de componentes como a cabeça, braços e perna. Até mesmo em classes de objetos, como motos, rostos humanos, aviões e carros.

Várias pesquisas relacionadas a detecção de objetos vem sendo realizadas, na maioria das vezes, para a sua identificação. Tais estudos serão abordados na seção 2.3.1 desta monografia, principalmente as que tratam de objetos em movimento.

2.2.3 Realce e Restauração

Muitas vezes antes da utilização de algoritmos de processamento em imagens, técnicas de pré-processamento tornam-se necessárias para restaurar imagens. Azevedo, Conci e Leta

(2008, p. 55) explicam que no processo de digitalização de uma imagem do mundo real, sempre há distorções, estas conhecidas como *aliasing*. Uma forma de minimizar o problema é aumentar o máximo possível a resolução, de acordo com o nível de detalhamento da imagem real.

Já o realce tem por objetivo destacar detalhes da imagem que são de interesse para análise ou que tenham sofrido alguma deterioração. Portanto, permite levantar informações que antes não eram percebidas visualmente, podendo assim aumentar a capacidade de extração de informações. Porém, estas podem variar em diferentes casos de estudo, uma vez que níveis de detalhe podem ser perdidos.

De acordo com Azevedo, Conci e Leta (2008, p. 55), quando a imagem encontra-se deteriorada, utilizam-se técnicas de restauração. Estas têm o objetivo de compensar a falta de contraste, efetuar a correção de foco, bem como imagens borradas por movimento. Geralmente ocasionadas no momento de aquisição, na transmissão ou em alguma etapa do processamento.

Deve-se verificar se os resultados são apenas qualitativos, o que pode acarretar em problemas de desempenho na aplicação. Em alguns casos, algoritmos matemáticos envolvidos com a restauração ou ao realce são extremamente complexos, podendo ter custo computacional alto.

2.3 PROCESSAMENTO DE IMAGENS DIGITAIS

Processar uma imagem consiste em transformá-la sucessivamente, utilizando técnicas para filtrar objetos e descartar dados irrelevantes. Gonzalez e Woods (2000, p. 1) explicam que o interesse desses métodos decorre de duas áreas: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para percepção automática através de máquinas.

Segundo Parker (1994, p. 15), o processamento de imagens está fortemente ligado a visão computacional, que tem como objetivo, aliada as técnicas de inteligência artificial, realizar tarefas semelhantes ao olho humano. Permitindo extrair informações relevantes para um determinado problema, para serem analisadas e aplicadas em uma solução.

Nas seções seguintes serão apresentados algoritmos de detecção de elementos em movimento e técnicas de segmentação de imagens. Também serão apresentados os descritores

de imagem e por fim, técnicas o conceito de interpretação de imagens e as áreas envolvidas (Redes Neurais Artificiais), sendo estes conceitos fundamentais para processamento de imagens digitais.

2.3.1 Detecção de Elementos em Movimento

A detecção de elementos em movimento vem sendo utilizada constantemente, como o objetivo de diferenciar, em uma seqüência de vídeo, os objetos dinâmicos dos estáticos. Porém, Gonzalez e Woods (2000, p. 195) já apontavam os problemas que dificultam em ter-se uma forma genérica para resolver este problema, uma vez que as dependem das condições do ambiente e da aplicação que se deseja criar, bem como custo computacional.

O problema é ainda maior quando são cenários reais ou em tempo real, onde variações no ambiente devem ser previstas. Um grande fator é a mudança de luminosidade, seja quanto a posição e a intensidade como na alteração de sombras, com objetos camuflados e superfícies espelhadas. Outros problemas podem também decorrer das alterações no posicionamento da câmera que implicam em oscilações ou objetos de alta freqüência, ou ainda das mudanças na geometria do fundo.

Para um melhor entendimento, nas seções seguintes serão mostrados métodos de detecção de elementos em movimento: remoção de fundo e segmentação adaptativa.

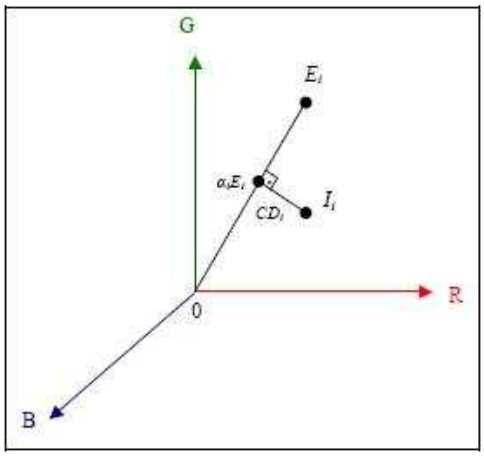
2.3.1.1 Remoção de Fundo

O método *Background Subtraction* (subtração de fundo) é uma das formas de descriminar um objeto em movimento, através de um fundo de cena. Fernandes (2002, p. 16) explica que este método consiste em subtrair a imagem atual de outra usada como referência, contendo apenas o fundo da cena e construída a partir de uma seqüência de imagens durante um período de treinamento. Portanto, deixa apenas objetos não estáticos e novos.

Como visto anteriormente, uma das facilidades da visão humana é a fidelidade à cor. Os seres humanos tendem a associar uma cor constante a um objeto, mesmo que esse esteja

sob a influência de variações de luminosidade. A fidelidade à cor é importante na remoção do fundo da cena, pois permite ao programa a correta classificação dos objetos, independente da projeção de sombras sobre eles. O que permite ignorar as sombras dos objetos, deixando de classificá-las como sendo pertencentes ao objeto em movimento.

Com base nas informações das cores dos objetos, o algoritmo de subtração de fundo foi desenvolvido para separar a distorção da cor da distorção da luminosidade, a partir de uma imagem de referência e da atual. Esse modelo é mostrado na figura 4, onde: *i* é o índice de um *pixel* nas imagens, *O* é o ponto de origem, onde as coordenadas para *R*, *G* e *B* são igual a zero; $E_i = [E_{R(i)}, E_{G(i)}, E_{B(i)}]$ representa o conjunto de valores RGB esperados para o *pixel* na imagem de referência; $I_i = [I_{R(i)}, I_{G(i)}, I_{B(i)}]$ representa o conjunto de valores RGB do *pixel* na imagem atual, da qual se deseja subtrair o fundo, e a linha OE_i, que passa pelos pontos *O* e E_i, é "linha cromática esperada", composta por pontos que representam o fundo da cena com ou sem a aplicação de luz e sombra (FERNANDES; 2002, p. 15).



Fonte: FERNANDES (2002, p. 16).

Figura 4 – Modelo de cor proposto no espaço tridimensional RGB

Usando a relação entre I_i e OE_I são encontrados os valores da distorção da luminosidade (α_i) e da distorção da cor (CD_i), sendo CD_i a distância ortogonal entre a cor observada em I_i e a linha cromática esperada; α_i é um escalar, se seu valor for igual a 1, indica a luminosidade idêntica ao fundo original, um valor maior que 1 indica o fundo mais iluminado e um valor menor que 1 indica o fundo menos iluminado.

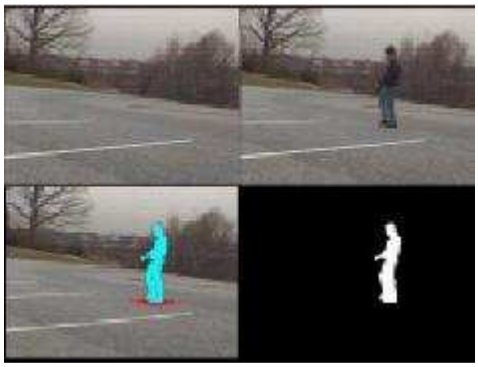
Para realizar a subtração do fundo, Horprasert, Harwood e Davis (2000) propuseram um modelo de cor pelas seguintes etapas:

a) modelagem do fundo (*background modeling*): onde a imagem de referência *E* é a construída a partir de uma série de imagens contendo apenas o fundo da cena;
b) seleção de limiar (*theshold selection*): onde são escolhidos os valores mínimo e

máximo aceitos para a distroção de luminosidade α e de cor *CD*;

c) operação de subtração (*subtraction operation*) ou classificação de *pixel* (*pixel classification*): onde os *pixels* das novas imagens *I* informadas são classificados como em movimento, pertencente ao fundo original, pertencentes ao fundo com sombra ou ao fundo mais iluminado.

Essa classificação gera uma imagem binária que indica se o *pixel* faz parte do fundo – original, sobreado ou iluminado - ou em movimento. A figura 5 ilustra todas as etapas do processo da subtração do fundo da cena, onde: no canto superior esquerdo é mostrada a imagem de referência construída, no canto superior direito é mostrada a imagem da qual será subtraído o fundo; no canto inferior esquerdo é mostrado o resultado da classificação dos *pixels*; e no canto inferior direito é mostrada a imagem binária gerada.



Fonte: FERNANDES (2002, p. 18).
Figura 5 – Exemplo de resultado de subtração de fundo

2.3.1.2 Algoritmo de Segmentação Adaptativa

Considerando o problema de videovigilância, um sistema sólido não deve depender de foco e iluminação, ou seja, deve ser robusto para qualquer que seja o seu campo de visão (STAUFER; GRIMSON, 1999, p. 1). Visando atender a essa necessidade, Staufer e Grimson (1999) criaram o algoritmo GMM, com o objetivo de definir uma região segmentada delimitando-se os *pixels* de interesse. O modelo é atualizado a cada nova observação obtida de acordo com pesos associados a cada distribuição, diminuindo a influência das observações

passadas permitindo adaptar-se às variações graduais de iluminação da cena. A imagem 6 mostra um exemplo do resultado desse algoritmo, onde apenas as pessoas que estão se movimentando são segmentadas, enquanto o resto foi interpretado como fundo.

Fonte: Staufer e Grimson (1999, p. 1).

Figura 6 – Segmentação utilizando algoritmo GMM

BUTLER et al (2005, p. 2) propôs um algoritmo mais simples, performático e com um resultado melhor, chamado de NHD. Este algoritmo foi também utilizado no sistema descrito no trabalho de DENMAN et al (2006). Ao contrário dos algoritmos apresentados anteriormente, a detecção deste utiliza imagens em cores Y'CbCr com formato 4:2:2 de entrada.

No sistema de cores Y'CbCr, a componente Y contém altas freqüências de luminância em escalas de cinza às quais o olho humano é sensível. Já as componentes de crominância Cb e Cr guardam, respectivamente, freqüências altas de informação de cores, às quais o olho humano é muito menos sensível.

Para compactar uma imagem no formato 4:2:2 os *pixels* devem ser agrupados em pares, dois de largura e um de altura. O valor do novo *pixel* será formado pelos valores de luminância dos *pixels* do grupo, juntamente com a média dos componentes de crominância. Portanto, possui quatro valores em sua estrutura, dois de luminância e um de crominância.

Para o processo de segmentação, o algoritmo utiliza grupos de *clusters*. A estrutura de um *cluster* é constituída por um peso e pelos valores da cor de um *pixel*. Os valores da cor de um *pixel* dentro de um *cluster* são chamados de centróide. O peso do *cluster* serve para denotar a freqüência da sua ocorrência, ou seja, quantas vezes a cor do centróide repete na imagem.

A cada quadro, o algoritmo percorre os *pixels* do mesmo com o objetivo de encontrar o *matching cluster*, percorrendo todos os *clusters* pertencentes ao grupo. Para isso, o cálculo da

distância de Manhattan é aplicado entre o *pixel* e o centróide do *cluster* atual, sobre os pares de luminância e de crominância. A distância de Manhattan aplicada para um plano cartesiano que contém os pontos *P1* e *P2*, com as respectivas coordenadas (x_1, y_1) e (x_2, y_2), define-se por $|x_1 - x_2| + |y_1 - y_2|$.

Se para um determinado *pixel* não for encontrado nenhum *matching cluster*, então o *cluster* de menor peso é substituído por um novo com seu centróide igual ao valor do *pixel* atual e com peso inicial (0,01). Se o *cluster* correspondente foi encontrado, então os pesos de todos os *clusters* pertencentes ao grupo são atualizados por:

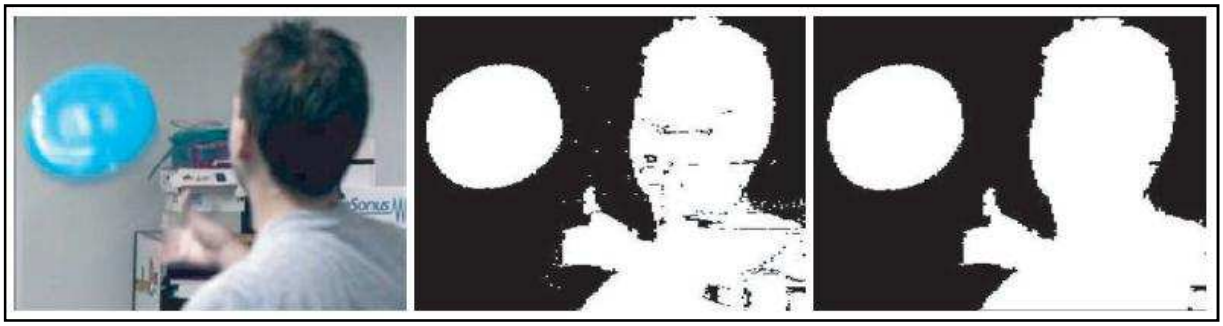
$$w'_k = \begin{cases} w_k + \frac{1}{L}(1 - W_k), & k = M_k \\ w_k + \frac{1}{L}(0 - W_k), & k \neq M_k \end{cases},$$

onde M_k é o índice do *cluster* correspondente. O parâmetro L é o inverso da taxa aprendizagem, α, e pode ser usado para controlar a velocidade em que a cena muda, quanto menor o valor de L mais rápida será a adaptação. O mais provável é que a cor do fundo seja estável, o que permite um limiar menor, representada por $T = T_{max} - \left(w_{max} \times (T_{max} - Th_{min})\right)$, onde T é o limite a ser utilizado para o *pixel* correspondente; T_{max} é o limite máximo; w_{max} é o peso da mais alta ponderada do *cluster*, e Th_{min} é a limiar mínima.

Aplica-se também um mecanismo estatístico, para determinar se um *pixel* é *background* ou se é *foreground*. Assume-se *background* os *pixels* pertencentes ao fundo e *foreground* tudo que não pertence ao fundo.

Ao adaptar os *clusters* seus pesos são reduzidos para impedir que estes sejam incorporadas ao *background*, enquanto os *clusters* não correspondentes têm seu peso aumentado. Isto pode ser usado para evitar um movimento lento, ou então incorporar objetos conhecidos ao *foreground*, através da fórmula $w_k = w_k \times A^{M_k}$, onde w_k é o peso do *cluster*, A é a taxa de adaptação, M_k é 1 se está em movimento e -1 caso contrário.

A figura 7 mostra os resultados obtidos com o algoritmo, onde o primeiro quadro mostra a imagem original, o segundo mostra a imagem após o processamento e a terceira depois do pós-processamento.



Fonte: Butler et al (2005, p. 6).

Figura 7 – Segmentação utilizando algoritmo NHD

2.3.2 Segmentação de Imagens

O processo de segmentação de uma imagem consistem em detectar sub-imagens a partir das bordas dos objetos. De maneira geral, esta etapa do processamento de imagens tem como objetivo isolar os objetos de interesse. Esse processo engloba algoritmos de detecção de pontos, linhas, bordas ou pela diferença da imagem.

A detecção de linhas analisa se um conjunto de pontos está adequadamente alinhado, verificando a variação de níveis de cor RGB. Já a detecção de borda tenta encontrar objetos através das suas variações locais, com base nos pontos vizinhos e contraste com o fundo da imagem. Conforme Gonzalez e Woods (2000, p. 295), os algoritmos de segmentação são geralmente baseados na descontinuidade e similaridade dos valores de nível de cinza. Nas seções seguintes, serão apresentados dois métodos de segmentação de imagens: limiarização e detecção de bordas

2.3.2.1 Limiarização

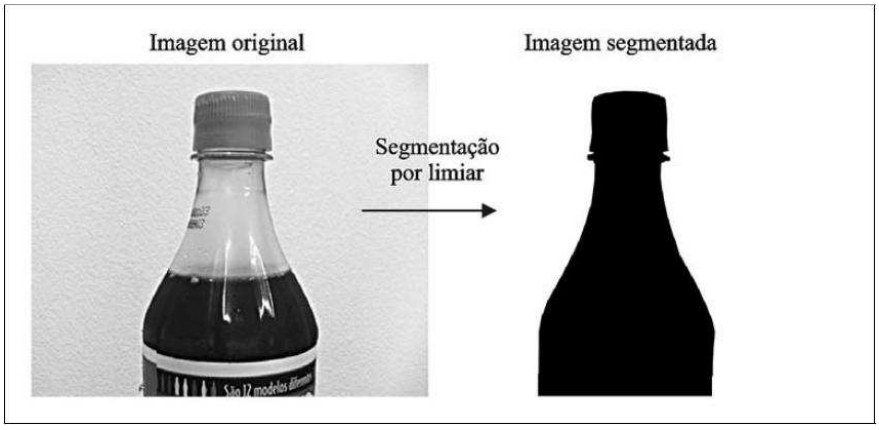
Uma abordagem muito utilizada para a segmentação de imagens através da similaridade é a limiarização. Ela consiste em converter imagens em tons de cinza para imagens binárias, por isso essa técnica é conhecida também como binarização e engloba

várias técnicas diferentes.

A limiarização global consiste em um limiar que representa um nível de cinza qualquer. Para cada *pixel* da imagem é realizado um teste para verificar se o nível de cinza deste *pixel* é maior que o limiar definido. Em caso positivo, o mesmo é rotulado com um valor, caso contrário é rotulado com outro valor. O resultado deste processo é uma imagem binária, ou seja, a cor de cada *pixel* é representada por apenas um bit. Cada *pixel* da imagem, portanto, apresenta a cor preta (valor 0) ou branca (valor 1).

Para aplicações de monitoramento, o resultado da segmentação desejado é uma imagem binária onde a forma do objeto a ser inspecionado esteja representada por um agrupamento de *pixels* com determinado valor e que o fundo esteja representado com *pixels* de outro valor.

A figura 8 exemplifica a segmentação através de limiarização global, onde o resultado obtido é a silhueta do frasco a ser inspecionado.



Fonte: STIVANELLO (2004, p. 24).

Figura 8 – Segmentação de imagem através de limiar

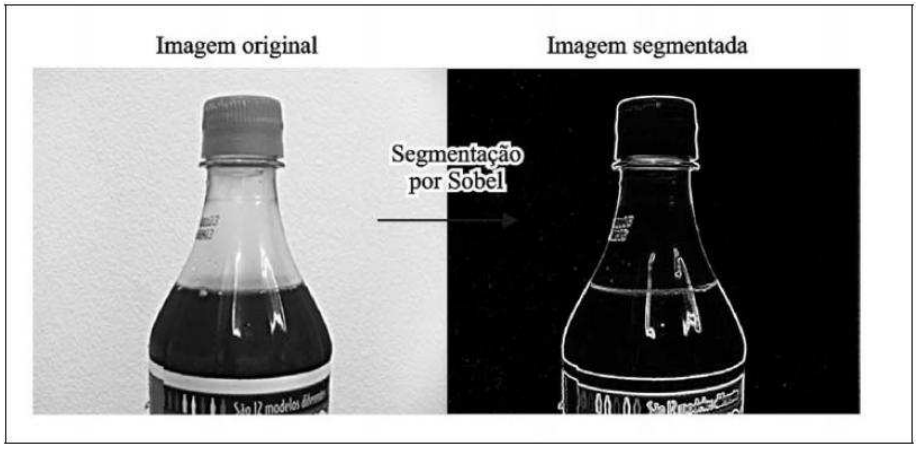
2.3.2.2 Detecção de Bordas

A detecção de bordas é uma das técnicas em que a segmentação é realizada com base na descontinuidade de valores de níveis de cinza. Gonzales e Woods (2000, p. 4) explicam que a detecção de bordas pode ser obtida pelo uso de filtros por derivada.

De acordo como Green (2003, p. 20), vários algoritmos de filtros por derivada foram desenvolvidos com a finalidade de detectar bordas, dentre eles pode-se destacar Roberts, Sobel e Canny. O algoritmo de Roberts é simples de programar e consome menos

processamento, pois trabalha com uma matriz 2x2. Porém é menos eficiente em imagens com ruído que o algoritmo de Sobel, que utiliza uma matriz 3x3, mas necessita de um maior poder de processamento. O algoritmo de Canny, por sua vez, tenta suavizar o ruído da imagem, fazendo a aproximação da borda detectada com a borda original e reduzindo ao máximo o número de pontos agrupados para uma mesma borda.

A partir das máscaras do operador de Sobel é possível conseguir uma imagem segmentada semelhante a figura 9. Tem-se que as derivadas baseadas nas mesmas podem ser obtidas pelas equações: $Gx = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$ e $Gy = (z_3 + 2z_6 + z9 - z1 + 2z4 + z7$.



Fonte: STIVANELLO (2004, p. 24).

Figura 9 – Aplicação do operador Sobel sobre a imagem

2.3.3 Descritores de Imagens

Azevedo, Conci e Leta (2008, p. 228) explicam que na etapa de segmentação, obtêm-se agrupamentos de *pixels* que representam os componentes da imagem a serem analisados. Porém, estes componentes devem ser descritos de maneira mais apropriada, já que uma análise direta sobre os *pixels* não é muito eficiente.

Os descritores são conjuntos de números, gerados para descrever uma forma. Os descritores podem não reconstruir completamente a forma descrita, mas devem ser suficientes para discriminar diferentes formas.

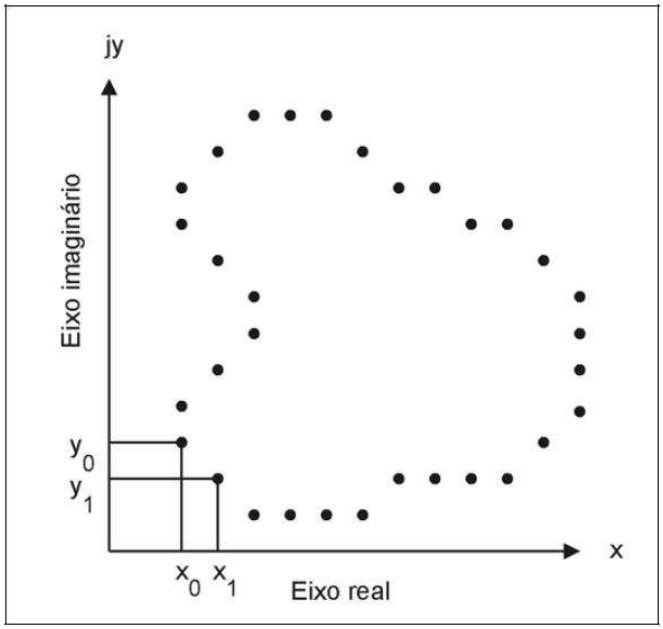
A descrição de componentes da imagem pode se dar através de descritores simples,

como a área. A área é obtida simplesmente através da contagem dos *pixels* que compõe a região. Porém, para a maioria das aplicações, informações simples como a área não são suficientes para a resolução dos problemas. Na seção seguinte será apresentado o método de descritores Fourier, muito utilizado nestes tipos de aplicação.

2.3.3.1 Descritores de Fourier

Descritores de Fourier são uma das formas de representar imagens, muito utilizados para identificação de objetos em visão computacional e em reconhecimento de padrões. Estes descritores não constituem um método simples, mas uma classe de métodos já que existem diferentes maneiras de definí-los.

Gonzales e Woods (2000, p. 355) demonstram como uma imagem pode ser representada através de descritores de Fourier. A figura 11 exibe uma fronteira digital de *N* pontos no plano *xy*.



Fonte: STIVANELLO (2004, p. 27).

Figura 11 – Código de cadeia

Por exemplo, se uma iteração iniciar de um ponto qualquer (*x0*, *y0*) no sentido anti-horário, pode-se encontrar os pares de coordenadas (*x0*, *y0*), (*x1*, *y1*), (*x2*, *y2*), ..., (*xn-1*, *yn-1*)

que representam a fronteira da forma. Estes pares podem ser tratados como números complexos da formas k = 0,1,2,...,N – 1, onde N é a quantidade de pontos que compõe a fronteira. Desta forma é possível reduzir problemas de duas dimensões em uma só, conforme $s(k) = x(k) + jy(k)$.

Esta representação possui a vantagem de reduzir um problema de duas dimensões a uma só dimensão, através da fórmula:

$$a(u) = \frac{1}{N}\sum_{k=0}^{N-1} s(k)^{-j2\pi uk/N}.$$

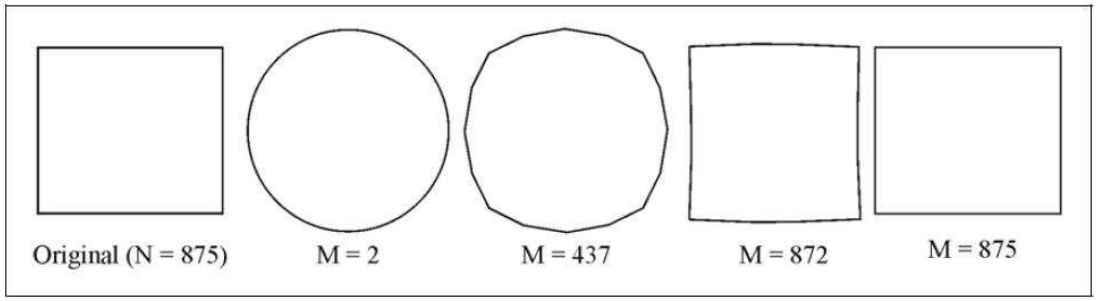
A transformada discreta de Fourier é definida por $a(u)$, para u = 0,1,2,...,N-1, onde N é a quantidade de pontos que compõe a fronteira. Os coeficientes complexos $a(u)$ são chamados de descritores de Fourier.

Portanto torna-se possível obter descritores e o número de postos que compõe a fronteira no plano *xy*, mas torna-se inviável em descrever a forma pela mesma quantidade de pontos. A vantagem deste método de descrição encontra-se justamente na possibilidade de representar uma forma com uma pequena quantidade de descritores.

A transformada inversa de Fourier de $a(u)$, para u = 0,1,2,...,N-1 onde N é a quantidade de pontos que compõe a fronteira. A transformada inversa de Fourier é utilizada para reconstruir $s(k)$ a partir dos coeficientes $a(u)$. A variável M representa o número de coeficientes utilizados na descrição da fronteira, sendo que este pode ser menor ou igual a N-1, através da fórmula:

$$s(k) = \sum_{u=a}^{M-1} a(u)^{j2\pi uk/N}.$$

Na figura 12 são exibidas as fronteiras obtidas pela reconstrução da fronteira original de um quadrado, geradas a partir de diferentes números de coeficientes. Através deste exemplo pode-se confirmar a premissa de que os coeficientes de baixa ordem capturam a essência geral da forma, sendo os detalhes mais finos capturados pelos coeficientes de alta freqüência.



Fonte: STIVANELLO (2004, p. 29).

Figura 12 – Forma original e reconstruções a partir de *M* coeficientes

Um trabalho que exemplifica a utilização de descritores de Fourier na descrição de elementos de imagens é o trabalho de STIVANELLO (2004). Neste trabalho, forma de produtos é descrita por descritores de Fourier, e em seguida é válida por uma rede neural para a identificação de possíveis defeitos.

2.3.4 Interpretação de Imagens

A interpretação consiste em identificar padrões através da análise das descrições da imagem realizadas nas etapas anteriores. As imagens podem ser de diferentes resoluções e escalas, mas independente disso, se caracterizam por apresentarem alguns elementos básicos que permitem a extração de informações.

A análise das descrições de imagens leva em conta padrões ou regras previamente definidas, porém, não há consenso sobre todos os elementos que devam ser considerados na análise visual de imagens. Gonzales e Woods (2000, p. 424) ressaltam que as propriedades estatísticas das classes de padrões são freqüentemente desconhecidas, ou não podem ser estimadas. Problemas de decisão teórica são mais bem tratados por métodos que levem às funções de decisão através de treinamento.

Facon (1993, p.174) exemplifica uma interpretação onde deverão ser classificados objetos dentre cinco classes existentes: perímetro, área, número de Euler, raio mínimo e raio máximo. Neste exemplo a classificação se dá diretamente pela comparação entre todas as propriedades do objeto analisado com cada um dos padrões existentes. Porém, há cenários onde os objetos não podem ser analisados por propriedades simples como estas. Neste caso, pode-se optar por Descritores de Fourier.

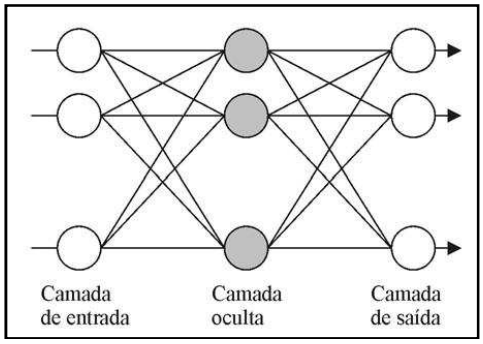
2.4 REDES NEURAIS ARTIFICIAIS

Segundo Loesch e Sari (1996, p. 5), redes neurais artificiais são sistemas computacionais implementados em hardware ou software que imitam as habilidades computacionais do sistema nervoso, usando um grande número de neurônios interconectados. Surgidas na década de 40, foram desenvolvidas com o objetivo de fazer uma analogia entre células nervosas vivas e o processo eletrônico.

Para redes neurais artificiais, o neurônio artificial é o elemento básico na sua formação, pois simula o funcionamento de um neurônio biológico. O neurônio possui um ou mais elementos da entrada. Esses elementos são considerados de maneira simultânea, ou seja, não existem situações onde o valor de apenas uma das entradas é considerado. Os valores dos pesos de cada entrada do neurônio artificial são obtidos no momento do treinamento da rede neural. Esses pesos representam o grau de importância para o neurônio e através de suas variações que se obtém o conhecimento.

A função de ativação, em modelos mais simples de redes neurais, simplesmente realiza a soma dos valores das entradas multiplicadas pelo respectivo peso, para então repassar o sinal obtido para a função de transferência. Esta então analisa o valor gerado e cria uma saída para o neurônio.

Uma rede neural pode possuir uma ou múltiplas camadas. Entre as camadas de entrada e saída podem existir camadas intermediárias ou ocultas, cuja diferença entre ela e a camada de saída é não possuir contato com o exterior. Estas camadas servem para melhorar o desempenho da rede. Por fim, tem-se a camada de saída, que é responsável por repassar o resultado do processamento da rede para o mundo exterior. A quantidade de neurônios desta camada é igual ao número de saídas esperadas da rede. A figura 13 ilustra uma rede neural com uma camada oculta entre as camadas de entrada e saída.



Fonte: STIVANELLO (2004, p. 20).
Figura 13 – Camadas da Rede Neural

O modelo supervisionado é a forma de treinamento mais utilizada nas redes neurais artificiais. Segundo Tafner (1996, p. 65), a rede neural será treinada com o auxílio de um treinador, que utilizará conjuntos de dados, para que cada entrada tenha uma saída desejada. Se for diferente a rede deverá ajustar os pesos de forma que armazene o conhecimento desejado. Essa iteratividade do treino deverá ser repetida com todo conjunto de treinamento, até que a taxa de acerto esteja dentro de uma faixa considerada satisfatória. A rede neural estará pronta para ser utilizada, lembrando que os pesos não podem ser ajustados nesta fase.

Dentre todas as arquiteturas conhecidas, a rede *Perceptron* multicamadas é a mais utilizada. Loesch e Sari (1996, p. 67) explicam que as capacidades da rede foram responsáveis pela popularização do modelo. “Além da capacidade de *abstração*, a rede possui capacidade de *generalização*, ou seja, é capaz de classificar corretamente um padrão complexo mesmo quando este não pertencer ao conjunto de treinamento da rede” (LOESCH; SARI, 1996, p. 67).

2.5 TRABALHOS CORRELATOS

Como visto na sessão 2.1, as técnicas utilizadas em sistemas de monitoramente por visão computacional podem ser utilizadas em várias áreas diferentes. Portanto foram escolhidos dois trabalhos correlatos com finalidades diferentes deste, para segmentar objetos em movimento e para identifica-los.

O trabalho de FERNANDES (2002) trata-se de estudo sobre captura de movimentos humanos através de uma câmera, sem que os indivíduos presentes na imagem possuam marcações especiais. A figura 14 mostra o resultado deste trabalho, que utilizou o algoritmo de remoção de fundo para encontrar a pessoa presente na imagem. O contorno em vermelho representa o resultado do algoritmo de detecção de borda.

Fonte: FERNANDES (2002, p. 76).
Figura 14 – Resultado do algoritmo de segmentação

Já o trabalho de STIVANELLO (2004) trata-se de uma ferramenta para controle de qualidade de produtos em uma fabrica. Através de uma Rede Neural Perceptron Multicamadas o sistema consegue diferenciar os produtos com problema dos com boa qualidade de fabricação. A figura 15 mostra na primeira imagem um produto considerado com boa qualidade, já a segunda imagem representa um produto com má qualidade de fabricação.

Fonte: STIVANELLO (2004, p. 68).
Figura 15 – Exemplo de produto válido e inválido

3 DESENVOLVIMENTO

Neste capítulo são abordadas as atividades relativas ao projeto e desenvolvimento do sistema proposto pelo trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O sistema de reconhecimento de veículos deverá:

a) disponibilizar uma interface para configurar um cenário a partir de uma imagem de vídeo (Requisito Funcional – RF);
b) permitir configurar parâmetros do algoritmo de segmentação (RF);
c) disponibilizar uma ferramenta de desenho de fronteiras, onde serão feitas as contagens dos veículos (RF);
d) efetuar pós-processamento das imagens segmentadas, permitindo definir o tamanho da área dos objetos encontrados (RF);
e) disponibilizar uma interface para permitir o treinamento da rede neural com exemplos de veículos do tipo automóvel (RF);
f) disponibilizar contador de veículos do tipo automóvel (RF);
g) ser desenvolvido utilizando programação orientada a objetos (RNF);
h) utilizar linguagem de programação C++ (Requisito Não Funcional – RNF);
i) utilizar ambiente de programação Borland Builder 6 (RNF).

3.2 ESPECIFICAÇÃO

O sistema apresentado utiliza alguns dos diagramas da *Unified Modeling Language* (UML). Foi utilizada a ferramenta Enterprise Architect para a elaboração dos diagramas de caso de uso, de classe e de sequência.

3.2.1 Diagrama de Caso de Uso

A figura 16 apresenta o diagrama de caso de uso do sistema, que permite ao usuário manipular as suas funcionalidades.

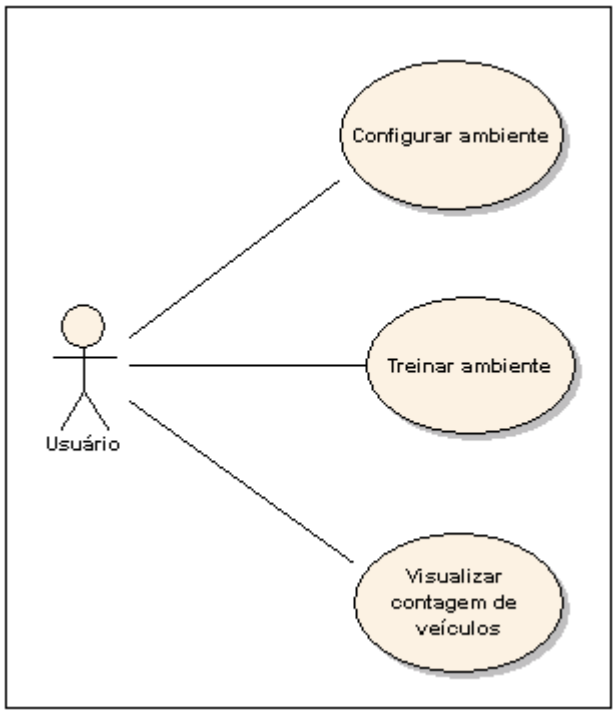


Figura 16 – Diagrama de caso de uso

O caso de uso `Configurar ambiente` representa a funcionalidade no que diz respeito a definição e configuração das variáveis pertinentes ao sistema. O usuário deve informar valores de *threshold* para os algoritmos de segmentação, bem como o tamanho da área dos objetos de interesse, carros ou não.

No caso de uso `Treinar ambiente`, o sistema deve permitir ao usuário delimitar uma faixa de interesse, onde serão feitos os reconhecimentos. Quando algum objeto ou ruído for detectado nesta faixa, o sistema deve questionar ao usuário se a área encontrada é um carro, ou algo que deva ser desconsiderando.

Já o caso de uso `Visualizar contagem de veículos` representa a funcionalidade de identificação e contagem de veículos. Após a rede estar devidamente treinada pelo usuário, o sistema deve mostrar a quantidade de carros que passaram pela faixa determinada pelo usuário.

3.2.2 Diagrama de Classe

O diagrama de classes apresentado na figura 17 exibe as classes utilizadas no sistema, para execução de vídeo, criação do ambiente, processamento de imagens, e analise de imagens através de redes neurais.

Basicamente as classes podem ser agrupadas em ambientes, processor, codecs e as restantes são de apoio com funcionalidades importantes ao sistema. No caso das classes codecs, estas representam as etapa de processamento de imagem.

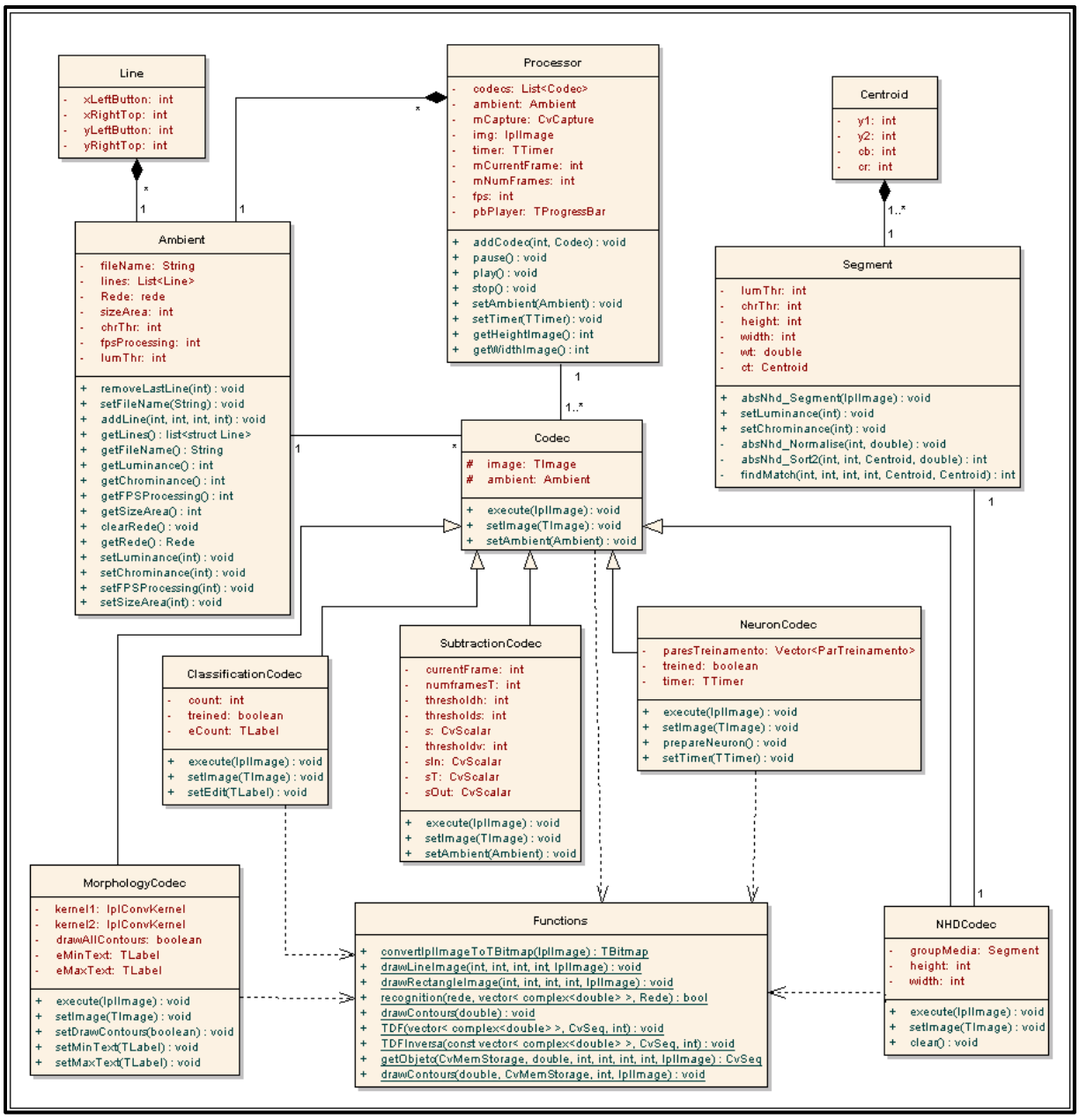


Figura 17 – Diagrama de classe

3.2.2.1 Classe `Ambient`

A classe `Ambient` representa o cenário a ser analisado, ou seja, tem o objetivo de agrupar todas as informações do ambiente. Esta classe poderia ser utilizada em implementações de outros sistemas, permitindo cadastrar vários ambientes em uma base de dados para serem utilizadas na aplicação.

Além da informação do caminho do arquivo de vídeo (atributo `fileName`), a classe `Ambient` contém a informação de quantos quadros por segundo as etapas processamento de imagens, pós-processamento, treinamento e reconhecimento de objetos serão executadas. Além deste atributo que é chamado de `fpsProcessing`, nela também são armazenados dados importantes para todas as etapas, que serão melhor detalhados nas seções que correspondem as estapas de manipulação de imagens.

Esta classe possui uma estrutura chamada `Line`, que contém os pontos de uma determinada linha. A classe `Ambient` armazena variáveis do tipo `Line`, criadas pelo usuário, no atributo `lines`, úteis para as etapas de treinamento e reconhecimento, bem na rotina de desenho.

A classe `Ambient` é também o ponto de ligação com a rede neural utilizada no sistema, através do atributo `rede`. O atributo rede é uma instância da classe `Rede` implementada no Trabalho de Conclusão de Curso de Stivanello (2004), cujas as classes utilizadas foram adaptadas para este trabalho. A rede implementada é do tipo Perceptron Multicamadas, pela sua capacidade de generalização. Para visualizar detalhes da rede, verificar o trabalho do mesmo.

3.2.2.2 Classe `Processor`

A classe `Processor` é responsável pela execução dos ambientes, ou seja, tem a função de *player* na aplicação, com auxílio de um *Thread* que é passado para esta. Possui informações do vídeo, como a quantidade de quadros por segundo, número total de quadros e resolução das imagens.

Ela também é responsável pela chamada das etapas de manipulação de imagens, que

estão contidas na lista `codecs`. Cada classe que pertence a hierarquia da classe `Codec` executa uma etapa de manipulação de vídeos, através do método `execute()` implementada na mesma. A estrutura da classe `Codec` será detalhada na seção seguinte.

As etapas de processamento de imagens são executadas a cada chamada da *Thread*, respeitando as informações contidas na classe `Ambient`, principalmente a classe responsável pela execução das etapas no intervalo de quadros por segundo. Desta forma o vídeo atualmente em execução tiver quadros, os mesmos são passados para as classes responsáveis pelo processamento, respeitando a ordem que foram adicionados na lista.

3.2.2.3 Classe `Codec`

A classe `Codec` é a classe base para as etapas de manipulação de imagens, contendo atributos e métodos comuns a todos. Esta possui a funcionalidade de mostrar as imagens atualmente processadas na tela, desenhando as linhas criadas pelo usuário descritas na classe `Ambient`.

Nenhuma manipulação ou processamento de imagens é realizada nesta classe, ela apenas mostra os resultados parciais que cada sub-classe, que a implementa, executou. O que permite criar uma instância direta da classe `Codec`, para visualizar a imagem original do vídeo.

3.2.2.4 Classe `NHDCodec`

A classe `NHDCodec` faz o controle do processo de segmentação de imagens, através da manipulação da classe `Segment`, que será apresentada na seção 3.2.2.9. A implementação das classes `NHDCodec` e `Segment` foram separadas, permitindo visualizar o algoritmo de segmentação isoladamente. O que possibilita que este algoritmo seja utilizado no desenvolvimento de outros sistemas.

3.2.2.5 Classe `SubtractionCodec`

A classe `SubtractionCodec` é responsável pelo algoritmo de subtração de fundo, que retorna uma imagem binária no formato RGB. Foi implementada para permitir a comparação entre os algoritmos de segmentação.

3.2.2.6 Classe `MorphologyCodec`

A classe `MorphologyCodec` faz os tratamentos de pós processamento na imagem, através de algoritmos de morfologia matemática. Através dos algoritmos de erosão e dilatação, os ruídos que ficaram na imagem após a etapa de processamento são reduzidos, facilitando a busca de objetos na imagem para as etapas seguintes (treinamento e reconhecimento de objetos). Essa rotina é necessária, pois podem existir bastante ruídos na imagem, por conseqüência, pequenos grupos de *pixels* são formados.

Após a aplicação dos algoritmos de morfologia, as rotinas de desenho de *boundingbox* das áreas e de realce dos contornos são executadas, após a extração dos componentes conexos da imagem. Estas rotinas respeitam o atributo que determina o menor valor de área que deverá ser desenhada, pertencente a classe `Ambient`. Além disso, a classe `MorphologyCodec` é responsável por informar o valor da menor e da maior área encontrada.

3.2.2.7 Classe `NeuronCodec`

A classe `NeuronCodec` é a responsável pelo treinamento da rede neural utilizada no sistema, portanto, é o ponto de ligação com a implementação da rede utilizada. A extração de componentes conexos da imagem serve para verificar qual dos contornos será considerado. Assume-se que o contorno que deve ser considerado seja aquele cuja a área cruza a linha desenhada pelo usuário, e que esta área seja maior que o mínimo informado na classe `Ambient`. Caso não haja objetos passando pela linha este processo é ignorado.

Com a lista de coordenadas dos pontos no plano xy, resultantes da etapa de pós-

processamento, já há informação suficiente para diferenciar veículos. Porém, a lista gerada geralmente é muito grande. Considerando-se que os quadros capturados por câmeras específicas para aplicações de CFTV são compostos por 320 *pixels* de largura por 240 *pixels* de altura, freqüentemente são obtidos contornos com mais de 1000 *pixels*. Esta quantidade de informação é inviável para a maioria das técnicas de reconhecimento.

Para resolver este problema foi implementada a técnica de descrição por Fourier apresentada na seção 2.3.3.2. Esta função foi criada na classe `Functions`, que será apresentada na seção 3.2.2.10. Após a redução da quantidade de informações que descrevem os carros, pode-se interpretar os mesmos através das técnicas de redes neurais descritas na seção 2.4.

Durante o treinamento da rede neural, a classe `NeuronCodec` dispara mensagens ao usuário, questionando se o objeto que passou pela reta cadastrada na classe `Ambient`, conforme dito na seção 3.2.2.1, é um carro ou não. Após o termino do vídeo, a rotina de treinamento da rede neural é chamada.

3.2.2.8 Classe `ClassificationCodec`

A classe `ClassificationCodec` efetua o processo de reconhecimento e contagem de veículos. O processo é bem parecido com o da classe `NeuronCodec`, leva em consideração apenas as componentes conexas cuja área está passando pela linha informada pelo usuário, respeitando o parâmetro de menor área da classe `Ambient`.

Porém, em vez de agrupar esses contornos em uma lista para posteriormente serem usados para treinar a rede, esta chama a execução de reconhecimento da rede, que está implementada na classe `Functions` (que será explicada na seção 3.2.2.10).

3.2.2.9 Classe `Segment`

A classe `Segment` é a implementação do algoritmo NHD, apresentado na seção 2.3.1.3. Como os quadros dos vídeos utilizados no sistema estão no formado RGB, o mesmo faz a conversão para o formato Y’CbCr, em seguida faz a compressão para o formato 4:2:2,

exigência da especificação do algoritmo NHD. A estrutura `Centroid`, internamente declarada, armazena os dados após a compressão.

O método `absNhd_Segment` executa o algoritmo sobre o quadro atual, varrendo todos os *pixels*, com o objetivo de agrupá-los, com base na constante `ORDER`, que define o peso de cada grupo. O processo de procura pelo *matching cluster* é executado pelo método `findMatch`, com base nas informações de luminância e cromância atribuídas para a classe. Após esta etapa, os grupos são ordenados por ordem de menor peso, através do método `absNhd_Sort2`.

Com os *clusters* ordenados, a etapa de classificação é executada. Está, define se o *pixel* atual pertence ao fundo ou se está em movimento. Para isso, uma nova imagem RGB é criada, onde os *pixels* que pertencem ao fundo recebem cor preta e os em movimento recebem cor branca, resultando em uma imagem binária.

3.2.2.10 Classe `Functions`

A classe `Functions` possui na sua implementação métodos abstratos, que são comuns a algumas instâncias da classe `Codec`. Estão presentes também, métodos de conversão de tipos de objetos do ambiente de programação utilizado, com a biblioteca de manipulação de imagens e vídeos utilizados no sistema.

Os métodos `drawContours` e `getObject` fazem a procura de componentes conexos dos objetos cujas as áreas cortaram a reta passada por parâmetro. Porém, o método `drawContours` apenas desenha a área e os contornos do primeiro componente conexo encontrado, enquanto que o `getObject` retorna as coordenadas dos *pixels* que compõe o contorno.

Os métodos `TDF` e `TDFInversa` são funções adaptadas do trabalho de conclusão de curso de Stivanello (2004), para o ambiente de programação Borland Builder 6.0. O método `TDF` gera os Descritores de Fourier a partir dos pontos do contorno, já o método `TDFInversa` reconstrói os pontos do contorno a partir dos Descritores de Fourier.

3.2.3 Diagrama de Seqüência

A figura 18 mostra o diagrama de seqüência, que permite visualizar o caso de uso `Configurar ambiente`.

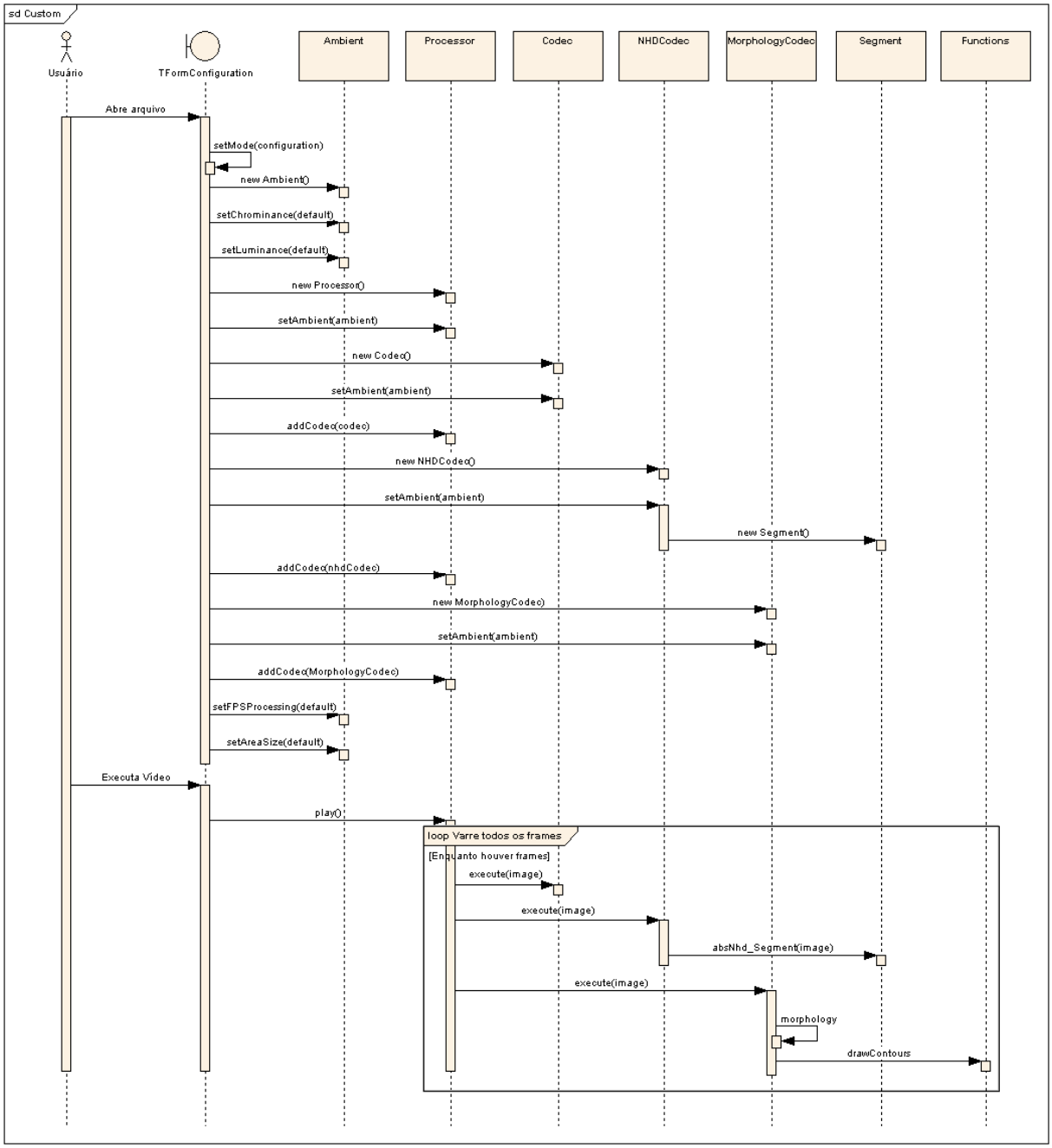


Figura 18 – Diagrama de seqüência configurar ambiente

O objeto da classe `Ambient` é criado no momento em que o usuário abre um arquivo de vídeo, atribuindo por padrão o modo configurar, para então atribuir os valores padrões de configuração que são passados para a classe `Ambient`. Depois que o ambiente está

configurado, o objeto da classe `Processor` é instanciado. Conforme explicado na seção seguinte, esta é a classe responsável pela execução dos vídeos, que fornece ao usuário as funções `Play`, `Stop` e `Pause`. Em seguida, os objetos responsáveis pelas etapas de manipulação de imagens são instanciados.

Inicialmente, o objeto da classe `Codec` é instanciado, para permitir ao usuário visualizar as imagens originais do vídeo. Em seguida, a classe responsável pela segmentação é instanciada. No diagrama acima, a classe responsável pela segmentação é a `NHDCodec` que implementa o algoritmo NHD. Porém, o usuário pode optar pela subtração de fundo, executado pela classe `SubtractionCodec`.

Para o modo configuração, as instâncias dos objetos de segmentação e `MorphologyCodec` são necessárias, para permitir ao usuário visualizar as imagens segmentadas, permitindo que o mesmo o configure caso necessário. O que muda no diagrama para os modos de treinamento e de contagem, são as instâncias dos objetos `NeuronCodec` ou `ClassificationCodec` respectivamente.

Quando o usuário seleciona a opção `Play`, a classe `Processor` itera sobre todas as instâncias de `Codec` que foram adicionadas nela, passando para elas uma cópia da imagem original, através do método execute. Como visto anteriormente, é na implementação deste método que cada etapa de manipulação das imagens é chamada.

A classe `NHDCodec` manipula a classe `Segment` para fazer a segmentação dos vídeos, pois é nesta classe que o histórico de cada *pixel* é armazenado. Após está etapa, a classe `MorphologyCodec` executa as funções de dilatação e erosão, na imagem binária resultante do processo anterior com auxilio da biblioteca do OpenCV. A classe `MorphologyCodec` chama a função `drawContours` da classe `Functions`, que procura as componentes conexas, desenha os contornos e a área dos objetos encontrados.

O cenário descrito no diagrama mostra apenas a etapa de configuração, porém na etapa de treinamento da rede neural a única diferença é a execução da classe `NeuronCodec`. Já na etapa de contagem, que realiza o reconhecimento dos objetos, esta é substituída pela classe `ClassificationCodec`.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

O ambiente de desenvolvimento utilizado para a implementação do sistema foi o Borland C++ Builder 6 (BORLAND, 2008). Esta ferramenta de desenvolvimento foi escolhida por oferecer diversos componentes necessários para a implementação, o que permitiu agilidade no desenvolvimento, com uma IDE que permite criar telas de sistema com facilidade.

A biblioteca utilizada para manipulação de imagens foi a OpenCV (SOURCEFORGE, 2008), que está descrita na seção 3.3.1.1.

A principal técnica implementada no sistema é o algoritmo NHD, cuja a rotina está descrita na seção 3.3.1.2.

3.3.1.1 OpenCV

O OpenCV é uma biblioteca livre multiplataforma, desenvolvida pela Intel no ano 2000. Escrita na linguagem C/C++, esta oferece várias ferramentas otimizadas para o desenvolvimento de aplicações de aplicativos na área da Visão Computacional. Encontra-se atualmente na versão 1.0 e segue a licença BSD da Intel, oferecendo os códigos fontes para que os desenvolvedores possam alterá-los para uso próprio.

A escolha desta biblioteca deve-se a facilidade de manipulação e pelo ótimo desempenho dos algoritmos implementados nela, o que torna o OpenCV é recomendado para aplicações em tempo real, onde a preocupação com performance é grande. Foram utilizados no sistema os módulos de Processamento de Imagens e Video I/O, Estrutura de dados, Álgebra Linear e algoritmos de Visão Computacional.

A classe utilizada para capturar os dados do vídeo, bem com os quadros do mesmo, foi o `CVCapture`. Além da captura destes dados, a instância desta classe permite capturar quadros e informações vindas de uma câmera. Quando as imagens são disponibilizadas através de uma câmera ou através de um arquivo de vídeo, é necessário obter cada um dos quadros de forma independente, através dos métodos `cvGrabFrame` e `cvRetrieveFrame`.

A estrutura de imagem utilizada no sistema foi a `IplImage`, por ser mais flexível que a classe `TBitmap` oferecida pelo ambiente de desenvolvimento. Além disso, ela é utilizada nos algoritmos de processamento de imagem pertencentes à biblioteca. Além dos algoritmos de processamento de imagem, a biblioteca possui funções de conversão entre formatos de imagens. As função utilizada para a conversão de formatos foi a `cvCvtColor`, utilizada na implementação do algoritmo NHD e nas funções de Visão Computacional oferecidas pela biblioteca.

O algoritmo de subtração de fundo foi implementado, utilizando apenas funções do OpenCV. O quadro 1 mostra a rotina de treinamento do algoritmo, utilizando para isso os primeiros quadros do vídeo, cuja a quantidade foi parametrizada pelo usuário.

```
cvSplit (img_hsv, img_h, img_s, imb_v, 0);
//Adiciona os frames de estimatica de background
cvConvertScate(img_h, img_h, 1.0/numframesT,0);
cvConvertScate(img_s, img_s, 1.0/numframesT,0);
cvConvertScate(img_v, img_v, 1.0/numframesT,0);
cvAdd(img_h, img_hA, img_hA);
cvAdd(img_s, img_sA, img_sA);
cvAdd(img_v, img_vA, img_vA);
// Caso seja o ultimo frame do treinamento
IF (currentFrame == numframes – 1)
     // merge entre as imagens
     cvMerge(img_hA, img_sA, img_vA, 0, img_hsvA);
```

Quadro 1 – Parte do algoritmo de Subtração de fundo

Outras funções de Visão Computacional oferecidas pela biblioteca foram utilizadas no sistema, como por exemplo as de morfologia matemática. O quadro 2 mostra a rotina de pós-processamento que utilizou estas funcões, implementada no método `execute` da classe `MorphologyCodec`.

```
// Função que cria os elementos estruturantes
IplConvKernel* kernel1 = cvCreateStructuringElementEx( 9, 9, 4, 4,
VC_SHAPE_ELLIPSE);
IplConvKernel* kernel2 = cvCreateStructuringElementEx( 3, 3, 1, 1,
VC_SHAPE_ELLIPSE);

//Função de dilatação baseada no primeiro elemento
cvDilate( img, img, kernel1, 1);
//Função de dilatação baseada no primeiro elemento
cvDilate( img, img, kernel1, 1);
//Função de dilatação baseada no primeiro elemento
cvDilate( img, img, kernel1, 1);
//Função de dilatação baseada no primeiro elemento
cvDilate( img, img, kernel1, 1);
```

Quadro 2 – Método de operações de morfologia matemática

Além de erosão e dilatação, a extração de componentes conexas presentes em uma imagem foram possíveis através da biblioteca OpenCV, utilizada no treinamento da Rede Neural utilizada no sistema. Também foram utilizadas funções de Álgebra Linear, para destacar contornos dos objetos e *boundigbox* que envolve os mesmos, conforme Quadro 3.

```
// Procura as componentes conexas na imagem
cvFindContours(img2, storage, &coutour, sizeof(CvContour), CV_RETR_COMP,
CV_CHAIN_APPROX_SIMPLE);
// Itera sobre as compontens conexas
while (contour != NULL) {
     int x1 = screenRect.x;
     int y1 = screenRect.y;
     int x2 = screenRect.x + screenRect.width;
     int y2 = screenRect.y + screenRect.heigth;
     // Desena boundingBox da area
     drawRectangleImage(image, x1, y1, x2, y2);
     // Desenha contornos
     cvDrawContours(image, contour, CV_RGB(0, 255, 0), CV_RGB(255, 0, 0),
0, 1, 8, cvPoint(0,0));
```

Quadro 3 – Método de desenho de área de contorno

3.3.1.2 Implementação do Algoritmo NHD

As etapas da implementação do algoritmo NHD, cuja a especificação foi apresentada na seção 2.3.1.2 e que foi implementada na classe `Segment`, está descrita no fluxograma apresentado na Figura 19.

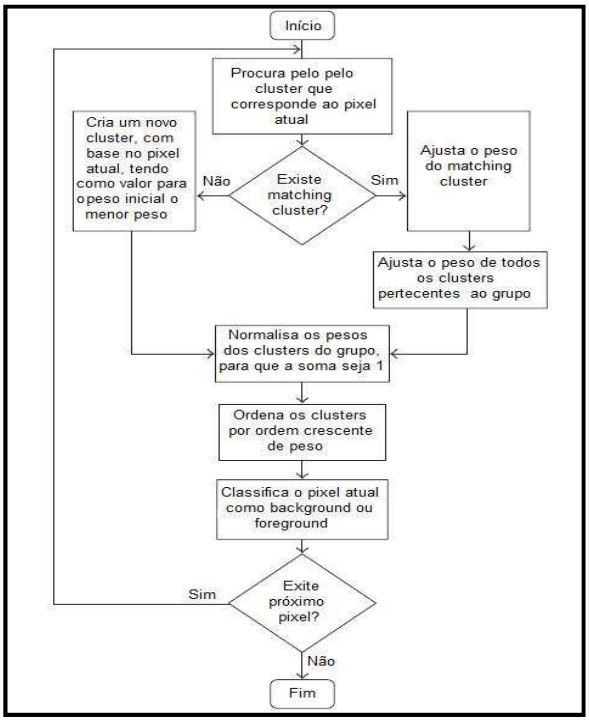


Figura 19 – Fluxograma do Algoritmo NHD

A principal etapa do algoritmo é a de procura do *matching cluster*. Esta é a rotina que define se o *pixel* pesquisado pertence ou não ao grupo de *clusters*. O quadro 4 mostra a implementação desta rotina.

```
int match;
// Varre todos os grupos de cluster
for (match = order – 1; match >= 0; match--) {
    // Calcula a distância de Manhattna entre o pixel e o cluster atual
    int lunDist = abs(ctPtr[match].y1 = ycbcr.y1) + abs(ctPtr[match].y2 –
ycbycr.y2);
    int chrDist = abs(ctPtr[match].cb = ycbcr.cb) + abs(ctPtr[match].cr –
ycbycr.cr);
    // Verifica se o resultado está dentro das faixas parametrizadas
    if (((lumDist >> (wBits)) <= lumThr) && ((chrDist >> (wBits)) <=
chrThr)){
          //Ajusta o valor do cluster atual
          ctPtr[match].y1 += ((ycbycr.y1 – ctPtr[match].y1) >> wBits);
          ctPtr[match].y2 += ((ycbycr.y2 – ctPtr[match].y2) >> wBits);
          ctPtr[match].cb += ((ycbycr.cb – ctPtr[match].cb) >> wBits);
          ctPtr[match].cr += ((ycbycr.cr – ctPtr[match].cr) >> wBits);
          break;
     }
}
```

Quadro 4 – Rotina de busca pelo Matching Cluster

.

3.3.2 Operacionalidade da implementação

Nesta seção é apresentada a operacionalidade da implementação, com suas funcionalidades. Mostra as funcionalidades de configuração, treinamento e contagem, bem como a de comparação entre os algoritmos de segmentação.

A tela exibida pela figura 20 mostra o menu da aplicação, que será utilizada por todas as telas do sistema. O usuário pode optar por uma das telas de segmentação para criar o ambiente e configurá-lo: `Algoritmo de subtração de fundo`, `Algoritmo NHD` e `Comparação de algoritmos`. Pode-se observar que a tela `Algoritmo NHD` é a que está aberta, que é similar a `Algoritmo de subtração de fundo`, possui as informações de configuração e de visualização do algoritmo.

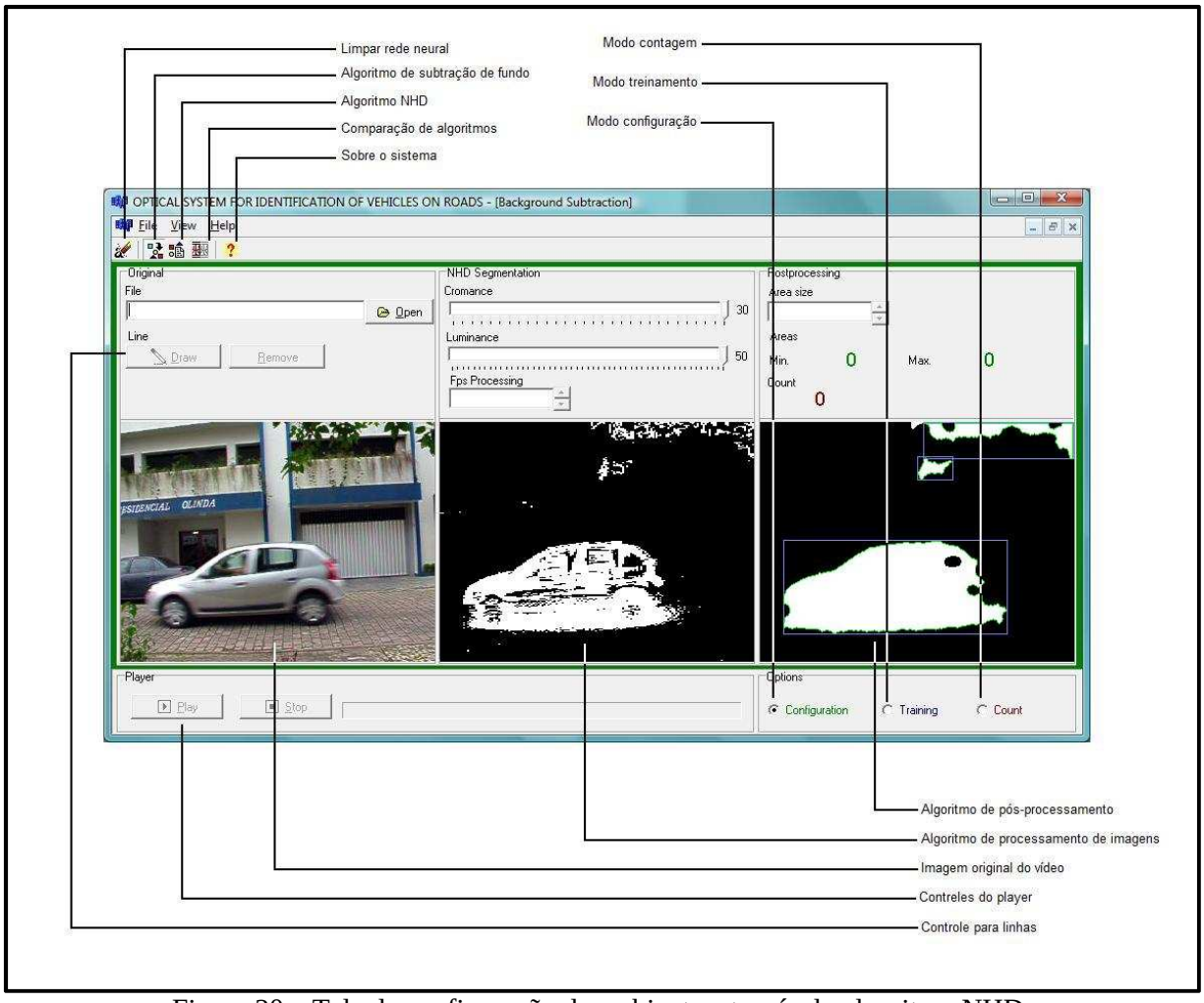


Figura 20 – Tela de configuração de ambiente, através do algoritmo NHD

Para limpar as informações da Rede neural o usuário deverá selecionar a opção `Limpar rede neural` no menu da aplicação, permitido que a mesma seja usada em ambientes diferentes, com arquivos de vídeo diferentes. A janela atualmente aberta possui a implementação dos casos de uso do sistema. Neles, o usuário pode escolher entre os modos de `configuração`, `treinamento` e `contagem`.

A aplicação fornece ao usuário, a possibilidade de desenhar as linhas que são usadas no modo `treinamento` e de `contagem`, bem como funções de `player`. Como dito anteriormente, as telas `Algoritmo NHD` e `Algoritmo de subtração de fundo` são similares, exceto quanto as informações de *threshold*.

Independente do modo escolhido, o usuário pode configurar as informações de configurações do ambiente, informações do vídeo atual e informações importantes para os algoritmos de segmentação e morfologia matemática. No modo `treinamento`, quando um objeto é encontrado, cruzando uma linha, o sistema dispara uma mensagem questionando se aquele objeto é um veículo ou não. Já o modo de contagem, mostra a quantidade de veículos

que cruzaram a linha.

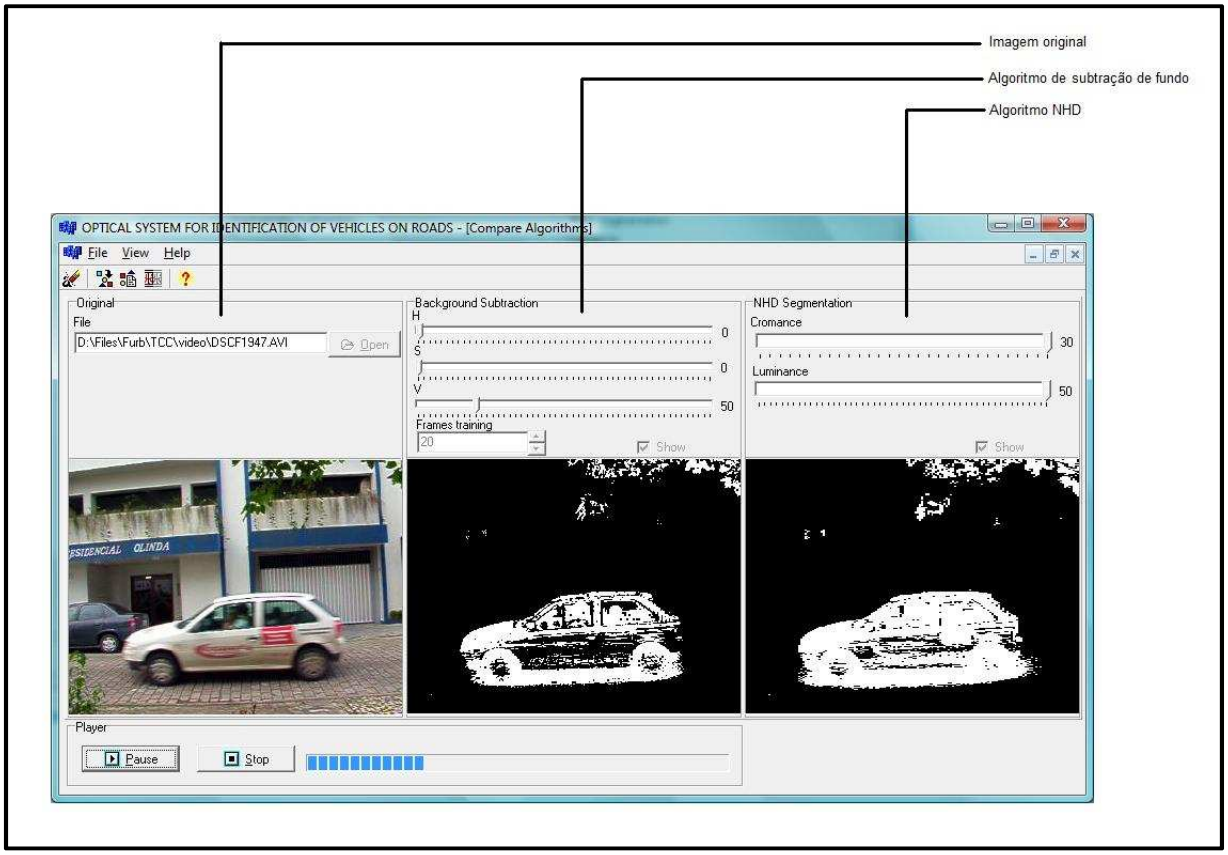


Figura 21 – Tela de comparação de algoritmos de segmentação

A tela exibida na figura 21 foi criada para permitir ao usuário comparar os algoritmos de segmentação. O primeiro quadro mostra o vídeo original, o segundo quadro mostra o resultado da segmentação do algoritmo de subtração de fundo, e o terceiro o resultado do algoritmo NHD. Existem botões do tipo *check* em cada quadro, para permitir ao usuário executar um algoritmo de cada vez.

3.4 RESULTADOS E DISCUSSÃO

Este trabalho permite observar como funciona um sistema de visão computacional internamente, mostrando as etapas de segmentação, pós-processamento e identificação através de redes neurais. Com base nessas informações, é possível identificar a viabilidade da utilização de técnicas de visão computacional podem ser aplicadas em sistemas de monitoramento.

Na seção 3.4.1 são apresentados os problemas encontrados durante o desenvolvimento do sistema, bem como os pontos positivos das estruturas utilizadas. Os testes realizados na aplicação são mostrados na seção 3.4.2, com a comparação dos algoritmos de segmentação e a eficácia da rede neural utilizada.

3.4.1 Condições da Implementação

Inicialmente, o sistema foi proposto para ser implementado na linguagem Java, utilizando o ambiente Eclipse. Porém, pela dificuldade em encontrar bibliotecas de processamento de imagens e de Visão Computacional, optou-se por utilizar a linguagem C++. Deve-se isso a utilização da biblioteca OpenCV, recomendada por desenvolvedores nesta área. Se todos os algoritmos utilizados no sistema tivessem sido programados, acarretaria no atraso do mesmo e o não atendimento no que foi proposto.

A linguagem C++ permite um controle maior sobre a memória utilizada pela aplicação. Isso foi importante para o desenvolvimento, pois manipular imagens em memória envolvem um grande custo computacional. Além disso é possível utilizar a estrutura da linguagem C, ou até mesmo Assembler, para se beneficiar da melhoria de performance destas linhas, utilizadas pelos criadores do algoritmo NHD.

A implementação do algoritmo NHD foi a maior dificuldade encontrada durante o desenvolvimento do sistema, pois para entendê-lo, foi necessário conhecer os conceitos utilizados por ele. Conceitos como o formato Y'CbCr, a compressão 4:2:2, a distância de Manhattan, entre conceitos comuns em algoritmos deste tipo.

Além destes conceitos, os artigos não traziam informações dos valores constantes utilizados pelos criadores, como tamanho dos grupos de c*luster* e os valores de *Threshold* reais utilizados. Para resolver essas questões, foi necessário entrar em contato com os autores destes artigos. Michael Bove Jr, um dos autores do artigo de BUTLER et al (2005), esclareceu dúvidas no desenvolvimento do algoritmo. Já Simon Denman, um dos autores do artigo de DENMAN et al (2006) descreveu os valores constantes utilizados, bem como técnicas de melhoria de performance.

3.4.2 Testes

Nos testes não foram utilizados ambientes complexos, visto que a quantidade de variáveis envolvidas, como número de objetos em uma mesma imagem, bem como a intersecção dos mesmos, poderá dificultar o estudo. Foram usados vídeos gerados em uma rua mão única, tendo a visão lateral dos objetos. A resolução dos vídeos gerados é de 320 X 240 *pixels*, com 30 quadros por segundo, gerados por uma câmera digital semi-profissional com auxilio de um tripé.

Os vídeos utilizados nos testes estão disponíveis em SANTOS (2008). O primeiro deles (video1.avi), mostra um ambiente comportado, havendo apenas veículos passando pelo mesmo. Já o segundo (video2.avi), possui um ambiente com veículos e pessoas passando, carro estacionando.

Para efetuar testes quanto a luminosidade do ambiente, foi criado um terceiro vídeo (video3.avi) através de uma alteração no primeiro vídeo, que após 200 quadros tem sua luminosidade aumentada. Para isso, foi utilizado o programa VirtualDub 1.7.8. Outra ferramenta utilizada foi o Super 2008 (ERIGHTSOFT, 2008), para remoção de som e compactação dos vídeos.

A seção 3.4.2.1 mostra os testes em relação aos algoritmos de segmentação, bem como pós-processamento. Na seção 3.4.2.2 são apresentados os resultados com os testes da rede neural utilizada.

3.4.2.1 Testes com Algoritmos de Segmentação

O objetivo destes testes é comparar os resultados dos algoritmos de segmentação, em diferentes luminosidades e com objetos que passaram a ser estacionários ao longo dos vídeos. Para esse teste, foram utilizados os arquivos se vídeo: video1.avi e video3.avi.

Tabela 1 – Resultados dos testes com algoritmos de Segmentação

TESTE DE PERFORMANCE			
Algoritmo	**FPS**	**Mem. Kb.**	**CPU**
Sub. Fundo	6	10.032	46%
NHD	4	14.780	53%

A tabela 1, mostra a comparação dos dados de performance dos algoritmos de segmentação. Pela tabela, é possível observar que ambos algoritmos estão lentos, mas o uso de memória e processador são baixos. A máquina onde foram realizados os testes possui sistema operacional Windows Vista, processador Athlon 64 X2 Dual Core 3800+ com 2GB de memória Ram.

A figura 22 mostra os resultados do algoritmo NHD quando um objeto que estava em movimento ficou parado. Nesta figura observa-se da esquerda para a direita, a primeira imagem mostrando o vídeo original, já a segunda imagem é de quando o objeto está começando a se adaptar ao fundo, e na terceira, mostra o veículo aparece adaptado ao fundo. Note que o motorista e a pessoa que está atrás do carro, marcados com o retângulo amarelo, ainda estão em movimento.

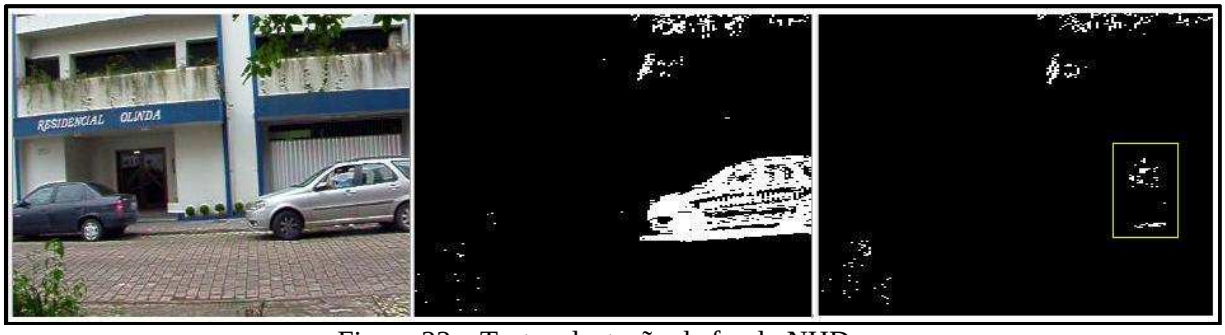


Figura 22 – Teste adaptação de fundo NHD

Na figura 23, é possível observar o momento em que o carro saiu do estado parado para movimento. A sua sombra ainda encontra-se parada, porém depois de alguns segundos a mesma é adaptada ao fundo. A primeira imagem mostra o vídeo original no momento em que o veículo mudou de estado, já a segunda imagem mostra o resultado do algoritmo NHD neste mesmo momento. A terceira e a quarta imagens mostram o veículo após a adaptação do fundo, com a imagem original do vídeo e o resultado do algoritmo NHD respectivamente. Como explicado na seção 2.3.1.2, o algoritmo de subtração de fundo não possui suporte a adaptação de fundo, o que o torna inviável para sistemas de monitoramento, uma vez que um veículo pode ficar parado por horas.

Figura 23 – Objeto entrando em movimento

O algoritmo NHD apresentou-se mais eficaz quando da alteração de luminosidade durante o monitoramento, conforme figura 24, onde a primeira imagem representa o vídeo original. A segunda imagem mostra o resultado do algoritmo de subtração de fundo após alguns segundos da alteração de luminosidade. Nota-se que o mesmo possui interferência de luz, enquanto que a terceira figura, que representa o algoritmo NHD no mesmo intervalo de tempo, foi totalmente adaptado.

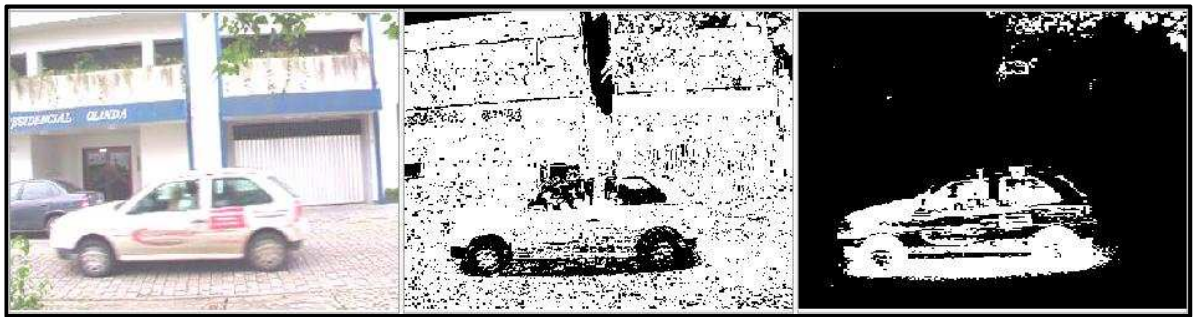


Figura 24 – Adaptação de alteração de luminosidade

3.4.2.2 Teste com a Rede Neural

Assim como no sistema desenvolvido por Stivanello (2004), para todos os testes foram utilizados 15 descritores de Fourier e a camada oculta da rede neural contendo 15 neurônios. Este mesmo número de neurônios foi utilizado na camada de entrada. Como a saída deve assumir um entre dois estados possíveis (é carro e não é carro), foi utilizado na camada de

saída apenas 1 neurônio.

Porém, a recomendação estabelecida para que o número de amostras seja 480 para esta rede não foi estabelecido, pois os vídeos utilizados são muito pequenos, possuindo um número de 10 amostras no máximo. Recomenda-se treinar a rede utilizando amostras positivas e negativas, portanto todas as etapas de treinamento da rede foram baseadas no segundo vídeo (video2.avi). Para testar a rede neural foram realizados quatro casos de teste, visando cobrir a maior quantidade de cenários de testes possíveis.

Para o primeiro caso de teste, a rede foi treinada utilizando o resultado do algoritmo de subtração de fundo. A figura 25 mostra as imagens segmentadas cujos objetos que cortam a linha vermelha não são considerados veículos (como ruídos e pessoas).

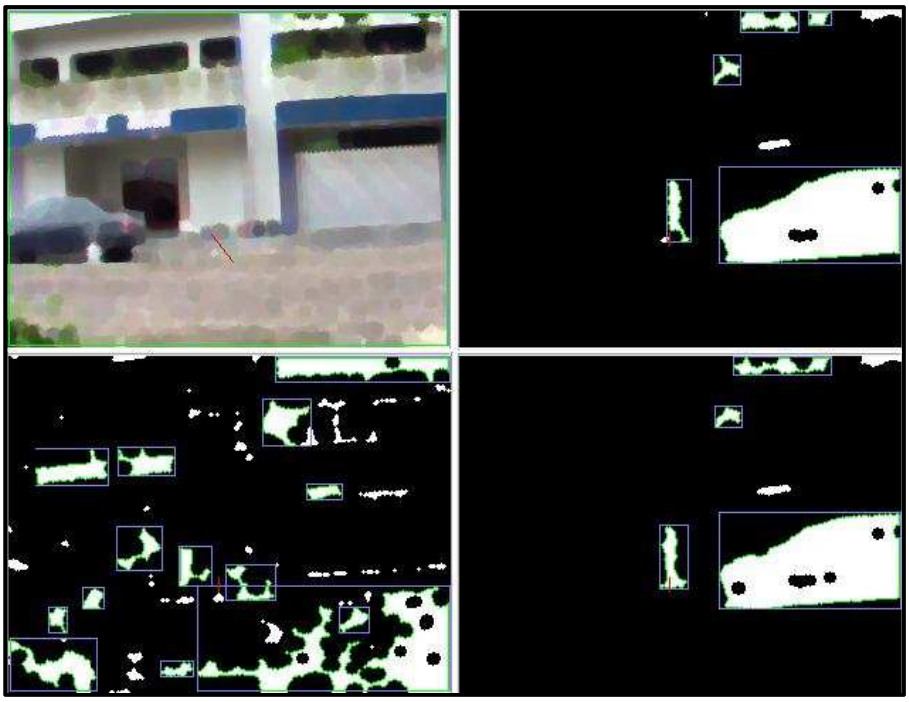


Figura 25 – Exemplos de objetos reprovados pela subtração de fundo

Já na figura 26 encontram-se os objetos considerados automóveis que passaram pela linha vermelha. Porém, como o algoritmo de subtração de fundo é sensível a luminosidade, deve-se levar em consideração que o ambiente deverá possuir sempre a mesma iluminação no momento de treinamento e no de reconhecimento.

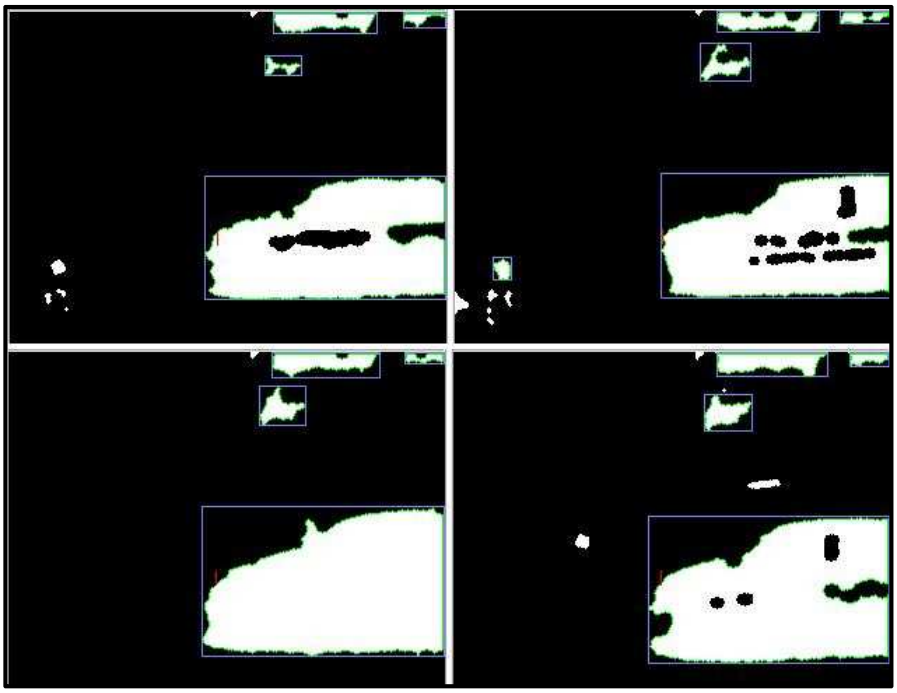


Figura 26 – Exemplos de objetos aprovados pela subtração de fundo

No primeiro caso de teste, o treinamento da rede foi um pouco lento, assim como descrito por Stivanello (2008), levando noventa segundos. Após o treinamento, quando verifica a contagem de veículos, a rede neural satisfez o desempenho previsto. Todos os objetos do tipo carro foram contados, e os que não eram foram ignorados.

Para o segundo caso de teste, foi utilizada a rede neural treinada pelo resultado do algoritmo de subtração de fundo, com a contagem de veículos geradas pelo algoritmo NHD. Uma ocorrência de veículo não pode ser identificada e alguns ruídos (provocados na finalização do vídeo) foram contados como sendo um carro, tendo apenas 66,6% dos objetos identificados corretamente. Como dito na seção anterior, o algoritmo NHD é mais sensível que o algoritmo se subtração de fundo quando a câmera treme.

No terceiro caso de teste, o objetivo é identificar a viabilidade de aplicar os algoritmos de segmentação podem ser efetuados em uma seqüência com um passo maior entre os frames. Para isso, a rede neural foi treinada através dos resultado do algoritmo NHD a cada 1 quadro por segundo. No momento da contagem, o ambiente foi configurado para que a segmentação, bem como o algoritmo de reconhecimento, fossem executados a cada 10 quadros por segundo. A figura 27 mostra as imagens segmentadas no momento do treinamento que não são considerados carros.

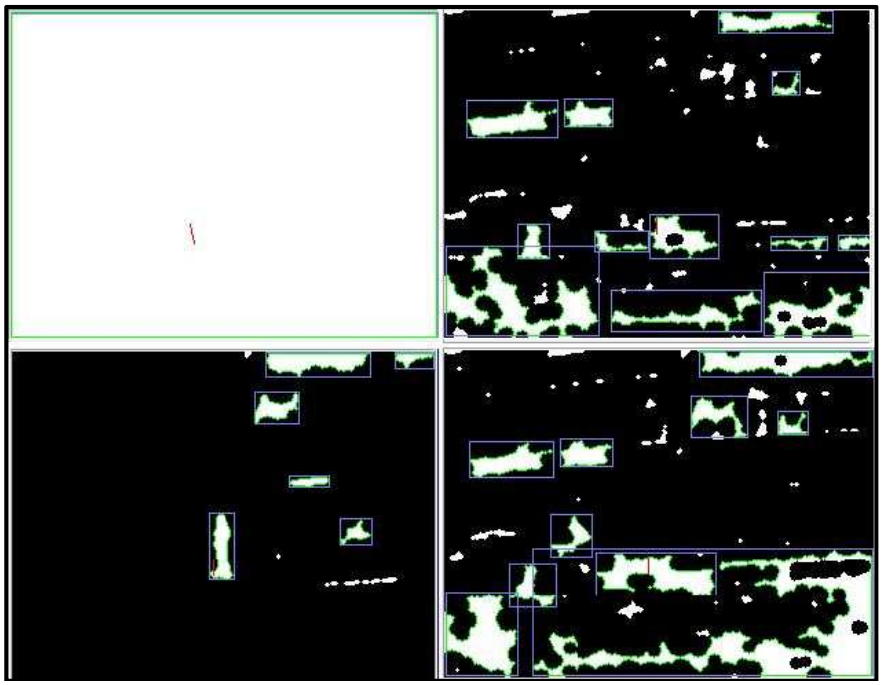


Figura 27 – Exemplos de objetos reprovados pelo algoritmo NHD

Já a figura 28 apresenta os objetos considerados veículos, após cruzarem a linha vermelha definida pelo usuário. Como previsto, alguns carros não puderam ser identificados e contados, pois os mesmos não apareceram totalmente no vídeo quando entraram em contato com a linha vermelha, deixando uma parte de sua área de fora. O percentual de acerto do terceiro caso de teste foi de 81,8%.

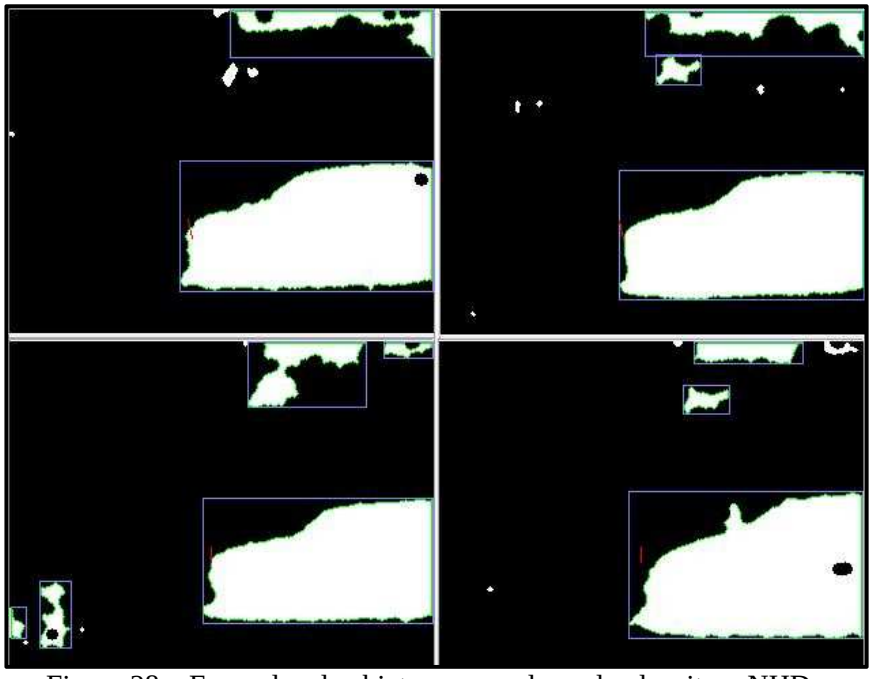


Figura 28 – Exemplos de objetos aprovados pelo algoritmo NHD

Para o ultimo caso de teste da rede neural, foi verificado se os exemplos de carros de um vídeo podem ser utilizados em outro. Para isso, a rede neural treinada no terceiro caso de

testes foi submetida ao primeiro arquivo de vídeo. Foi possível observar que, mesmo com uma pequena quantidade de objetos utilizados no treinamento, o algoritmo conseguiu contar a maioria dos veículos, possuindo 90,9% de acerto.

Os dados de cada caso de teste realizado podem ser visualizados na tabela 2.

Tabela 2 – Resultados dos casos de teste com rede neural

RESULTADOS DOS TESTE		
Caso de teste	**Número de objetos**	**Percentual de acerto**
Teste 1	9	100%
Teste 2	9	66,6%
Teste 3	11	81,8%
Teste 4	11	90,9%

4 CONCLUSÕES

Este trabalho apresentou dois métodos para identificação de veículos em ruas, bem como o desenvolvimento de um sistema para investigação, através da contagem de carros em uma estrada. Para realizar tal tarefa foram utilizadas diversas técnicas de processamento de imagens, além de uma rede neural para o reconhecimento.

A escolha do algoritmo de Adaptação de Fundo NHD mostrou-se melhor que Remoção de Fundo, que deixam a aplicação “presa” ao ambiente observado. O algoritmo de Remoção de Fundo falha quando existe a possibilidade de um objeto ficar parado na cena, dificultando a segmentação dos objetos em movimento. O algoritmo NHD mostrou-se eficaz em ambientes reais, permitindo uma flexibilidade quanto a luminosidade, tanto causadas pelo sol quanto por objetos que geram iluminação.

Aliada a etapa de segmentação, a etapa de pós-processamento contribui para a diminuição de ruídos na imagem, o que foi fundamental para a etapa de pesquisa por componentes conexos. O parâmetro que permite ignorar imagens com tamanho inferior a um determinado valor permite um filtro ainda maior, em relação a quantidade de objetos relevantes a imagem.

A utilização dos descritores de Fourier para a descrição de formas, baseado no TCC de Stivanello (2004), permitiu representar as formas a partir de uma quantidade pequena de descritores, útil para etapa de interpretação. Além disto, o fato da técnica ser invariante quanto a translação e a rotação dos veículos, amplia a possibilidade do sistema ser utilizado em cruzamentos.

Já a utilização de redes neurais do tipo Perceptron Multicamadas, implementada por Stivanello (2004), baseou-se na generalização da rede e pela facilidade de adaptação ao sistema implementado. A velocidade do treinamento é proporcional ao grau de diferença entre os objetos tidos como aprovados e os reprovados.

Finalmente, é importante comentar que, apesar dos resultados do sistema serem satisfatórios, o mesmo é muito lento para ser executado em uma aplicação de tempo real. A etapa de segmentação dos objetos é lenta, principalmente quando usado o algoritmo NHD. Como visto na seção 3.4.2.2, torna-se fundamental utilizar toda a área do veículo para utilizar a rede, pois se apenas uma área do mesmo for considerada pode acarretar em falhas no reconhecimento, como ocorreu no salto de quadros.

4.1 EXTENSÕES

A abrangência dos sistemas de visão computacional possibilita que o presente trabalho seja estendido com diferentes objetivos, principalmente para objetos em movimento. Isso foi levado em consideração durante o desenvolvimento do sistema, na escolha das ferramentas utilizadas e na independência dos principais algoritmos utilizados.

Inicialmente, para aumentar a flexibilidade da aplicação e proporcionar a portabilidade do código, torna-se necessário o estudo de outras toolkits para o desenvolvimento da sua interface, tais como IUP, wxWidgets, GLUI. Como visto anteriormente, o ambiente Borland Builder utilizado foi escolhido devido as suas facilidades (já apresentadas) e pelo conhecimento deste ambiente pelo desenvolvedor

A rotina de segmentação deve ser estudada, visando melhorar a performance da mesma, para que seja possível utilizá-la em tempo real. Estas rotinas ocuparam pouca memória durante suas execuções, o que permite investigar se a criação de mais estruturas temporárias podem melhorar a performance. Mesmo sabendo que nem sempre a relação de aumentar o uso de memória ira refletir numa melhora da performance.

Além disso, o sistema pode ser melhorado, implementando outros algoritmos, como para perseguição de objetos na imagem e tratamento de colisão de objetos. Ou até mesmo, adaptar outro algoritmo de segmentação, visando comparar com os que foram utilizados, bastando apenas estender a classe `Codec`.

A rede neural poderia ser melhorada, permitindo identificar diferentes tipos de objetos, como carros, motos e pessoas dentro do cenário proposto, permitindo que o mesmo seja usado em cruzamentos e vias complexas. Existem também outras técnicas para identificação de objetos que poderiam ser estudadas, algumas destas permitem diferenciar modelos de automóveis conforme XIAOXU (2004).

Em sistemas de monitoramento inteligente é comum o estudo de comportamento de objetos, como barreiras virtuais (semelhante a utilizada para contar veículos), velocidade e trajetória dos objetos. A estrutura poderia ser aproveitada também para analisar outros tipos de objetos e em cenários diferentes, como foi descrito na seção 2.1. As técnicas utilizadas aqui podem ser aplicadas também em sistemas multicâmeras, onde um objeto pode passar de uma câmera para a outra e o sistema se encarrega de acompanhá-lo, conforme trabalho realizado por DENMAN et al. (2006).

REFERÊNCIAS BIBLIOGRÁFICAS

AZEVEDO, Eduardo; CONCI, Aura; LETA, Fabiana R. **Computação gráfica:** teoria e prática. Rio de Janeiro: Elsevier, 2008.

BORLAND. **C++ Builder**. [S.l], 2008. Disponível em: <http://www.codegear.com/products/cppbuilder>. Acesso em: 03 nov. 2008.

BUTLER, Darren E. et al. **Real-time adaptive background segmentation.** Hindawi, 2005. Disponível em: <http://www.hindawi.com/getarticle.aspx?doi=10.1155/asp.2005.2292>. Acesso em: 10 set. 2008.

CÉSAR JR, Roberto M.; COSTA, Luciano F. **Shape analysis and classification.** 1. ed. Boca Raton: CRC Press, 2001.

COMPANHIA HIDRO ELÉTRICA DO SÃO FRANCISCO. **Adaptação de sistemas CFTV para aplicações de monitoramento inteligente.** [S.l.], 2004. Disponível em: <http://www.chesf.gov.br/downloads/pesquisadesenvolvimento/Projetos/PP_1.pdf>. Acesso em: 03 abr. 2008.

DENMAN, Simon et al. **Multi-view Intelligent Vehicle Surveillance System.** Brisbane, 2006. Disponível em: <http://portal.acm.org/citation.cfm?id=1190541>. Acesso em: 11 nov. 2008.

ERIGHTSOFT. **Super 2008**. [S.l], 2008. Disponível em: <http://www.erightsoft.com/SUPER.html >. Acesso em: 03 nov. 2008.

FACON, Jacques. **Processamento e análise de imagens.** 1. ed. Embalse: EBAI, 1993.

FERNANDES, Leandro A. F. **Protótipo de sistema óptico de captura do movimento humano, sem a utilização de marcações especiais.** 2002. 99 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GARCIA, Mary L. **The design and evaluation of physical protection systems.** Burlington: Bristsh Libary, 2001.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais.** Tradução de Roberto Marcondes Cesar Junior, Luciano da Fontoura Costa. São Paulo: Edgard Blücher, 2000.

GREEN, Michael. **Recognising a pattern-featured object in a RoboCup environment.** [S.l.]: Lund Universty, 2003. Disponível em: <http://ai.cs.lth.se/xj/MichaelGreen/report0815.pdf >. Acesso em: 01 abr. 2008.

HORPRASERT, Thanarat; HARWOOD, David; DAVIS, Larry S. **A statical approach for real-time robust background subtaction and shadow detection**. Germantown, 2000. Disponível em: <http://www.ces.clemson.edu/~stb/ece904/spring2006/HorprasertACCV2000.pdf>. Acesso em: 06 set. 2008.

INVISYS. **Sistemas de visão computacional.** [S.l], 2008. Disponível em: <http://www.invisys.com.br/vcbi.html>. Acesso em: 26 ago. 2008.

LOESCH, Cláudio; SARI, Solange T**. Redes neurais artificiais:** fundamentos e modelos. Blumenau: EdiFURB, 1996.

OBJECT VIDEO. **Homeland:** critical times require critical protection. [S.l.], 2008. Disponível em: <http://www.objectvideo.com/solutions/homeland/>. Acesso em: mar. 2008.

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

PORTAL DA AUTOMAÇÃO. **Segurança:** setor de segurança cresce 20% ao ano no Grande ABC. [S.l.], [2004?]. Disponível em: <http://www.quimera.com.br/materia_266.asp>. Acesso em: 01 abr. 2008.

PROGRAMA institucional. Produção partido Democratas. [S.l.], fev. 2008. 1 vídeo (10 min). Disponível em: <http://www.democratas.org.br/videos?>. Acesso em: 05 mar. 2008.

REIS JUNIOR, Ademar S.; SOARES FILHO, Milton. **Aplicação de processamento de imagens a sistemas de segurança.** Curitiba, 2002. Disponível em: <http://www.ademar.org/texts/seguranca-processamento-imagens.pdf>. Acesso em: 31 mar. 2008.

RIBEIRO, Fernando F. S.; LIMA, Antonio C. C. Detecção de volume de tráfego de veículos proporcionada por visão computacional via redes neurais. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 4., 1999, São José dos Campos. **Anais...** São José dos Campos: ITA, 1999. p. 96-101. Disponível em: <http://www.ele.ita.br/cnrn/artigos-4cbrn/4cbrn_024.pdf>. Acesso em: 03 abr. 2008.

SANTOS, Daniel. **Sistema óptico para identificação de veículos em estradas**. [S.l], nov. 2008. Disponível em: <http://www.inf.furb.br/~ddsbl/>. Acesso em: 03 nov. 2008.

SOURCEFORGE. **Open computer vision library**. [S.l], out. 2008. Disponível em: <http://sourceforge.net/projects/opencvlibrary>. Acesso em: 03 nov. 2008.

STAUFFER, Chris; GRIMSOM, Eric L. W. **Adaptive background mixture models for real-time tracking.** Cambridge, 1999. Disponível em: <http://www.ai.mit.edu/projects/vsam/Publications/stauffer_cvpr98_track.pdf>. Acesso em: 01 abr. 2008.

STIVANELLO, Maurício E. **Inspeção industrial através de visão computacional.** 2004. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TAFNER, Malcon A. **Redes neurais artificiais:** introdução e princípios de neurocomputação. Blumenau: EdiFURB, 1996.

XIAOXU, Ma; GRIMSON, Eric. **Edge-based rich representation for vehicle classification.** Cambridge, 2004. Disponível em: <people.csail.mit.edu/xiaoxuma/proj/vehi_reco/MaGrimson_ICCV05_VehicleReco.pdf>. Acesso em: 16 nov. 2008.

ZHANG, Hongming et al. **Object detection using spatial histogram features**. Harbin, 2006. Disponível em: <http://www.jdl.ac.cn/project/faceId/articles/RRJDL_HongmingZhang_IJCNN05.pdf>. Acesso em: 06 set. 2008.