UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Um Equipamento Eletrônico de Monitoramento de Velocidade

por

MARCO ANTÔNIO AMARAL FÉRIS

Trabalho de conclusão de mestrado apresentado como requisito parcial para a obtenção do grau de Mestre em Informática

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, dezembro de 2003

CIP – Catalogação na Publicação

Féris, Marco Antônio Amaral
Um Equipamento Eletrônico de Monitoramento de Velocidade / por Marco Antônio Amaral Feris. – Porto Alegre: PPGC da UFRGS, 2003.
77p.: il.
Trabalho de Conclusão de Mestrado – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Geyer, Cláudio Fernando Resin.
1. Tecnologia. 2. Tráfego. 3. Controle. 4. Mercado. 5. Imagem. I. Geyer, Cláudio Fernando Resin. II. Título

Agradecimentos

Antes de tudo, gostaria de fazer os seguintes agradecimentos às pessoas que contribuíram de uma forma significativa neste trabalho:

Aos meus pais, Dulce Helena e Antônio, que durante toda a vida se esforçaram ao máximo para fornecer a melhor educação e sempre orientaram a crescer através do trabalho, retidão e equilíbrio;

E ao Márcio Faccin, um profissional dedicado à área de processamento de imagens, por estar sempre à disposição para troca de idéias.

Sumário

Lista de Abreviaturas

A/D	Analógico-Digital
ANSI	*American National Standards Institute*
API	Interface para Programação de Aplicações
ATM	*Asynchronous Transfer Mode*
BD	Banco de Dados
BDE	*Borland Database Engine*
DCL	Data Control Language
DDL	*Data Definition Language*
DEC	*Digital Equipament Corporation*
DIB	*Device Independent Bitmap*
DML	*Data Manipulation Language*
INMETRO	Instituto Nacional de Metrologia, Normalização e Qualidade Industrial
POO	Programação Orientada a Objetos
RGB	Sigla formada pelas iniciais das seguintes cores em inglês: Red (vermelho), Green (verde) e Blue (azul).
SGBD	Sistema Gerenciador de Bancos de Dados
SGBDR	Sistema Gerenciador de Bancos de Dados Relacionais
VESA	*Video Eletronics Standards Association*

Lista de Figuras

Lista de Tabelas

Resumo

A velocidade de veículos em vias públicas pode ser obtida de diversas formas. A técnica mais usada é de laços magnéticos, onde se instalam sensores sob o asfalto. Entretanto, esta técnica apresenta desvantagens, tais como, a não detecção de motocicletas (o campo magnético gerado por este tipo de veículo é imperceptível ao sistema) e dificuldade de manutenção da via (se o órgão publico tiver que mexer numa rede cloacal que passa perto dos sensores, por exemplo, pode ser necessário reinstalá-los). Nesse contexto, este trabalho propõe-se a discutir uma nova maneira de se calcular a velocidade de veículos, através do processamento de imagens.

Para isto, torna-se fundamental conhecer os conceitos que envolvem as técnicas mais utilizadas (além dos laços magnéticos, a captura de dois quadros consecutivos e o sistema Doppler), os equipamentos disponíveis no mercado (Pardais, Lombadas Eletrônicas, Bandeiras, Caetanos e Radares) e a forma como o INMETRO faz a aferição destes equipamentos.

O estudo apresenta, igualmente, os principais fundamentos relacionados ao processamento digital de imagens, com especial atenção para detecção de bordas, de forma que seja possível avaliar a nova técnica proposta, que calcula a velocidade a partir de um único quadro. O presente trabalho objetiva apresentar o Pardalzito como alternativa técnica inovadora para aplicação de um, sistema que implementa esta idéia na prática.

Palavras-chave: tecnologia, tráfego, controle, mercado, imagem.

TITLE: “AN ELECTRONIC DEVICE TO CALCULATE VEHICLE’S SPEED”

Abstract

There are many ways to calculate vehicle’s speed in roads and streets. The most popular technique applies magnetic sensors. However, this technique presents some disadvantages: it doesn’t detect motorcycles (due to low magnetic field) and the maintenance is expensive (due to sensors are installed under the asphalt). In this context, the present work intends to propose a new technique to calculate vehicle’s speed through image processing.

To achieve this goal it is very important to review the most applied technologies (besides magnetic sensors, a capture of two consecutives snapshots and Doppler system), to know the equipments available in the market (such as radars) and how INMETRO, government’s department, verifies the accuracy of these equipments.

The main concepts involving digital processing of images are presented, with special attention for detention of edges. The objective of this study is to present Pardalzito as an innovative technique for vehicle’s speed calculation.

Keywords: technology, traffic, control, market, image.

1 Introdução

É apresentado um processo para cálculo da velocidade de veículos através do processamento de imagens, utilizando apenas um quadro gerado por uma câmera digital e compara-se com sistemas similares disponíveis hoje no mercado.

1.1 Apresentação

Este trabalho refere-se às atividades realizadas pelo mestrando Marco Antônio Amaral Féris, sob a orientação do professor Dr. Cláudio Fernando Resin Geyer, durante os anos de 2001 a 2003 no Instituto de Informática da Universidade Federal do Rio Grande do Sul.

Os equipamentos eletrônicos de monitoramento de velocidade são destinados principalmente à segurança no trânsito e o objetivo principal é fazer com que os motoristas reduzam a velocidade nos locais onde estiverem instalados sob pena de serem multados. Além disto, geram relatórios que contribuem para controlar o fluxo de veículos na via.

As características técnicas desejáveis neste tipo de equipamento são:

- Gabinete: com tratamento anticorrosivo, à prova d'água, refrigerado, com abertura chaveada para acesso aos seus componentes, protegido contra altas temperaturas, exposição a sol intenso, umidade, salinidade e vibração:
- Computador: que seja confiável, com disco rígido de alta capacidade;
- No-break: para garantir a integridade dos dados e sem precisar interromper as atividades no caso de falta de energia elétrica;
- Imagens com grande precisão e nitidez;
- Compressão de dados e imagens;
- Proteção contra sobrecarga de tensão e corrente;
- Fácil manutenção.

1.2 Motivação

Durante as aulas do Mestrado Profissional em Engenharia da Computação da UFRGS, houve uma discussão bastante fértil na turma a respeito de qual seria o tema da dissertação.

Os professores apresentavam os projetos que estavam sendo desenvolvidos e uma das áreas mais interessantes foi a de trânsito, coordenada pelo professor Cláudio Geyer.

Não era objetivo fazer um trabalho apenas teórico e sim desenvolver um sistema que trouxesse algo novo para a sociedade e que pudesse ser implementado na prática. E trabalhar nesta área significava aprimorar conhecimentos, dada a diversidade de tecnologias utilizadas.

Além do mais, o fato de voltar a projetar *software* também era um fator estimulante e saudosista, pois há seis anos que isto não mais ocorria em função das novas responsabilidades profissionais assumidas (na Altus-www.altus.com.br, como coordenador de projetos na área de automação industrial e na Dell-www.dell.com, como *project manager* na área de introdução de novos produtos).

Compondo estes fatores, surgiu a idéia de se desenvolver um novo processo para se calcular a velocidade de veículos, com características superiores aos equipamentos encontrados no mercado como, por exemplo, a eliminação da instalação de sensores sob a via pública, que proporciona maior mobilidade e a diminuição de custos de instalação e manutenção.

1.3 Objetivos

O objetivo deste trabalho é propor uma nova forma para detectar a velocidade de veículos.

Para isto, discutiremos as tecnologias usadas e equipamentos disponíveis hoje no mercado (bem como suas limitações), as tecnologias a serem empregadas nesta nova forma (com ênfase ao processamento de imagens) e na seqüência será descrito o **Pardalzito**, um sistema que implementa esta idéia.

1.4 Estrutura do Texto

O calculo de velocidade de veículos utilizando processamento de imagens baseou-se em tecnologias conhecidas, mas combinadas de tal forma que permitiram obter um resultado inédito.

Este trabalho está dividido em seis capítulos:

- O capítulo dois apresenta as tecnologias usadas nos equipamentos detectores de Velocidade e os modelos existentes no mercado;
- O capítulo três aborda o processamento de imagens, com destaque para as técnicas de reconhecimento de bordas;
- O capítulo quatro apresenta um processo para se calcular a velocidade utilizando processamento de imagens, dando destaque à formula baseada no reconhecimento de padrões;
- O capítulo cinco apresenta o Pardalzito, uma aplicação desenvolvida para mostrar de forma prática como se calcula velocidade de veículos utilizando processamento de imagens;
- Para finalizar, segue a conclusão, os caminhos que este trabalho abre para pesquisas futuras, a bibliografia.

E três apêndices:

- O apêndice A apresenta as técnicas complementares que embasaram este trabalho e as ferramentas utilizadas para o desenvolvimento do aplicativo;
- O apêndice B apresenta a portaria nro115 INMETRO, que regulamenta a aferição dos equipamentos;
- E o apêndice C contém o Manual de Operação do Pardalzito.

2 Equipamentos Detectores de Velocidade

Este capítulo tem como objetivo apresentar as tecnologias existentes para detectar velocidade de veículos, os tipos de equipamentos disponíveis no mercado e comentar sobre a aferição dos mesmos.

2.1 Estado da Arte

Existem três formas de detecção de velocidade: através de laços magnéticos, através de processamento de imagem, capturando dois quadros consecutivos e sistema Doppler, que serão discutidas a seguir:

2.1.1 Laços Magnéticos

Instalam-se laços magnéticos sob o asfalto como sensores para detectar veículos. Junto a eles existe um sistema computadorizado que fica gerando uma freqüência constante. Quando um veículo passa sobre os sensores, esta freqüência altera-se e detecta-se a velocidade do veículo. Veja na figura 2.1 a localização dos laços magnéticos colocados sob a via:

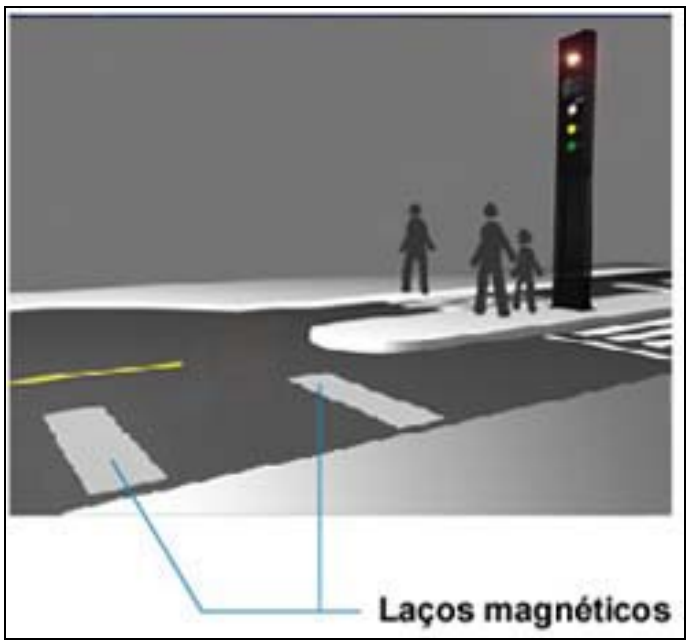


FIGURA 2.1 - Laços magnéticos para detectar velocidade

Observação: para evitar falhas de detecção, colocam-se três laços magnéticos para cada pista de rolamento obtendo-se então três cálculos distintos e consecutivos da velocidade empreendida por cada veículo monitorado.

Desvantagens

As desvantagens deste sistema são:

- **Tolerância** de até 7km/h;
- **Sistema fixo**, pois é necessário embutir os sensores magnéticos na pista;
- Custo para instalação destes sensores;
- **Pode não detectar veículos leves**, tais como motos, em função de não gerar campo magnético suficiente para detecção;
- **Dificuldade de manutenção**: Se o órgão público precisar recapear o asfalto ou abrí-lo para colocação de canos de esgoto, por exemplo, pode ser necessário reinstalar estes sensores.

2.1.2 Captura de Dois Quadros Consecutivos

A imagem de cada veículo é captada duas vezes, de forma superposta e consecutiva, na ordem de milisegundos. Com a comparação entre os quadros, verifica-se a quantidade de pixels deslocados para calcular a velocidade do veículo. Veja na figura 2.2 as imagens capturadas sobrepostas:

FIGURA 2.2 - Imagem Entrelaçada

Observação: as capturas de imagens são feitas por câmeras apontadas para cada pista de rolamento. Por exemplo: se a via tiver 4 pistas, serão necessárias 4 câmeras.

Desvantagens

As desvantagens deste sistema são:

- **Tolerância** de até 7km/h;
- **Possibilidade de não calcular ou calcular a velocidade de forma errada.** Caso o computador seja lento ou estiver executando uma tarefa interna mais prioritária (afinal normalmente não se utilizam sistemas operacionais de tempo real), existe a possibilidade de quando o sistema for capturar o segundo quadro o veículo já tenha passado e não consiga calcular a velocidade. Também pelos mesmos motivos, se aparecer um outro veículo em seguida, o sistema pode se confundir e calcular da velocidade de forma errônea (ele pode considerar o primeiro quadro sendo um veículo e segundo sendo outro).

2.1.3 Sistema Doppler

Uma antena fica propagando permanentemente ondas de freqüência tipo Doppler que quando em contato com qualquer corpo, retornam a este mesmo emissor possibilitando o cálculo da velocidade. Veja na figura 2.3 um radar que utiliza o sistema Doppler:

FIGURA 2.3 - Radar Doppler

Observação: também para evitar falhas de detecção, o cálculo é realizado três vezes.

Desvantagens

As desvantagens deste sistema são:

- **Tolerância** de até 7km/h;
- **Necessita que haja alguém operando** este equipamento, que pode estar instalado num tripé ou dentro de um automóvel.

Basicamente estes radares podem ser instalados num tripé ou dentro de um veículo. Veja estas duas formas nas figuras 2.4 e 2.5:

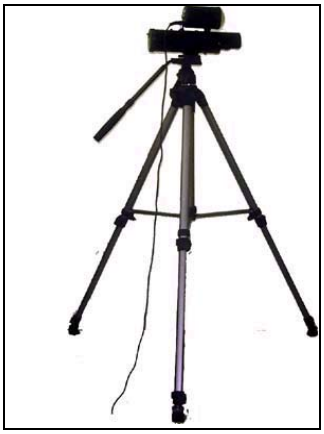


FIGURA 2.4 - Radar instalado num tripé

FIGURA 2.5 - Radar Instalado dentro de um automóvel

2.2 Tipos de Equipamentos

As tecnologias acima empregadas geraram diversos equipamentos para detecção de velocidade: Pardais, Lombadas Eletrônicas, Bandeira, Caetano e Radar, que serão descritos a seguir:

2.2.1 Pardais

Os Pardais detectam a velocidade através de laços magnéticos e procuram ficar escondidos ao lado da via. Veja na figura 2.6 um pardal instalado numa rodovia e a câmera em destaque:

FIGURA 2.6 – Pardal

2.2.2 Lombadas Eletrônicas

As Lombadas Eletrônicas são equipamentos que substituem as lombadas tradicionais (também conhecidas como quebra-molas), mas que preservam o automóvel de possíveis danos na suspensão e evitam acidentes. Permitem também a captação de imagens durante a noite, de veículos na contramão e acima de 100 km/h. Veja na figura 2.7 uma Lombada Eletrônica instalada numa rodovia:

FIGURA 2.7 – Lombada Eletrônica

Modelos

Existem dois tipos de lombadas: Totem e Pórtico, ambas medem a velocidade de veículos em uma ou mais faixas de rolamento de mesmo sentido ou oposto. Veja nas figuras 2.8 e 2.9 os modelos de lombadas do tipo totem e pórtico:

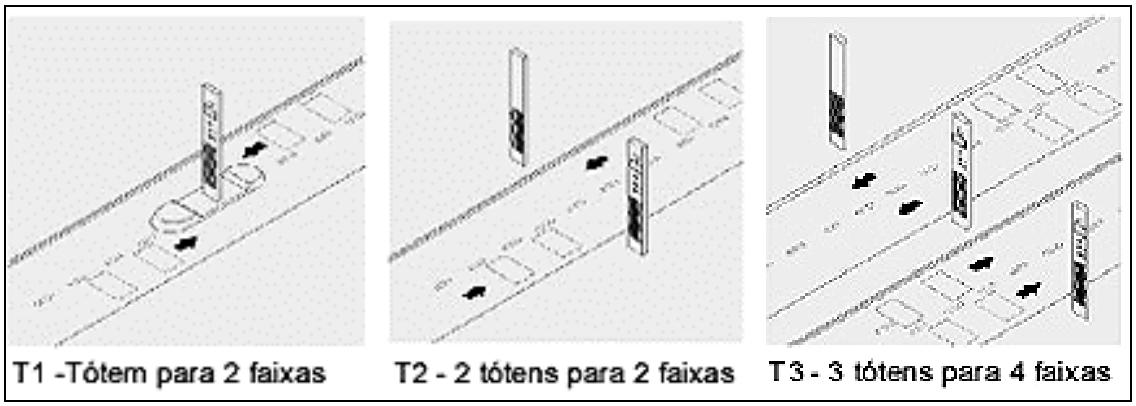


FIGURA 2.8 – Lombadas Eletrônica modelo totem

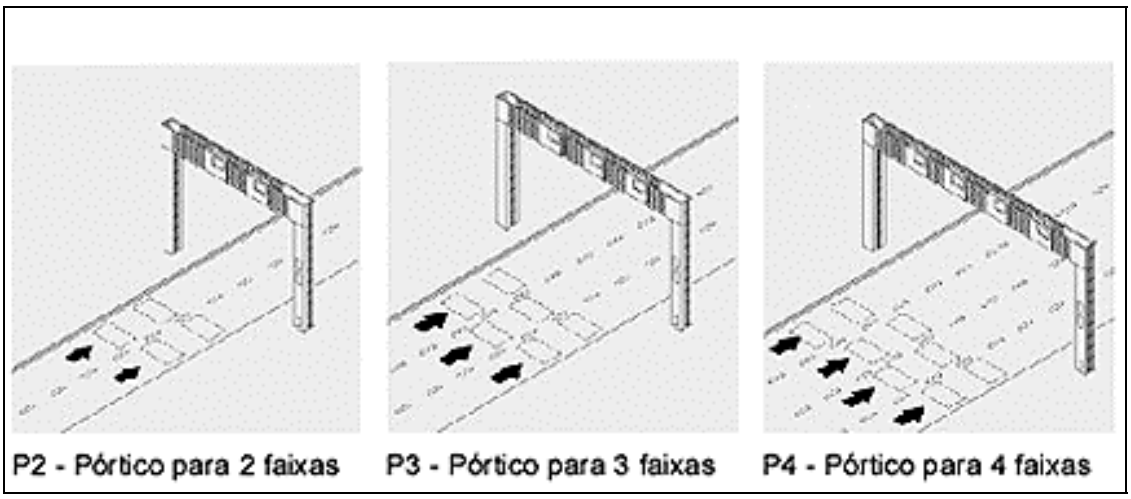


FIGURA 2.9 – Lombadas eletrônica modelo pórtico

Indicação

A velocidade é indicada no display e são emitidos sinais luminosos e sonoros (veículo em velocidade abaixo, igual ou superior à máxima permitida) informando motoristas e pedestres. Veja na figura 2.10 um exemplo de sinalização:

FIGURA 2.10 – Sinalizações de uma Lombada Eletrônica

2.2.3 Bandeiras

As Bandeiras são equipamentos que funcionam da mesma maneira que Pardais ou Lombadas Eletrônicas e ficam visíveis ao lado da via. Veja na figura 2.11 um modelo de Bandeira:

FIGURA 2.11 – Bandeira

2.2.4 Caetanos

Os controladores eletrônicos de infrações em semáforos, mais conhecidos como Caetanos, registram a invasão da faixa de segurança e passagem no sinal vermelho. Detectam a velocidade através de laços magnéticos e possuem uma interface com o semáforo para que seja conhecido seu estado. Veja na figura 2.12 um Caetano instalado num cruzamento de vias:

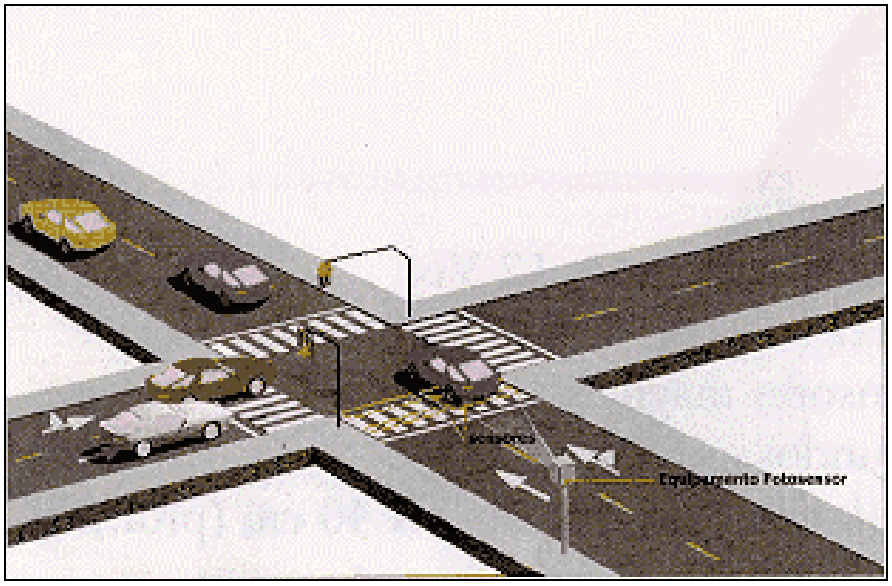


FIGURA 2.12 – Caetano

2.2.5 Radares

Os radares são equipamentos móveis, que necessitam de uma pessoa para operar, saem de fábrica aferidos e lacrados pelo INMETRO e detectam a velocidade através do sistema Doppler. São compostos por um computador portátil básico, antena Doppler, conversor 12Vdc para a tensão da rede, controlador Doppler e uma câmera digital instalada num tripé. Veja na figura 2.13 um típico radar:

FIGURA 2.13 – Radar

2.3 Aferição

Para garantir que os equipamentos estejam calculando a velocidade correta em que um veículo passa na via, o INMETRO, Instituto Nacional de Metrologia, Normalização e Qualidade Industrial, é órgão responsável pela realização de testes e liberação e cada equipamento, conforme portaria nº 115, de 29 de junho de 1998.

2.3.1 Verificações

Para equipamentos fixos, normalmente realiza-se uma primeira verificação no local de instalação antes de ser colocado em uso. São feitas 10 medições para a velocidade nominal máxima permitida na via para qual o instrumento está ajustado.

Também ocorrem verificações periódicas que são realizadas anualmente, quando houver modificação da velocidade nominal máxima permitida da via, reparo do equipamento, modificação da programação ou solicitação de detentor do instrumento. Neste caso normalmente se fazem 05 medições.

2.3.1.1 Erros máximos admitidos

A tabela 2.1 apresenta os erros máximos admitidos durantes os testes pelo INMETRO:

TABELA 2.1 – Erros máximos admitidos pelo INMETRO

Verificação	Medidor de velocidade fixo ou estático		Medidor de velocidade móvel	
	Vel ≤ 100 km/h	Vel > 100 km/h	Vel ≤ 100 km/h	Vel > 100 km/h
Verificação inicial	± 3 km/h	± 3%	± 5 km/h	± 5%
Verificação periódica/eventual	± 5 km/h	± 5%	± 7 km/h	± 7%

Observações:

a) Para radares móveis, o lacre do INMETRO já sai de fábrica;
b) O INMETRO publica o Diário Oficial da União cada modelo de equipamento aprovado.

3 Processamento Digital de Imagem

Este capítulo abordará o processamento digital de imagem 2D, que é a base deste trabalho; as demais tecnologias utilizadas no sistema desenvolvido serão abordadas no apêndice A.

De acordo com Falcão (2003), o interesse em processamento de imagem digital surgiu de duas áreas:

- Interpretação humana usando melhoramento de imagens;
  - Percepção de máquina através da extração de informações das imagens de forma tratável pelo computador.

De acordo com Brito (1999), uma das primeiras aplicações foi o melhoramento de imagens de jornal transmitida via cabo submarino entre London e New York. Em 1920, o tempo requerido para esta transmissão foi reduzido de mais que uma semana para menos que três horas graças ao sistema Bartlane. Melhoramentos continuaram, mas os avanços expressivos em processamento de imagem digital se iniciaram apenas em 1964 com o uso do computador digital em larga escala e do programa espacial, através do processamento de imagens da lua. Hoje, processamento de imagem digital encontra diversas aplicações em: medicina, biologia, astronomia, automação industrial, engenharia, geologia, agronomia, etc.

Uma imagem é uma forma compacta de representar muitas informações e desde a captura até seu entendimento, ela passa por várias etapas. Veja na figura 3.1:

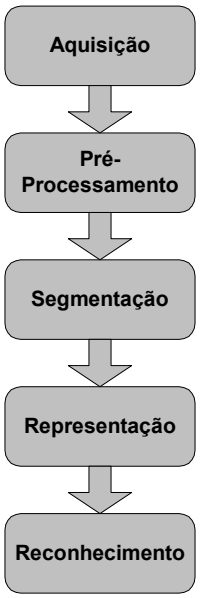


FIGURA 3.1 – Etapas do Processamento de Imagens

A seguir iremos estudar cada uma destas etapas e introduzir os conceitos necessários para compreensão à medida do necessário.

3.1 Aquisição de imagem

De acordo com Curtin (2003), as interfaces para aquisição de imagens são utilizadas para transferir as informações de câmeras ou qualquer fonte de sinal de vídeo em imagens, que poderão ser tratadas como dados num computador. Estas interfaces, chamadas de *frame grabbers*, possuem uma alta taxa de transferência de dados, requisito fundamental para as aplicações mais modernas de processamento e análise de imagens. Em função da aplicação, vamos estudar as câmeras digitais.

Existem diferentes técnicas para se fazer uma câmera digital, que resultam em diferentes qualidades e aplicabilidades.

A qualidade da imagem produzida pode ser medida de diversas formas, que incluem: resolução, intervalo dinâmico e fidelidade nas cores. Também existem atributos importantes: obturador, equivalência ISO, portabilidade, tempo de vida da bateria, luminosidade, etc. Tudo isto deve ser considerado na escolha da câmera para a aplicação.

Observação: a solução proposta necessita de uma câmera digital de boa resolução, que funcione com pouca luminosidade e, principalmente, que gere imagens entrelaçadas.

A seguir, vamos estudar os principais conceitos sobre câmeras digitais:

3.1.1 Sensores de Imagem, *Photosites* e *Pixels*

As câmeras digitais usam sensores de imagem, compostos por milhões de diodos sensíveis à luz, chamados de *photosites*, que armazenam o brilho da luz que incide sobre cada um deles. Veja na figura 3.2 os *photosites* organizados em linhas e colunas:

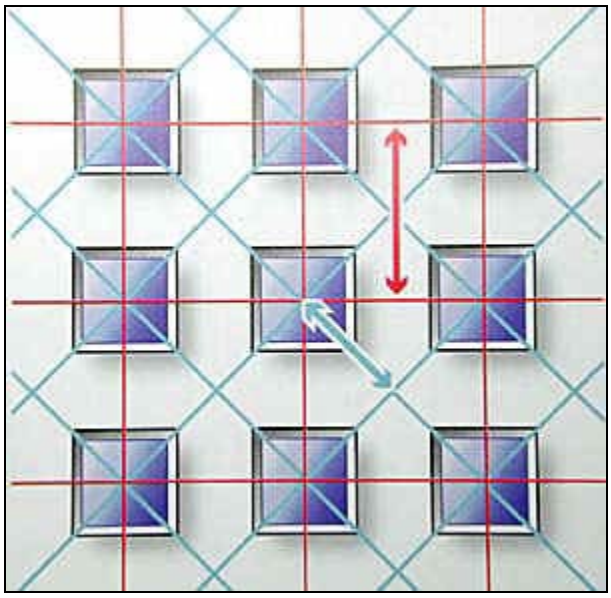


FIGURA 3.2 –*Photosites* organizados em linhas e colunas

Quando o obturador fecha e a exposição completa, os *photosites* armazenam cargas diretamente proporcionais à luz que incide sobre eles de forma que possam ser convertidas em números digitais de modo a recriar a imagem. Veja na figura 3.3 a seguir um exemplo de armazenamento de cargas nos *photosites*, representado de forma binária:

1	1	1	1	1	1	1	1	1
1	1	5	5	5	5	5	1	1
1	1	5	10	10	10	5	1	1
1	1	5	10	10	10	5	1	1
1	1	5	10	10	10	5	1	1
1	1	5	10	10	10	5	1	1
1	1	5	10	10	10	5	1	1
1	1	5	5	5	5	5	1	1
1	1	1	1	1	1	1	1	1

FIGURA 3.3 – Exemplo de armazenamento de cargas nos *photosites*

3.1.2 Intervalo Dinâmico

O intervalo dinâmico pode ser considerado como o número de cores que podem ser selecionadas. Por exemplo, se a câmera tiver 24 bits de cor, teremos 16,777,216 cores.

Isto está diretamente relacionado com a precisão do conversor A/D, que transforma a luz que incide sobre um pixel e o ruído que incide sobre o sistema em um número digital. Tipicamente, as câmeras digitais possuem conversores A/D de 8 bits. Quanto mais bits forem usados para converter a imagem, melhor será a qualidade da câmera.

Já o ruído é um fator que prejudica a qualidade. É difícil de medir e aparece na imagem através de diversas fontes, tais como fonte de alimentação, ruído das lentes ou dos próprio sensor, por exemplo.

3.1.3 *Blooming*

Blooming é o efeito visual causado pela superexposição dos *photosites*, que provoca distorção na imagem e/ou na cor (muita luz sobre os sensores).

3.1.4 Resolução Espacial e Profundidade da imagem

A qualidade da imagem digital, mostrada na tela ou impressa, depende em parte do número de *pixels* usados para criá-la (isto é chamado de resolução).

Mas a resolução pode ser melhorada através de software, usando o recurso da interpolação (o software varre toda a figura e, *pixel* por *pixel*, verifica a vizinhança para descobrir que cor deve a ser a do novo *pixel*). Isto é chamado de zoom digital.

Observação: quanto mais *photosites* existirem, melhores serão as imagens, mas devem ser observados os seguintes fatores:

a) O chip pode ficar grande demais e os *photosites* menores, o que significa dificuldade de fabricação, aumento do custo e necessidade de se ter *photosites* mais sensíveis à luz;

b) *Photosites* maiores podem criar problemas para o armazenamento em função do tamanho final das imagens;

c) O tamanho da imagem é dito de duas formas: ou pelas dimensões em pixels ou pelo total de pixels que ela contém. Por exemplo: uma mesma imagem pode ser dita que tem 1800 x 1600 pixels ou contém 2.88 milhões de pixels (1800 x 1600).

Por exemplo: sendo x=linha e y=coluna, considere uma imagem *f*(*x*,*y*) contida em uma região retangular de 30*cm* em *x* por 20*cm* em *y*. Se obtivermos amostras uniformemente espaçadas a cada 1*mm* em *x* e em *y* (dimensões do pixel são 1mm x 1mm, teremos 200 x 300 mm (que é igual a 60.000 *pixels*). Dizemos então que a *resolução espacial* da imagem é 200x300pixels.

O número *L* de níveis de quantização da função *f*(*x*,*y*) é normalmente uma potência de 2 (i.e. *L*=256,1024,4096). Digamos que neste exemplo *L*=256. Isto significa que a cada pixel pode ter associado um valor de cinza entre 0 e 255, que requer no máximo 8 bits para ser armazenado na memória do computador. Dizemos então que a *profundidade da imagem* é de 8 bits por pixel (ou 1 byte por pixel). Podemos então observar que necessitamos 200x600x1=60 kbytes de memória para armazenar esta imagem.

A figura 3.4a mostra uma imagem de 256x256 *pixels* e 256 níveis de cinza. As figuras 3.4b e 3.4c mostram o resultado de reduzir a resolução espacial de 256x256 para 128x128 e 64x64 *pixels*, respectivamente. Em todos os casos o número de níveis de cinza permanece 256. Para manter a mesma área de display da figura 3.4a, os *pixels* das imagens de mais baixa resolução são replicados. Observe a degradação quadriculada sofrida pelas imagens devido à perda de resolução espacial (o pixel começa a ser perceptível).

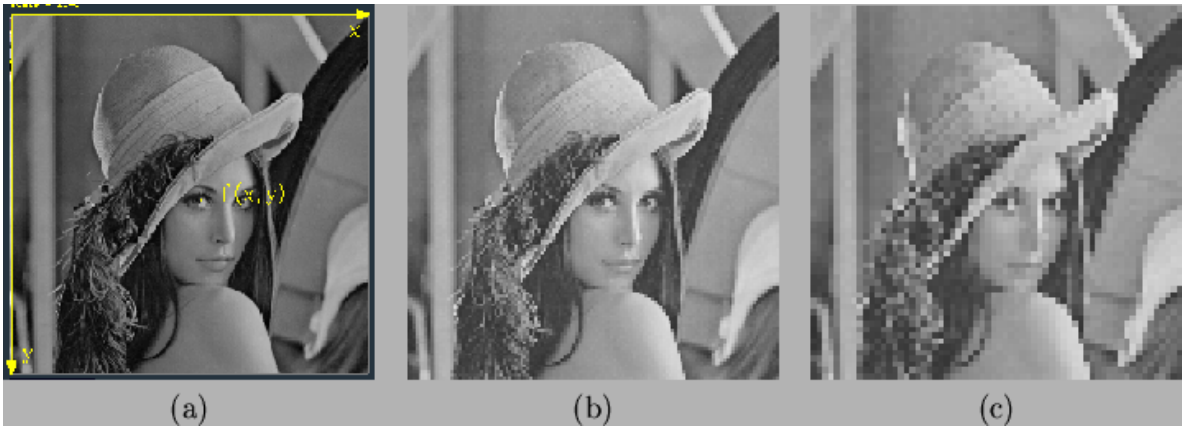


FIGURA 3.4 – A mesma imagem com resoluções diferentes

O efeito de aparecer os pixels também ocorre se ampliarmos uma figura não vetorial. Veja na figura 3.5 este efeito:

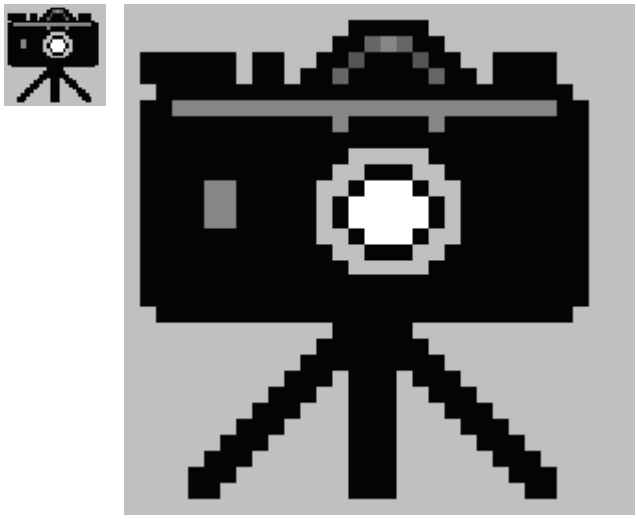


FIGURA 3.5 –*Pixels* aparecem num zoom

3.1.4.1 Varreduras Progressivas e Entrelaçada

Depois que um sensor captura a imagem, ela pode ser lida, convertida para um sinal digital e armazenada. Mas esta captura não se realiza de uma única vez e sim através de dois tipos de varreduras: progressiva ou entrelaçada.

Varredura Progressiva

A imagem é formada por linhas que são processadas uma após a outra, em seqüência. Possui melhor resolução dinâmica e não necessita de muitas linhas como a varredura entrelaçada. Atualmente, a maioria dos DVDs utilizam varredura progressiva em função de que os filmes têm sido gerados usando esta técnica.

Varredura Entrelaçada

Com o objetivo de oferecer maior resolução, sem aumentar consideravelmente os custos, foi desenvolvida a técnica de entrelaçamento, que corresponde a um método onde a linha de varredura começa a partir da extremidade superior esquerda do quadro e faz a varredura das linhas ímpares até a extremidade inferior direita; em seguida executa o mesmo procedimento para as linhas pares e depois retorna à extremidade superior esquerda para compor um único quadro. Como apenas parte das linhas é refeita por vez, é possível apresentar o dobro de linhas por ciclo de renovação, aumentando conseqüentemente a resolução. A

desvantagem dessa técnica fica por conta do tempo de resposta menor (crítico em aplicações de animação e vídeo) e do possível efeito de *flicker*.

Veja na figura 3.6 como funciona esta varredura:

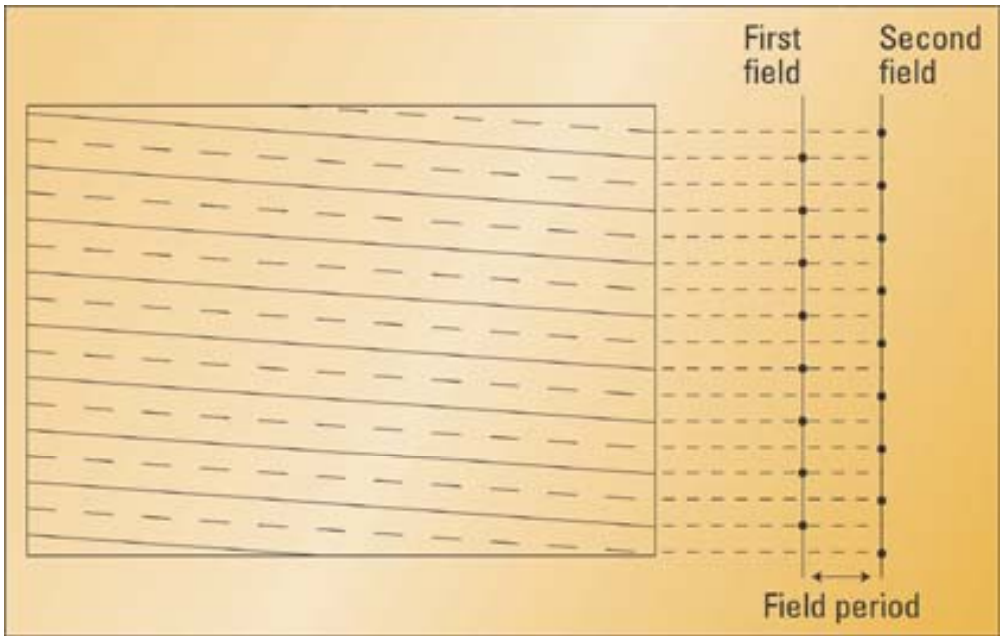


FIGURA 3.6 – Varredura entrelaçada

Na figura 3.7 temos uma imagem capturada por uma câmera entrelaçada:

FIGURA 3.7 – Exemplo de imagem entrelaçada.

Observação: a solução proposta usa câmeras de varredura entrelaçada (para poder obter a diferença entre as linhas ímpares e pares) e captura imagens sem compressão (em função de que o tempo para cálculo é crítico e este recurso exige mais processamento da máquina).

3.1.4.2 Sistemas de Cores RGB e CMYK

RGB

As cores que aparecem numa imagem em geral são oriundas de três cores básicas: vermelho, verde e azul. Isto é conhecido como RGB, pois é um sistema que se baseia no conceito de adicionar cores. Para obter branco, por exemplo, basta combinar as cores básicas em igual quantidade. Veja na figura 3.8 a representação do sistema RGB:

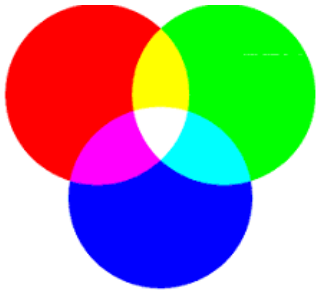


FIGURA 3.8 – Sistema de cores RGB

Em um monitor, cada *pixel* é formado por um grupo das três cores básicas. Veja na figura 3.9:

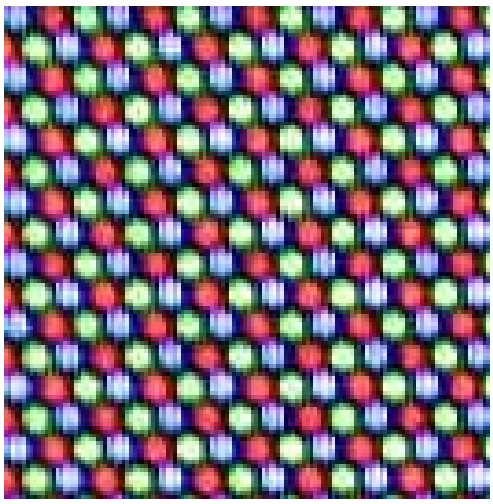


FIGURA 3.9 - Sistema de cores RGB usado num monitor para exibir imagem

CMYK

As impressoras e algumas câmeras e a utilizam o sistema CMYK, que possui as seguintes cores: Ciano (C), Magenta (M), Amarelo (Y), e preto (K) como extra; Quando estas cores são combinadas em quantidades iguais, o resultado é preto, porque elas se subtraem (ao invés do RGB que é um sistema aditivo). Veja na figura 3.10 a representação do sistema CYMK:

FIGURA 3.10 – Sistema de cores CYMK

Observação: a solução proposta usa o sistema de cores RGB.

3.1.4.3 Tipos de Câmeras

As tecnologias para capturar imagens são diversas. Iremos estudar 4 métodos mais usados no cotidiano.

- **Um chip, um sensor**: Existem diferentes filtros de cores, com um padrão pré-determinado, sobre cada *photosite* para capturar as três cores básicas com uma única exposição. Este padrão, chamado de Bayer, usa um quadrado de 4 células onde 2 são verdes (na diagonal), uma para vermelho e outra para azul. A vantagem deste sistema é seu baixo custo; a desvantagem é o *aliasing*, fenômeno ocorre quando o objeto que está sendo fotografado é tão pequeno o filtro não consegue processar, ocasionando perda de cores. É usada na web, em controle de acesso e uso doméstico.
- **Um chip, três sensores**: Existem 4 sensores monocromáticos, cada um com um filtro diferente. Estes sensores estão agrupados e são rotacionados durante a exposição. A posição neutra é utilizada para compor a imagem e focá-la e as demais são para capturar cada uma das cores (vermelho, verde e azul). As vantagens deste sistema são a alta resolução e o fato de permitir capturar imagens em ambientes com luz estroboscópia; a desvantagem é que não serve para capturar objetos em movimento, a não ser que use apenas preto e branco. É usada principalmente em estúdios fotográficos.
- **Dois chips**: Captura a crominância usando um sensor (usualmente equipado com um filtro para cada cor) e a luminância com um segundo sensor (usualmente o que captura a luz verde). A vantagem deste sistema é seu custo moderado associado com uma resolução muito alta; A desvantagem é o tempo de varredura e o processamento dos dados, que torna a captura

de imagens lenta. É usada em fotografias artísticas de pequenos objetos parados onde o objetivo é mostrar os detalhes.

- **Três chips**: Este sistema separa a imagem , através de sem-transparentes espelhos, para ser tratada por um sensor de cor separadamente. A vantagem deste sistema é permitir capturar objetos em movimento, com excelente resolução e sem perder cores (sem *aliasing*), em praticamente todos os ambientes (a única exceção é são para ambientes iluminados com luzes florescentes de baixa freqüência); a desvantagem é seu alto custo, decorrente da triplicação de sensores e a precisão para alinha-los. É principalmente usada para fotografia artística.
- **Sensores lineares**: Scanners e algumas câmeras profissionais usam sensores de imagens com *photosites* arrumados em formato de linha ou de árvores. E muitos dispositivos movimentam estes sensores para poder cobrir toda a imagem.

3.2 Pré-processamento

3.2.1 Filtros

De acordo com Brito (1999), a filtragem é uma típica aplicação da Transformada de Fourier. Com o objetivo de eliminar ou realçar determinadas componentes da imagem, tais como ruído ou contornos, a filtragem no domínio da freqüência explora a característica reversível da Transformada de Fourier e a informação contida neste domínio.

O princípio de funcionamento dos filtros que operam em domínio espacial baseia-se em relações de vizinhança entre os elementos de uma região de tamanho e formato predeterminado. Por questões de simetria usam-se, na definição dos núcleos dos filtros, vizinhanças *n*x*n*, onde *n* é um número ímpar. Por questões de eficiência computacional, preferem-se valores pequenos para *n*.

Domínio espacial refere-se ao plano da própria imagem, sendo que nesta categoria trabalha-se diretamente com o valor dos *pixels* de uma imagem. No processo de filtragem são atribuídos valores aos elementos da imagem destino em função dos elementos presentes na imagem fonte.

A utilização de filtros com formatos diferentes e valores dependentes da posição na imagem, é conhecida como filtragem por máscara de deslocamento ou janela móvel.

O formato da máscara de deslocamento mais comumente utilizado é o quadrado. Dependendo do algoritmo, pode-se ter sub-máscaras com outros formatos, como triangulares, pentagonais ou formatos irregulares. A máscara pode ter formatos diferentes, como cruz ou retângulo. Alguns algoritmos admitem a escolha do

formato da máscara de deslocamento e outros são projetados para um tipo específico de máscara, existindo também algoritmos onde até o tamanho da máscara é predeterminado. Geralmente, o deslocamento da máscara é realizado da esquerda para a direita sobre cada linha da imagem. Finalizando esta linha, o centro da máscara é posicionado no início da linha seguinte, e o processo é repetido até que a máscara alcance o canto inferior direito da imagem. Com a máscara centralizada em um ponto da borda, alguns elementos da janela não terão valores definidos por estarem fora dos limites da imagem. Para contornar este problema, pode-se restringir o posicionamento da máscara de modo que a mesma não seja sobreposta a pontos não pertencentes à imagem ou atribui-se aos pontos fora da imagem um valor predeterminado. No primeiro caso, após a execução do filtro, costuma-se preencher as bordas não calculadas com uma constante ou com o valor do ponto mais próximo. Este último procedimento é chamado de repetição de bordas. Outras abordagens podem ser seguidas, como definir os operadores de forma que tratem todos os casos especiais, assumir a imagem como sendo ciclicamente fechada ou atribuir o valor zero aos pontos fora da imagem.

O princípio geral da filtragem, no domínio da freqüência, pode ser indicado pela seguinte transformação:

$$G(u,v) = H(u,v)F(u,v)$$

Onde:

$F(u,v)$ é a transformada de Fourier da imagem de entrada
$G(u,v)$ é a imagem de saída filtrada
$H(u,v)$ é a respectiva *função de transferência* do filtro.

O problema consiste em definir a função $H(u,v)$ que conduza à imagem desejada $G(u,v)$ e a transformada inversa define a imagem filtrada no domínio espacial $g(x,y)$:

$$\mathcal{F}^{-1}[G(u,v)]$$

3.2.1.1 Filtros Passa-Baixas

Os filtros *passa-baixa* deixam passar as componentes de baixa freqüência, atenuando as de alta freqüência relacionadas com as transições bruscas da imagem. Estas transições são representadas por ruídos ou contornos, o que significa que ao eliminarmos o ruído indesejável da imagem, a partir de um filtro passa-baixas, estamos atenuando os seus contornos nas mesmas proporções, caso típico do processamento linear. Veja na figura 3.12 um filtro passa-baixas:

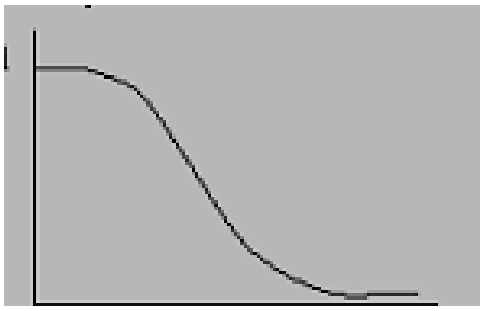


FIGURA 3.12 – Filtro passa-baixas

Um *filtro ideal passa-baixas* 2-D pode ser representado pela seguinte função de transferência:

$$H(u,v) = \begin{cases} 1 & \text{se } D(u,v) \leq D_0 \\ 0 & \text{se } D(u,v) > D_0 \end{cases}$$

onde D_0 >=0 é a *freqüência de corte* e $D(u,v)$ é a distância do ponto (u,v) até a origem do plano da freqüência, ou seja,

$$D(u,v) = (u^2 + v^2)^{\frac{1}{2}}$$

Observe que as freqüências contidas no círculo de raio D_0 não sofrem atenuações, daí o termo filtro ideal. A figura 3.13 apresenta uma imagem original com ruído, uma máscara correspondente à função $H(u,v)$ e a imagem resultante do filtro passa-baixas:

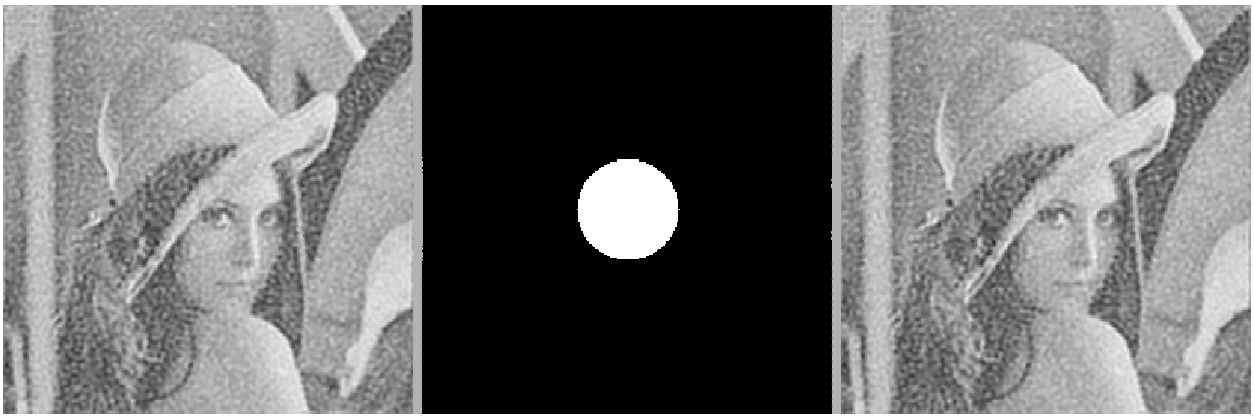


FIGURA 3.13 – Imagem com ruído, máscara e com o filtro passa-baixas.

O filtro passa-baixas de Butterworth de ordem *n* é dado pela seguinte função de transferência:

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$

Esta função define um filtro sem grande descontinuidade, não estabelecendo uma transição brusca na freqüência de corte. Nestes casos, costuma-se definir um valor para D_0 correspondendo a uma porcentagem do máximo valor $H(u,v)$. Podemos ver facilmente que quando $D_0 = D(u,v)$ o valor de $H(u,v)$ cai para 50% do seu valor inicial.

Exemplos

O filtro da mediana é um exemplo típico de um filtro passa-baixas no domínio espacial. Consiste em substituir o valor de um pixel *m*, centrado numa determinada vizinhança, pelo valor médio dos pixels desta vizinhança ordenados de acordo com suas magnitudes. Elimina eficientemente ruído impulsivo, representando descontinuidades abruptas e isoladas na imagem, não introduz valores de níveis de cinza diferentes daqueles contidos na imagem original e, por afetar menos os contornos, pode ser aplicado iterativamente.

É fácil constatar o efeito de redução do ruído após tal filtragem, assim como o problema de suavização dos contornos da imagem que, naturalmente, se torna mais grave à medida que a máscara *h* aumenta de dimensão. A figura 3.14 apresenta uma imagem com ruído e uma iteração do filtro da mediana com uma vizinhança 5x5:

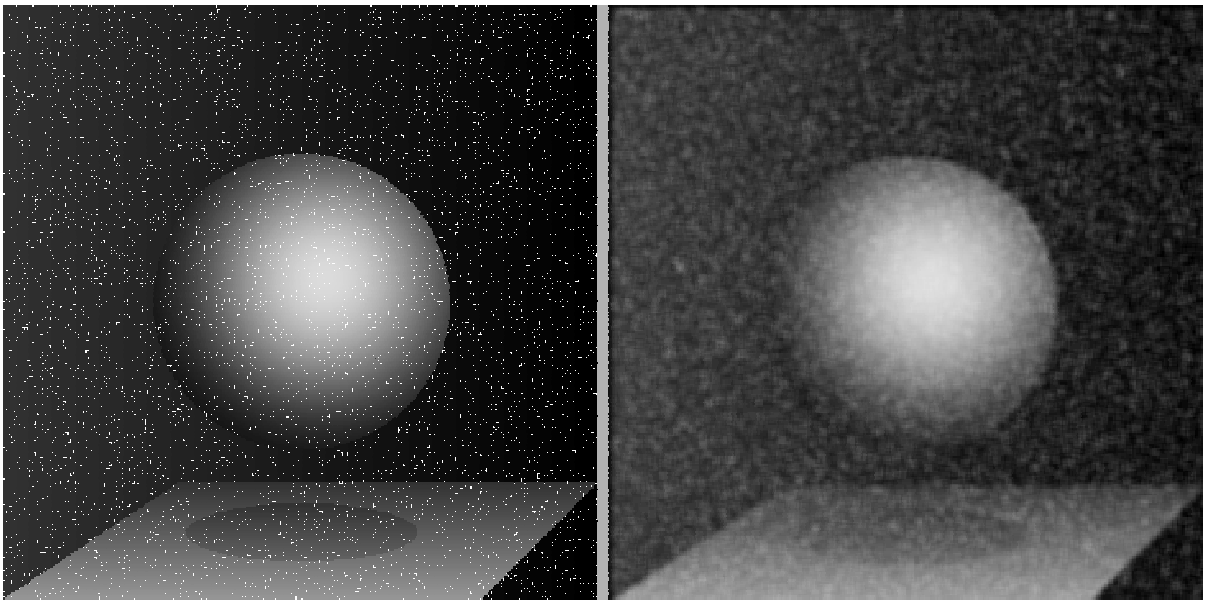


FIGURA 3.14 – Imagem com ruído e depois do filtro da mediana 5x5

Os valores de *h* podem ser definidos a partir da consideração do modelo de um determinado tipo de ruído, visando expressar aproximações deste modelo. As máscaras, a seguir, são definidas considerando o ruído com uma distribuição de probabilidade gaussiana, distribuição esta muito utilizada na prática.

$$h_1 = \frac{1}{16}\begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix}$$

3.2.1.2 Filtros Passa-Altas

Os filtros *passa-altas* atenuam ou eliminam as componentes de baixas freqüências. O efeito evidente desta filtragem é a obtenção de um realce nas zonas de alta freqüência do sinal, isto é, dos seus contornos e - infelizmente - ruído. Veja na figura 3.15 um filtro passa-altas:

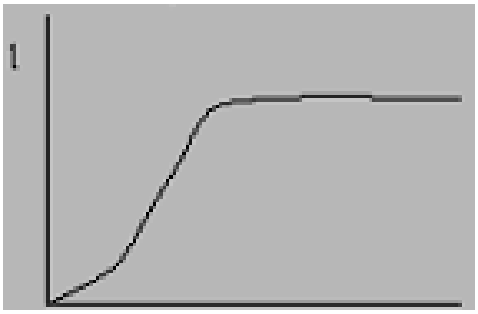


FIGURA 3.15 – Filtro passa-altas 1D

Um *filtro ideal passa-altas* 2-D é dado pela seguinte função de transferência:

$$H(u,v) = \begin{cases} 0 & \text{se } D(u,v) \leq D_0 \\ 1 & \text{se } D(u,v) > D_0 \end{cases}$$

Onde:

D_0 é a freqüência de corte medida a partir da origem, no plano da freqüência
$D(u,v)$ é definida como anteriormente.

A figura 3.16 apresenta uma imagem original com ruído, uma máscara correspondente à função $H(u,v)$, e a imagem resultante do filtro passa-altas:

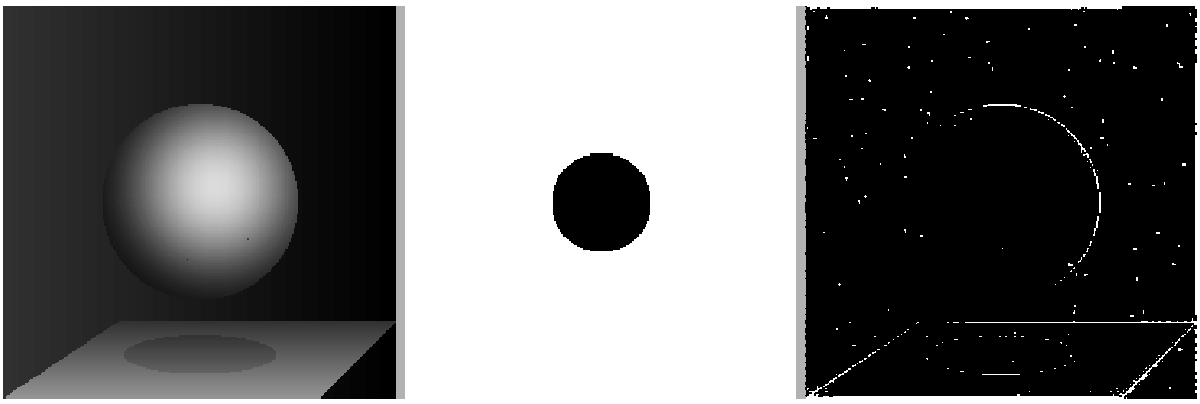


FIGURA 3.16 – Imagem com ruído, a máscara e depois do filtro passa- altas.

O filtro passa-altas de Butterworth de ordem n é dado por

$$H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}}$$

Novamente, quando $D_0 = D(u,v)$, $H(u,v)$ cai 50% do seu valor máximo.

3.2.1.3 Filtros Passa-Faixas

Num projeto de filtros lineares podemos considerar, ainda, os filtros *passa-faixas* associados às regiões compreendidas entre as baixas e altas freqüências. Estes filtros são empregados, por exemplo, em problemas de restauração de imagens. A figura a seguir apresenta o modelo de filtros para o caso 1-D (uma rotação destas funções define os mesmos filtros para o caso 2-D). Veja na figura 3.17 um filtro passa-faixas:

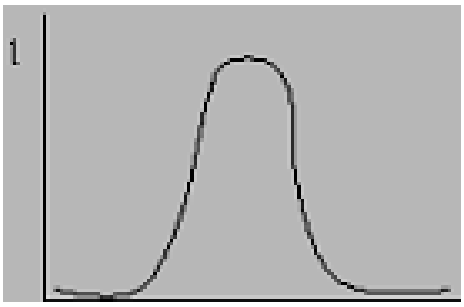


FIGURA 3.17 – Filtro passa-faixas

3.3 Segmentação

De acordo com Brito (1999), por segmentação de uma imagem entende-se a extração ou identificação de objetos contidos na imagem, onde o objeto' é toda característica com conteúdo semântico relevante para a aplicação desejada. As descrições são constituídas basicamente de uma lista de objetos, seus rótulos e relações entre os objetos. Podem-se distinguir dois tipos de objetos: objetos ``complexos'', que são formados por outros objetos, e objetos ``simples'', onde isto não acontece.

O processo de segmentação consiste em uma divisão ou separação de uma imagem em regiões de atributos similares. Os atributos básicos utilizados são a amplitude e a luminância, isto quando se trata de imagem monocromática, e componentes de cores (brilho,contraste,etc) para imagens coloridas. Contornos (Bordas) e textura também são atributos úteis à segmentação. Além disso, este processo não envolve classificar cada segmento.

O segmentador apenas subdivide a imagem, não se dedicando a reconhecer os segmentos individuais ou relações (similaridades) de um pixel com uma certa região ou a outros pixels. Portanto, a segmentação faz parte, em geral, de um processo maior que é o de obter uma descrição da imagem. Um dos passos na análise da imagem é a identificação dos objetos simples que correspondem, em geral, a linhas ou regiões (grupos de pontos conectados).

3.3.1 Detecção de Bordas

As bordas de uma imagem são o resultado de mudanças em alguma propriedade física ou espacial de superfícies iluminadas. A maioria das técnicas de detecção de bordas emprega operadores diferenciais de primeira ou de segunda ordem. Os operadores diferenciais ressaltam os contornos das bordas, mas também amplificam o ruído da cena. Grande parte dos operadores de borda utiliza algum tipo de suavização da imagem antes da operação diferencial.

Mudanças ou descontinuidades em algum atributo da amplitude de uma imagem, tal como a luminância, são características primitivas fundamentalmente importantes de uma imagem, pois elas freqüentemente provêem indicações da extensão física de objetos na imagem. Uma borda numa imagem monocromática, por exemplo, é uma mudança súbita do nível de cinza entre duas regiões relativamente homogêneas. Cada região é uniforme e homogênea com relação a alguma propriedade da amplitude, tais como tom ou textura, e o valor desta propriedade difere de maneira significante de acordo com a vizinhança de cada região. Idealmente, a seção transversal de uma borda apresenta a forma de uma função degrau. Uma linha caracteriza-se por ter um nível de cinza relativamente constante ao longo de uma faixa estreita e alongada. A seção transversal de uma linha tem a forma de um pico (ou depressão) estreito. Tanto as bordas quanto as linhas são descontinuidades da imagem.

Existem diversas formas de se detectar bordas mas podemos separa-las em duas categorias principais: gradiente e Laplaciano, que serão estudados a seguir.

Suponha que temos que procurar uma borda cujo sinal é representado pela figura 3.18.

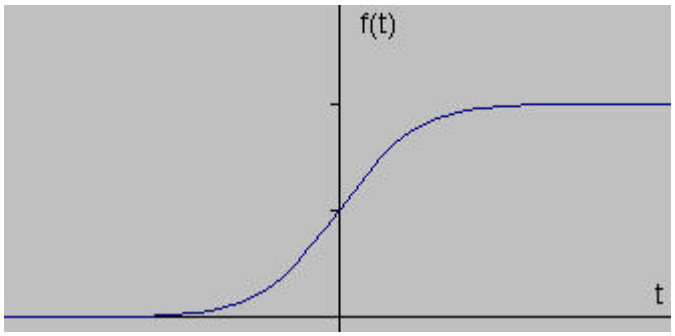


FIGURA 3.18 – Exemplo de borda

Se for calculado o gradiente deste sinal, na qual se for de apenas uma dimensão é a derivada de primeira ordem, nós teremos o seguinte (figura 3.19):

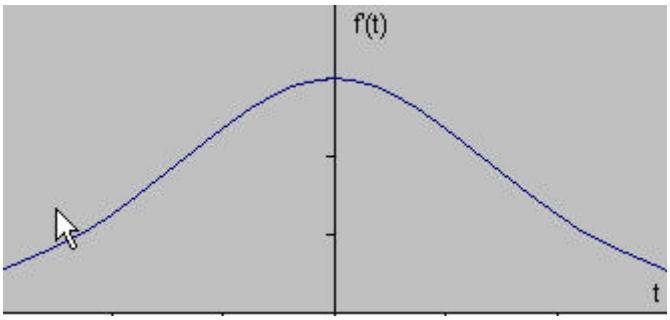


FIGURA 3.19 – Derivada de primeira ordem

Claramente a derivada mostra a posição máxima no centro da borda; Este método é caracterizado pelo filtro de gradiente. Robert e Sobel são algoritmos que usam esta técnica.

Mas quando a derivada de primeira ordem está no máximo, a derivada de segunda ordem é zero e também se pode localizar bordas assim. É o método Laplaciano (figura 3.20).

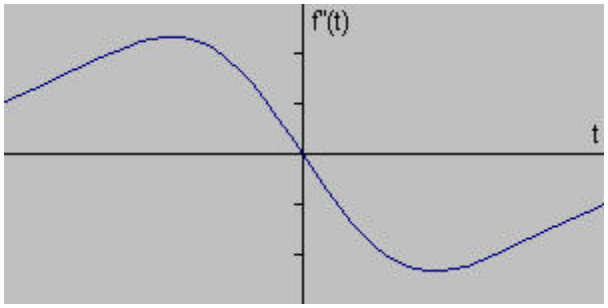


FIGURA 3.20 – Derivada de segunda ordem

3.3.1.1 Rastreamento de Contorno

Há, pelo menos, dois modos de fazer o rastreamento do contorno: segundo a varredura e por busca em todas as direções. Ambos trabalham coma imagem gradiente, e não com a original.

Segundo a Varredura

No rastreamento segundo a varredura é varrida uma linha da imagem gradiente por vez e várias curvas são rastreadas de uma só vez. O processo é iniciado pela detecção de um ponto inicial do contorno, que é feito considerando valores de borda (gradiente) maiores que um limiar L. Para cada ponto (x,y) são examinados os vizinhos anteriores (já visitados e rotulados) e testado se este ponto corresponde à continuação de algum contorno já detectado. No teste de

continuidade, o valor da borda em (x,y) é comparado com um limiar D (D $\leq$ L) e é verificado se a direção desta no ponto (x,y) está alinhada com a da borda num de seus vizinhos. Os passos do procedimento são mostrados neste algoritmo:

$k \leftarrow 0$
percorra a imagem no sentido da varredura;
para todo ponto g, tal que $A(g) > D$
(a) se existir borda em p ($borda(p) \neq 0$), p vizinho anterior de g, $| \phi(p) - \phi(g) | <$ *ALFA*, então continue em g a borda cuja direção coincide melhor com g: $borda(g) \leftarrow borda(p)$;
(b) caso contrário, *se* $A(g) > L$, inicie nova borda: $k \leftarrow k+1$; $borda(g) \leftarrow k$.

Onde:
g é (x,y); os vizinhos anteriores de g são (x-1,y),(x,y-1),(x-1,y-1) e (x-1,y+1);
$A(g)$ é a magnitude da borda em g; $\phi(p)$ é a direção da borda em p;
borda (p) indica qual a borda em p (se igual a zero então não há borda em p ou p ainda não foi visitado).

Deve-se incluir no algoritmo descrito anteriormente, uma linha e uma coluna (com bordas=0) antes da primeira linha e coluna da imagem. Usam-se limiares diferentes (L e D) para a detecção do contorno e sua continuação. Como D $\leq$ L, o critério para a continuidade fica sendo (quanto à magnitude) menos rigoroso que para a detecção. Isto faz com que as bordas extraídas sejam diferentes, dependendo do sentido e direção da varredura.

Em Todas as Direções

No rastreamento por busca em todas as direções, segue-se um único contorno por vez. Inicialmente, detecta-se um ponto de contorno (por exemplo, escolhe-se o ponto da imagem gradiente com maior magnitude de borda) e examinam-se seus vizinhos procurando possíveis continuidades. Um vizinho é sua continuação se sua magnitude for maior que um limiar (d) e a borda estiver alinhada com a do ponto escolhido. Das possíveis continuidades (se houver) escolhe-se uma e repete-se o processo. As outras possibilidades são armazenadas para investigação posterior. Entre os pontos examinados são excluídos aqueles que pertencem a algum contorno. Os passos do procedimento são mostrados neste algoritmo:

$k \leftarrow 0$
escolha um ponto g, tal que $A(g) > L$ e a *borda* (g) = 0; se não existe ponto nestas condições, então pare (fim).
inicie novo contorno: $k \leftarrow k+1$; $borda(k) \leftarrow k$;
ache todos os pontos p da vizinhança de g, tais que $A(p)>D$, a *borda* (p) = 0 e $| \phi(p) - \phi(g) | <$ *ALFA*; para todos estes pontos p faça a $borda(g) \leftarrow borda(p)$ e guarde estes pontos num conjunto X;
se X está vazio , vá para 2;

escolha e extraia um ponto *g* de *X*; vá para 4.

Onde:
g é (*x*,*y*); os vizinhos anteriores de *g* são *(x-1,y), (x,y-1), (x-1,y-1) e (x-1,y+1)* é a magnitude da borda em *g*;
$\phi(p)$ é a direção da borda em *p*;
borda (*p*) indica qual a borda em *p* (se igual a zero então não há borda em *p* ou p ainda não foi visitado).

Observação: ao invés de usar a direção no ponto *g* ($\phi(g)$) como referência, pode-se usar a "direção do contorno" definida como a média das direções dos últimos pontos do contorno (3 últimos, por exemplo). Isto faz o algoritmo menos sensível ao ruído. O mesmo pode ser feito com relação ao algoritmo anterior.

3.3.1.2 *Threshold*

Após a formação da imagem gradiente, através de métodos de detecção de bordas derivativos, é necessário que a imagem gradiente seja comparada a um valor de limiar de nível de cinza (*threshold*) para determinar se existem bordas. O valor de limiar determina a sensibilidade do detector de bordas.

Para imagens sem ruídos, este limiar pode ser escolhido de modo que todas as descontinuidades de amplitude a partir de um nível de contraste mínimo sejam reconhecidas como bordas; para imagens com ruído, esta seleção torna-se difícil, pois o valor de limiar tanto pode deixar despercebido bordas existentes quanto pode designar falsas bordas induzidas pelo ruído.

Vários métodos estão disponíveis para a escolha do limiar. Um deles é aproximar o histograma pela soma de duas normais. O limiar ótimo corresponde à interseção destas duas normais. Outro método bastante usado e conhecido consiste em suavizar o histograma e tomar o mínimo entre os dois picos. Em vários problemas práticos, é interessante usar um limiar que varia de acordo com a posição na imagem para corrigir o efeito da iluminação desigual.

3.3.1.3 Sobel

De acordo com Green (2002), o Operador de Sobel usa um par de máscaras de convolução 3x3: uma para calcular o gradiente na direção x (colunas) e outra para calcular na direção y (linhas). Veja na figura 3.21 as máscaras para o cálculo do gradiente de Sobel:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

FIGURA 3.21 - Máscaras para o cálculo do gradiente de Sobel

O valor absoluto do gradiente bidimensional no ponto *G(x,y)* é calculado por:

$$|G| = |Gx| + |Gy|$$

A máscara desliza sobre a imagem original, modifica o valor do pixel e o desloca para a direita, até que a linha acabe. Em seguida inicia a próxima linha. A figura 3.22 mostra a máscara aplicada numa região da imagem original e o resultado gerado:

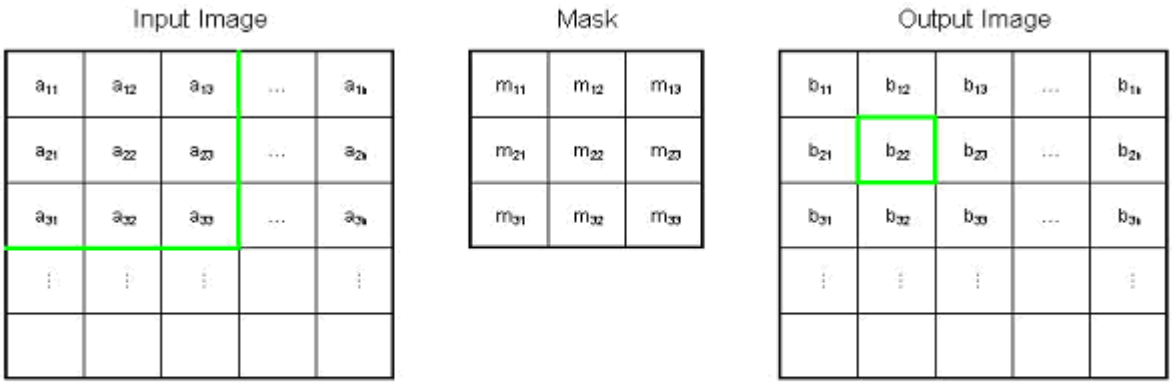


FIGURA 3.22 – Obtenção da borda através da magnitude de GX e GY

3.3.1.4 Roberts

De acordo com Green (2002), o Operador de Roberts provavelmente seja o algoritmo mais simples de detecção de bordas. Ele utiliza duas máscaras: uma para calcular o valor gradiente da primeira diagonal (gr1) e outra para calcular o gradiente da segunda diagonal (gr2): Veja na figura 3.23 as máscaras para o cálculo do operador gradiente de Roberts:

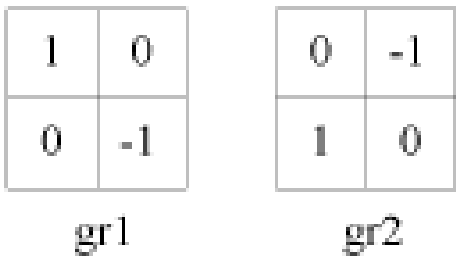


FIGURA 3.23 - Máscaras para o cálculo do operador gradiente de Roberts

O valor exato do gradiente bidimensional na posição (*x*,*y*), correspondente ao ponto superior esquerdo da janela 2x2, seria calculado por:

$$g_r(x,y) = \sqrt{g_{r1}^{\ 2} + g_{r2}^{\ 2}}$$

Devido ao custo computacional, as operações de elevar ao quadrado e raiz quadrada são muitas vezes substituídas pela seguinte aproximação que é mais eficiente:

$$g_r(x,y)=\alpha.(|g_{r1}(x,y)| + |g_{r2}(x,y)|)$$

Uma desvantagem do operador de Roberts é a sua assimetria. Dependendo da direção, certas bordas são mais realçadas que outras, mesmo tendo igual magnitude.

3.3.1.5 Canny

De acordo com Bueno (2000), o algoritmo de Canny é um operador gaussiano de primeira derivada que suaviza os ruídos e localiza as bordas. Para desenvolver este algoritmo, primeiramente concentraram seus estudos em bordas ideais, cuja representação pode ser feita por funções em uma dimensão (1-D). Na prática este não é um modelo exato, mas representa muito bem o efeito causado pelos ruídos.

Considerando a função gaussianica em uma dimensão, podemos expressar:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}$$

a primeira derivada é:

$$G'(x) = \frac{-x}{\sqrt{2\pi}\sigma^3} e^{\frac{-x^2}{2\sigma^2}}$$

e a segunda derivada é:

$$G''(x) = -\frac{1}{\sqrt{2\pi}\sigma^3} e^{\frac{-x^2}{2\sigma^2}} \left[1 - \frac{x^2}{\sigma^2}\right]$$

Veja a figura 3.24 que mostra os gráficos das funções de Gauss (a), a primeira (b) e a segunda derivada (c):

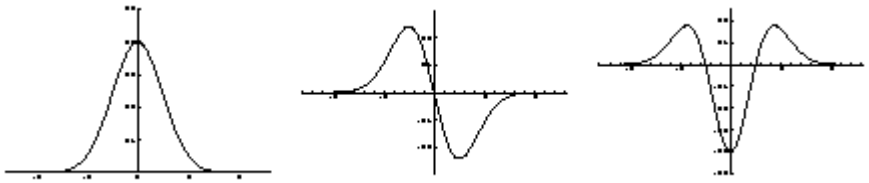


FIGURA 3.24 - Gráficos das funções de Gauss, primeira e segunda derivada.

As funções bidimensionais (2-D) são melhores expressadas em coordenadas polares onde

$$r = \sqrt{x^2 + y^2}$$

representa uma distância radial da origem ao ponto. A função é simétrica e independente de ϕ. Assim,

$$G(r) = \frac{1}{2\pi\sigma^2} e^{\frac{-r^2}{2\sigma^2}}$$

a primeira derivada é

$$G'(r) = \frac{-r}{2\pi\sigma^4} e^{\frac{-r^2}{2\sigma^2}}$$

e a segunda derivada é

$$G''(x) = -\frac{1}{2\pi\sigma^4} e^{\frac{-r^2}{2\sigma^2}} \left[1 - \frac{r^2}{\sigma^2}\right]$$

Veja a figura 3.25 que mostra os gráficos das funções em 2-D de Gauss (a), a primeira (b) e segunda derivada (c):

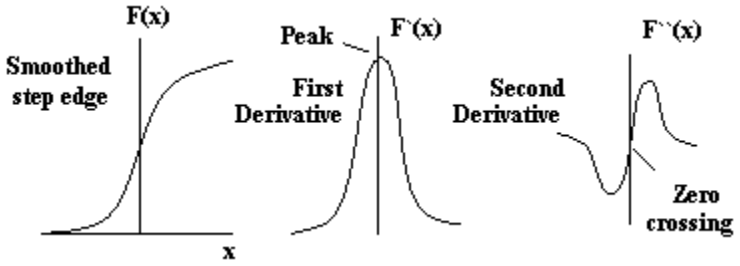


FIGURA 3.25 - Gráficos das funções em 2-D de Gauss, 1ª e 2ª derivada.

De fato, a primeira derivada da imagem convolucionada da função gaussianica,

$$g(x,y) = D[Gauss(x,y) * f(x,y)]$$

é equivalente a imagem convolucionada da primeira derivada da função de Gauss,

$$g(x,y) = D[Gauss(x,y)] * f(x,y)$$

Características:

- **Boa detecção**: habilidade para localizar e marcar todas as bordas realmente existentes.
- **Boa localização**: minimiza a distância entre a borda detectada e a borda real.
- **Boa resposta**: para cada borda deve haver somente uma resposta.

Para isto, implementa-se este algoritmo em quatro estágios:

a) **Uniformização da imagem**: a imagem é uniformizada por uma função gaussianica bidimensional (2-D) de tamanho especificado por um parâmetro usual. Na prática, convoluções gaussianicas bidimensionais de tamanho elevado, levam muito tempo para serem processadas, portanto é comum aproximá-las por duas funções gaussianicas unidimensionais, uma no eixo x e a outra no eixo y. Isso resulta em dois valores para cada pixel.

b) **Diferenciação**: considerando a convolução bidimensional apresentada no estágio anterior, a imagem uniformizada é separada nas direções x e y. Assim é possível calcular o gradiente da superfície uniforme da imagem convolucionada. Considerando também a aproximação unidimensional apresentada no estágio 1, os valores uniformizados na direção x são convolucionados utilizando a primeira derivada da função gaussianica unidimensional com mesmo alinhamento na direção y. Da mesma maneira, os valores uniformizados na direção y são convolucionados utilizando a primeira derivada da função gaussianica unidimensional com mesmo alinhamento, agora na direção x. Para somar os valores dos gradientes de x e y, a magnitude e o ângulo da inclinação podem ser calculados através da hipotenusa e arctg do ângulo, similarmente ao operador de Sobel.

c) **Omissão de pontos de mínima intensidade**: encontrado a medida da intensidade de cada ponto da imagem, precisa-se localizar agora as bordas. Isso é possível localizando os pontos de máxima intensidade, ou de maneira inversa, pelos pontos de mínima intensidade, que precisam ser omitidos. Um valor de máxima intensidade ocorre no local mais alto da função gradiente ou onde a derivada da função gradiente possui valor zero. Entretanto, desejamos omitir os pontos de mínima intensidade ou pontos de mínimas direções perpendiculares com a borda. Aproximações são freqüentemente usadas, além da diferenciação perpendicular para cada borda. Cada pixel em volta forma o centro de um novo pixel na vizinhança. Interpolando os valores dos pixels ao redor, as magnitudes dos gradientes são calculados pelos limites das fronteiras vizinhas em ambas as direções

perpendiculares do pixel central. Se o pixel considerado na figura abaixo possuir valor menor do que os valores dos pixels vizinhos, ele será omitido.

d) **Limiarização da borda (threshold)**: a limiarização usado no algoritmo Canny usa o método chamado "histerese". Considerando um segmento de borda, para todo valor situado acima do limite superior de limiarização, ele é imediatamente aceito. Para todo valor situado abaixo do limite inferior de limiarização, ele é imediatamente rejeitado. Pontos situados entre os dois limites serão aceitos se eles estiverem relacionados com pixels que apresentem respostas fortes.

3.3.1.6 Spline

De acordo com Green (2002), este algoritmo implementa um novo operador gradiente, o qual é baseado em splines cúbicas para o realce de contornos em imagens digitais. A aproximação para o gradiente em um determinado ponto é obtida através da convolução da imagem com quatro máscaras. Cada máscara aproxima no ponto central a derivada da spline cúbica que interpola em uma das quatro direções possíveis: uma para a horizontal, uma para a vertical e duas para as diagonais. O desempenho do algoritmo proposto é comparado posteriormente com o dos dois outros métodos de cálculo do gradiente já mencionados.

Assim como nos gradientes de Roberts e Sobel, o gradiente Spline é derivado de gradientes unidimensionais e pode ser descrito usando quatro máscaras, como já mencionado, resultando em um gradiente horizontal, g_{px}, um gradiente vertical, g_{py} e dois gradientes nas diagonais, g_{p1} *e* g_{p2}. Veja na figura 3.26 as máscaras para o cálculo do gradiente de Spline

0	0	0	0	0
0	0	0	0	0
-1	8	0	-8	1
0	0	0	0	0
0	0	0	0	0

g_{px}

0	0	-1	0	0
0	0	8	0	0
0	0	0	0	0
0	0	-8	0	0
0	0	1	0	0

g_{py}

-1	0	0	0	0
0	8	0	0	0
0	0	0	0	0
0	0	0	-8	0
0	0	0	0	1

g_{p1}

0	0	0	0	1
0	0	0	-8	0
0	0	0	0	0
0	8	0	0	0
-1	0	0	0	0

g_{p2}

FIGURA 3.26 - Máscaras para o cálculo do gradiente de Spline

O valor absoluto do gradiente Spline bidimensional é calculado por:

$$g_p(x,y)=\alpha.(3.(|g_{px}|+|g_{py}|)+2.(|g_{p1}|+|g_{p2}|))$$

Observe que o peso dos valores de gradiente g_{px} *e* g_{py} são maiores que os pesos para g_{p1} *e* g_{p2} por um fator de *3/2*. Esta é uma aproximação de um fator de raiz

quadrada de 2 que corresponde a relação das distâncias entre os pontos vizinhos nas direções vertical e horizontal.

Os fatores de escala são determinados de acordo com o máximo valor de todos os pixels da imagem gradiente considerando, primeiramente, o fator de escala igual a 1. É calculado da seguinte forma: primeiramente se faz uma convolução dos filtros do operador gradiente com a imagem de entrada e com o fator de escala $\alpha=1$. Obtida a imagem gradiente, observamos qual o maior valor de nível de cinza existente nesta e conseqüentemente calculamos um fator de escala que multiplicado pelo maior valor de nível de cinza seja igual a 2^n-1 onde n é o número de bits por pixel da imagem. Por exemplo, se o maior valor de nível de cinza da imagem gradiente for 134 em uma imagem representada com 6 bits por pixel, ou seja, a faixa de valores de níveis de cinza da imagem original é [0,63], então o fator de escala pode ser calculado da seguinte forma:

$$134.\alpha=63 \Leftrightarrow \alpha=63/134$$

Como pudemos observar o fator de escala da fórmula do gradiente não depende de forma alguma da resolução da imagem original.

3.3.1.7 Laplaciano

De acordo com Green (2002), o algoritmo Laplaciano funciona da mesma forma do que o algoritmo de Sobel, pois utiliza apenas uma máscara ao invés de duas. E por ser mais simples, é mais sensível a ruídos do que Sobel.

3.4 Representação

De acordo com Falcão (2003), representação consiste das várias formas de armazenar a fronteira e o interior de objetos segmentados. Contém informações sobre a forma, topologia dos objetos e a descrição quantitativa destas informações.

O termo imagem refere-se a uma função bidimensional da intensidade da luz, denotada por $f(x,y)$, onde o valor da amplitude de f no ponto de coordenadas espaciais (x,y) diz a intensidade (brilho) da imagem naquele ponto. As imagens percebidas pelas pessoas normalmente no seu dia-a-dia consistem da luz refletida pelos objetos. A natureza básica de $f(x,y)$ pode ser caracterizada por duas componentes:

Quantidade de luz incidente na cena vista.

Quantidade de luz refletida pelos objetos presentes na cena. Elas são chamadas de componente de iluminação e refratância, e são denotadas por $i(x,y)$ e $r(x,y)$

respectivamente. As funções anteriores se combinam como um produto para formar $f(x,y)$:

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

Onde $0<i(x,y)<\infty$ e $0<r(x,y)<1$.

A natureza de $i(x,y)$ é determinada pela fonte de luz, e a de $r(x,y)$ pelas características dos objetos presentes na cena. A partir desta seção chamaremos a intensidade de uma imagem monocromática *f* no ponto de coordenadas espaciais (x,y) como sendo o *nível de cinza (l)* da imagem naquele ponto. A faixa de variação de (l) em uma imagem é chamada de escala de cinza. Na prática esta variação corresponde ao intervalo [0, L], onde $l = 0$ é considerado preto e $l = L$ é considerado branco na escala.

Para ser adequada ao processamento computacional, uma imagem representada pela função $f(x,y)$ precisa ser digitalizada tanto espacialmente como em amplitude. A digitalização das coordenadas espaciais é chamada de amostragem da imagem, e a digitalização da amplitude é chamada de quantização dos níveis de cinza. Suponha que uma imagem contínua $f(x,y)$ seja aproximada por amostras igualmente espaçadas arranjadas na forma de uma matriz N x M, como mostra a equação abaixo. Assim, uma imagem digital pode ser vista como uma matriz de pontos com N linhas e M colunas onde cada elemento da matriz representa uma quantidade discreta pertencente a um intervalo $[0,k-1]$:

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(0,M-1) \\ \vdots & \ddots & & \vdots \\ f(N-1,0) & f(N-1,1) & \ddots & f(N-1,M-1) \end{bmatrix}$$

A matriz acima representa, o que normalmente é conhecido como uma imagem digital. Cada elemento da matriz é chamado de pixel. Os termos imagem e pixel serão utilizados nas discussões seguintes para denotar uma imagem digital e seus elementos.

A resolução de uma imagem depende extremamente da quantidade de amostras e de níveis de cinza utilizados na digitalização da imagem. Se aumentarmos a quantidade desses parâmetros a matriz digitalizada se aproximará muito da imagem original, contudo isso implicaria em uma grande quantidade de bits para representar a imagem, o que inviabiliza o armazenamento dessas informações.

É difícil definir o que seria uma boa imagem, pois isso varia de acordo com a aplicação requerida. Imagens com baixa resolução espacial apresentam geralmente replicação de pixels, o que produz um efeito tipo tabuleiro de damas, enquanto que imagens com baixo número de níveis de cinza geralmente

apresentam um efeito de falso contorno, mas geralmente imagens com grande quantidade de detalhes podem ser bem representadas utilizando poucos níveis de cinza.

Durante ao longo deste documento, vários conceitos foram introduzidos, de forma a embasar a compreensão do mesmo. Vamos agora tratar de vizinhança, conexidade, adjacência, objeto e histograma:.

- **Vizinhança**: em uma imagem digital 2D um pixel $p(x,y)$ tem quatro vizinhos que compartilham uma aresta com $p(x,y)$: $p(x+1,y)$, $p(x-1,y)$, $p(x,y+1)$ e $p(x,y-1)$. Este conjunto é chamado de vizinhança 4 de p ($N_4(p)$). Considerando os vizinhos que compartilham pelo menos um vértice com $p(x,y)$ temos um conjunto vizinhança 8 de p ($N_8(p)$). Este conjunto é formado pelos pixels de $N_4(p)$ e os pixels diagonais $p(x+1,y+1)$,$p(x-1,y+1)$,$p(x+1,y-1)$ e $p(x-1,y-1)$. Um tratamento especial é normalmente dado aos pixels que pertencem às bordas da imagem, pois alguns de seus vizinhos vão estar fora da imagem.
- **Conexidade**: Dois pixels são ditos conexos-4 (ou conexos-8) se eles são vizinhos-4 (ou vizinhos-8) e se ambos satisfazem algum critério de classificação. Por exemplo, se o nível de cinza deles está acima de um dado valor limiar.
- **Adjacência**: Dois pixels são ditos adjacentes-4 (ou adjacentes-8) se eles são conexos-4 (ou conexos-8). Dois subconjuntos de pixels são ditos adjacentes se pelo menos um pixel do primeiro conjunto for adjacente a um pixel do segundo.
- **Caminho**: Um caminho na imagem que vai de um pixel $p(x_0,y_0)$ a um pixel $p(x_n,y_n)$ é uma seqüência de pixels distintos com coordenadas (x_0,y_0), (x_1,y_1) , ..., (xn,yn), onde n é o comprimento do caminho, e (x_{i-1},y_{i-1}) e (x_i,y_i), i=1,2,..,n, são adjacentes. Dizemos então que este é um caminho-4 (ou caminho-8) dependendo do grau mais alto de adjacência ao longo do caminho.
- **Objeto**: Considerando o exemplo da figura a seguir, podemos particionar uma imagem em componentes conexas disjuntas de acordo com diferentes critérios de classificação.. Cada objeto pode ser definido como uma componente conexa cuja vizinhança é composta por outros objetos. Neste caso temos três objetos, O_1, O_2 e O_3. O interior de um objeto é definido pelos pixels que pertencem a sua componente conexa e o exterior pelos outros pixels da imagem. A fronteira do objeto é definida pelo conjunto de pixels de seu interior que possui pelo menos um vizinho no exterior do objeto. Uma borda de um objeto 2D pode ser definida como o subconjunto de pixels da fronteira que forma um caminho fechado. Um objeto, portanto, pode possuir várias bordas. As bordas do objeto O_2, por exemplo, são B_2, B_3 e B_4. As bordas possuem uma relação hierárquica: B_2 é a borda mais externa e contém duas bordas internas B_3 e B_4. Dizemos então que o

número de bordas internas é igual ao número de "buracos" do objeto. Veja na figura 3.27 uma imagem e suas componentes conexas:

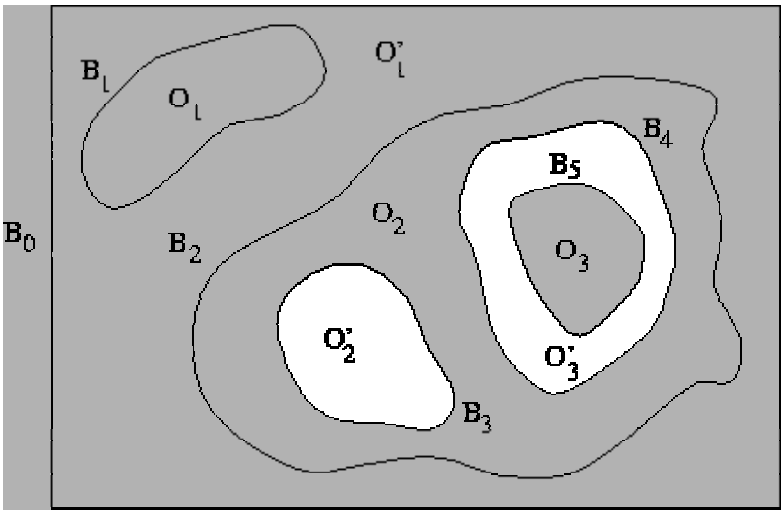


FIGURA 3.27 – Uma imagem e suas componentes conexas.

- **Histograma**: o histograma dos níveis de cinza de uma imagem constitui uma operação global indicando a freqüência de ocorrência dos níveis de cinza ou brilho. Fornece uma idéia global da dinâmica de uma cena e tem aplicações em filtragem, segmentação, reconhecimento de padrões etc. A figura 3.28 apresenta uma imagem com seu respectivo histograma:

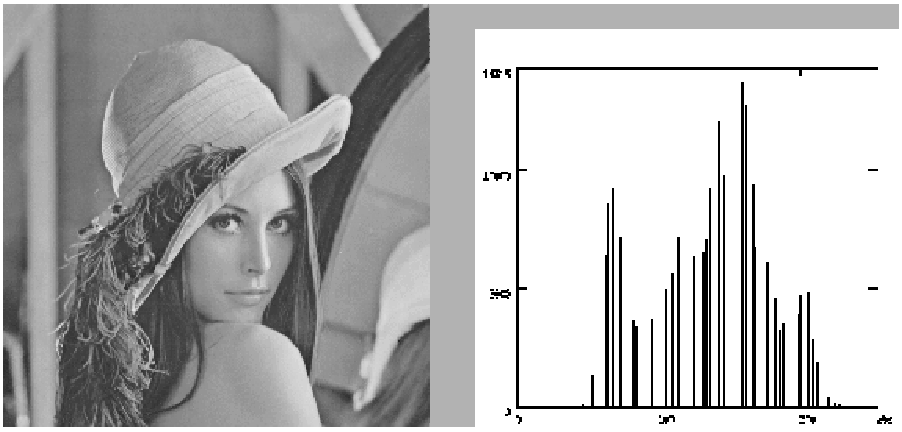


FIGURA 3.28 – Histogramas de imagens

4 Cálculo da Velocidade

Este capítulo trata do processo proposto para cálculo de velocidades de veículos, através de apenas uma única foto, utilizando processamento de imagem, que é a idéia principal do trabalho.

Basicamente, aponta-se uma câmera para a pista de rolamento que se deseja monitorar a velocidade, com foco na placa dos veículos. Esta câmera fica gerando quadros com imagens entrelaçadas e calculando a velocidade. Os que estiverem dentro do permitido serão desconsiderados; caso contrário, os dados sobre o veículo irão para um banco de dados que registra as infrações cometidas.

Veja o fluxo do processo na figura 4.1 e a seguir a explicação detalhada de cada etapa do processo:

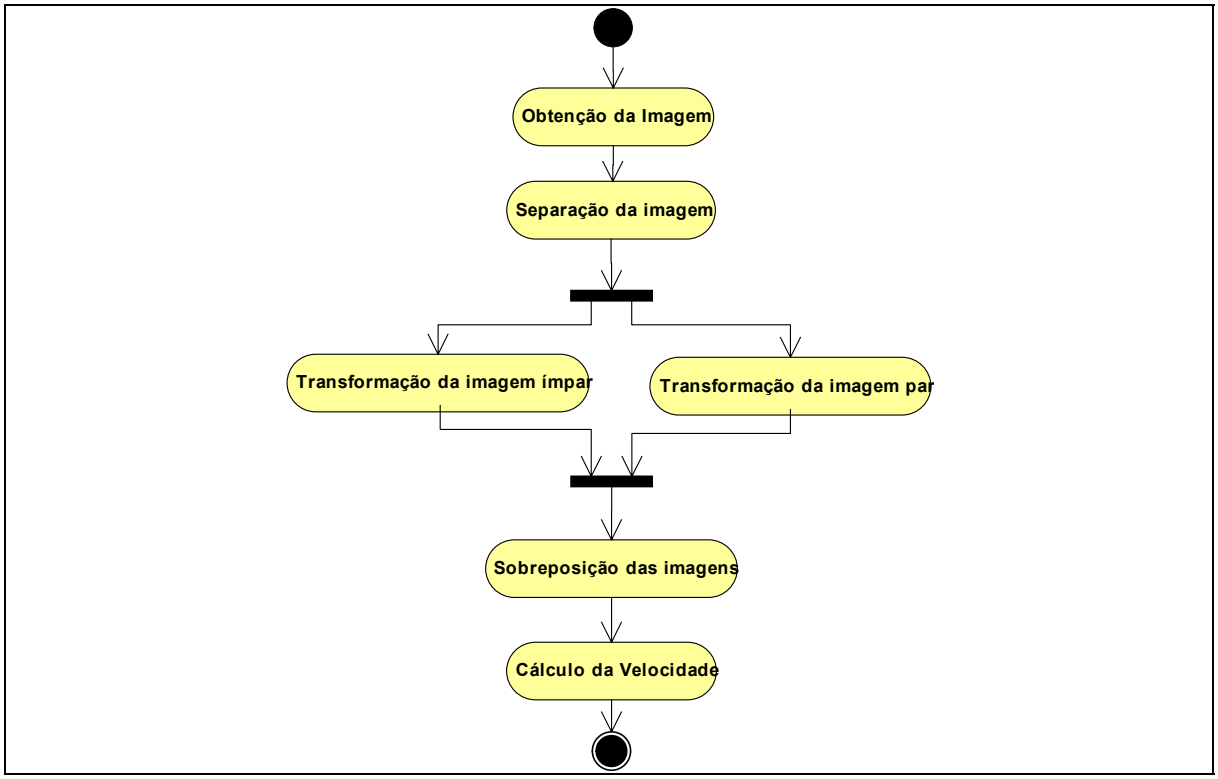


FIGURA 4.1 – Cálculo da velocidade usando processamento de imagem.

4.1 Obtenção da Imagem

Uma câmera fica permanentemente filmando a via pública. Esta câmera deve ser capaz de gerar imagens entrelaçadas, pois o princípio para o cálculo é justamente a diferença de varredura entre as linhas ímpares e pares. Veja na figura 4.2 um exemplo de imagem entrelaçada:

FIGURA 4.2 – imagem entrelaçada

4.2 Separação da imagem

A partir desta imagem, geram-se outras duas, uma só formada pelas linhas pares, outra só formada pelas linhas ímpares.

Como as linhas intermediárias são retiradas, as imagens ímpares e pares depois de separadas ficam nítidas e compactadas. Veja este efeito nas figuras 4.3 e 4.4:

FIGURA 4.3 – Imagem formada somente pelas linhas ímpares

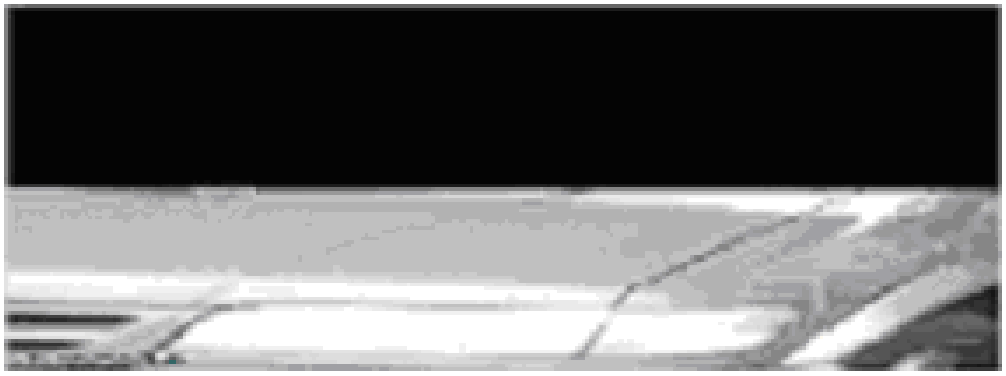


Figura 4.4 – Imagem formada somente pelas linhas pares

4.3 Transformação das imagens ímpar e par

Aplica-se um algoritmo de detecção de bordas nas imagens ímpar e par. No caso da aplicação, o escolhido foi o algoritmo de Sobel, explicado anteriormente. Veja nas figuras 4.5 e 4.6 as imagens resultantes após a aplicação deste algoritmo:

FIGURA 4.5 – Bordas da imagem de linhas ímpares

Figura 4.6 – Bordas da imagem de linhas pares

Observação: Sobel é um algoritmo de detecção de bordas simples, que atendeu aos experimentos realizados que embasam esta idéia. Para transformar o protótipo desenvolvido (veja o capítulo 5) em um produto comercial, é necessário homologar este algoritmo em ambientes de baixa luminosidade, pois é provável que ele não consiga detectar bordas de noite, por exemplo.

4.4 Sobreposição de imagens

A imagem capturada, separada em 2 e em cada uma delas aplicou-se o algoritmo de detecção de bordas. O próximo passo é sobrepor estas imagens para poder calcular a diferença entre as bordas geradas pelas linhas ímpares e pelas pares.

Para sobrepor estas imagens usam-se os planos de uma imagem RGB, colocando cada componente desta maneira:

a) **R** = **zero** (sem cor vermelha – este plano não será usado);
b) **G** = imagem formada pelas linhas **ímpares**;
c) **B** = imagem formada pelas linhas **pares**.

A figura 4.7 ilustra esta técnica:

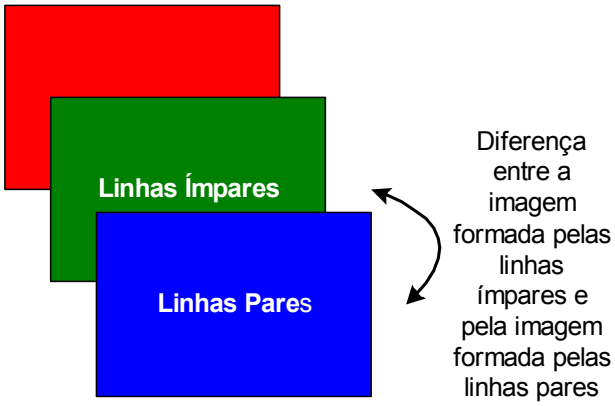


FIGURA 4.7 – Imagens sobrepostas usando o sistema RGB

E na prática, as figuras ímpar e par ficam como mostradas na figura 4.6:

FIGURA 4.8 – Sobreposição das imagens obtidas

Veja o processo completo na figura 4.9:

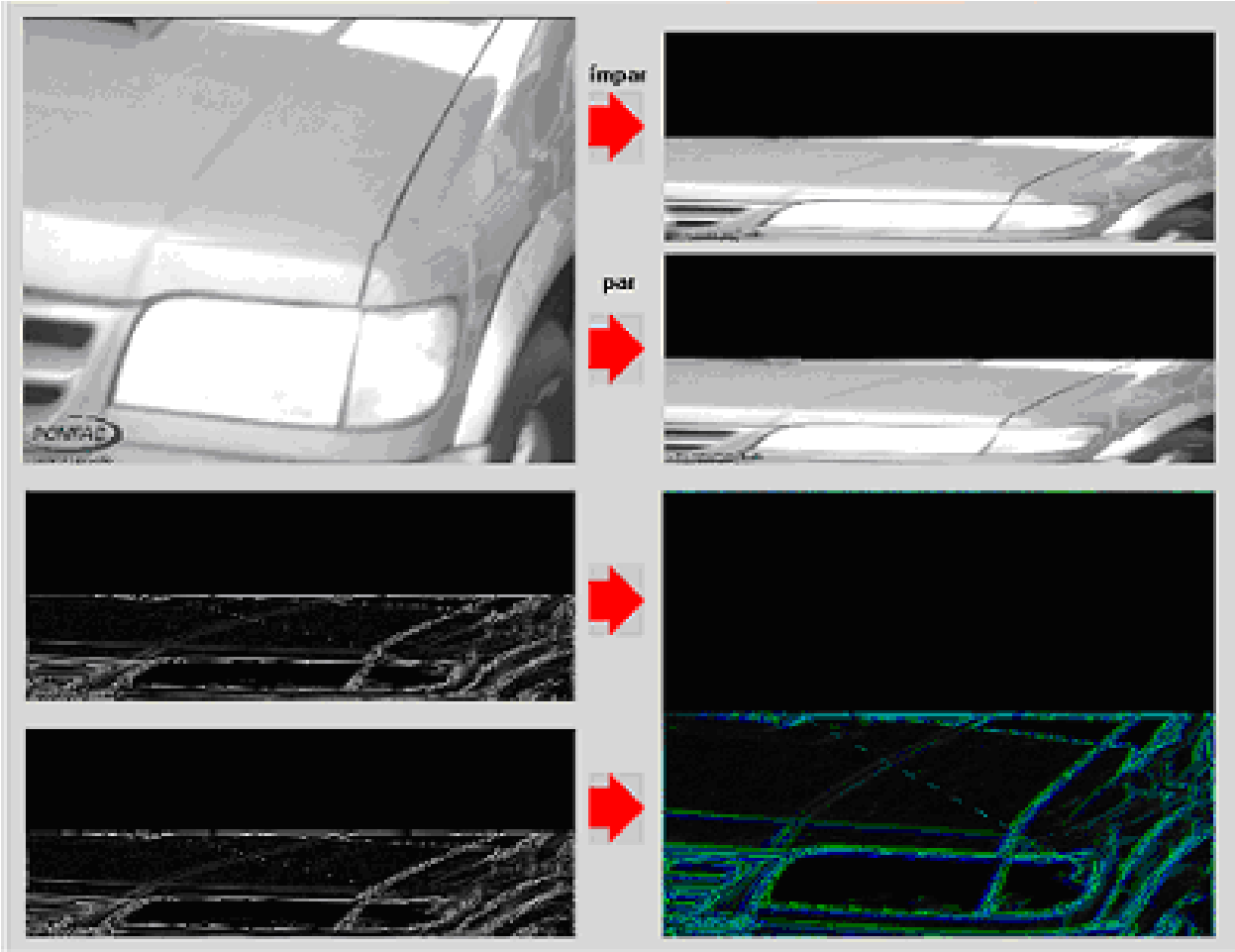


FIGURA 4.9 – Processo completo para transformação da imagem adquirida

4.5 O Processo de Cálculo da Velocidade

Para calcular a velocidade pode-se tentar identificar o tipo de veículo que foi fotografado. Mas motos, carros e caminhões, por exemplo, têm formatos distintos e uma solução para isto seria reconhecer padrões através de redes neurais. A desvantagem é que teríamos um algoritmo complexo que iria aumentar o tempo de processamento.

Outra solução, mais simples e eficaz, é não considerar qual o tipo de veículo e sim calcular a diferença entre o instante t e t+1, que na realidade é a diferença entre as bordas das imagens ímpar e par.

Como elas são bastante similares (os processos e algoritmos aplicados em casa imagem foi o mesmo), podemos deslocar os planos para cima e para baixo e ir

contando quantas linhas existem entre eles e o grau de similaridade, descontando as linhas iniciais.

A figura 4.10 mostra os planos das imagens ímpares e pares se deslocando de forma a obter o maior grau de similaridade e descobrir quantas linhas existem de diferença entre elas:

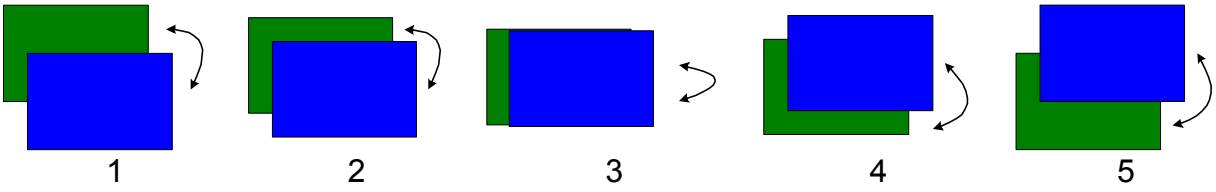


FIGURA 4.10 – Descobrindo a maior similaridade entre pares e ímpares

Para ficar mais claro, vamos transformar estas imagens: como a intensidade da cor varia de 0 a 255 (varia em cada componente RGB), coloca-se um X para todas aquelas que tiverem valores superiores a 128 e um espaço em branco nas demais. Veja nas figuras 4.11 e 4.12 como os padrões entre as figuras ímpares e pares são bastante similares, mas estão deslocados entre si:

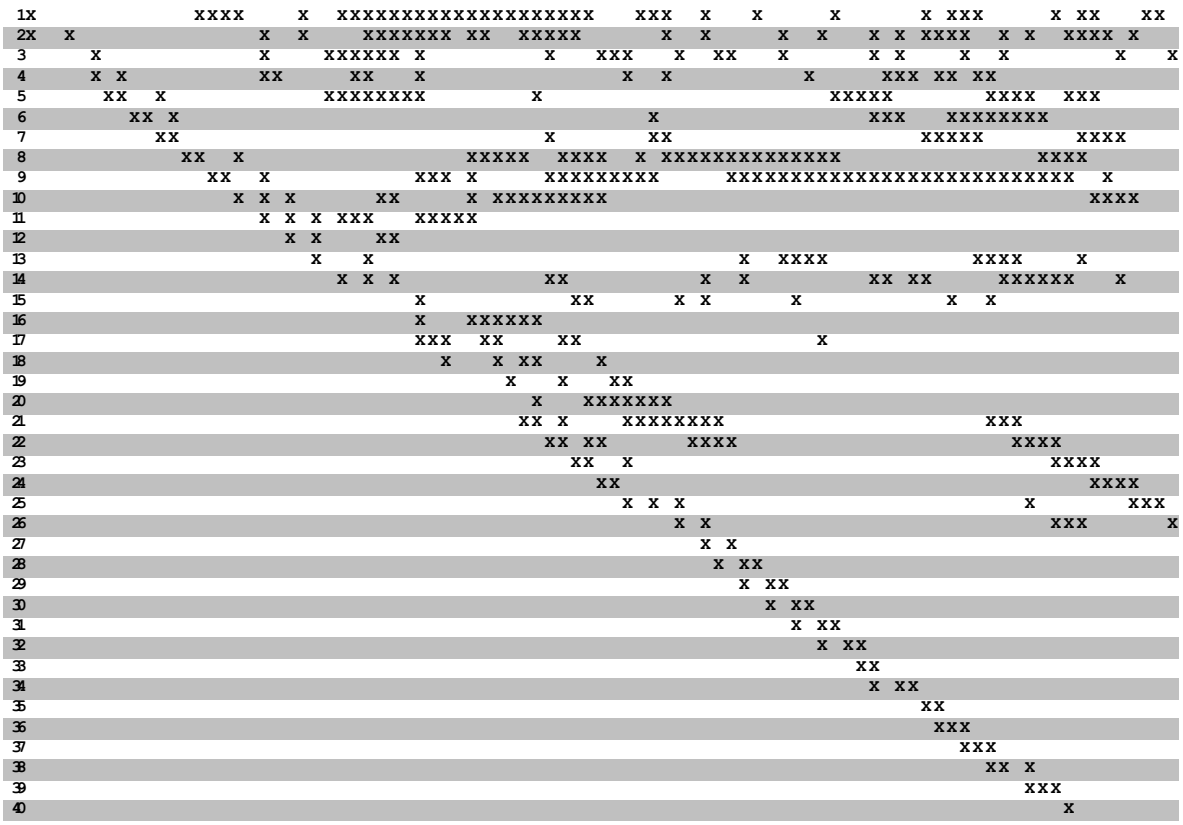


FIGURA 4.11 – Binário das bordas da imagem de linhas ímpares

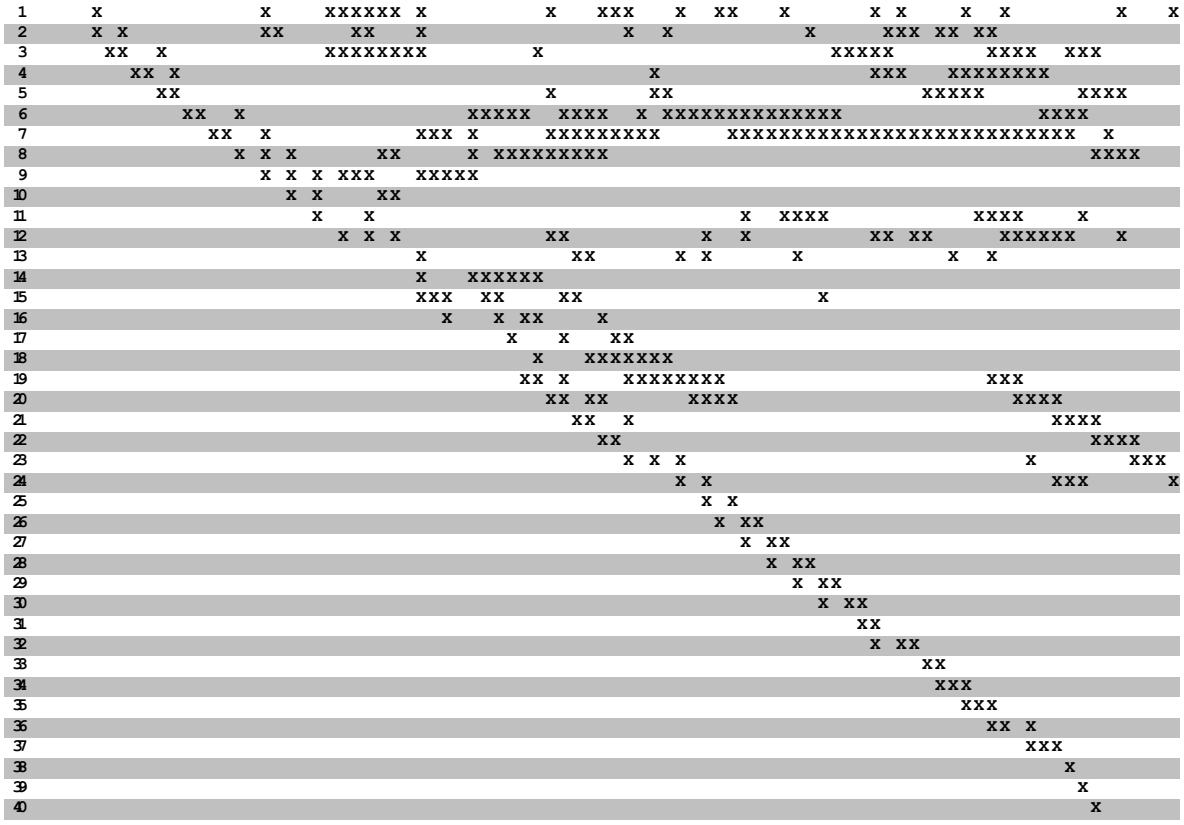


FIGURA 4.12 – Binário das bordas da imagem de linhas pares

Observe que são resultados praticamente idênticos, mas deslocados entre si (número de linhas encontradas entre uma imagem e outra).Neste exemplo, o veículo se deslocou 02 linhas.

Com o úmero de linhas deslocadas, a distância da câmera em relação à pista e uma tabela de correspondência, chega-se a velocidade atingida pelo veículo.

Obs: esta tabela de correspondência deve ser formada por dados obtidos na prática (depois de se instalar o sistema pela primeira vez, realizam-se várias medições para preenchê-la), que pode ficar como mostra a tabela 4.1:

TABELA 4.1 – Exemplo com o número de pixels em relação à câmera

Velocidade (em km/h)			
#pixels deslocados	distância da câmera até a pista		
	d<5	5 <=d<10	d>=10
1	20	22	24
2	25	27	29
3	30	32	34
4	35	37	39
5	40	42	44
6	45	47	49
7	50	52	54
8	55	57	59
9	60	62	64
10	65	67	69
11	70	72	74
12	75	77	79
13	80	82	84
14	85	87	89
15	90	92	94
16	95	97	99
17	100	102	104
18	105	107	109
19	110	112	114
20	115	117	119
21	120	122	124
22	125	127	129
23	130	132	134
24	135	137	139
25	140	142	144
>25	999	999	999

4.6 Considerações Finais

A grande vantagem deste método é a simplicidade, pois se consegue calcular a velocidade de veículos sem necessitar de algoritmos complexos, utilizando apenas tecnologias conhecidas, mas agrupadas de forma diferente.

Além disto, se a câmera estiver bem posicionada, numa única imagem capturada se obtém a velocidade e a placa do veículo.

5 Pardalzito

Este capítulo tem como objetivo descrever o funcionamento do Pardalzito, um sistema que implementa na prática a técnica proposta para se calcular a velocidade de veículos.

Observação: o manual de usuário está anexado a este documento e mostra todas as funcionalidades do sistema.

5.1 Descrição do Funcionamento

São três os módulos que compõem o sistema: Cálculo de Velocidade, Interface Homem-Máquina e Banco de Dados, que serão descritos a seguir.

5.1.1 Módulo de Cálculo de Velocidade

Desenvolvido em C++ Builder (veja maiores detalhes no apêndice A.7), é o módulo responsável pelo cálculo da velocidade. Possui um driver para adquirir imagens da câmera e transmite via socket para o módulo de Interface Homem-Máquina a velocidade calculada e a foto dos veículos que ultrapassaram o permitido.

5.1.2 Módulo de Interface Homem-Máquina

Desenvolvido em Delphi (veja maiores detalhes no apêndice A.8), é o módulo responsável pela interface com o usuário. É nele que se configura quantas câmeras estão disponíveis, qual a velocidade permitida, qual via será monitorada, etc. Possui um driver para adquirir imagens da câmera e gera um arquivo em formato ASCII para que o módulo de Cálculo de Velocidade possa se ajustar à configuração feita pelo usuário.

5.1.3 Banco de Dados

Usando o Interbase (veja maiores detalhes no apêndice A.9), armazena-se os dados sobre os veículos que ultrapassaram a velocidade permitida e o log de eventos do sistema via SQL.

5.2 Diagrama de Componentes

Veja na figura 5.1 o diagrama de componentes deste sistema:

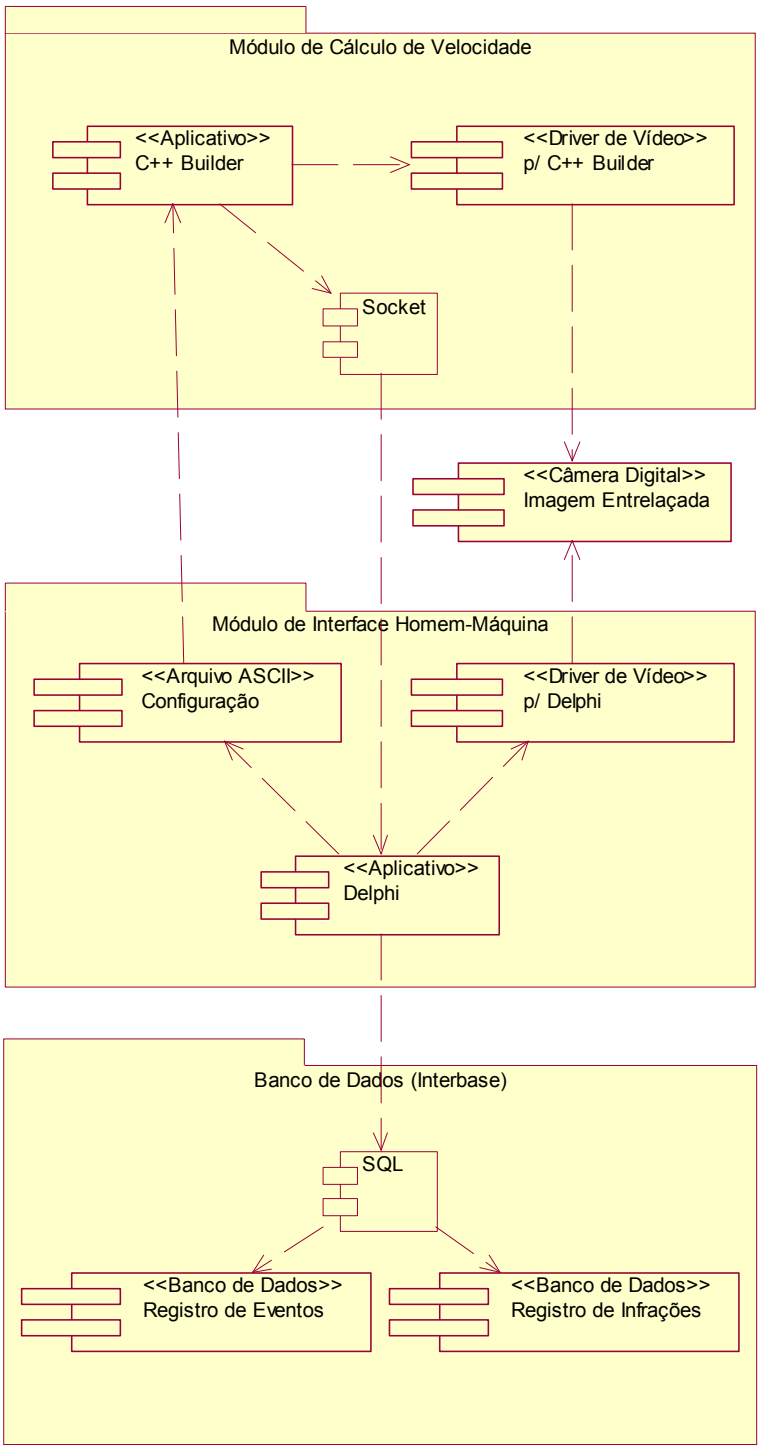


FIGURA 5.1 – Diagrama de Componentes do Pardalzito

5.3 Use Cases

O diagrama de *Use Cases* a seguir mostra as funções que os atores executam no sistema (figura 5.2):

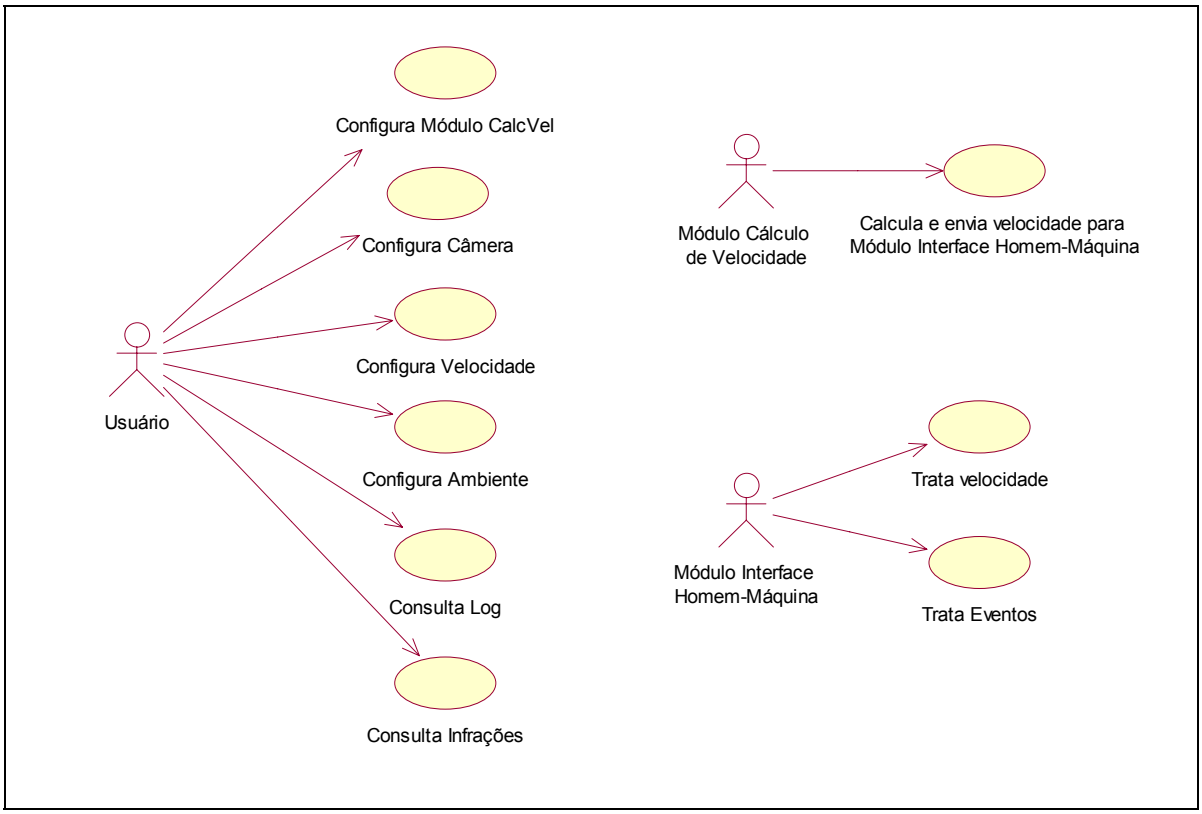


FIGURA 5.2 – Use Cases do Pardalzito

Atores

- **Usuário**: Realiza a configuração do módulo de cálculo de velocidade, da câmera, das velocidades permitidas (e em que faixas de horários e dias) e do ambiente (onde os arquivos estão dispostos). Também consulta o registro de eventos (log) e os veículos que ultrapassaram a velocidade estipulada.
- **Módulo CalcVel**: O módulo Cálculo de Velocidade recebe as imagens da câmera, analisa-as, calcula a velocidade dos veículos e envia as informações daqueles que ultrapassarem o limite permitido.
- **Módulo IHM**: O módulo Interface Homem-Máquina recebe as informações sobre os veículos que ultrapassaram a velocidade e armazena no banco de dados, além dos registros de eventos (log).

5.4 Hardware necessário

Para executar este sistema, é necessário que o hardware seja PC compatível e com esta configuração mínima:

- Processador: 1.8GHz
- Memória: 256MB
- HD: 30GB
- Placa de vídeo com resolução de 1024x768 pixels

6 Conclusão

Este trabalho apresentou o funcionamento e características de Pardais, Caetanos, Lombadas Eletrônicas, Bandeiras e Radares, que são equipamentos disponíveis no mercado para se calcular a velocidade de veículos. Mas que precisam de obras de infra-estrutura, gente operando ou possuem problemas de imprecisão, pois se baseiam no tempo de captura de quadros consecutivos mas sem utilizar um sistema operacional de tempo real.

Também apresentou uma proposta para se desenvolver um novo equipamento, através de tecnologias e ferramentas conhecidas, mas combinadas de uma forma diferente, com a idéia de reduzir custos, ter mais mobilidade e ser de fácil manutenção.

A idéia é utilizar uma câmera digital para filmar a via pública e ficar gerando imagens entrelaçadas que são analisadas permanentemente. A partir de cada quadro capturado, se separa a imagem par da imagem ímpar e aplica-se um algoritmo de reconhecimento de bordas em cada uma delas. As imagens resultantes são sobrepostas e a partir da diferença entre elas se calcula a velocidade de veículos (a distância da câmera da via e a taxa de aquisição de quadros são outros fatores que influenciam neste cálculo).

Para embasar esta proposta, toda a teoria e ferramentas utilizadas (processamento de imagens, algoritmos de detecção de bordas, sockets, TCP/IP, SQL, banco de dados, Delphi, C++, etc) foram explicadas detalhadamente e direcionadas à aplicação.

Além disto, foi desenvolvido um *software* aplicativo, cujo nome é Pardalzito, que utiliza todas as técnicas e ferramentas apresentadas e demonstra na prática o funcionamento do método proposto.

Mas além das melhorias sugeridas para o Pardalzito existem outros projetos que podem complementar este trabalho de forma a torná-lo mais completo:

a) Aferição: o sistema utiliza o entrelaçamento da imagem, associado com a quantidade de quadros que a câmera captura durante um determinado período e a distância que ela está instalada, mas não se sabe a resolução do que foi calculado. Exemplo: o sistema calculou que a imagem ímpar está deslocada em 2 linhas da imagem par. Qual a velocidade que isto representa? 5 km/h? 10 km/h? A partir desta conclusão, pode-se trabalhar para que este sistema seja qualificado pelo INMETRO;

b) Transmissão de dados: atualmente os órgãos públicos utilizam a chamada "Combi-net". Ou seja, periodicamente uma equipe recolhe em cada equipamento os dados armazenados. É um processo manual, demorado e dispendioso. A idéia é utilizar os períodos de ociosidade do equipamento (quando não tem carro passando) para transmitir os dados armazenados automaticamente de forma compactada. Isto pode ser feito através de celular ou da própria rede elétrica;

c) Linux: De forma a não precisar pagar licença para sistemas operacionais proprietários, pode-se implementá-lo em Linux e as ferramentas utilizadas (C++Builder, Delphi e Interbase) têm versões para este sistema operacional (Kylix e Interbase para Linux);

d) Melhorias diversas na aplicação, tais como ter apenas um banco de dados e os aplicativos calculando a velocidade de veículos em diversos pontos da cidade, criar módulo de segurança (login), para controlar o acesso de usuários no sistema e implementar um software de instalação do sistema.

Bibliografia

[BAL 99] BALL, H. **Usando Linux**. Rio de Janeiro: Campus, 1999. 650p. ISBN85-352-0385-0.

[BAR 94] BARKAKATI, Nabajyoti; **Visual C++** - Guia de Desenvolvimento Avançado. [S.l: s.n.], 1994. 1210p.

[BOR 2003] BORLAND. **Practical UML**: A Hands-On Introduction for Developers. Disponível em: <http://www.togethersoft.com/services/practical_guides/umlonlinecourse>. Acesso em: 12 fev. 2003

[BOV 2000] BOVIK, A. **Handbook of Image and Video Processing**. [S.l]: Academic Press, 2000. 891p.

[BRA 2000] BRAGA, A. de P.; CARVALHO, A. P. de L; LUDERMIR, T. B. **Redes Neurais Artificiais**: teoria e Aplicações. [S.l]: LTC, 2000. 262p.

[BRI 99] BRITO, S. F. **Algoritmos para Segmentação**. Universidade Federal da Paraíba. 1999. Disponível em: <http://www.dee.ufcg.edu.br/~acsi>. Acesso em: 08 fev. 2003.

[BUE 2000] BUENO, M. L. **Detecção de Bordas através de Algoritmo Canny**. Disponível em: <http://www.inf.ufsc.br/~visao/2000/Bordas>. Acesso em: 22 mar. 2004.

[BUI 2003] BUILDING, J. C. M. The University of Edinburg. **Graphics File Formats**. Disponível em: <http://www.dcs.ed.ac.uk/home/mxr/gfx>. Acesso em: 08 fev. de 2003

[CAL 2000] CALCULADOR DE LENTES. Disponível em: <http://www.gus.com.br/>. Acesso em: 08 fev. 2003.

[CAN 86] CANNY, J., A Computational Approach To Edge Detection, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Los Alamitos v.8, n.6, p 679 – 698, 1996.

[CAS 98] CASACURTA, A.; OSÓRIO, F. F.; MUSSE, S. R. **Computação Gráfica – Introdução**. Disponível em: <http://www.inf.unisinos.br/~osorio/CG-Doc/CG-Web/cg.html>. Acesso em: 08 fev. 2003

[CUR 2003] CURTIN, D. P. **Image Sensors - Capturing the Photograph**. Disponível em: <http://www.shortcourses.com/how/sensors/sensors.htm>. Acesso em: 22 mar. 2004.

[FAL 2003] FALCÃO, A. X. **Fundamentos de processamento de Imagem Digital**. Disponível em: <http://www.dcc.unicamp.br/~cpg/material-didatico/mo815/9802/curso/curso.html>. Acesso em: 22 mar. 20004.

[FER 2001] FÉRIS, M. A. A. **Estudo sobre Equipamentos Eletrônicos de Monitoramento de Velocidade**. Trabalho Individual (Mestrado em Engenharia da Computação) – Instituto de Informática, UFRGS, 2001

[GON 92] GONZALEZ, R. C. **Digital Image Processing**. Reading: Addison Wesley, 1992. 716p.

[GRE 2002] GREEN, B. **Canny Edge Detection Tutorial**. Disponível em: <http://www.pages.drexel.edu/~weg22/edge.html>. Acesso em: 24 mar. 2004.

[INM 2000] INMETRO. Disponível em: <http://www.transitoweb.hpg.com.br/sociedade/31/legis/INMETRO1.html>. Acesso em: 08 jul. 2001.

[JUN 2000] CESAR JUNIOR, R. M. **Análise e Reconhecimento de Formas**: Teoria e Prática. Disponível em: < http://www.ime.usp.br/~eira/mac5749/>. Acesso em: 08 fev 2003.

[JUN 2001] BRAZ JUNIOR, O. O. **Apostila Borland Delphi**. Disponível em <http://www.pegar.com.br/pegar_tutor.asp?link_id=77>. Acesso em: 09 jul. 2003.

[MAN 2001] MANN, S. **Intelligent Image Processing**. [S.l]: Wiley, 2001.

[MCR 97] MCREYNOLDS, T. **Convolutions**. Disponível em: <http://www.sgi.com/software/opengl/advanced97/notes/node145.html#SECTION0001>. Acesso em: 08 fev. 2003.

[PAR 97] PARKER, J. R. **Algorithms for Image Processing and Computer Visio**. New York: John Witey & Sons, 1997, 417p.

[PON 2003] PONFAC Sistemas de Visão. Disponível em: <http://www.ponfac.com.br>. Acesso em: 23 fev 2003.

[PRA 2089] PRATT, W. K. **Digital Image Processing**. New York: Addison-Wesley, 1989.

[PRO 2003] THE CODE PROJECT. **Edge Detection Using C**, Win32 SDK and GDI+. Disponível em: <http://www.codeproject.com/vcpp/gdiplus/edge_detection_in__c.asp>. Acesso em: 08 fev. 2003.

[PUC 2002] PUC-RIO, **Fundamentos da Imagem Digital**. Disponível em: <http://www.tecgraf.puc-rio.br/~scuri/pub/fid.pdf>. Acesso em: 17 set 2002.

[RUS 2003] RUSS, J. **The Image Processing and Measurement Cookbook**. Disponível em: <http://www.reindeergraphics.com/tutorial/index.shtml>. Acesso em: 08 fev. 2003.

[UTH 2003] UTHSCSA. **ImageTool**. Disponível em: <http://ddsdx.uthscsa.edu/dig/itdesc.html>. Acesso em: 08 fev. 2003.

[VIE 99] VIEIRA NETO, H.. MARQUES FILHO, O. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. 406p.

[WIG 2003] WIGGEN, E. **Elementary Digital Filtering**. Disponível em: <http://www.gamedev.net/reference/programming/features/edf>. Acesso em: 08 fev 2003.

[YOU 2003] YOUNG, I.T; GERBRANDS, J.J.; VAN VLIET, L.J. **Image Processing Fundamentals** - Contents. Disponível em: <http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip.html>. Acesso em: 08 fev. 2003.

[ZHO 2003] ZHOU, W. **A New Method of Edge Detection**. Disponível em: <http://www.prettyview.com/edge/>. Acesso em: 08 fev. 2003.

Apêndice 1 Técnicas e Ferramentas

Este Apêndice tem como objetivo comentar sobre teorias que embasaram este trabalho (Programação Orientada a Objetos, TCP/IP, sockets, arquivos BMP, Banco de Dados) e ferramentas utilizadas no desenvolvimento do Pardalzito (C++ Builder, Delphi, Interbase, BDE e IBConsole).

A.1. Programação Orientada a Objeto

A Programação Orientada a Objetos é relativamente nova, pois por volta de 1970 surgiram as primeiras publicações, mas o seu "boom" se deu nos anos 90, quando tornou-se a principal metodologia de desenvolvimento de software. A orientação a objetos permite modelar de forma mais natural o mundo real, pois as estruturas de dados são vistas como objetos, ou seja, têm características e funções. Seu maior objetivo é aumentar a produtividade do desenvolvimento de software através de uma maior expansibilidade e reutilização de código, além de controlar a complexidade e o custo da manutenção do mesmo.

A Programação Orientada ao Objeto, POO é uma extensão natural da programação estruturada. Requer que se usem boas técnicas de programação e o resultado é um código limpo, expansível, reutilizável e simples de se efetuar manutenção.

De acordo com Osmar de Oliveira Braz Junior, quando a metodologia orientada a objetos é utilizada, a fase de projeto do desenvolvimento do software está mais intimamente ligada à fase de implementação. Um dos pontos chaves da metodologia orientada a objetos é centralização das atenções nas Estruturas de Dados, ao contrário da metodologia estruturada, onde a atenção era centralizada nos procedimentos. Na orientação a objetos há uma maior aproximação entre dados e procedimentos, pois procedimentos são definidos em termos dos dados.

Vamos apresentar algumas definições importantes sobre POO:

- **Ancestral:** Super classe ou classe de base, a partir da qual outras classes podem ser criadas.
- **Atributos:** Variáveis de instância. São os dados de um objeto.
- **Classes -** a classe representa um tipo ou categoria de objetos, o modelo a partir do qual um objeto pode ser construído. É a estrutura propriamente dita, que define os dados e métodos daquela classe de objetos. O objeto em si, é uma instância da classe. Na programação estruturada podemos fazer uma analogia com os tipos e variáveis, onde a classe equivale ao tipo e o objeto à variável desse tipo.
- **Descendente:** Subclasse.

- **Encapsulamento**: é a capacidade de "esconder" detalhes de implementação (abstração). Seu principal objetivo é tornar o objeto independente de sua implementação interna, para isso a implementação das suas propriedades e métodos são "escondidas" de forma que o usuário precise apenas conhecer a interface do objeto para poder utilizá-lo. Desta forma o desenvolvedor poderá alterar tranqüilamente a implementação de um objeto (Classe) sem causar transtornos ao usuários.
- **Herança -** É a capacidade que uma classe de objetos tem de herdar variáveis e métodos de outra classe. Esta capacidade permite que o código já escrito seja reutilizado de maneira muito mais eficiente e simples do que na programação estruturada. Um programa orientado a objeto costuma implementar verdadeiras árvores genealógicas de classes, com vários níveis de herança.
- **Hierarquia de Classes:** Conjunto de classes ancestrais e descendentes, geralmente representadas em uma árvore hierárquica.
- **Interface**: Conjunto de mensagens que define o comportamento de um objeto (Protocolo).
- **Instância**: É o objeto propriamente dito. Possui características próprias.
- **Mensagem**: Representa uma ação do objeto ou uma mudança de estado. Define a comunicação entre objetos.
- **Métodos:** Funções e procedimentos de um objeto.
- **Objeto**: É a característica mais importante de uma linguagem orientada ao objeto. Pode ser definido como uma entidade lógica que contém dados e códigos para manipular esses dados. Dentro de um objeto, alguns códigos e/ou dados podem ser privados ao objeto e inacessíveis diretamente para qualquer elemento fora dele. Dessa maneira, um objeto evita significativamente que outras partes não relacionadas do programa modifiquem ou usem incorretamente as partes privadas do objeto. Essa ligação dos códigos e dos dados é freqüentemente referenciada como encapsulação. Para todas as intenções e propósitos, um objeto é uma variável de um tipo definido pelo usuário. Pode parecer estranho à primeira vista encarar um objeto, que une tanto código como dados, como sendo uma variável. Contudo, em programação orientada ao objeto, esse é precisamente o caso. Quando se define um objeto, cria-se implicitamente um novo tipo de dado.
- **Polimorfismo:** Capacidade de redefinir métodos e propriedades de uma classe em seus descendentes.
- **Propriedade**: Define as característica dos objetos de uma classe.

A.2. TCP/IP

De acordo com Luiz Carlos dos Santos, a arquitetura TCP/IP surgiu por causa do Departamento de Defesa do governo dos Estados Unidos da América, com objetivo principal de manter conectados mesmo que, apenas em parte, órgãos do governo e universidades.

A ARPANET, surgiu como uma rede que permaneceria intacta caso um dos servidores perdesse a conexão, e para isso, ela necessitava de protocolos que assegurassem tais funcionalidades trazendo confiabilidade, flexibilidade e que fosse fácil de implementar. Foi desenvolvida então, a arquitetura TCP/IP.

O modelo TCP/IP quando comparado com o modelo OSI, tem duas camadas que se formam a partir da fusão de algumas camadas, elas são: as camadas de Aplicação (Aplicação, Apresentação e Sessão) e Rede (Link de dados e Física). Veja na figura A1:

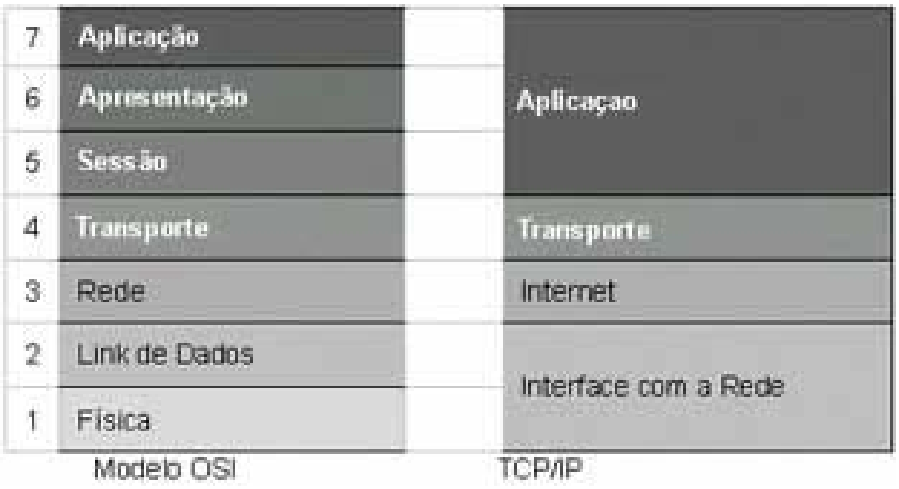


FIGURA A.1 – Modelo OSI e TCP/IP

A figura A2 mostra o modelo TCP/IP com suas camadas, seus protocolos e sua ligação física. Em seguida, temos uma breve explicação de cada uma delas:

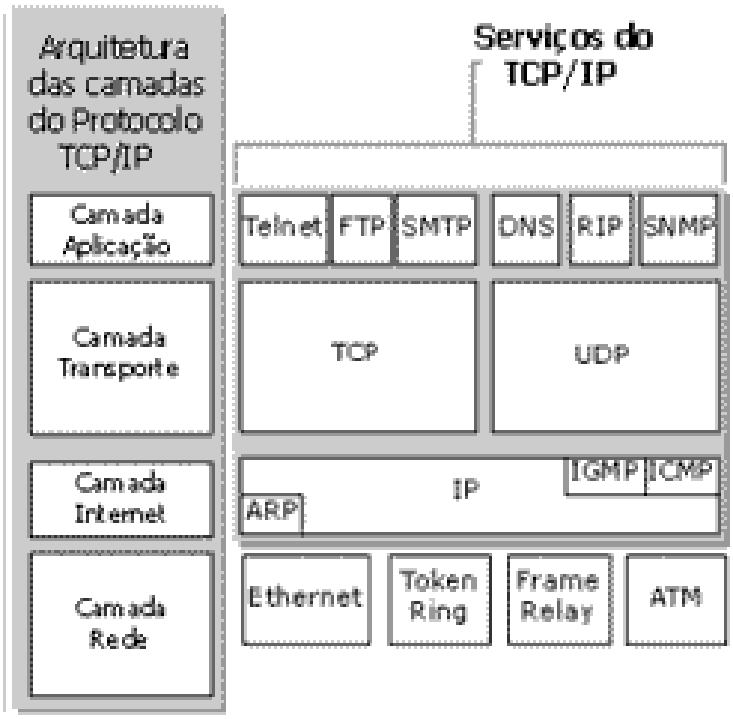


FIGURA A.2 – Modelo TCP/IP

Camada 7 - Aplicação

É formada pelos protocolos utilizados pelas diversas aplicações do modelo TCP/IP. Esta camada não possui um padrão comum. O padrão é estabelecido por cada aplicação. Isto é, o FTP possui seu próprio protocolo, assim como o TELNET, SMTP, POP3, DNS e etc.

Camada 4 - Transporte

Camada fim-a-fim, isto é, uma entidade desta camada só se comunica com a sua entidade-par do host destinatário. É nesta camada que se faz o controle da conversação entre as aplicações intercomunicadas da rede. Dois protocolos aqui são usados: o TCP e o UDP. O TCP é orientado à conexão e o UDP não. O acesso das aplicações à camada de transporte é feito através de portas que recebem um número inteiro para cada tipo de aplicação.

O TCP é um protocolo da camada de transporte confiável, ele é baseado em conexão encapsulada no IP. O TCP garante a entrega dos pacotes, assegura o seqüenciamento dos pacotes, e providencia um *checksum* que valida tanto o cabeçalho, quanto os dados do pacote. No caso da rede perder ou corromper um pacote TCP/IP durante a transmissão, é tarefa do TCP retransmitir o pacote faltoso ou incorreto. Essa confiabilidade torna o TCP/IP o protocolo escolhido para transmissões baseadas em sessão, aplicativos cliente-servidor e serviços críticos.

Os cabeçalhos dos pacotes TCP requerem o uso de bits adicionais para assegurar o correto seqüenciamento da informação, bem como um *checksum* obrigatório para garantir a integridade do cabeçalho e dos dados. Para garantia da entrega dos pacotes, o protocolo requisita que o destinatário, informe através do envio de um *acknowledgement*, para que seja confirmado o recebimento.

O protocolo UDP é a segunda opção da camada de transporte, sendo que ele não é confiável, pois não implementa *acknowledgements*," janelas" e nem seqüenciamentos, o único controle feito é um *checksum* opcional que está dentro do seu próprio *header*, ele é utilizado por aplicações que não vão gerar altos volumes de tráfego na Internet.

Camada 3 - Internet

Essa camada é a primeira normatizada do modelo. É responsável pelo endereçamento, roteamento e controle de envio e recepção. Ela não é orientada à conexão, se comunica através de datagramas.

O IP é o protocolo da camada Internet. Ele é encarregado da entrega de pacotes para todos os outros protocolos da família TCP/IP. Ele oferece um sistema de entrega de dados sem conexão. Isto é, os pacotes IP não são garantidos de chegarem ao seu destino, nem de serem recebidos na ordem em que foram enviados. O "checksum" do IP confirma apenas a integridade do cabeçalho do pacote.

O endereço IP é formado por um número de 32 bits no formato "nnn.nnn.nnn.nnn" onde cada "nnn" pode variar de 0 até 255 (1 octeto = 8 bits). Os endereços possuem uma classificação que varia de acordo com o número de sub-redes e de hosts. Tal classificação tem por finalidade otimizar o roteamento de mensagens na rede.

O protocolo IGMP, é o responsável por implementar a facilidade "IP multicasting", utilizada em empresas que tem diversos sites interligados por "Gateways" através de circuitos ponto a ponto.

O protocolo ICMP fornece mecanismos para reporte de erros, fazendo com que os "Gateways", possam informar ao host originador da requisição, a ocorrência de algum erro. Como conclusão, o ICMP apenas notifica a fonte original sobre determinada ocorrência de erro, sendo que esta fonte é responsável por efetuar o relato do mesmo à aplicação correspondente.

Quando um host remetente precisa saber o endereço físico do host destinatário, ele envia um pacote ARP na rede em broadcast contendo todos os campos conhecidos preenchidos, e o destinatário retorna uma réplica ARP após preencher os campos desconhecidos pelo remetente, ficando então, ambos os hosts e suas tabelas atualizadas.

Camada 2 – Rede

Camada de abstração de *hardware*, tem como principal função a interface do modelo TCP/IP com os diversos tipos de redes (X.25, ATM, FDDI, Ethernet, Token Ring, Frame Relay, PPP e SLIP). Por causa da grande variedade de tecnologias de rede, ela não é normatizada pelo modelo, o que provê a possibilidade de interconexão e interoperação de redes heterogêneas.

Cada serviço corresponde a um protocolo específico. No caso de e-mails, este serviço é atendido pelo protocolo SMTP, que, ao ser feita uma solicitação de e-mail (envio ou recebimento) ao TCP/IP, este é atendido pelo SMTP. No caso do www, usado para visualização de páginas, o protocolo usado é o HTTP. Existem ainda inúmeros outros.

O Ethernet (ANSI/IEEE 802.3 [ISO 8802-3]) é um padrão para redes em barra utilizando o CSMA/CD como método de acesso.

O Token Ring (ANSI/IEEE 802.5 [ISO 8802-5]) é um padrão para redes em anel utilizando passagem de permissão como método de acesso.

O Asynchronous Tranfer Mode (ATM) é um padrão para construção de redes de banda larga com integração de serviços digitais (RSDI/DVI).

Veja na figura A3 o esquema de um computador operando com TCP/IP.

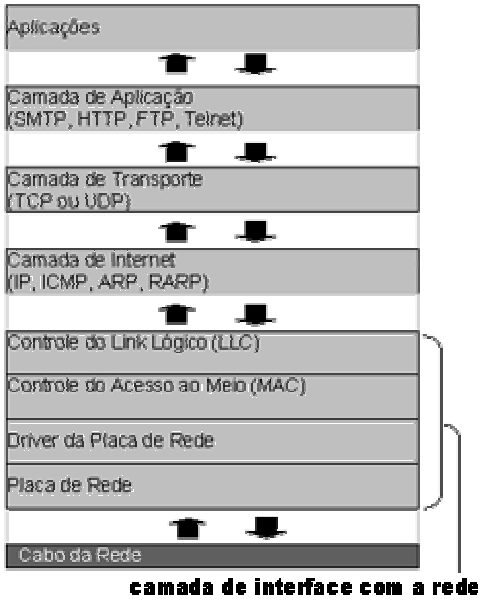


FIGURA A.3 - Esquema de um computador operando com TCP/IP

A.3. Sockets

De acordo com Antonio Marcelo, as grandes ferramentas utilizadas por especialistas de segurança, hackers e crakers tem como base a linguagem C ANSI ou C ++. Muitos dos scanners, sniffers, backdoors, etc. exploram um recurso muito conhecido na programação-cliente servidor: os sockets.

Um socket é nada mais nada menos que um programa, ou rotinas de programas que permitem a comunicação, interligação e troca de dados entre aplicações. Por exemplo: quando é feita uma conexão de FTP, um socket é estabelecido entre a origem e o destino.

Basicamente um socket pode ser declarado mediante três headers básicos :

```
#include <sys/types.h>
#include <sys/ socket.h>
#include <arpa/inet.h>
```

Estes três headers permitem que utilizemos as funções para a montagem de uma conexão.

A definição de um socket é feita da seguinte maneira em C :

```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
main(){
int e_socket;
...
}
```

Com isto começamos o nosso trabalho.Vamos começar utilizando os dois tipos de sockets, mais utilizados em aplicações, baseados no o protocolo TCP (Stream Sockets) e os que utilizam o protocolo UDP (Datagram Sockets). Estes sockets também são conhecidos como "SOCK_STREAM" e "SOCK_DGRAM", respectivamente.

A estrutura padrão em C de um socket pode ser definida da seguinte maneira:

```
struct sockaddr_in {
short int sin_family;
unsigned short int sin_port;
struct in_addr sin_addr;
unsigned char sin_zero[8];
}
```

Cada item destas linhas possui uma característica importante:

```
short int sin_family;
```

Tipo de família do socket, sendo que os padrões mais comuns seriam os seguintes:

```
AF_INET - ARPA INTERNET PROTOCOLS
AF_UNIX - UNIX INTERNET PROTOCOLS
AF_ISSO - ISO PROTOCOLS
AF_NS - XEROX NETWORK SYSTEM PROTOCOLS
```

```
unsigned short int sin_port; // Número da porta TCP ou UDP a ser utilizada para a comunicação dos programas.
struct in_addr sin_addr; // Endereço IP do host destino. Pode ser colocado de maneira direta ou por uma entrada de dados.
unsigned char sin_zero[8]; // Zera a estrutura do socket.
```

A declaração do socket é feita da seguinte maneira:

```
e_socket=socket(sin_family, tipo_do_socket_desejado,número_do_protocolo);
```

Traduzindo para o C ANSI ficaria assim:

```
e_socket = socket (AF_INET,SOCK_STREAM,0)
```

Onde 0 é o número do protocolo e pode ser substituído pelo seguinte :

```
  0 - IP - INTERNET PROTOCOL
  1 - ICMP - INTERNET CONTROL MESSAGE PROTOCOL
  2 - IGMP - INTERNET GROUP MULTICAST PROTOCOL
  3 - GGP - GATEWAY-GATEWAY PROTOCOL
  6 - TCP - TRANSMISSION CONTROL PROTOCOL
  17 - UDP - USER DATAGRAMA PROTOCOL
```

Exemplo mais completo agora:

```
main(){
int e_socket;
struct sockaddr_in destino;
e_socket = socket(AF_INET,SOCK_STREAM,0);
if(e_socket < 0)
{perror("Socket");
exit(1);
}destino.sin_family = AF_INET;
destino.sin_port = htons(2048);
destino.sin_addr.s_addr = inet_addr("10.0.0.1");
bzero(&(destino.sin_zero),8);
...
}
```

Quando um programa vai se comunicar com outro a função `connect()` do socket é utilizada para testar a conexão e iniciar o *processo de comunicação*. O protótipo da função é o seguinte:

```
int connect(socket,(struct sockaddr * )&destino, sizeof(destino));
```

```
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
int e_socket;
struct sockaddr_in destino;
int conexao;
main()
{
e_socket = socket (AF_INET,SOCK_STREAM,0);
if(e_socket < 0)
{
perror("ERRO !");
exit(1);
}
destino.sin_family = AF_INET;
destino.sin_port = htons(22);
destino.sin_addr.s_addr = inet_addr("10.0.0.20");
bzero(&(destino.sin_zero),8);
conexao = connect(e_socket,(struct sockaddr * )&destino,
sizeof(destino));
if(conexao < 0) {
perror("Porta fechada !\n");
close(e_socket);
exit(1);
}
printf("A PORTA 22 DO SSH ESTA ABERTA !\n");
close(e_socket);
}
```

Eis o programa que testa se aporta 22 está aberta. Ele funciona da seguinte maneira:

```
int e_socket;
struct sockaddr_in destino;
int conexao;
```

Declaração das variáveis do sockets.

```
e_socket = socket (AF_INET,SOCK_STREAM,0);
if(e_socket < 0)
{
perror("ERRO !");
exit(1);
}
```

Em seguida vamos declarar um socket do tipo TCP (SOCK_STREAM) e testamos se as funções de sockets estão ativas .

```
if(e_socket < 0)
{
perror("ERRO !");
exit(1);
}
```

Neste ponto declaramos o tipo de socket (AF_INET) a porta que quermos testar se está aberta (destino.sin_port = htons(22);) o endereço do host que quermos testar (destino.sin_addr.s_addr = inet_addr("10.0.0.20");) e zeramos a estrutura

```
(
bzero(&(destino.sin_zero),8);)
destino.sin_family = AF_INET;
destino.sin_port = htons(22);
destino.sin_addr.s_addr = inet_addr("10.0.0.20");
bzero(&(destino.sin_zero),8);
```

E no final do programa, testamos se a conexão está ativa ou não, utilizando a função CONNECT().

```
conexao = connect(e_socket,(struct sockaddr * )&destino,
sizeof(destino));
if(conexao < 0) {
perror("Porta fechada !\n");
close(e_socket);
exit(1);
}
printf("A PORTA 22 DO SSH ESTA ABERTA !\n");
close(e_socket);
}
```

A.4. Arquivos BMP

De acordo com Stefan Hetzl, o formato de arquivos .bmp (algumas vezes salvos como.dib) é o padrão do Windows desde suas versões iniciais; pode ser compactado mas não suporta animações.

A estrutura básica de dados é a seguinte:

```
BITMAPFILEHEADER    bmfh;
BITMAPINFOHEADER    bmih;
RGBQUAD             aColors[];
BYTE                aBitmapBits[];
```

Onde:

bmfh contém informações sobre o arquivo bitmap;
bmih contém informações sobre a figura;
aColors contém a tabela de cores;
aBitmapBits contém a figura de acordo com o formato definido em *bmih*.

Vamos a seguir detalhar cada um destes elementos.

A tabela A1 mostra a estrutura BITMAPFILEHEADER:

TABELA A.1 – Estrutura BITMAPFILEHEADER

Byte Inicial	Tamanho (bytes)	Nome	Valor padrão	Propósito
1	2	BfType	19778	Deve conter 'BM' para indicar que é um arquivo .bmp
3	4	BfSize	??	Especifica o tamanho do arquivo em bytes
7	2	bfReserved1	0	Zero
9	2	bfReserved2	0	Zero
11	4	BfOffBits	1078	Especifica o deslocamento em bytes desde o início do arquivo até os dados bitmap

A tabela A2 mostra a estrutura BITMAPINFOHEADER:

TABELA A.2 – Estrutura BITMAPINFOHEADER

Byte Inicial	Tamanho (bytes)	Nome	Valor padrão	Propósito
15	4	biSize	40	Especifica o tamanho da estrutura BITMAPINFOHEADER, em bytes.
19	4	biWidth	100	Especifica a largura da imagem, pixels.
23	4	biHeight	100	Especifica a altura da imagem, pixels.
27	2	biPlanes	1	Zero
29	2	biBitCount	8	Especifica o número de bits por pixel.
31	4	biCompression	0	Especifica o tipo de ocmpressão (usualmente é zero, que significa que não há compressão).
35	4	BiSizeImage	0	Especifica o tamanho da imagem, em bytes. Se não tem compressão, pode-se colocar zero.
39	4	biXPelsPerMeter	0	Usualmente zero
43	4	biYPelsPerMeter	0	Usualmente zero
47	4	BiClrUsed	0	Especifica o número de cores usados no bitmap. Deve ser zero se o número de cores é calculado através do biBitCount member.
51	4	BiClrImportant	0	Usualmente zero (todas as cores são importantes)

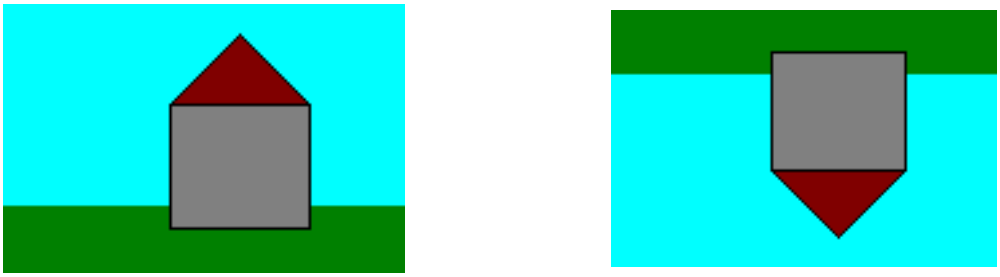
A tabela A3 mostra a estrutura RGBQUAD:

TABELA A.3 – Estrutura RGBQUAD

Byte Inicial	Tamanho (bytes)	Nome	Valor padrão	Propósito
1	1	RgbBlue	-	Especifica a intensidade da cor azul.
2	1	RgbGreen	-	Especifica a intensidade da cor verde.
3	1	RgbRed	-	Especifica a intensidade da cor vermelha.
4	1	RgbReserved	-	Zero

Armazenamento da Imagem

A imagem é armazenada de forma invertida ao mostrado na tela. Veja na figura A4.



Pixels mostrados na tela *Pixels* armazenados num arquivo .bmp

FIGURA A.4 – Estrutura RGBQUAD

A.5. Bancos de Dados

Todos nós sabemos que existem gigantescas bases de dados gerenciando nossas vidas. De fato sabemos que nossa conta bancária faz parte de uma coleção imensa de contas bancárias de nosso banco. Nosso Título Eleitoral ou nosso Cadastro de Pessoa Física, certamente estão armazenados em Bancos de Dados colossais. Sabemos também que quando sacamos dinheiro no Caixa Eletrônico de nosso banco, nosso saldo e as movimentações existentes em nossa conta bancária já estão à nossa disposição.

Nestas situações sabemos que existe uma necessidade em se realizar o armazenamento de uma série de informações que não se encontram efetivamente isoladas umas das outras, ou seja, existe uma ampla gama de dados que se referem a relacionamentos existentes entre as informações a serem manipuladas.

Estes bancos de dados, além de manterem todo este volume de dados organizado, também devem permitir atualizações, inclusões e exclusões do volume de dados, sem nunca perder a consistência. E não podemos esquecer que na maioria das vezes estaremos lidando com acessos concorrentes a várias tabelas de nosso banco de dados, algumas vezes com mais de um acesso ao mesmo registro de uma mesma tabela!

Um banco de dados é antes de mais nada uma coleção logicamente coerente de dados com determinada significação intrínseca. Em outras palavras um arquivo contendo uma série de dados de um cliente, um arquivo com dados aleatoriamente gerados e dois arquivos padrão dbf (dBase) que tem uma relação definida entre ambos, não pode ser considerada uma base de dados real.

Os Administradores de Banco de Dados (DBA) são responsáveis pelo controle ao acesso aos dados e pela coordenação da utilização do BD. Já os projetistas de Banco de Dados (DBP) são analistas que identificam os dados a serem armazenados em um Banco de Dados e pela forma como estes serão representados.

Os Analistas e Programadores de Desenvolvimento, criam sistemas que acessam os dados da forma necessária ao Usuário Final, que é aquele que interage diretamente com o Banco de Dados.

Resumindo, o Banco de Dados garante a independência lógica e física dos dados, permitindo alterar o esquema conceitual dos dados, sem alterar as visões dos usuários. Podemos também alterar o esquema interno, sem contudo alterar seu esquema conceitual.

Tipos de Banco de Dados

De acordo com *Alexandre Xavier Falcão*, os principais tipos de banco de dados são os orientados a objetos, universais, hierárquicos, de rede, semânticos e os relacionais. Iremos estudar este último tipo:

Relacional

Os bancos de dados relacionais representam os dados através de relações, que contém informações sobre as entidades representadas e seus relacionamentos. É baseado no conceito de matrizes, onde as chamadas linhas são os registros e as colunas os campos.

Vamos conhecer alguns de seus conceitos principais:

- **Tupla**: linha da relação;
- **Atributo**: coluna da relação, não são passíveis de novas divisões;
- **Domínio**: o conjunto de valores passíveis de serem assumidos por um atributo;
- **Esquema de uma relação**: campos existentes em uma tabela. Não podem ser duplicadas e a ordem de entrada não deve ter qualquer importância no que concerne ao seu tratamento;
- **Instância da relação**: consiste no conjunto de valores que cada atributo assume em um determinado instante;
- **Chave Primária**: atributo que definir um resgistro, dentre uma coleção de registros;

- **Chave Secundária (Terceária, etc):** são chaves que possibilitam pesquisas ou ordenações alternativas (diferentes da ordem criada a partir da chave primária ou da ordenação física da tabela;
- **Chave Estrangeira**: chave que permite a ligação lógica entre uma tabela (onde ela se encontra) com outra na qual ele é chave primária.

Componentes

Existem os seguintes componentes num Banco de Dados:

- **Gerenciador de Acesso ao Disco**: O SGBD utiliza o Sistema Operacional para acessar os dados armazenados em disco, controlando o acesso concorrente às tabelas do Banco de Dados. Controla todas as pesquisas solicitadas pelos usuários no modo interativo, os acessos do compilador DML, os acessos feitos pelo Processador do Banco de Dados ao Dicionário de Dados e também aos próprios dados;
- **Compilador DDL** (Data Definition Language): processa as definições do esquema do Banco de Dados, acessando quando necessário o Dicionário de Dados do Banco de Dados;
- **Dicionário de Dados:** contém o esquema do banco de dados, suas tabelas, índices, forma de acesso e relacionamentos existentes;
- **Processador do Banco de Dados**: manipula requisições à própria base de dados em tempo de execução. É o responsável pelas atualizações e integridade da base de dados;
- **Processador de Pesquisas** (queries): analisa as solicitações dos usuários e se forem consistentes, aciona o processador do banco de dados para acesso efetivo aos dados;
- **Compilador DML**: As aplicações fazem seus acessos ao pré-compilador DML (Data Manipulation Language) da linguagem hospedeira, que os envia ao Compilador DMLonde são gerados os códigos de acesso ao Banco de Dados.

Forma Normal

Num BD relacional, existem três formas normais:

- **Primeira Forma Normal**: Uma relação se encontra na primeira forma normal se todos os domínios de atributos possuem apenas valores atômicos (simples e indivisíveis), e que os valores de cada atributo na tupla seja um valor simples. Assim sendo todos os atributos compostos devem ser divididos em atributos atômicos.
- **Segunda Forma Normal**: Uma relação se encontra na segunda forma normal quando estiver na primeira forma normal e todos os atributos que não participam da chave primária são dependentes desta. Assim devemos verificar se todos os atributos são dependentes da chave primária e retirar-se da relação todos os atributos de um grupo não dependente que dará origem a uma nova relação, que conterá esse atributo como não chave. Desta maneira, na segunda forma normal evita inconsistências devido a duplicidades.
- **Terceira Forma Normal**: Uma relação estará na terceira forma normal, quando estiver na primeira forma norma e todos os atributos que não participam da chave primária são dependentes desta porém não transitivos. Assim devemos verificar se existe um atributo que não depende diretamente da chave, retirá-lo criando uma nova relação que conterá esse grupo de atributos, e defina com a chave, os atributos dos quais esse grupo depende diretamente.

A.6. SQL

De acordo com Anderson Haertel Rodrigues, quando os Bancos de Dados Relacionais estavam sendo desenvolvidos, foram criadas várias linguagens destinadas à sua manipulação. O Departamento de Pesquisas da IBM desenvolveu a SQL como forma de interface para o sistema de Banco de Dados relacional denominado SYSTEM R, no inicio dos anos 70. Em 1986 o American National Standard Institute (ANSI), publicou um padrão SQL e ela se estabeleceu como linguagem padrão de Banco de Dados Relacional.

SQL disponibiliza uma série de comandos que permitem a definição dos dados, chamada de DDL (Data Definition Language), que é destinado a criação do banco de dados, das tabelas que o compõe, além das relações existentes entre elas. Como exemplo de comandos da classe DDL temos os comandos Create, Alter e Drop.

Os comandos da série DML (Data Manipulation Language), destinados a consultar, inserir, excluir e alterar em um ou mais registros de uma ou mais tabelas de maneira simultânea.

Vale destacar algumas características importantes de SQL:

a) A capacidade de gerenciar índices sem a necessidade de controle individualizado de índice corrente;

b) A capacidade de construção de visões, que são formas de visualizarmos os dados na forma de listagens independente das tabelas e organização lógica dos dados;

c) A capacidade que dispomos de cancelar atualizações em andamento, através dos comandos Commit e Rollback são responsáveis por estas facilidades.

Devemos notar que a linguagem SQL consegue implementar estas soluções, somente pelo fato de estar baseada em banco de dados, que garantem por si mesmo a integridade das relações existentes entre as tabelas e seus índices.

A.7. C++Builder

O C++ Builder da Borland é um ambiente visual, orientado a objetos que tem por finalidade desenvolver aplicações rapidamente para o MS Windows. Estas aplicações podem ser de propósitos gerais ou cliente/servidor, com o mínimo de codificação manual. Além disto, C++ Builder disponibiliza uma extensa biblioteca de componentes reutilizáveis e um ambiente de ferramentas RAD (Desenvolvimento de Aplicações Rápida).

A.8. Delphi

O Delphi é um ambiente de desenvolvimento de aplicações, orientado a objeto, que permite o desenvolvimento de poderosas aplicações baseadas no MS Windows com o mínimo de codificação. O Delphi também oferece ferramentas de desenvolvimento, tais como templates de aplicações e forms, que lhe permitem criar e testar rapidamente o protótipo de suas aplicações. Você pode utilizar o conjunto de componentes e código gerado para transformar seus protótipos em aplicações robustas que satisfaçam suas necessidades. O Delphi também oferece ferramentas de bancos de dados que lhe permitem desenvolver aplicações Client/Server e relatórios. As ferramentas de bancos de dados permitem que se visualize seus dados dinamicamente durante o desenvolvimento para que verifique imediatamente se os resultados de suas queries estão de acordo com suas necessidades.

Observação: Kylix é um ambiente de desenvolvimento similar e voltado para Linux. Permite que se desenvolvam aplicações em C++ e Delphi diretamente para este sistema operacional e migrar códigos projetados em C++ Builder e Delphi para Windows sem maiores dificuldades.

A.9. Interbase

O Interbase® é um poderoso banco de dados Cliente/Servidor relacional, compatível com SQL-ANSI-92, de fácil instalação e uso e projetado para ser independente de plataformas e de sistemas operacionais. Tem como características principais acesso nativo a driver JDBC, sombreamento do Banco de Dados, replicação, tratamento de blobs, sistema de Eventos e executa em diversos sistemas operacionais.

Embora tenha características avançadas, o Interbase® não é tão reconhecido como o Oracle, o Microsoft SQL Server e outros servidores SQL. Isto se deve em parte a falta de *marketing* e divulgação por parte do fabricante nos meios especializados. No entanto, isto vem mudando pelo fato de que este produto está disponível na internet para *download* gratuitamente (Open Source). Veja na figura A5 como o Interbase e comunica com outros componentes:

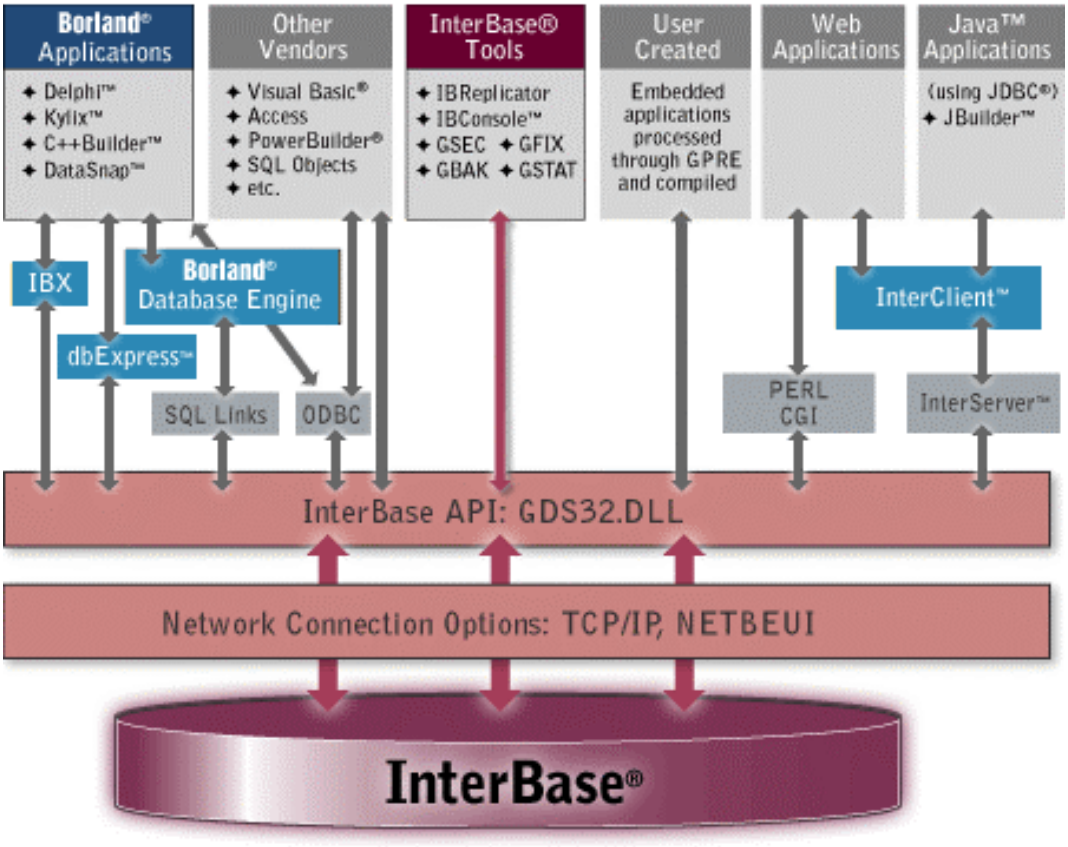


FIGURA A.5 – Como o Interbase e comunica com outros componentes

A.11. BDE

De acordo com Oliveira Braz Junior, o BDE, Borland Database Engine, é um núcleo de Banco de Dados que fornece a capacidade de acesso a banco de dados para os softwares Delphi, Paradox, dBase e C++, oferecendo um grande conjunto de características previamente testadas para auxiliar desenvolvedores de aplicações Cliente-Servidor. Tem como características principais à orientação a objetos e possui drivers específicos para cada SGBD. Veja na figura A6 o relacionamento entre o BDE e outros produtos:

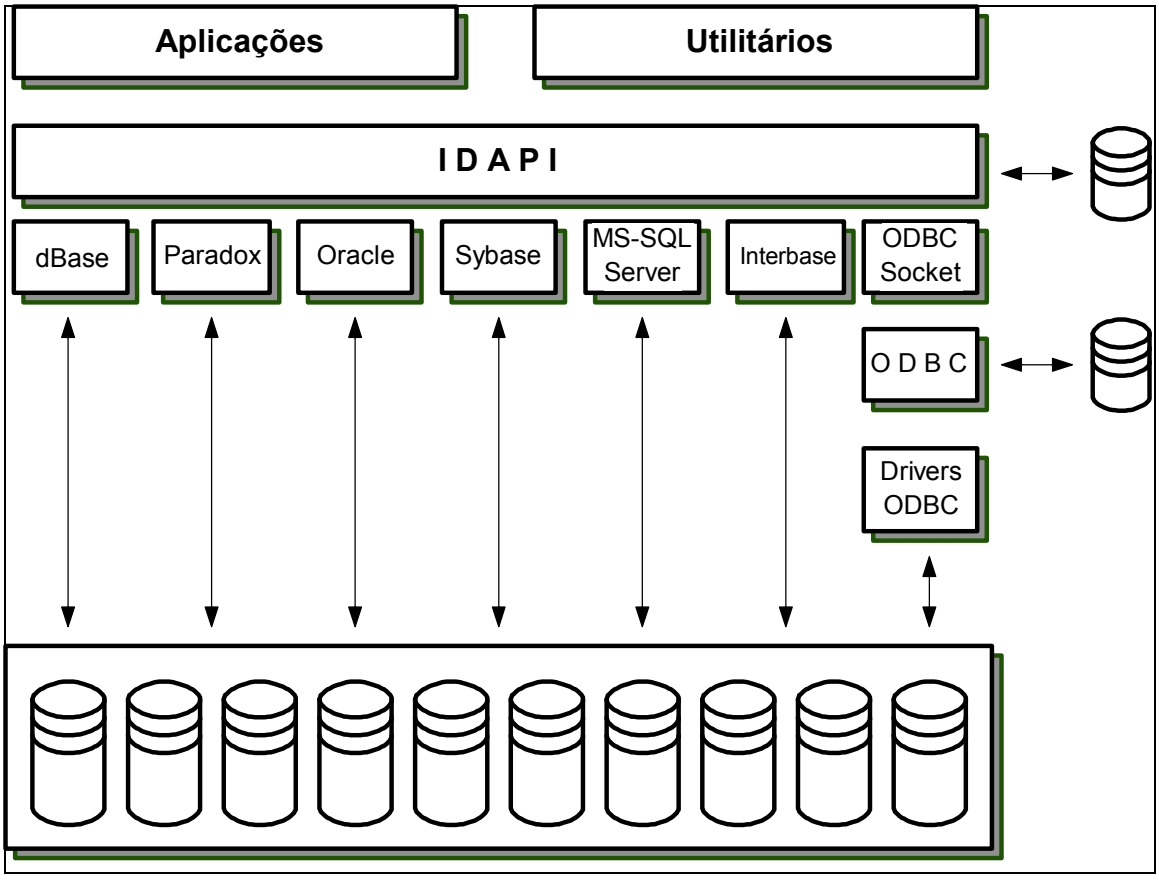


FIGURA A.6 – Relacionamento entre o BDE e outros produtos

A.12. IBConsole

O IBConsole é o gerenciador de dados que acompanha o InterBase. É uma excelente ferramenta para aprender a linguagem SQL, pois se usa para criação, relacionamento e manutenção através da linha de comando.

Apêndice 2 Portaria #115 do INMETRO

Apêndice 3 Manual de Operação do Pardalzito