

ARTHUR FERREIRA BAILÃO

**MÉTODO AUTOMÁTICO DE CONTAGEM
VOLUMÉTRICA DE VEÍCULOS BASEADO EM
VISÃO COMPUTACIONAL**

ORIENTADOR: PROF. HERMES AGUIAR MAGALHÃES

SUPERVISOR: PROF^A. LEISE KELLI DE OLIVEIRA

Belo Horizonte

Novembro de 2014

“A ciência pode ser descrita como a arte da sistemática simplificação.”

(Karl Popper)

Resumo

A Contagem Volumétrica consiste em quantificar o volume de veículos trafegando em uma determinada via, durante um intervalo de tempo definido. A informação coletada é de grande importância, pois seus resultados são subsídios básicos para estudos econômicos, projetos rodoviários e planejamento de tráfego. Neste trabalho é proposto e validado um método para contagem volumétrica de veículos baseado em Visão Computacional. Se diferencia de outros trabalhos pela simplicidade de implementação, aproveitando de recursos de *software* já consolidados na área, com resultados semelhantes aos de soluções mais complexas. Os índices de desempenho mostram que o método é bastante preciso, sendo o melhor resultado o índice Kappa 0.86.

Palavras-chave: Visão Computacional, Contagem Volumétrica de Veículos, Engenharia de Transporte.

Abstract

The Volumetric Count is a method used to quantify the amount of vehicles transiting on certain way, during a known period of time. This information is important due to the fact that its conclusions can be used to economical studies, highway projects and traffic planing. In this document it is proposed and validated a method to volumetrically count the vehicles on a way using computer vision. It is different from other approaches because it is simpler, taking advantage of software resources already available and its results are similar to the ones acquired by more complex solutions. The performance indices show that the method is very precise, with the best result Kappa index of 0.86.

Keywords: Computer Vision, Vehicle Counting, Transportation Engineering.

Lista de Figuras

2.1	Seis laços indutivos instalado em uma rodovia, dois por faixa [Goldner, 2009].	10
2.2	Representação esquemática de um cabo piezoelétrico [Goldner, 2009].	12
2.3	Esquema ilustrativo da arquitetura de um sistema de monitoramento do tráfego baseado em sensores micro-ondas [Goldner, 2009].	13
2.4	Tipos de montagem para um sensor ultrassônico [Goldner, 2009].	14
3.1	O ser humano é capaz de determinar a forma e a transparência de cada pétala de uma flor [Szeliski, 2010].	20
3.2	Para um computador, o retrovisor de um carro é apenas uma matriz composta por pixels [Bradski & Kaehler, 2008].	21
3.3	Convenção dos eixos para representação de imagens digitais [Gonzalez & Woods, 2000].	22
3.4	(a) Cubo de cores RGB. Os pontos ao longo da diagonal principal têm valores da escala de cinza que vão do preto (0, 0, 0) ao branco (255, 255, 255) [Gonzalez & Woods, 2000]; (b) Representação em cores do cubo RGB [Schouten, 2003].	24
3.5	Exemplo de conversão para <i>grayscale</i> . (a) Imagem original RGB; (b) Imagem em escala de cinza.	25
3.6	Histograma de uma imagem <i>grayscale</i> gerado pelo software GIMP [2013]. .	25
3.7	Filtragem linear por convolução: a imagem da esquerda convolução com o filtro no centro produz a imagem da direita. Os pixels azuis indicam a vizinhança usada para gerar o pixel verde na saída [Szeliski, 2010].	26
3.8	Percebe-se o comportamento passa-baixa do filtro atenuando componentes de alta frequência na imagem (bordas). (a) Imagem original; (b) Aplicação de filtragem linear gaussiana [Laganière, 2011].	27
3.9	Segmentação de uma maçã utilizando limiarização binária [OpenCV Development Team, 2013].	28

4.1	(a) Posicionamento da câmera para captura de imagens. A área em destaque simboliza o campo de visão obtido na posição escolhida. (b) Cena capturada na Avenida Presidente Carlos Luz.	34
4.2	Fluxograma com a representação global do método de contagem.	35
4.3	<i>Frame</i> obtido a partir do vídeo de entrada.	37
4.4	Resultado da etapa de pré-processamento, uma imagem filtrada em escala de cinza.	38
4.5	Detecção do <i>foreground</i> na etapa de subtração de fundo.	39
4.6	Resultado da etapa de binarização. (a) Imagem binarizada utilizando um limiar simples; (b) Operação morfológica de fechamento.	41
4.7	Resultado da etapa de detecção de <i>blobs</i>	42
4.8	Máscara binária que define a região de contagem do algoritmo.	43
4.9	Fluxograma do algoritmo de rastreamento e contagem de veículos.	44
4.10	Resultado da etapa de rastreamento e contagem, ilustrando <i>trackers</i> , <i>key-points</i> e a região de contagem.	45
4.11	Diagrama UML do <i>software</i>	46
5.1	Resultado do método de contagem para dois tipos de cena. (a) Av. Carlos Luz, sentido Pampulha; (b) Av. Carlos Luz, sentido Centro.	52
5.2	Tipo de tráfego encontrado nas vias. (a) Veículos muito grandes causam problemas como oclusão e vibrações na passarela. (b) Quando a cor de um veículo é muito próxima da cor do <i>background</i> , problemas na detecção do objeto podem acontecer.	53
5.3	Problemas encontrados no processamento de imagens. (a) A operação de subtração de <i>background</i> fica comprometida quando veículos com área muito grande aparecem na cena. (b) Veículos próximos são detectados como um único objeto, gerando eventos falso negativo FN.	54

Lista de Tabelas

2.1	Vantagens e desvantagens de tecnologias detectoras de veículos [Magalhães, 2008].	7
2.2	Comparação entre resultados de tempo de execução do método pelo <i>laptop</i> e celular [Feitosa, 2012].	17
4.1	Matriz de confusão para um classificador de duas classes.	48
4.2	Nível de exatidão de uma contagem, conforme o valor do índice Kappa.	50
5.1	Vídeos utilizados nos testes.	52
5.2	Resultado obtido da contagem nas amostras de teste, índices de desempenho e índice Kappa (K).	53

Sumário

Resumo	v
Abstract	vii
Lista de Figuras	ix
Lista de Tabelas	xi
1 Introdução	1
1.1 Objetivos	3
1.2 Organização do Texto	3
2 Contagem de Veículos e Monitoramento do Tráfego	5
2.1 Tecnologias detectoras de veículos	7
2.1.1 Detector de laço indutivo	9
2.1.2 Tubo pneumático	10
2.1.3 Sensor magnético	11
2.1.4 Sensor piezoelétrico	11
2.1.5 Câmera de vídeo	12
2.1.6 Sensores micro-ondas	13
2.1.7 Sensores ultrassônicos	14
2.2 Aplicações de Visão Computacional	15
2.2.1 Contagem volumétrica utilizando dispositivos móveis	15
3 Visão Computacional	19
3.1 Imagem digital	22
3.2 Modelo RGB de cores	23
3.3 Escala de cinza	23
3.4 Histograma	25

3.5	Filtragem linear	26
3.6	Limiarização	28
3.7	Biblioteca OpenCV	29
3.8	Aplicações existentes	30
4	Metodologia	33
4.1	Características de captura	33
4.2	Fluxo de processos	35
4.2.1	Entrada de dados	36
4.2.2	Pré-processamento	37
4.2.3	Subtração de <i>background</i>	38
4.2.4	Binarização	39
4.2.5	Detecção de <i>blobs</i>	41
4.2.6	Rastreamento e contagem	43
4.3	Estrutura do <i>software</i>	46
4.4	Avaliação dos resultados	47
4.4.1	Matriz de confusão	47
4.4.2	Índice Kappa (K)	49
5	Testes e Resultados	51
5.1	Testes	51
5.2	Resultados e discussão	52
6	Considerações Finais	55
6.1	Limitações	55
6.2	Trabalhos futuros	56
	Referências Bibliográficas	57

Capítulo 1

Introdução

O tráfego de veículos representa um fenômeno de grande importância socioeconômica, principalmente nos grandes centros urbanos cujos deslocamentos, no menor tempo possível, são uma necessidade cotidiana. Projetar sistemas viários que absorvam toda a frota veicular desses grandes centros e minimizam os congestionamentos exige ferramentas cujo desenvolvimento ainda representa objeto de estudo para diversos grupos de pesquisadores.

Vários fatores interferem na qualidade do tráfego, sendo o principal o crescente número de veículos nos centros urbanos. Segundo FENABRAVE [2013] - Federação Nacional da Distribuição de Veículos Automotores, as vendas de veículos cresceram cerca de 381% entre janeiro de 2003 e janeiro de 2011. A facilitação ao crédito pessoal, os longos financiamentos oferecidos pelas concessionárias e a redução do IPI (Imposto sobre Produtos Industrializados) incentivaram o consumo de carros no Brasil, explicando essa tendência ascendente no número de automóveis.

A falta de planejamento urbano e o crescente aumento de veículos, a péssima qualidade do transporte público no Brasil e o excessivo número de acidentes de trânsito são umas das principais causas de congestionamentos. Nos últimos anos, milhões de pessoas perderam tempo e dinheiro devido a problemas relacionados ao trânsito [Tancredi, 2012]. Diante desse fato, a administração pública deve priorizar o assunto mobilidade, como tem sido feito nas principais regiões metropolitanas brasileiras. Através de pesquisas pode-se levantar dados que contribuem para análise e simulação do tráfego. Informações como fluxo de veículos, volume de tráfego e matriz O/D (origem/destino) são de grande relevância no estudo dessa área, formando uma base teórica sólida para ser utilizada na resolução de problemas dessa natureza.

Na área de simulação do tráfego existem várias empresas atuantes: PTV [2013] é atualmente a líder mundial de mercado em simulação de fluxo de tráfego multi-modal

de microregiões. Outras companhias, TRL Software [2013] e CITILABS [2013], também fornecem produtos com capacidades de predição, simulação de congestionamentos, atrasos, acidentes, além de gerar relatórios detalhados e animações 2D e 3D. SUMO - *Simulation of Urban MObility* é uma alternativa *open source*¹ e gratuita para simulação do tráfego, diferente dos produtos proprietários e de alto custo citados anteriormente [Behrisch et al., 2011].

Segundo DNIT [2011], o PNCT - Plano Nacional de Contagem de Trânsito vem armazenando nos últimos anos importantes dados de volume de tráfego e já possui uma significativa série histórica desses dados. A informação coletada é de grande importância pois seus resultados são subsídios básicos para estudos econômicos, projetos rodoviários e planejamento de tráfego, além do papel essencial no estabelecimento de critérios para o cumprimento das seguintes finalidades:

- Planejar o sistema rodoviário;
- Programar necessidades e prioridades de melhorias no sistema rodoviário;
- Estabelecer as tendências de tráfego no futuro;
- Avaliar o fluxo existente de tráfego em relação ao sistema rodoviário atual;
- Justificar e planejar o policiamento;
- Estudos de localização de postos de pesagem, socorro médico emergencial, etc.;
- Projetar pavimento, obras de arte, seção transversal e outros elementos de rodovia [DNIT, 2011].

Existem métodos manuais e automáticos para realização das contagens volumétrica e classificatória, que normalmente são de alto custo financeiro. De acordo com a forma de instalação dos equipamentos de contagem, tais métodos podem ser invasivos ou não-invasivos. Os métodos invasivos necessitam de instalações junto ou sob a camada asfáltica. Já os métodos não-invasivos normalmente utilizam câmeras, sensores ou instalações sobre o solo [Goldner, 2009].

Este trabalho tem como propósito desenvolver um método computacional de simples implementação, configuração, instalação e operação, capaz de realizar a contagem volumétrica de veículos de forma não-invasiva e que possa ser utilizado como uma alternativa de baixo custo financeiro, se comparado aos sistemas de contagem volumétrica convencionais.

¹O termo código aberto, ou *open source* em inglês, foi criado pela OSI e refere-se a *software* livre. Mais informações em <http://opensource.org>

1.1 Objetivos

O objetivo principal do trabalho é desenvolver um sistema de visão computacional que auxilie na análise das condições do tráfego urbano. São os objetivos específicos deste estudo:

- Entender a utilização de um sistema de visão para problemas de transporte;
- Realizar a contagem volumétrica dos veículos através de um método não-invasivo de simples implementação;
- Analisar as condições do tráfego urbano através de imagens coletadas por uma câmera digital;
- Através da análise dos resultados, determinar a qualidade do método e identificar pontos de acerto e erro que podem ser trabalhados.

1.2 Organização do Texto

A monografia está dividida em seis capítulos:

- Capítulo 1 é introdutório e são apresentados alguns pontos que motivam e justificam o desenvolvimento do projeto, além de uma listagem dos objetivos específicos que definem o escopo do mesmo.
- Capítulo 2 apresenta as tecnologias de contagem volumétrica e classificatória existentes. É também apresentado uma pesquisa bibliográfica sobre sistemas de visão computacional inseridos no contexto da Engenharia de Transporte.
- Capítulo 3 introduz a teoria de visão computacional, contextualizando o leitor. Trata-se de uma apresentação da tecnologia com foco em conceitos e técnicas de processamento de imagens.
- Capítulo 4 apresenta a metodologia para contagem volumétrica automática de veículos. As técnicas utilizadas, a sequência de operações, as imagens de processamento, as condições de captura, a estrutura do software e os métodos para análise de resultados são mostrados e explicados.
- Capítulo 5 apresenta os resultados e a validação das contagens; uma análise crítica do método proposto mostrando como as características da cena influenciam no resultado.

- Capítulo 6 são feitas as conclusões e as considerações finais do trabalho, bem como recomendações para trabalhos futuros.

Capítulo 2

Contagem de Veículos e Monitoramento do Tráfego

De acordo com Dnit [2011], Departamento Nacional de Infra-Estrutura de Transportes, a contagem volumétrica consiste em quantificar o volume de veículos que trafegam por um determinado trecho da rodovia, durante um intervalo de tempo definido. A contagem de veículos é importante para o cumprimento de diversas finalidades, dentre elas: planejamento do sistema rodoviário, estabelecimento de tendências de tráfego futuro, determinação do volume de viagens de forma a proporcionar justificativa econômica aos investimentos programados, avaliação do fluxo existente de tráfego em relação ao sistema rodoviário atual, planejamento e justificativa do policiamento, realização de análise estatística de acidentes, estudos de localização de postos de pesagem, socorro médico emergencial, entre outros.

O volume de veículos, obtido através da contagem, pode ser classificado como [Goldner, 2009]:

- **Volume de tráfego:** quantidade de veículos em tráfego na seção de uma via, em um período de tempo definido;
- **AADT ou VMDA:** volume médio diário de veículos durante o ano, ou seja, a somatória anual, dividido por 365;
- **ADT ou VMD:** volume diário do tráfego ou volume médio diário. Também é usado para intervalos de tempo diferentes, representando o volume total durante dado período, dividido pela quantidade de dias do período. Assim tem-se:
 - **VMDm:** volume médio diário mensal. Número total de veículos em um mês, dividido pelo número de dias do mês;

- **VMDs:** volume médio diário semanal. Número total de veículos em uma semana, dividido por 7. É sempre acompanhado pelo nome do mês a que se refere;
- **VMDd:** Volume médio diário em um dia de semana. Deve ser sempre acompanhado pela indicação do dia de semana e do mês correspondente;
- **Composição do tráfego:** porcentagem dos diferentes tipos de veículos que compõem o tráfego, por exemplo, automóveis, caminhões, ônibus e motos;
- **Volume abreviado:** volume do fluxo de um período menor do que 1 hora;
- **Variações do volume de tráfego:** mudanças no volume em um determinado período, divididas em: variações sazonais ou mensais ao longo do ano; variações diárias ao longo da semana; variações horárias ao longo do dia; e variações dentro da hora.

Goldner [2009] define dois métodos de contagem: **manual** e **mecânica**. A contagem manual utiliza recursos humanos, ou seja, pesquisadores observam o fluxo de veículos e manualmente fazem suas anotações. Esse procedimento é vantajoso pela precisão e variedade nas informações de tráfego obtidas, como tipo e tamanho dos veículos, além de proporcionar flexibilidade, simplicidade e rapidez de execução. No entanto, é um método com limitações de cobertura e de alto custo. A contagem mecânica utiliza detectores de tráfego de instalações permanente ou móvel. Possui algumas vantagens, dentre elas: baixo custo operacional, grande amplitude de tempo de cobertura e boa precisão. Por outro lado não fornece muitas informações e demanda investimento inicial alto.

A classificação feita por Goldner [2009] peca ao usar o termo contagem mecânica para caracterizar métodos não manuais. Grande parte dos equipamentos de contagem volumétrica são dispositivos eletrônicos, por exemplo sensores micro-ondas, ultrassônicos e piezoelétricos, descaracterizando o uso dessa nomenclatura. Até mesmo o tubo pneumático, que pode ser considerado um dispositivo mecânico, utiliza sinais elétricos para transmissão dos dados. Por esse motivo, será adotado o termo **contagem automática** para descrever os métodos de contagem que não dependem diretamente da participação humana durante o processo de obtenção dos dados.

Por sua vez, os métodos automáticos de contagem podem ser classificados como **invasivos** e **não-invasivos**. Os métodos invasivos necessitam de instalações junto ou sob a camada asfáltica. Já os métodos não-invasivos não modificam a estrutura da via, com instalações acima ou às margens da faixa de tráfego [Goldner, 2009].

Nas seções seguintes são apresentados alguns equipamentos automáticos de contagem volumétrica, descritos em Goldner [2009], Almeida [2010] e DNIT [2011], bem como algumas aplicações de visão computacional inseridas na área de Engenharia de Transporte.

2.1 Tecnologias detectoras de veículos

Atualmente existem diversas tecnologias de sensores capazes de efetuar a detecção de veículos [of Transportation et al., 1997]. A Tabela 2.1 lista grande parte delas, apresentando vantagens e desvantagens para cada uma [Magalhães, 2008]. Nas subseções a seguir alguns sensores que utilizam essas tecnologias são detalhados.

Tabela 2.1: Vantagens e desvantagens de tecnologias detectoras de veículos [Magalhães, 2008].

Tecnologia	Vantagens	Desvantagens
Ultra-sônico	<ul style="list-style-type: none"> • Tamanho compacto, fácil instalação 	<ul style="list-style-type: none"> • Pode ser sensível à temperatura e turbulência do ar
Microondas Doppler	<ul style="list-style-type: none"> • Bom para condições meteorológicas adversas • Mede velocidade dos veículos diretamente 	<ul style="list-style-type: none"> • Não consegue detectar veículos parados ou com velocidade inferior a 6 km/h
Microondas - presença real	<ul style="list-style-type: none"> • Bom para condições meteorológicas adversas • Detecta veículos parados • Opera com visada lateral da via 	<ul style="list-style-type: none"> • Requer uma antena de feixe estreito para definir área de interesse a uma única faixa no modo de visualização no sentido da via
Infravermelho passivo (só mantece recebe sinal)	<ul style="list-style-type: none"> • Distância de alcance sob neblina maior do que sensores de imagem na área da luz visível 	<ul style="list-style-type: none"> • Degradação potencial por chuva ou neve intensa • Imprecisão na medida de velocidade

Continua na próxima página

Tabela 2.1

Tecnologia	Vantagens	Desvantagens
Infravermelho ativo (transmite e recebe)	<ul style="list-style-type: none"> • Distância de alcance sob neblina maior do que sensores de imagem na área da luz visível • Mede velocidade diretamente 	<ul style="list-style-type: none"> • Degradação potencial por substâncias que obscurecem a atmosfera e por condições meteorológicas adversas
Processadores de imagem e vídeo no espectro de luz visível	<ul style="list-style-type: none"> • Provê dados para gerenciamento de tráfego e imagem para gerenciamento de incidentes • Uma única câmera e processador podem cuidar de muitas faixas • Riqueza de informações sobre tráfego torna-se disponível 	<ul style="list-style-type: none"> • Degradação potencial por condições meteorológicas adversas • Veículos grandes podem ocluir veículos menores • Sombras, reflexos do chão molhado e transições dia/noite podem resultar em detecções falsas ou perdidas
Laser pulsado	<ul style="list-style-type: none"> • Alta direcionalidade • Grande alcance • Portátil, normalmente instalado na lateral da via 	<ul style="list-style-type: none"> • Custo elevado • Degradação potencial por condições meteorológicas adversas • Obstrução por outros veículos
Laser de varredura	<ul style="list-style-type: none"> • Excelente classificação dos veículos 	<ul style="list-style-type: none"> • Custo elevado • Necessidade de ser colocado sobre a via, verticalmente
Fibra ótica	<ul style="list-style-type: none"> • Imunidade a adversidades meteorológicas • Imunidade a distúrbios eletromagnéticos 	<ul style="list-style-type: none"> • Custo elevado • Necesita interrupção da via e cortes no pavimento
Acústico passivo	<ul style="list-style-type: none"> • Instalação lateral à via • Auxilia na classificação 	<ul style="list-style-type: none"> • Adequado apenas à detecção de passagem de veículos

Continua na próxima página

Tabela 2.1

Tecnologia	Vantagens	Desvantagens
Cabos piezo-elétricos	<ul style="list-style-type: none"> • Sensível ao peso do veículo, permite peso em movimento • Imune a adversidades meteorológicas 	<ul style="list-style-type: none"> • É preciso interromper a via e fazer cortes no pavimento • Não permite detecção de presença (a menos que o veículo pare exatamente sobre o sensor)
Pneumático	<ul style="list-style-type: none"> • Baixo custo • Usado em contagem de veículos para gerenciamento de tráfego 	<ul style="list-style-type: none"> • Componentes mecânicos envolvidos: manutenção freqüente • Instalação normalmente provisória
Microloop (3M)	<ul style="list-style-type: none"> • Cortes no pavimento são mínimos • Fácil instalação 	<ul style="list-style-type: none"> • Não permite instalação em latitudes próximas à linha do equador, pois detecta variações no campo magnético da terra
Laços indutivos (ILD - <i>Inductive Loop Detectors</i>)	<ul style="list-style-type: none"> • Baixo custo • Imune a adversidades meteorológicas • Robusto • Permite classificação pela assinatura magnética 	<ul style="list-style-type: none"> • É preciso interromper a via e fazer cortes no pavimento • Instalação permanente

2.1.1 Detector de laço indutivo

São os sensores mais utilizados para a coleta de dados de tráfego. É composto basicamente por um circuito elétrico oscilador, uma unidade eletrônica de processamento, cabos de transmissão e o laço indutivo propriamente dito, que é formado por uma ou mais voltas de um cabo isolado instalado sob o pavimento da via (Figura 2.1), caracterizando-o como equipamento invasivo.

O laço funciona como indutor de um circuito oscilador de frequência fixa, normalmente entre 10 KHz e 200 KHz. Quando objetos de metal transitam nas proximidades do equipamento instalado no solo, acontece uma interação entre os campos magnéticos induzidos no laço e no objeto metálico, que normalmente é um veículo, provocando variações na indutância do laço e, portanto, variações na frequência de oscilação do circuito ressonante. A presença de um veículo na região do laço provoca um aumento

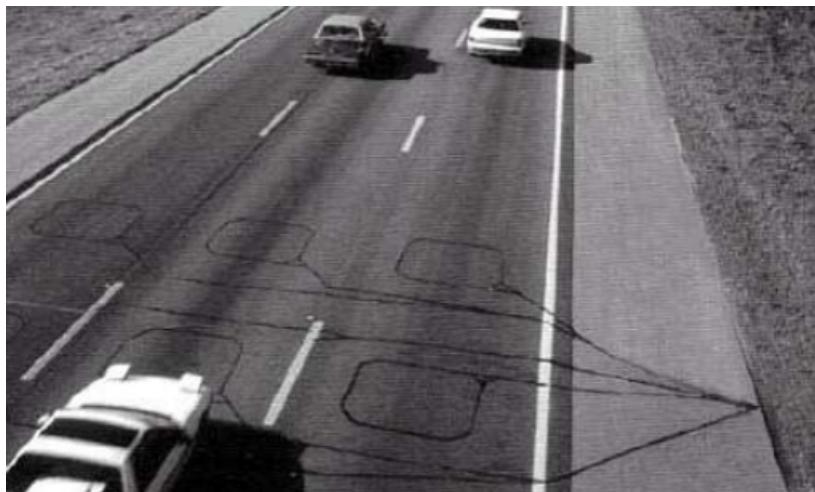


Figura 2.1: Seis laços indutivos instalado em uma rodovia, dois por faixa [Goldner, 2009].

na frequência de oscilação do circuito, comportamento que é interpretado pela unidade de processamento capaz de realizar a contagem.

Cada veículo que passa pelo laço indutivo possui um perfil magnético característico que depende de vários fatores, dentre eles, seu tamanho, formato, condutividade e orientação em relação ao laço. Isso possibilita, além da contagem, realizar uma classificação veicular através de métodos de clusterização por redes neurais, como descrito em [Almeida, 2010].

A vantagem de se usar laços indutivos para monitoramento é o bom funcionamento na captura de dados básicos do tráfego, como volume e velocidade, além de apresentar baixo custo se comparado à equipamentos não-invasivos. Por outro lado, o processo de instalação do equipamento junto ao solo gera a paralisação no tráfego local e muitas vezes causa transtornos no trânsito.

2.1.2 Tubo pneumático

Os sensores de contagem volumétrica por tubo pneumático funcionam a partir da pressão exercida pelos pneus de um veículo assim que passam sobre o tubo de borracha. A massa de ar é então deslocada ao longo do tubo até o ponto de conversão, onde um interruptor converte o sinal pneumático em sinal elétrico, para que esse possa ser transmitido a um *software* de análise ou a um contador.

O tubo é instalado sobre a via, perpendicular à direção do fluxo do tráfego, caracterizando-se como um equipamento invasivo. É normalmente utilizado para contagem de tráfego em períodos curtos, classificação dos veículos por número de eixos e

espaçamento, medição de velocidade e pesquisas acadêmicas.

Apesar de ser a primeira tecnologia de detecção de tráfego, inventada em 1920, é ainda muito utilizada devido ao baixo custo e a simples instalação e operação. No entanto, o equipamento é impreciso na contagem de caminhões ou ônibus muito largos, além de apresentar baixa durabilidade e sensibilidade à temperatura.

2.1.3 Sensor magnético

O princípio de funcionamentos dos sensores magnéticos é baseado em variações das linhas de fluxo do campo magnético terrestre. Tal efeito é conhecido como anomalia magnética, causada pela aproximação de objetos metálicos (veículos) da área de cobertura do sensor. Por indução eletromagnética, variações de tensão são geradas e posteriormente amplificadas, digitalizadas e finalmente transmitidas ao controlador eletrônico que realiza a contagem.

São utilizados para medir volume, direção, presença e velocidade dos veículos. Podem ser divididos em dois tipos: os magnetômetros de indução, que se baseiam em mudanças nas linhas de fluxo do campo magnético devido ao movimento, não detectam veículos parados na via; e os magnetômetros de eixo duplo, que detectam mudanças nos componentes horizontais e verticais do campo magnético terrestre de acordo com a densidade do metal do veículo, e são capazes de identificar objetos em movimento ou parados.

A utilização destes sensores é vantajosa devido a pouca sensibilidade à temperatura ambiente e ao tráfego intenso. Suas desvantagens são a pequena zona de detecção, a dificuldade em considerar veículos parados e a sua característica de instalação invasiva, fazendo-se necessárias paralizações do tráfego e interferências na camada asfáltica.

2.1.4 Sensor piezoeletrico

Um material piezoeletrico é capaz de converter energia cinética em energia elétrica. Esses materiais geram uma tensão quando submetidos a choques mecânicos ou vibrações. Portanto, quando um veículo passa sobre um detector, o sensor gera uma tensão proporcional ao peso do veículo. É construído a partir de um cabo piezoeletrico do tipo coaxial, com um núcleo de metal, seguido pelo material piezoeletrico e uma camada externa de metal, como ilustrado na Figura 2.2.

Devido a sua grande precisão, esse tipo de sensor é capaz de medir volume do tráfego, velocidade, peso e ainda classificar os veículos baseado no espaçamento e contagem de eixos. Sua instalação é feita junto à via, caracterizando-o como invasivo, e não

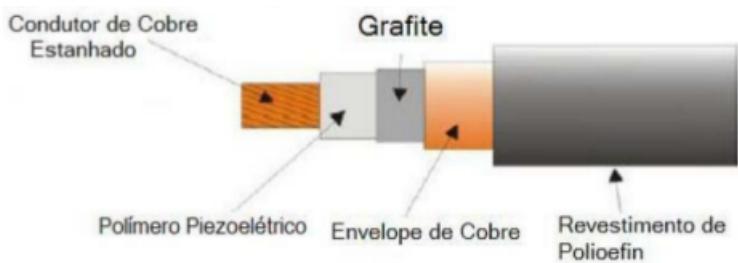


Figura 2.2: Representação esquemática de um cabo piezoelétrico [Goldner, 2009].

é recomendado em solos irregulares. Apresenta alto custo se comparado a outros equipamentos invasivos, mas em contrapartida são mais precisos e adquirem informações mais completas.

2.1.5 Câmera de vídeo

As câmeras de vídeo são utilizadas no monitoramento do tráfego, na maioria das vezes, como uma simples ferramenta de vigilância urbana, com capacidades de transmissão e gravação de imagens. Nesse cenário, as imagens devem ser interpretadas por um operador humano, que tem a função de extrair dados de tráfego. Com o avanço das áreas de processamento digital de imagens e visão computacional, surgiram diversos sistemas de gerenciamento do tráfego capazes de extrair automaticamente informações de tráfego importantes, como velocidade, volume, presença, ocupação, densidade, movimentos de conversão, mudança de faixa, aceleração, classificação de veículos e outros [Martinsky, 2007; Feitosa, 2012].

Um sistema inteligente de processamento de imagens de vídeo consiste em uma ou mais câmeras, um *hardware*¹ para digitalização e processamento e um *software* para interpretação e conversão de imagens em dados de tráfego, numéricos ou simbólicos. Segundo Feitosa [2012], esse tipo de sistema pode, com apenas uma câmera, substituir vários equipamentos invasivos, como laços indutivos, e proporcionar detecção de veículos em múltiplas faixas com menores custos de manutenção.

Dentre as vantagens de uso de câmeras está a instalação, que é totalmente não-invasiva e não gera qualquer tipo de transtorno ao trânsito. Uma vez capturadas, as imagens do tráfego podem ser analisadas posteriormente por profissionais capazes de, por exemplo, realizar a contagem volumétrica manual ou periciar um acidente. Mudanças naturais na cena, como chuvas, tempo nublado, o dia e a noite, podem interferir na visibilidade, dificultando tanto a interpretação humana quanto o processamento di-

¹Parte física de computadores ou quaisquer equipamentos eletrônicos.

gital por visão computacional. Deslocamentos indesejados da câmera e problemas de oclusão também são comuns.

2.1.6 Sensores micro-ondas

Esse tipo de equipamento transmite radiação de micro-ondas de baixa energia na direção de uma área do pavimento utilizando uma antena localizada na parte superior do radar. Quando um veículo atravessa um feixe, uma parte dessa energia é refletida de volta para o detector, também localizado na parte superior do radar, e dados como velocidade e presença são calculados por um controlador. A Figura 2.3 ilustra a configuração mais comum de um sistema de monitoramento baseado em sensores micro-ondas.

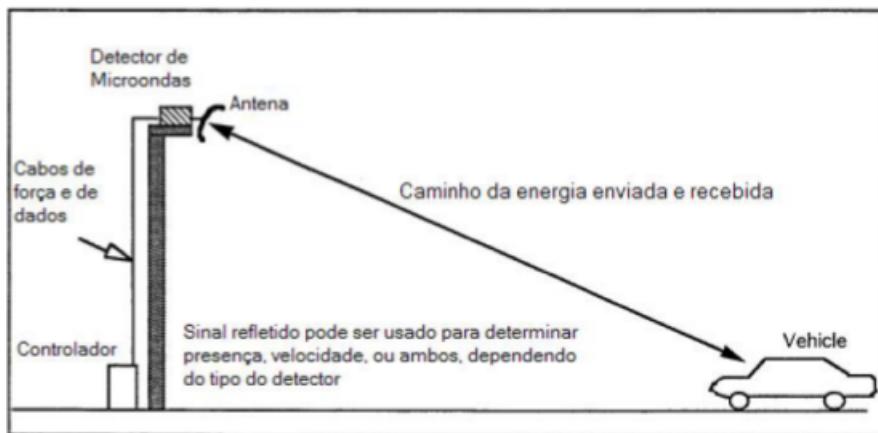


Figura 2.3: Esquema ilustrativo da arquitetura de um sistema de monitoramento do tráfego baseado em sensores micro-ondas [Goldner, 2009].

Esses sensores podem ser de dois tipos:

- **Doppler:** mede a presença de um veículo em função do movimento relativo entre uma fonte sonora e seu receptor e que, por efeito *Doppler*, varia a frequência recebida na volta. Como é um método baseado em movimento, não considera veículos parados ou em regime de “anda-pára”.
- **Radar:** utiliza um sinal de frequência ou fase modulada para calcular o atraso de tempo da onda refletida, obtendo a distância do veículo. Nesse método é possível detectar a presença de veículos parados, além de medir velocidade, monitorar filas e ocupação.

Os sensores micro-ondas são não-invasivos e não interferem no tráfego de veículos no processo de instalação. Geralmente não apresentam sensibilidade em más condições climáticas e podem ser construídos para suportar múltiplas pistas. No entanto, um dos tipos não é capaz de detectar veículos parados.

2.1.7 Sensores ultrassônicos

Esses detectores transmitem ondas de pressão de energia sonora acima da frequência audível humana, que é de 20 KHz. Estas ondas sonoras refletem no pavimento ou no veículo, são captadas por um receptor e processadas para fornecer informações de passagem e presença de veículos. A montagem não-invasiva do equipamento pode ser feita acima ou ao lado da via, como ilustra a Figura 2.4.

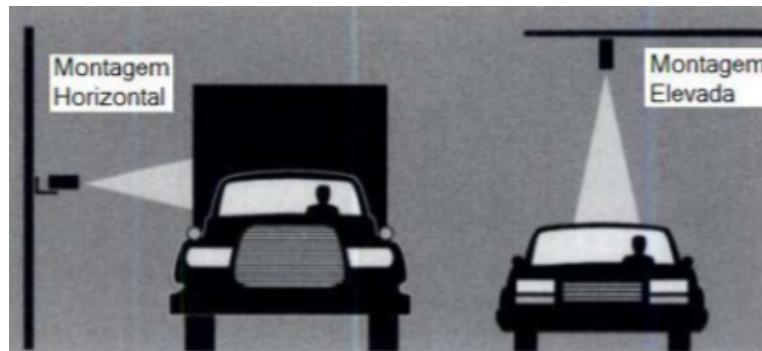


Figura 2.4: Tipos de montagem para um sensor ultrassônico [Goldner, 2009].

Existem dois tipos de sensores ultrassônicos:

- **Pulso ultrassônico:** são emitidos pulsos de energia com largura e período conhecidos. Se o tempo do pulso refletido for menor que o valor conhecido, a presença do veículo é acusada, fornecendo informações de altura, largura, ocupação, volume e classificação.
- **Onda ultrassônica contínua:** usam o princípio de *Doppler*, que leva em conta a variação da frequência causada pela velocidade relativa entre onda emitida e refletida, para determinar a presença, volume e velocidade veicular.

O uso de sensores ultrassônicos é vantajoso pois a instalação é não-invasiva e pode monitorar mais de uma pista ao mesmo tempo. Como desvantagem, as mudanças de temperatura e ventanias podem afetar o seu desempenho.

2.2 Aplicações de Visão Computacional

Com o desenvolvimento de pesquisas na área de processamento de imagens digitais nos últimos tempos, impulsionado pelas facilidades de implementação de aplicações de visão computacional utilizando a biblioteca OpenCV [Bradski, 2000], cada vez mais trabalhos de monitoramento do tráfego que utilizam visão computacional são publicados. A seguir, uma breve revisão bibliográfica de projetos de Engenharia de Transporte que utilizam conceitos de visão computacional.

2.2.1 Contagem volumétrica utilizando dispositivos móveis

Feitosa [2012] propõe em sua Tese de Mestrado em Ciência da Computação pela UFG - Universidade Federal de Goiás, um método para contagem volumétrica de veículos baseado em visão computacional. A pesquisa é focada em otimização de algoritmos de processamento de imagens para execução em dispositivos móveis.

Alguns requisitos são estabelecidos nesse trabalho: as imagens são capturadas da lateral de uma via, com uma câmera fixada em uma baixa altura, identificando e contando veículos que trafegam em ambas as direções; as imagens utilizadas são de baixa resolução; e o processo de contagem é realizado em tempo real.

Apesar de o objetivo do projeto ser a contagem volumétrica via dispositivos móveis, o estudo inicial das técnicas de processamento de imagem e visão computacional é feito em um *laptop*². A metodologia adotada pode ser dividida em seis etapas principais. São elas:

1. **Entrada de dados:** os *frames*³ do vídeo são disponilizados em tempo real ou por um arquivo em disco, que foi gravado anteriormente.
2. **Pré-processamento de imagem:** conversão para escala de cinza, caso ainda não esteja nesse formato; equalização do histograma; e suavização da imagem por filtragem linear.
3. **Reconhecimento da imagem referência de fundo:** obtenção de um modelo adaptativo da imagem de fundo da cena, utilizando um método de rápido processamento.
4. **Definição da área de movimento:** determinação da região da cena com maior movimentação que, provavelmente, é a área de tráfego de veículos. Essa região

²Computador portátil, que no Brasil também é conhecido como *notebook*

³Cada um dos quadros de um vídeo.

de interesse também elimina movimentações indesejadas, como por exemplo, o movimento da vegetação.

5. **Segmentação de objetos:** subtrai a imagem referência de fundo do *frame* corrente, segmentando os objetos de interesse. Também descarta objetos sem interesse.
6. **Acompanhamento de objetos segmentados:** rastreamento dos objetos segmentados ao longo dos quadros, evitando que o mesmo seja contado mais de uma vez.

O método proposto foi implementado em forma de aplicativo na linguagem Java, utilizando a biblioteca de processamento de imagens JavaCV, que utiliza a maior parte das funções da OpenCV [Bradski, 2000]. Como o trabalho de Feitosa [2012] tem por objetivo a execução em dispositivos móveis, o desempenho do algoritmo é um fator determinante. Por isso, foi necessário que alguns algoritmos de processamento de imagens, já existentes na OpenCV, fossem reimplementados com foco em velocidade de execução, o que elevou muito a complexidade do código. Na etapa de reconhecimento da imagem referência de fundo, por exemplo, seria muito mais simples, do ponto de vista de desenvolvimento, utilizar as funções de subtração de *background* já implementadas na OpenCV. No entanto, por ser um método mais completo e complexo, acarretaria em menor desempenho de execução se comparado à implementação de Feitosa [2012].

Os testes foram realizados a partir de 15 amostras de vídeos, sendo 14 com resolução 640×480 e apenas 1 com resolução 320×240 , e duração média de 09:30 minutos. Para avaliar o desempenho do método, num primeiro momento foi feita uma contagem volumétrica manual dos veículos em cada amostra, para que assim o percentual de acerto do método automático, executado no *laptop*, fosse calculado. Além disso, o tempo de execução de cada amostra foi medido.

Em seguida, o aplicativo desenvolvido em Java foi adaptado para ser executado por um dispositivo móvel com sistema operacional Android. Os mesmos testes foram realizados para as 15 amostras e o percentual de acerto permaneceu o mesmo, resultado esperado tendo em vista que os algoritmos de processamento de imagem e visão computacional não mudaram. Nesse novo cenário, o tempo de execução também foi medido. A Tabela 2.2 lista as medições de cinco amostras.

Os percentuais de acerto obtido nas contagens foi satisfatórios e aplicativo desenvolvido para executar em um *laptop* foi adaptado e portado para um dispositivo móvel Android com sucesso. No entanto, o tempo de execução do método no dispositivo móvel não foi satisfatório, inviabilizando que essa aplicação funcione em tempo real.

Tabela 2.2: Comparação entre resultados de tempo de execução do método pelo *laptop* e celular [Feitosa, 2012].

Amostra	% acerto	Duração	Tempo celular	Tempo <i>laptop</i>
1	90,58	00:15:08	01:00:56	00:03:54
2	92,45	00:05:15	00:19:55	00:01:36
3	91,30	00:02:44	00:11:55	00:00:27
4	98,15	00:10:18	00:48:18	00:03:39
5	97,71	00:13:07	01:09:27	00:03:48

Capítulo 3

Visão Computacional

Segundo Shapiro & Stockman [2001], visão computacional é um campo da Ciência da Computação que inclui métodos para a aquisição, processamento e análise de imagens. Morris [2004] define processamento de imagem e visão computacional separadamente. Para Morris, processamento de imagem é a manipulação de uma imagem digital, gerando como resultado uma nova representação da mesma imagem, ao passo que visão computacional é a extração de informações numéricas ou simbólicas de imagens. Em linhas gerais, visão computacional é a tecnologia que transforma imagens do mundo real em uma representação que computadores são capazes de interpretar e processar e, com isso, produzir informações úteis para aplicações de Engenharia. Contempla uma base teórica e tecnológica para a construção de sistemas artificiais que obtém informações de imagens ou quaisquer dados multidimensionais. Mas como os computadores podem entender o mundo visual dos humanos? Quais vantagens e quais tipos de aplicações podem existir a partir desse conceito?

Segundo Szeliski [2010], o ser humano é perfeitamente capaz de perceber a estrutura tridimensional do mundo ao seu redor. Ao visualizar a Figura 3.1, percebe-se que a visão humana interpreta variações de transparência e sombra, além de diferenciar o objeto do *background*¹ com facilidade. Isso acontece porque o cérebro humano divide o sinal de visão em muitos canais, gerando um fluxo de diferentes tipos de informação. Ele é capaz de identificar as partes importantes de uma imagem a serem examinadas e ao mesmo tempo suprimir a atenção para regiões menos importantes [Szeliski, 2010]. Além disso, o cérebro possui um sistema de realimentação poderoso que implementa um controle em malha fechada, composto por sensores (visão, audição, olfato, tato e paladar) e atuadores (íris para controlar a entrada de luminosidade nos olhos) [Bradski & Kaehler, 2008].

¹Segundo plano ou plano de fundo



Figura 3.1: O ser humano é capaz de determinar a forma e a transparência de cada pétala de uma flor [Szeliski, 2010].

Diante da facilidade com que o ser humano enxerga o mundo ao seu redor é intuitivo pensar que o processamento de imagens por visão computacional é simples. Mas isso não é verdade. Em um sistema de visão, o computador recebe, na maioria das vezes, apenas uma matriz de números inteiros, em que cada posição é denominada pixel, como mostrado na Figura 3.2. Todos aqueles padrões de interpretação de informação presentes no cérebro não existem aqui. Além disso, deve-se considerar os ruídos existentes num sistema de visão, que diminuem ainda mais a quantidade de informação dos dados. Esse tipo de problema pode ser causado devido a variações no ambiente (luminosidade, clima, reflexos, movimentações), imperfeições na captura de imagem (lente, configuração mecânica), ruídos elétricos no sensor óptico da câmera e compressão das imagens após a captura [Bradski & Kaehler, 2008].

Mesmo com todos esses desafios, por incrível que pareça, é possível desenvolver sistemas baseados em visão computacional bastante robustos e com alto grau de confiabilidade. Isso começa a se tornar possível quando as imagens capturadas estão inseridas no contexto de uma determinada aplicação. Por exemplo: se um sistema de visão computacional tem por objetivo rastrear carros, não faz nenhum sentido buscar os veículos em áreas que não sejam as ruas, avenidas e rodovias. Caso um objeto seja encontrado em uma área verde ou no azul do céu, a probabilidade de que esse

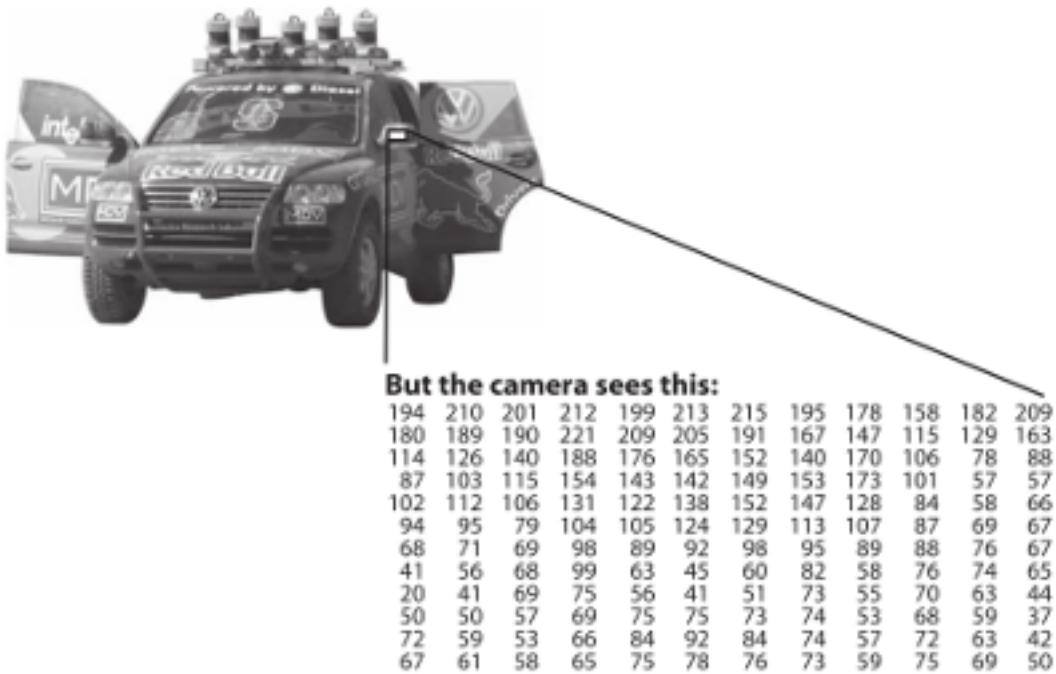


Figura 3.2: Para um computador, o retrovisor de um carro é apenas uma matriz composta por pixeis [Bradski & Kaehler, 2008].

objeto seja um carro é muito baixa. É uma análise óbvia mas de suma importância no processamento de imagens.

O uso de métodos estatísticos em visão computacional também são primordiais, convergindo para o problema de ruído discutido anteriormente. Considerar a média dos pixeis no tempo é uma abordagem estatística, que pode vir a ser uma solução para problemas que envolvem imagens ruidosas. Outra técnica bastante comum é a construção de modelos das câmeras, capazes de caracterizá-las matematicamente através de seus parâmetros intrínsecos e extrínsecos. Os parâmetros internos da câmera como distância focal, distorções de lente e tamanho do pixel correspondem aos parâmetros intrínsecos. Os parâmetros extrínsecos estão relacionados à orientação e posição da câmera em relação a um sistema de referência no mundo. Com esses parâmetros fica simples corrigir imperfeições nas imagens, que podem ocorrer devido a lente ou alguma configuração mecânica [Bradski & Kaehler, 2008].

Essas e mais uma série de técnicas, descritas nas seções seguintes, são objetos de estudo no campo de visão computacional. Elas em conjunto são combinadas e organizadas de maneira a transformar quaisquer dados multidimensionais em informações de alto nível, como por exemplo ausência e presença de um componente, dimensão e cor de objetos. Como regra geral: quanto mais restrito é o escopo de uma aplicação

de visão computacional, mais o problema pode ser simplificado e mais confiável será a solução final [Bradski & Kaehler, 2008].

A seguir é realizado uma breve revisão bibliográfica sobre conceitos, técnicas e métodos de processamento de imagens digitais aplicados ao campo da visão computacional.

3.1 Imagem digital

De acordo com Gonzalez & Woods [2000], uma imagem pode ser definida como uma função bidimensional de intensidade de luz $f(x, y)$, em que x e y são coordenadas espaciais de um plano, e a amplitude de f é proporcional ao brilho (ou níveis de cinza) da imagem naquele ponto. A Figura 3.3 ilustra a convenção dos eixos adotada nesse e na maioria dos trabalhos da área.

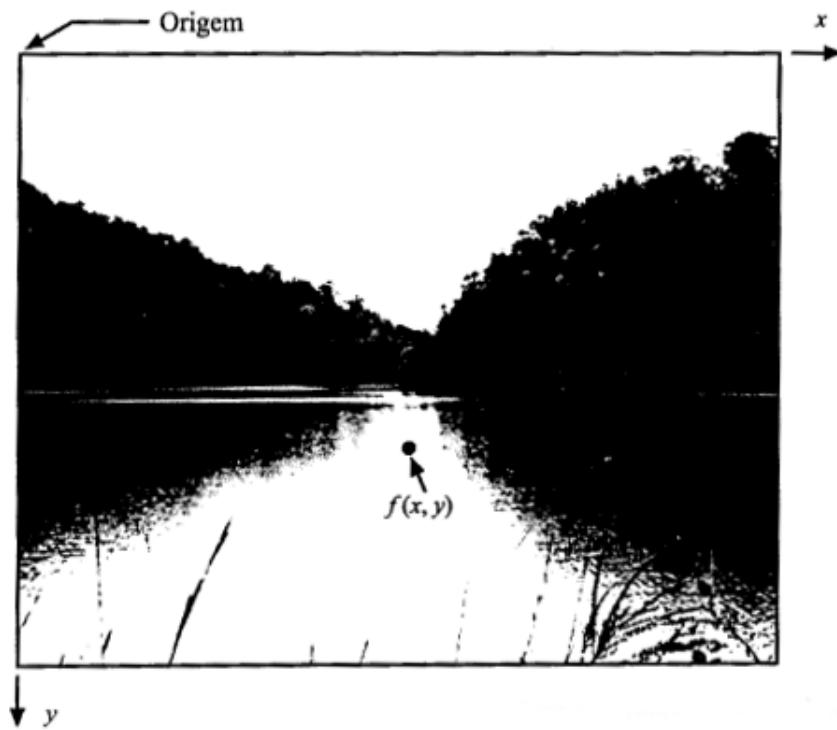


Figura 3.3: Convenção dos eixos para representação de imagens digitais [Gonzalez & Woods, 2000].

Uma imagem digital é a discretização de uma imagem $f(x, y)$, tanto em coordenadas espaciais quanto em intensidade de brilho. A OpenCV 2 [Bradski, 2000], uma biblioteca de visão computacional descrita na Seção 3.7, considera a imagem digital como sendo uma matriz cujos índices de linhas e colunas determinam um ponto na

imagem, e o valor correspondente do elemento da matriz identifica o nível de cinza naquele ponto. Os elementos dessa matriz digital são chamados de pixels, embora existam outros nomes como elementos da imagem, elementos da figura ou pels [Gonzalez & Woods, 2000].

3.2 Modelo RGB de cores

Segundo Gonzalez & Woods [2000], é comprovado que o olho humano possui entre 6 e 7 milhões de cones², que podem ser divididos em três principais categorias de sensoriamento, aproximadamente correspondentes às cores vermelho, verde e azul. Desse total de cones existentes, cerca de 65% são sensíveis à luz vermelha, 33% são sensíveis à luz verde e apenas 2% deles são sensíveis à luz azul, sendo os cones azuis os que possuem maior grau de sensibilidade à luz. Dessa forma, as cores são vistas pelo olho humano como combinações das chamadas cores primárias: vermelho (R, *red*), verde (G, *green*) e azul (B, *blue*), fato que inspirou estudos do modelo RGB de cores.

O modelo RGB é um sistema de coordenadas cartesianas, onde o sub-espaço das cores de interesse é representado pelo cubo da Figura 3.4a. Nesse modelo, as cores são pontos sobre ou dentro do cubo, definidas por vetores que estendem-se a partir da origem. A escala de cinza é definida pela diagonal principal desse cubo e assume valores do preto, localizado na origem, ao branco, no vértice mais distante [Gonzalez & Woods, 2000].

Em representações computacionais desse modelo os componentes R, G e B, denominados canais de cor, assumem valores inteiros de uma escala que vai de 0 a 255, possibilitando que cada valor seja armazenado por 1 *byte*³. Dessa forma, é possível gerar exatamente 16.777.216 (ou 256^3) combinações de cores diferentes. A Figura 3.4b ilustra o gradiente formado na superfície do cubo RGB quando os pontos assumem valores inteiros num intervalo de 0 a 255.

3.3 Escala de cinza

A escala de cinza é muito utilizada em processamento de imagens digitais por ter menor custo computacional. Em imagens RGB são necessários 3 *bytes* (um por canal) para definir a cor de um pixel, enquanto que em imagens *grayscale*⁴ a representação

²Células fotorreceptoras, localizadas na retina do olho, que são responsáveis pela visão das cores.

³Unidade de medida de armazenamento de memória em computação. Cada *byte* contém 8 *bits*, suficiente para armazenar 2^8 números binários.

⁴Escala de cinza.

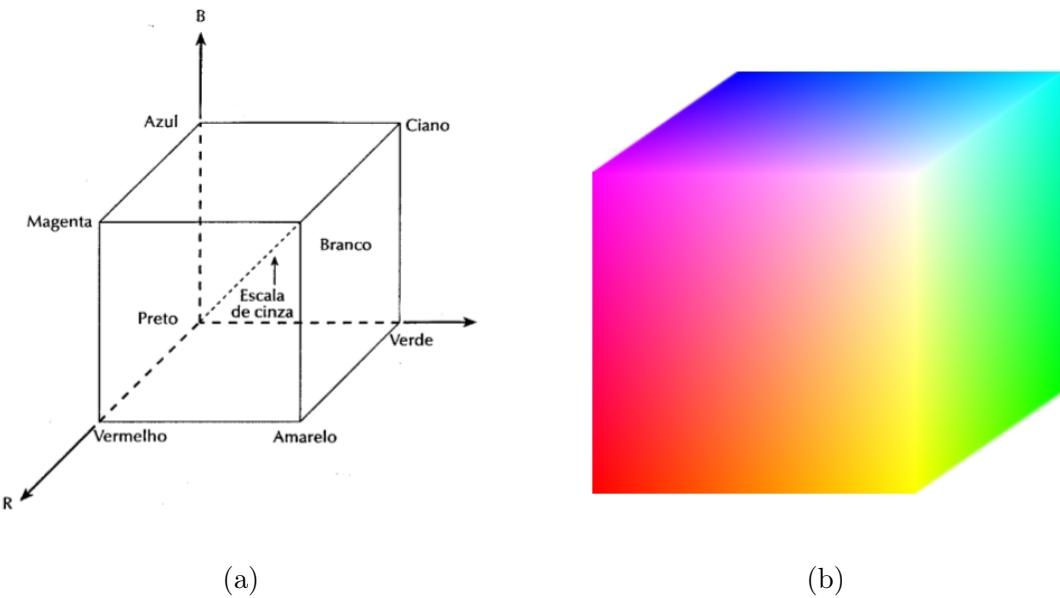


Figura 3.4: (a) Cubo de cores RGB. Os pontos ao longo da diagonal principal têm valores da escala de cinza que vão do preto ($0, 0, 0$) ao branco ($255, 255, 255$) [Gonzalez & Woods, 2000]; (b) Representação em cores do cubo RGB [Schouten, 2003].

da cor de um pixel utiliza apenas 1 *byte*, um terço de uma imagem RGB. De acordo com Gonzalez & Woods [2000], sistemas de visão computacional em que a cor não é um requisito, é essencial converter as imagens RGB para *grayscale* antes de qualquer operação, como mostrado na Figura 3.5, diminuindo assim o peso do processamento digital.

Para obter uma imagem *grayscale* a partir de uma imagem RGB deve-se considerar a média da intensidade de cada canal. Essa conversão é dada pela equação

$$C_{xy} = (k_R \times R_{xy}) + (k_G \times G_{xy}) + (k_B \times B_{xy}), \quad (3.1)$$

$$k_R, k_G, k_B > 0 \text{ e } k_R + k_G + k_B = 1,$$

onde x e y representam a posição do pixel no plano espacial da imagem segundo a convenção de eixos descrita na Seção 3.1; C_{xy} a intensidade resultante dos pixels na escala de cinza; R_{xy} , G_{xy} e B_{xy} a intensidade dos pixels dos canais vermelho, verde e azul, respectivamente; e k_R , k_G e k_B constantes relacionadas à sensibilidade de incidência de luz de cada canal de cor [Johnson, 2006].



Figura 3.5: Exemplo de conversão para *grayscale*. (a) Imagem original RGB; (b) Imagem em escala de cinza.

3.4 Histograma

Histograma é um gráfico matemático que representa a distribuição de frequências de uma massa de medições [Lancaster, 1973]. Normalmente é representado por retângulos na vertical, onde cada retângulo corresponde a um intervalo e sua altura é proporcional ao número de ocorrências naquele intervalo.

Em visão computacional, os histogramas são utilizados para representar o número de ocorrências de cada cor em uma imagem, como ilustra a Figura 3.6. Normalmente, é gerado um histograma para cada canal de cor em imagens coloridas, já que pode existir uma infinidade de cores possíveis. O histograma pode proporcionar um melhor entendimento da imagem pois facilita a visualização de parâmetros como contraste e luminosidade.

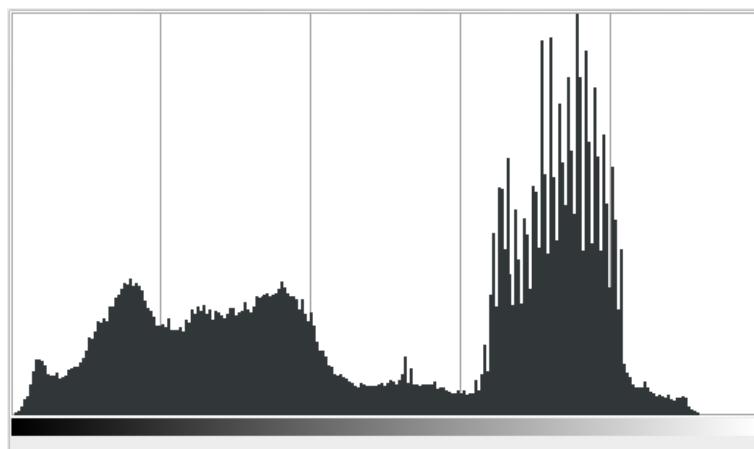


Figura 3.6: Histograma de uma imagem *grayscale* gerado pelo *software* GIMP [2013].

3.5 Filtragem linear

A filtragem linear é um tipo de operação local que considera pixels vizinhos de um determinado ponto para calcular seu novo valor, como ilustrado na Figura 3.7. Operadores locais, ou operadores de vizinhança⁵ como são mais conhecidos, são muito utilizados no processamento digital de imagens e podem ser usados como filtros espaciais lineares para gerar efeitos de borramento, acentuação de bordas, remoção de ruídos, entre outros.

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$f(x,y)$

$h(x,y)$

$g(x,y)$

Figura 3.7: Filtragem linear por convolução: a imagem da esquerda convolução com o filtro no centro produz a imagem da direita. Os pixels azuis indicam a vizinhança usada para gerar o pixel verde na saída [Szeliski, 2010].

De acordo com Szeliski [2010], filtros lineares são os operadores de vizinhança mais usados, em que o valor de saída $g(i, j)$ é determinado por uma soma ponderada de valores de pixels de entrada $f(i + k, j + l)$ (Figura 3.7),

$$g(i, j) = \sum_{k,l} f(i + k, j + l)h(k, l). \quad (3.2)$$

A ponderação é dada pelo *kernel* ou máscara $h(k, l)$, que contém os coeficientes do filtro, responsáveis pelas características de suavização, borramento ou remoção de ruídos inerentes a cada tipo de filtragem. A Equação (3.2) é conhecida como operador de **correlação**, que na forma compacta pode ser escrito como

$$g = f \otimes h. \quad (3.3)$$

⁵Tradução livre de *neighborhood operators*

Uma variação muito comum de (3.2) é dada por:

$$g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l) = \sum_{k,l} f(k, l)h(i - k, j - l), \quad (3.4)$$

onde o sinal dos deslocamentos em f foram invertidos. Essa é a chamada operação de **convolução**,

$$g = f * h. \quad (3.5)$$

As operações de correlação e convolução são equivalentes, a não ser pela fase de 180° que gera uma rotação no processo de filtragem por correlação. Elas são a base matemática para implementação de qualquer tipo de filtro linear, como por exemplo, **gaussiano**.



(a)

(b)

Figura 3.8: Percebe-se o comportamento passa-baixa do filtro atenuando componentes de alta frequência na imagem (bordas). (a) Imagem original; (b) Aplicação de filtragem linear gaussiana [Laganière, 2011].

Segundo OpenCV Development Team [2013], o filtro gaussiano é talvez o mais útil entre todos. É largamente utilizado no processamento de imagens digitais para remoção de ruídos e redução de detalhes, como ilustrado na Figura 3.8, onde pode-se verificar um suave borramento e a diminuição de detalhes nas bordas, características típicas de um filtro passa-baixa. A filtragem é feita através da convolução de cada pixel da imagem de entrada por um *kernel* cujos valores obedecem uma distribuição gaussiana bidimensional, representada por

$$h(k, l) = Ae^{-\frac{-(k-\mu_k)^2}{2\sigma_k^2}}e^{-\frac{-(l-\mu_l)^2}{2\sigma_l^2}}, \quad (3.6)$$

onde μ é a média e σ o desvio padrão. Nessa configuração, os pixels mais próximos de μ têm maior peso no resultado, portanto são mais significativos para o cálculo da intensidade de cada ponto. O peso dos pixels vizinhos decai com a distância e é influenciado também pelo desvio padrão σ .

3.6 Limiarização

Segundo Bradski & Kaehler [2008], a operação de limiarização, também conhecida por *thresholding*, é o método de segmentação em imagens digitais mais simples que existe. Essa técnica pode ser aplicada para separar regiões correspondentes a objetos que se deseja analisar, como ilustrado na Figura 3.9.



Figura 3.9: Segmentação de uma maçã utilizando limiarização binária [OpenCV Development Team, 2013].

A separação se dá pela diferença de intensidade entre os pixels do objeto e os pixels pertencentes ao plano de fundo. Para diferenciar os pixels de interesse é feito uma comparação do nível de cinza de cada pixel com um limiar (ou valor de *threshold*), que pode ser uma constante determinada de acordo com as necessidades do problema ou uma variável em algoritmos de limiarização adaptativos. Uma vez que os pixels de interesse foram segmentados, basta determinar um valor comum para identificá-los, sendo mais comum o branco 255 (Figura 3.9).

No caso geral, a operação de limiarização binária é dada por

$$dst(x, y) = \begin{cases} maxVal & \text{se } src(x, y) > thresh \\ 0 & \text{caso contrário.} \end{cases} \quad (3.7)$$

Portanto, se a intensidade do pixel $src(x, y)$ é maior que o limiar $thresh$, então a nova intensidade do pixel $dst(x, y)$ será $maxVal$; caso contrário, é atribuído a $dst(x, y)$ o valor 0 (preto) [Bradski & Kaehler, 2008].

3.7 Biblioteca OpenCV

OpenCV (*Open Source Computer Vision*) [Bradski, 2000] é uma biblioteca de visão computacional *open source*⁶ disponível em: <http://sourceforge.net/projects/opencvlibrary/>. O código é escrito em C e C++ e pode ser compilado e executado em ambientes Linux, Windows e Mac OS X. Existem também interfaces em Python, Ruby, Matlab e outras linguagens. Contém mais de 500 algoritmos otimizados para processamento de imagens e vídeos, cobrindo diversas áreas da visão computacional, como: inspeção industrial, imagens médicas, segurança, interface com o usuário, calibração de câmera, visão estéreo, robótica e *machine learning* [Bradski & Kaehler, 2008].

Segundo Laganière [2011], desde o lançamento da OpenCV em 1999, ela vem sendo largamente adotada como a principal ferramenta de desenvolvimento pela comunidade de pesquisadores e programadores em visão computacional. A biblioteca foi originalmente desenvolvida pela Intel [Intel, 2013], por um time liderado por Gary Bradski e com o propósito de avançar em pesquisas na área de visão. Depois de uma série de versões *Beta*, a versão 1.0 foi lançada em 2006. O segundo maior lançamento aconteceu em 2009 com a OpenCV 2, propondo importantes modificações em sua estrutura e especialmente a nova interface C++. Atualmente a OpenCV encontra-se na versão 2.4.

Bradski & Kaehler [2008] afirmam que a licença *open source* da OpenCV permite que aplicações comerciais possam ser construídas utilizando parte ou toda a biblioteca, sem a necessidade de que o código da aplicação seja aberto. Devido a essa política liberal de uso, gigantes como IBM, Microsoft, Intel, SONY, Siemens, Google, além dos centros de pesquisa Stanford, MIT, CMU, Cambridge e INRIA, utilizam a biblioteca em seus projetos e pesquisas. A OpenCV foi peça chave no sistema de visão de um robô desenvolvido em Stanford, conhecido como Stanley, que ganhou a corrida de carros autônomos *\$2M DARPA Grand Challenge*.

⁶O termo código aberto, ou *open source* em inglês, foi criado pela OSI e refere-se a *software* livre. Mais informações em <http://opensource.org>

3.8 Aplicações existentes

Invent Vision [2013] possui diversas aplicações concebidas utilizando os conhecimentos de visão computacional. O fato dessa tecnologia ser não intrusiva, ou seja, não altera em nada o meio em que está sendo utilizada, torna os sistemas de visão realizáveis em grande parte dos processos industriais, urbanos e ambientais.

As aplicações em análise dimensional se caracterizam por efetuarem medidas em objetos, sendo elas lineares e/ou angulares. A análise dimensional por imagem é vantajosa pois possibilita que as medidas sejam feitas à distância, quando não é possível ou desejável o contato direto com o objeto. Câmeras de alta resolução se aplicam nesses projetos, garantindo medições precisas e com repetibilidade. Muito comum na siderurgia, esse tipo de aplicação possibilita a medição em objetos que se encontram em altas temperaturas. As câmeras infravermelho tem sido utilizadas nesse tipo de aplicação.

O uso de reconhecimento de padrões permite a identificação de características de um produto comparando-o com um modelo predeterminado. Essas características a serem inspecionadas são escolhidas para identificar e diferenciar um tipo de modelo de outro. Assim é possível dizer se um objeto está conforme um padrão ou não.

Inspeção de nível é bastante comum na indústria de bebidas e na indústria farmacêutica para inspeção de enchimento de ampolas, vidros de medicamentos ou qualquer recipiente translúcido que contenha líquido.

A inspeção por análise de cores possibilita a criação de soluções para separar produtos por cor ou verificar se a tonalidade está igual a uma amostra. Tem a vantagem de ser um método determinístico em relação a uma inspeção de cores realizada por seres humanos, onde pode existir subjetividade. No entanto, variações de luminosidade podem dificultar a realização desse tipo de inspeção.

Existem também aplicações de rastreabilidade que envolvem leituras de códigos de barras ou de códigos bidimensionais como *Data Matrix*⁷ e o *QR Code*⁸. Permitem identificar e rastrear os produtos, bem como armazenar grandes quantidades de dado. São sistemas frequentemente utilizados na indústria automobilística mas são aplicáveis em qualquer projeto que necessite de controle de produção.

As aplicações para contagem, seleção e classificação podem instrumentar diversas

⁷Código de barras matricial bi-dimensional que consiste em células brancas ou pretas arranjadas em forma de quadrado ou retângulo. Se caracteriza pelas bordas em formato de L. Armazena no máximo 2335 caracteres alfanuméricos.

⁸*Quick Response Code* é um código de barras bi-dimensional criado pela empresa japonesa Denso-Wave em 1994. Pode armazenar até 7089 caracteres, dependendo do tipo de dado. Possui redundância em sua codificação, possibilitando correção e recuperação de informação.

indústrias, como por exemplo a indústria farmacêutica para contagem de medicamentos. Também é comum na produção de componentes eletrônicos para verificação de pinos e conectores. Na engenharia de transporte podem contribuir na análise do tráfego realizando contagem volumétrica de veículos e pessoas.

Capítulo 4

Metodologia

Como visto no Capítulo 3, existem diversas técnicas de processamento digital de imagens que viabilizam a construção de sistemas de visão computacional. A Seção 2.2 descreve exemplos de sistemas de visão computacional voltados para o monitoramento do tráfego, deixando claro o alto nível de complexidade dessas aplicações do ponto de vista de desenvolvimento. São trabalhos com ótimos resultados, mas de elevada complexidade algorítmica, exigindo um conhecimento profundo na área para um entendimento completo das metodologias propostas.

Este trabalho se diferencia dos demais por adotar uma metodologia simplificada, utilizando algoritmos de processamento de imagem já implementados pela biblioteca OpenCV 2.4.4 [Bradski, 2000], diminuindo a complexidade da solução. Nas seções a seguir, o fluxo de processos é descrito e detalhado, bem como as características de captura e cena especificadas.

4.1 Características de captura

A captura das imagens do tráfego foram feitas com a câmera que equipa um aparelho celular Samsung OMNIA W GT-I8350 [Samsung, 2013], com as seguintes especificações:

- CPU: 1,4 GHz
- Memória: 8 GB
- Sistema operacional: Windows Phone 7.8
- Câmera: 5.0 Megapixel

- Gravação de vídeos HD 720p @30fps¹

Inicialmente, as imagens foram capturadas e salvas na memória interna do aparelho, e num segundo momento transferidas para um *Laptop* onde seriam processadas. Todas as imagens foram capturadas durante o dia, em condições de iluminação estável e boa luminosidade. Portanto, o método proposto neste trabalho não está validado para grandes variações de iluminação ou cenas com pouca luz.

Em sistemas de monitoramento do tráfego que utilizam câmeras é muito comum a ocorrência de oclusão, ou seja, dependendo do ângulo de captura, alguns veículos não aparecem em parte ou por completo nas imagens. Alguns sistemas automáticos de contagem mais complexos tratam o problema de oclusão utilizando modelos para a forma dos veículos e calibração de câmera [Song & Nevatia, 2005]. O método de contagem desenvolvido neste trabalho não considera a possibilidade de oclusão e, com intuito de minimizar esse efeito, buscou-se um ângulo de captura superior e frontal à via. Portanto, as imagens foram capturadas na passarela para pedestres localizada na Avenida Presidente Carlos Luz 3001, Pampulha, Belo Horizonte - MG, nas proximidades do Shopping Del Rey. A Figura 4.1 ilustra a posição de captura e a cena obtida.

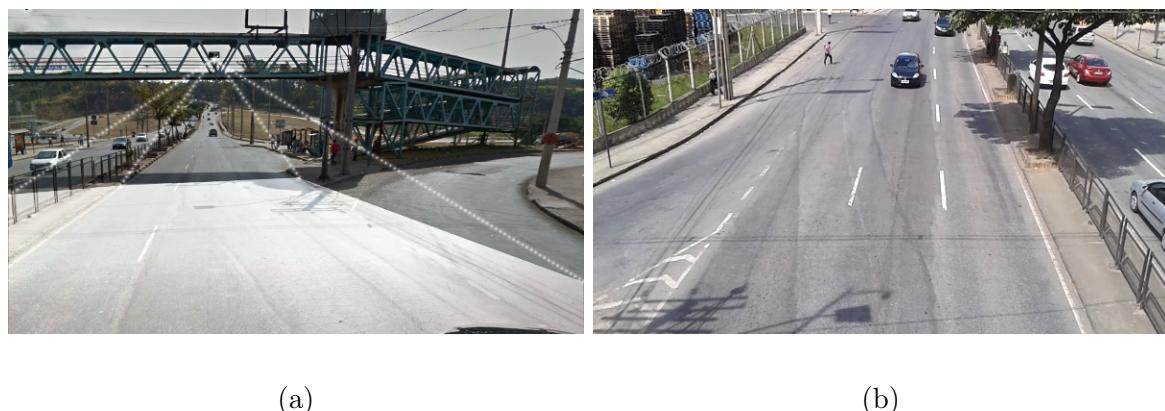


Figura 4.1: (a) Posicionamento da câmera para captura de imagens. A área em destaque simboliza o campo de visão obtido na posição escolhida. (b) Cena capturada na Avenida Presidente Carlos Luz.

¹Frames per second, ou frames por segundo.

4.2 Fluxo de processos

O método aqui apresentado pode ser dividido em seis etapas globais: entrada de dados, pré-processamento, subtração de *background*, binarização, detecção de *blobs* e rastreamento e contagem. A Figura 4.2 representa o fluxograma global do processo.

Para cada etapa de processamento de imagens existe o código C++ correspondente às operações realizadas pelas mesmas, com o objetivo de mostrar a simplicidade de uso da biblioteca OpenCV. O resultado obtido em cada etapa é apresentado por meio de imagens intermediárias do processo. Na última etapa, o fluxograma da Figura 4.9 descreve o algoritmo implementado para rastreamento e contagem dos veículos.

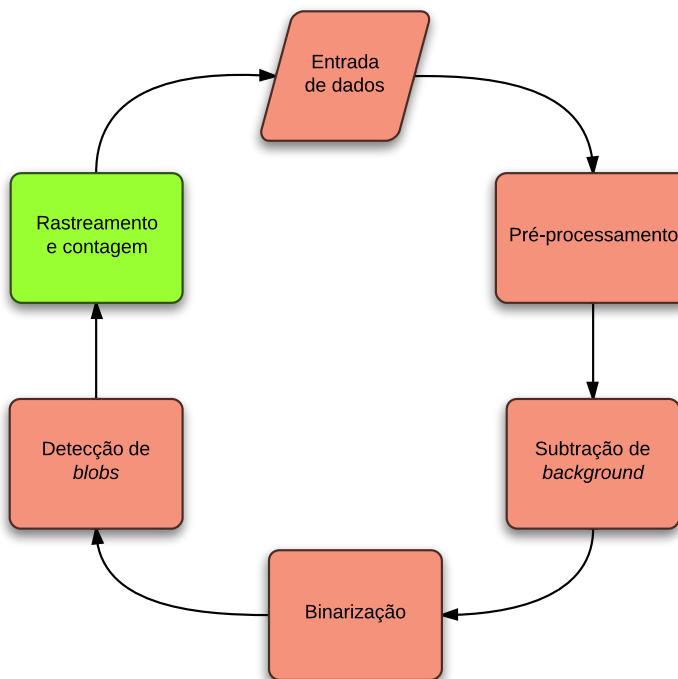


Figura 4.2: Fluxograma com a representação global do método de contagem.

De forma resumida, o método proposto considera o vídeo como um conjunto de imagens sequenciais, obtidas uma a uma na etapa de entrada de dados. Em seguida, cada imagem, chamada também de quadro ou *frame* de vídeo, passa por uma etapa de pré-processamento, responsável pela conversão para escala de cinza e uma filtragem linear. O resultado é usado para criação de um modelo do plano de fundo da cena e que, por meio de uma subtração do quadro atual, consegue-se isolar os objetos em movimento que encontram-se em primeiro plano. Na etapa de binarização, um limiar de intensidade de cor segmenta a imagem em regiões brancas, que são os objetos em movimento, e pretas, que representam o plano de fundo. Posteriormente, os objetos

brancos em movimento são detectados como componentes conectados ou *blobs*, e seu centro é tido como ponto chave para a etapa de rastreamento e contagem. Por último, um algoritmo considera os pontos chave, ou *keypoints*, para realizar o rastreamento e contagem dos veículos.

4.2.1 Entrada de dados

A obtenção dos dados em um sistema de visão computacional pode ser feita em tempo real ou a partir de imagens já capturadas previamente. No trabalho em questão, esse processo se dá por meio de vídeos já salvos em disco, através do trecho de código C++ a seguir:

```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>

int main(int argc, char const *argv[])
{
    cv::VideoCapture capture("video.avi");
    if (!capture.isOpened()) {
        std::cout << "can not open camera or video file" << std::endl;
        return;
    }
    ...
    for (;;) {
        cv::Mat frame;
        capture >> frame;

        if (frame.empty()) break;
        ...
    }

    return 0;
}
```

A classe `cv::VideoCapture` da OpenCV faz todo o trabalho de obtenção dos *frames*, abstraindo um arquivo de vídeo por uma sequência de imagens, obtidas uma a uma na linha: `capture >> frame`. A Figura 4.3 representa um *frame* do vídeo, usado para exemplificar o resultado do processamento de cada etapa subsequente.

Vale ressaltar que a biblioteca OpenCV utiliza objetos do tipo `cv::Mat` para armazenar em memória os dados de uma imagem, sendo este também o tipo de entrada

e saída para a maioria das funções de processamento de imagem.



Figura 4.3: *Frame* obtido a partir do vídeo de entrada.

4.2.2 Pré-processamento

A principal função da etapa de pré-processamento é a conversão da imagem de entrada para *grayscale*, uma vez que informações de cor não são relevantes nas etapas seguintes e o processamento de imagens em escala de cinza, que possuem apenas um canal, é mais rápido. No código, a conversão é feita pela chamada da função `cv::cvtColor`, passando como parâmetros as imagens de entrada e saída, e o *flag* `CV_BGR2GRAY`, que indica o tipo de operação desejada.

```
...
for (;;) {
    ...
    cv::Mat gray;
    cv::cvtColor(frame, gray, CV_BGR2GRAY);

    cv::GaussianBlur(gray, gray, cv::Size(7, 7), 3);
    ...
}
```

Em seguida, utiliza-se uma filtragem linear gaussiana, descrita na Seção 3.5, com o objetivo de atenuar ruídos oriundos do processo de captura e provocar um efeito

suave de borramento na imagem. A chamada da função `cv::GaussianBlur` realiza a operação de filtragem, que é parametrizada pelas imagens de entrada e saída, o tamanho do *kernel* utilizado como elemento de convolução e o desvio padrão nas direções X e Y. O resultado desse procedimento é ilustrado na Figura 4.4.



Figura 4.4: Resultado da etapa de pré-processamento, uma imagem filtrada em escala de cinza.

4.2.3 Subtração de *background*

A subtração de *background* é a operação mais complexa e custosa, do ponto de vista computacional, entre todas as etapas de processamento de imagens do trabalho em questão, mas o seu uso, a partir da implementação da OpenCV, é extremamente simples. Internamente, a OpenCV constrói um modelo adaptativo de mistura de gaussianas para subtração de *background* com detecção de sombras, baseado em Zivkovic [2004] e Zivkovic & van der Heijden [2006]. Nesse método, cada pixel é modelado como uma mistura de gaussianas e, em cada iteração, a probabilidade do pixel pertencer ao plano de fundo é calculada.

```

...
cv :: BackgroundSubtractorMOG2 model;
for (;;) {
    ...
    cv :: Mat foreground;
```

```

model(gray, foreground);
...
}
...

```

No código C++, essa operação pode ser realizada em duas linhas: a declaração do objeto `model` da classe `cv::BackgroundSubtractorMOG2` e a extração da imagem de *foreground*, ou imagem de primeiro plano, com a chamada `model(gray, foreground)`. O resultado é mostrado na Figura 4.5, onde os pixels pretos representam o plano de fundo, os brancos os objetos em movimento do primeiro plano e os pixels cinza suas sombras.



Figura 4.5: Detecção do *foreground* na etapa de subtração de fundo.

4.2.4 Binarização

Em visão computacional, imagens binárias são aquelas que possuem apenas dois valores de intensidade, normalmente 0 (preto) e 255 (branco). Geralmente, os pixels branco pertencem aos objetos de interesse da cena, que no escopo desse trabalho fazem parte dos veículos em movimento. A etapa de binarização é responsável por segmentar as regiões de interesse da cena para que, nas próximas etapas, sejam detectadas e rastreadas.

A imagem de primeiro plano obtida na etapa anterior (Subseção 4.2.3) possui pixels cinza, que representam as sombras dos objetos. Analisando a Figura 4.5, percebe-se

que os pixels branco sozinhos não definem muito bem um veículo, mas o conjuntos dos pixels branco e cinza sim. Então, é feita uma operação de limiarização, também conhecida por *thresholding* (Seção 3.6), com intuito de elevar o valor de todos os pixels cinza para branco. Isso é feito pela chamada da função `cv::threshold`, parametrizada pelas imagens de entrada e saída, um valor de limiar baixo, que garante a elevação à branco de praticamente todos os tons de cinza, e o tipo de limiarização a ser aplicada, nesse caso binária.

Percebe-se na Figura 4.6a que a operação de *thresholding* segmentou corretamente os veículos. No entanto, para detecção de objetos em visão computacional, é interessante que as regiões de interesse sejam uniformes e não contenham orifícios. Nesse tipo de situação é comum o uso de operações morfológicas, capazes de alterar a forma dos objetos segmentados em imagens binárias. Como o objetivo é uniformizar a região de segmentação dos objetos, deve-se aplicar uma operação de fechamento, que diminui orifícios e descontinuidades nas regiões de interesse, alongando os segmentos de cor branca pelas imperfeições de cor preta. A Figura 4.6b mostra uma melhor uniformidade na região dos objetos segmentados.

```

...
for (;;) {
    ...
    cv::threshold (foreground , foreground , 5 , 255 , CV_THRESH_BINARY);

    cv::Mat morph;
    cv::Mat element = cv::getStructuringElement (cv::MORPH_ELLIPSE,
                                                cv::Size (5,5));
    cv::morphologyEx (foreground , morph , CV_MOP_CLOSE, element ,
                      cv::Point (-1,-1), 3);
    ...
}
...

```

As operações morfológicas, assim como os filtros, também utilizam um elemento estrutural que percorre toda a imagem. Nesse caso, utilizou-se um *kernel* elíptico, que é a forma que mais se aproxima de um veículo na imagem binarizada. A criação desse elemento se dá pela chamada da função `cv::getStructuringElement`, que recebe como parâmetro a forma e o tamanho desejados. A operação de fechamento propriamente dita é feita pelo método `cv::morphologyEx`, parametrizado pelas imagens de entrada e saída, o tipo de operação morfológica, o elemento estrutural, o ponto de ancoragem com o *kernel* (`cv::Point(-1,-1)` representa o centro) e o número de iterações.

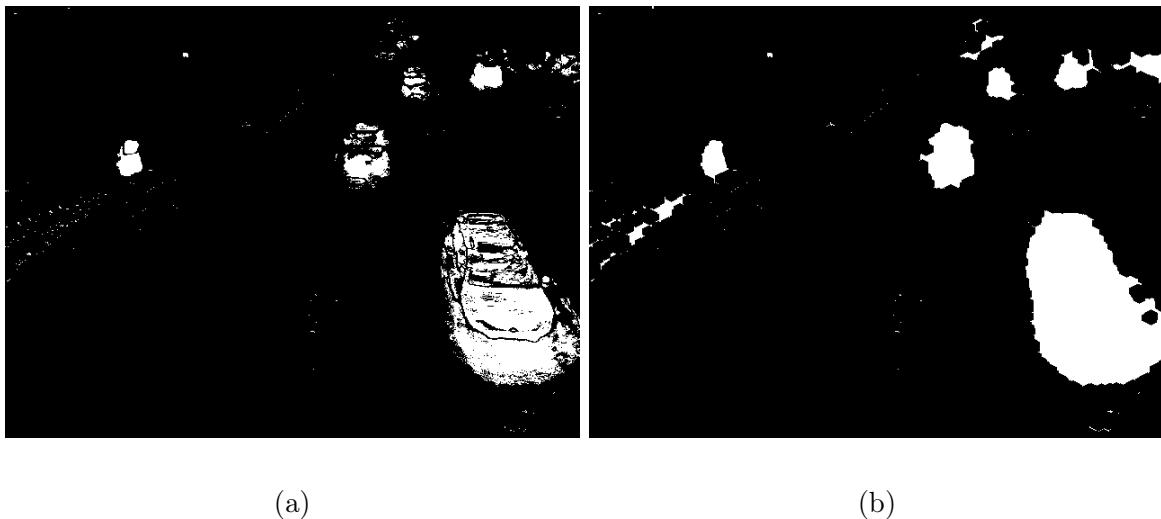


Figura 4.6: Resultado da etapa de binarização. (a) Imagem binarizada utilizando um limiar simples; (b) Operação morfológica de fechamento.

4.2.5 Detecção de *blobs*

Uma vez segmentados, os objetos de interesse são detectados nessa etapa como *blobs*, que são regiões de pontos com valor 255 (branco), conectados entre si, definidas pelo seu ponto central e diâmetro. A biblioteca OpenCV utiliza a entidade `cv::KeyPoint` para representá-los. Nesse caso, a principal função dos *keypoints* é definir a posição de cada objeto segmentado, para que na próxima etapa esses pontos sejam rastreados.

A OpenCV disponibiliza uma interface abstrata para detecção de *keypoints*, chamada `cv::FeatureDetector`. Uma de suas especializações é a classe `cv::SimpleBlobDetector`, capaz de extrair *blobs* de uma imagem. Em sua contrução, a classe `cv::SimpleBlobDetector` recebe uma estrutura de parâmetros de filtragem, responsáveis por descartar pontos que não atendam as especificações desejadas. O método proposto habilita apenas dois tipos de filtro: de cor, que considera apenas *blobs* brancos, e área, que abrange regiões com área mínima de 500 pixels e máxima de 80000.

```
...
cv::SimpleBlobDetector::Params params;
params.filterByInertia = false;
params.filterByConvexity = false;
params.filterByColor = true;
params.blobColor = 255;
params.filterByCircularity = false;
params.filterByArea = true;
params.minArea = 500.0f;
params.maxArea = 80000.0f;
```

```
cv::Ptr<cv::FeatureDetector> detector =  
    new cv::SimpleBlobDetector(params);  
detector->create("SimpleBlob");  
...  
for (;;) {  
    ...  
    std::vector<cv::KeyPoint> keypoints;  
    detector.detect(morph, keypoints)  
}  
...
```

Uma vez criado, o objeto `detector`, através do método `detect`, recebe como parâmetro a imagem binarizada da Figura 4.6b e retorna um vetor de *keypoints*, contendo todos os *blobs* detectados naquele *frame*. A Figura 4.7 ilustra os pontos detectados, representados pelos círculos em vermelho.

Esta é a última operação que envolve processamento de imagem. Com a posição e o tamanho dos *blobs* detectados, definidos por um vetor de *keypoints*, é possível criar um algoritmo de rastreamento que garanta que cada objeto seja contado apenas uma vez, como descrito logo a seguir na Subseção 4.2.6.



Figura 4.7: Resultado da etapa de detecção de *blobs*.

4.2.6 Rastreamento e contagem

Assim como em todas as etapas anteriores, buscou-se a implementação de um método simples de rastreamento e contagem, mas que fosse capaz de obter bons resultados. O algoritmo utiliza o vetor de *keypoints*, obtido na etapa anterior, e uma máscara binária, que define a região de contagem. Um objeto só é contabilizado quando o *keypoint* que o define entra na região de contagem, definida pelos pixels branco da Figura 4.8.



Figura 4.8: Máscara binária que define a região de contagem do algoritmo.

O rastreamento dos *blobs*, aqui representados por *keypoints*, é feito por um perseguidor, denominado *tracker*. Um *tracker* é um objeto circular, definido por sua posição e raio. Um *keypoint* está contido em um *tracker* se a distância entre eles for menor que o raio do *tracker*, ou seja, se o *keypoint* pertencer à área definida pelo *tracker*. Na Figura 4.10, o *tracker* é representado pelos círculos azuis. Para que um *keypoint* seja contabilizado na contagem ele deve, além de ingressar na região de contagem, ser contido por pelo menos um *tracker*. Internamente, o algoritmo armazena uma lista com todos os *trackers* válidos, para que na próxima iteração esses dados sirvam como validação para a operação de contagem.

O fluxograma da Figura 4.9 ilustra os passos do algoritmo de rastreamento e contagem. Pra cada *frame* processado é gerado uma lista de *keypoints*, que tem seus itens percorridos individualmente. Para cada *keypoint*, obtém-se todos os *trackers* que o contenham. Nesse momento, três situações podem ser identificadas:

1. Ainda não existe nenhum *tracker* que contenha *keypoint*;
2. Existe apenas um *tracker* que contenha *keypoint*;
3. Existe mais de um *tracker* que contenha *keypoint*.

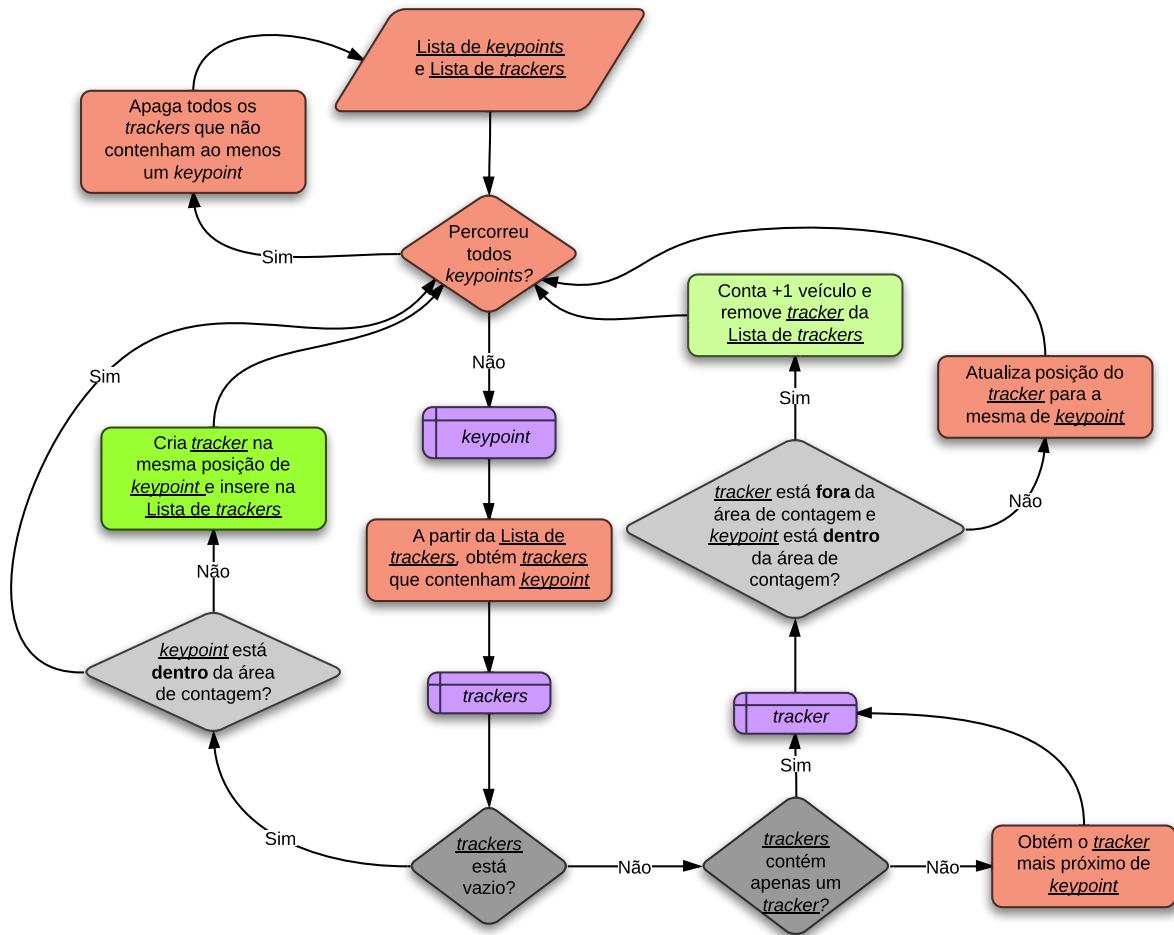


Figura 4.9: Fluxograma do algoritmo de rastreamento e contagem de veículos.

O primeiro caso acontece quando um *keypoint* acabou de ser detectado na cena. Um *tracker* para rastreá-lo é criado na mesma posição de *keypoint* e inserido na lista interna de *trackers*, a menos que o *keypoint* encontre-se na área de contagem. Ou seja, não faz sentido rastrear objetos que surgiram subtamente dentro da área de contagem.

Na situação 2, o *tracker* encontrado indica que o *keypoint* já estava sendo rastreado nas iterações anteriores. A posição do *tracker* representa exatamente a posição do objeto na última iteração. Logo, se o *tracker* encontra-se **fora** da área de contagem e o *keypoint* **dentro**, significa que o objeto acabou de entrar em uma condição

válida de contagem, ou seja, ingressou na região de contagem definida pela máscara binária, e mais um veículo é contabilizado. Uma vez contado, o *tracker* daquele objeto é removido da lista de *trackers*, uma vez que não é mais necessário rastreá-lo. Caso a condição de contagem não seja atendida, a posição do *tracker* é atualizada para a mesma de *keypoint*.

A situação 3 acontece quando um *keypoint* está contido em mais de *tracker*, devido à proximidade de dois ou mais objetos detectados na cena. Esse caso é tratado da mesma forma que na situação 2, adicionando apenas uma etapa inicial que identifica qual o *tracker* mais relevante para *keypoint* dentre os selecionados, ou seja, o *tracker* mais próximo de *keypoint*.

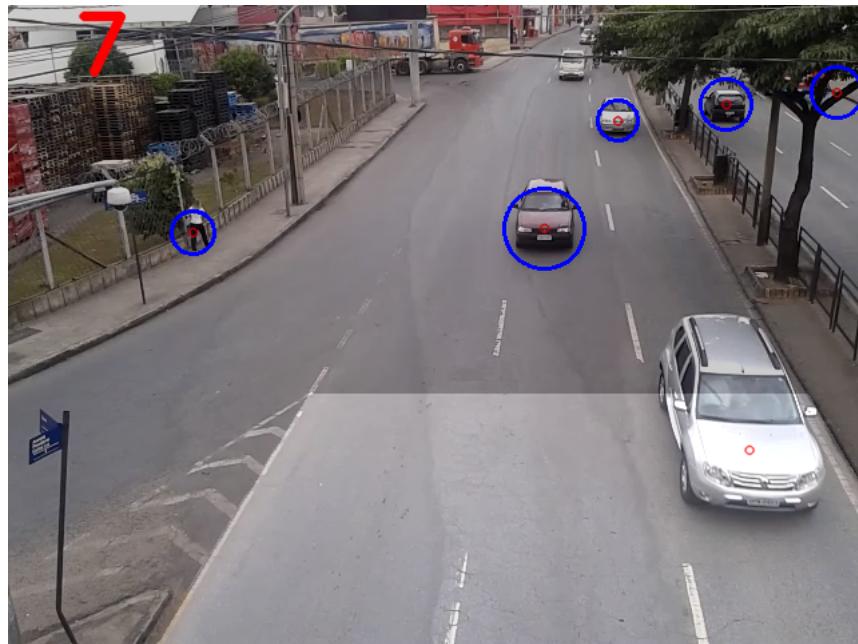


Figura 4.10: Resultado da etapa de rastreamento e contagem, ilustrando *trackers*, *keypoints* e a região de contagem.

Depois de percorrer todos os *keypoints*, o algoritmo remove da lista os *trackers* que não contenham ao menos um *keypoint*. Isso é necessário pois a detecção de objetos não é perfeita, ou seja, se em algum *frame* um objeto já anteriormente rastreado não for detectado como um *keypoint*, aparece na lista de *trackers* um elemento que não contém um *keypoint*, configurando uma condição inválida.

A Figura 4.10 ilustra um veículo que entrou na região de contagem, teve seu *tracker* removido e foi contabilizado corretamente. Percebe-se também que qualquer objeto em movimento é detectado e rastreado, como uma pessoa caminhando. No entanto, somente objetos que possuam um *keypoint* contido por um *tracker* e que

entrem na região de contagem são contabilizados.

4.3 Estrutura do software

Para que o método aqui proposto pudesse ser testado e validado foi desenvolvido um *software* multiplataforma na linguagem C++ [cplusplus.com, 2013], com intuito de obter um bom desempenho computacional, fácil implementação e a possibilidade de utilizar os conceitos de programação orientada a objetos. A Figura 4.11 ilustra a estrutura e o relacionamento entre classes através de um diagrama UML².

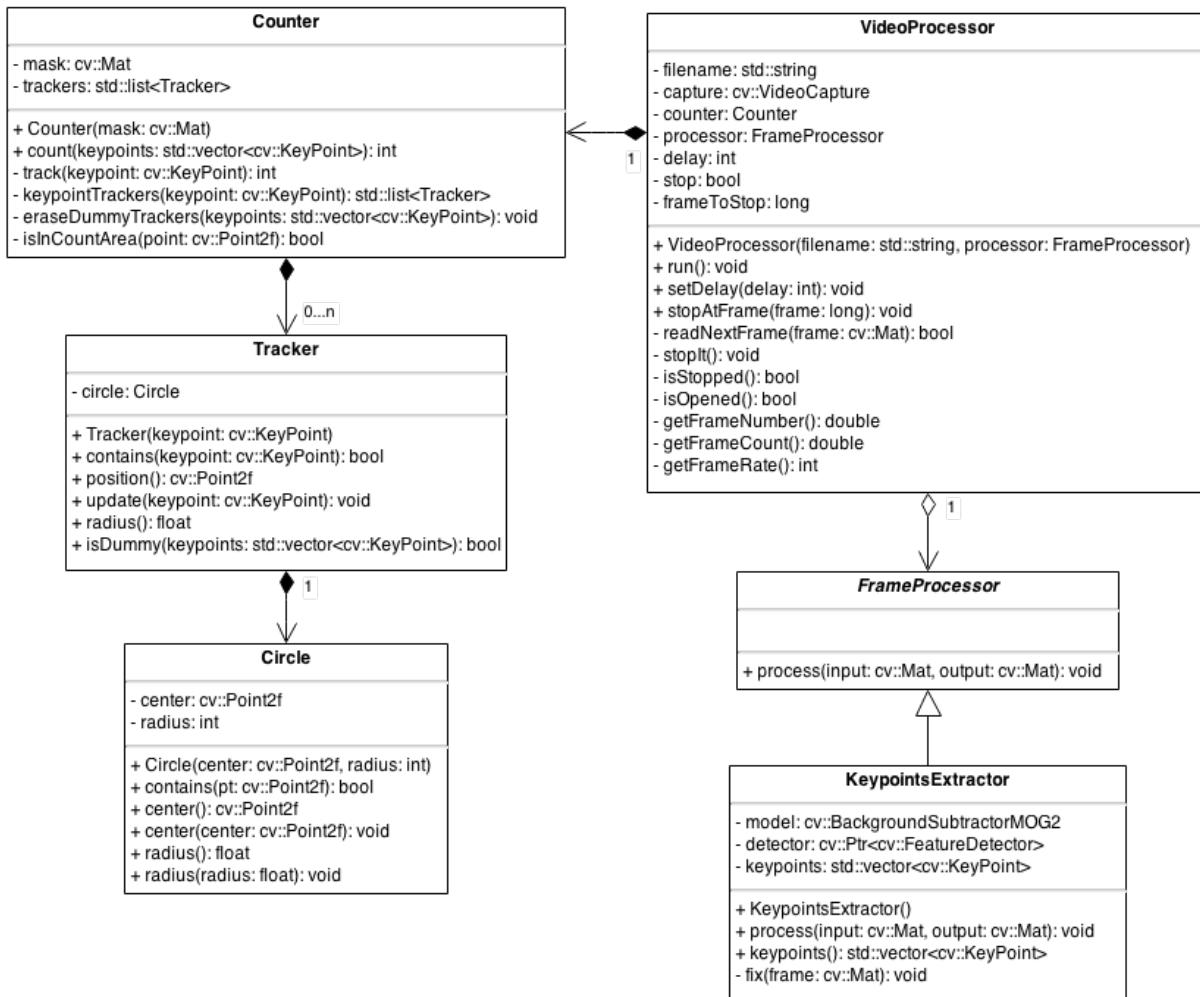


Figura 4.11: Diagrama UML do *software*.

²Unified Modeling Language - um conjunto padronizado de diagramas com notação formalmente definida, adotado na área de engenharia de software em um esforço para facilitar a concepção de sistemas orientados a objetos [Magalhães, 2008].

A classe `VideoProcessor` é a classe central do sistema. Ela é responsável por abrir e ler o arquivo de vídeo através do objeto `capture` do tipo `cv::VideoCapture`; interfacear com a classe abstrata `FrameProcessor` através da chamada do método `process`; e executar o algoritmo de rastreamento e contagem implementado pela classe `Counter`.

A classe abstrata `FrameProcessor` define uma interface cuja função é receber uma imagem de entrada, executar algum processamento sobre ela e disponibilizar o resultado. A classe `KeypointsExtractor` implementa essa interface abstrata com as operações descritas nas seguintes subseções: Entrada de dados, Pré-processamento, Subtração de *background*, Binarização e Detecção de *blobs*. Ou seja, essa classe concentra todas as operações de visão computacional e através do método `keypoints` retorna os pontos que potencialmente representam veículos em movimento.

A classe `Counter` implementa o algoritmo descrito na subseção Rastreamento e contagem. Utilizando os *keypoints*, uma máscara binária que define a região de contagem e uma lista interna de *trackers*, o método `count` retorna quantos eventos de contagem ocorrem para cada iteração.

A classe `Circle` é utilizada pela classe `Tracker` para representar computacionalmente um círculo com centro e raio definidos. Já a classe `Tracker` é utilizada na forma de lista pela classe `Counter`, na implementação do algoritmo de rastreamento e contagem.

O código dessa aplicação pode ser encontrado no seguinte repositório: <https://github.com/arthurbailao/potential-ironman>.

4.4 Avaliação dos resultados

Para avaliar o método de rastreamento e contagem de veículos aqui proposto foram utilizadas algumas métricas já conhecidas na literatura. Nas subseções a seguir são descritas as métricas de análise de resultados usadas nesse trabalho.

4.4.1 Matriz de confusão

A matriz de confusão contém informações sobre classificações reais e preditas feitas por um classificador, como mostrado na Tabela 4.1. Ela permite que a partir de seus elementos diversas medidas de desempenho sejam calculadas.

Um verdadeiro positivo (VP) é um evento predito como pertencente à classe positiva, sendo este realmente pertencente à classe positiva. Um falso negativo (FN) acontece quando o evento predito como pertencente da classe negativa pertence à classe

Tabela 4.1: Matriz de confusão para um classificador de duas classes.

		Classe verdadeira	
		Positivo	Negativo
Classe predita	Positivo	VP	FP
	Negativo	FN	VN

positiva. Analogamente, o verdadeiro negativo (VN) é um evento da classe negativa predito corretamente e o falso positivo (FP) um evento da classe positiva predito de forma incorreta.

Para evitar o viés estatístico na montagem da matriz de confusão ao caracterizar as quatro possibilidades, estabelece-se uma janela de tempo baseada no tempo médio em que um veículo leva para percorrer toda a cena, dada uma determinada condição de tráfego. A contagem pura e simples, sem considerar esta janela de tempo, levaria à não consideração dos eventos verdadeiro negativo VN. Dessa forma, os elementos da matriz de confusão podem ser descritos como:

- Verdadeiro positivo (VP): mais um veículo foi contabilizado quando o mesmo adentrou a região de contagem em uma janela de tempo;
- Falso negativo (FN): mais um veículo não foi contabilizado quando o mesmo adentrou a região de contagem em uma janela de tempo;
- Verdadeiro negativo (VN): mais um veículo não foi contabilizado quando nenhum veículo adentrou a região de contagem em uma janela de tempo;
- Falso positivo (FP): mais um veículo foi contabilizado quando nenhum veículo adentrou a região de contagem em uma janela de tempo.

Vale ressaltar que dentro de uma janela de tempo pode acontecer um ou mais eventos, sejam eles do mesmo tipo ou não. A única restrição é que, para cada janela, no mínimo um evento deve obrigatoriamente acontecer. Na prática o evento verdadeiro negativo VN ocorre apenas quando nenhum dos outros três eventos acontece.

Algumas medidas de desempenho podem ser calculadas a partir dos elementos da matriz de confusão. São elas: Especificidade (Equação 4.1), Valor Preditivo Negativo (Equação 4.2), Precisão (Equação 4.3), *Recall* (Equação 4.4), Acurácia (Equação 4.5) e *F-Measure* (Equação 4.6) [Powers, 2007].

$$E = \frac{VN}{VN + FP} \quad (4.1)$$

$$VPN = \frac{VN}{VN + FN} \quad (4.2)$$

$$P = \frac{VP}{VP + FP} \quad (4.3)$$

$$R = \frac{VP}{VP + FN} \quad (4.4)$$

$$A = \frac{VP + VN}{VP + FP + FN + VN} \quad (4.5)$$

$$FM = \frac{2 * R * P}{R + P} \quad (4.6)$$

4.4.2 Índice Kappa (K)

O índice Kappa é utilizado como uma medida apropriada da exatidão por representar inteiramente a matriz de confusão. Ele toma todos os elementos da matriz ao invés de apenas aqueles que retratam a quantidade de classificações verdadeiras, o que ocorre quando se calcula a exatidão global da classificação [Rosenfield, 1986].

O índice Kappa pode ser encontrado com base na Equação 4.7:

$$K = \frac{\theta_1 - \theta_2}{1 - \theta_2}, \quad (4.7)$$

onde

$$\theta_1 = \frac{VP + VN}{VP + FP + FN + VN}, \quad (4.8)$$

e

$$\theta_2 = \frac{\alpha + \beta}{\gamma^2}, \quad (4.9)$$

onde $\alpha = (VP + FN) * (VP + FP)$, $\beta = (VN + FN) * (VN + FP)$ e $\gamma = VP + VN + FP + FN$.

O nível de exatidão obtido pode ser classificado conforme a Tabela 4.2, seguindo o estabelecido por Landis & Koch [Landis & Koch, 1977].

Tabela 4.2: Nível de exatidão de uma contagem, conforme o valor do índice Kappa.

Índice Kappa (K)	Qualidade
$K < 0.2$	Ruim
$0.2 \leq K < 0.4$	Razoável
$0.4 \leq K < 0.6$	Bom
$0.6 \leq K < 0.8$	Muito bom
$K \geq 0.8$	Excelente

Capítulo 5

Testes e Resultados

O método proposto foi implementado em forma de um *software* multiplataforma, desenvolvido em linguagem C++ [cplusplus.com, 2013], com intuito de obter um bom desempenho de execução e fácil implementação utilizando os conceitos de programação orientada a objetos.

Como biblioteca para processamento das imagens, foi utilizado a OpenCV 2.4.4 [Bradski, 2000], contribuindo para que operações complexas de visão computacional pudessem ser realizadas com poucas linhas de código. O desenvolvimento do método não está preso à tecnologia escolhida, podendo ser empregado utilizando outras linguagens de programação, bem como outras bibliotecas. A tecnologia foi escolhida visando a simplicidade no processo de desenvolvimento das operações de processamento de imagens, que estão bem detalhadas pelos trechos de código na Seção 4.2.

5.1 Testes

Todos os vídeos foram capturados por um aparelho celular Samsung OMNIA W GT-I8350 [Samsung, 2013], como descrito na Seção 4.1. As imagens foram feitas na passarela localizada na Av. Presidente Carlos Luz, nos dois sentidos da via, como ilustra a Figura 5.1. Duas configurações de captura foram utilizadas:

- HD 720p: gerando vídeos com resolução 1280×720 , e *framerate* de 29 FPS;
- VGA: gerando vídeos com resolução 640×480 , e *framerate* de 29 FPS.

Com intuito de equiparar as resoluções dos vídeos gerados a partir das duas configurações de captura, as amostras em HD foram redimensionadas para ter altura

e largura metade do valor original, resultando em vídeos com resolução 640×360 . A Tabela 5.1 lista as amostras utilizadas.

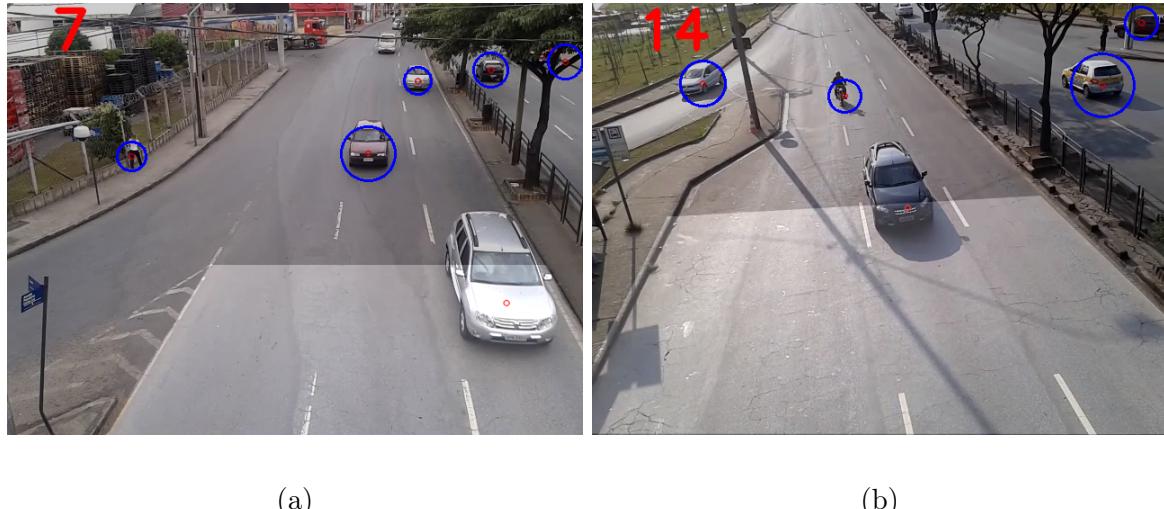


Figura 5.1: Resultado do método de contagem para dois tipos de cena. (a) Av. Carlos Luz, sentido Pampulha; (b) Av. Carlos Luz, sentido Centro.

Tabela 5.1: Vídeos utilizados nos testes.

Nº	Nome do vídeo	Resolução	Duração	FPS
1	carlos_luz_centro_vga	640×480	00:05:23	29
2	carlos_luz_centro_hd_resized	640×360	00:05:25	29
3	carlos_luz_pampulha_vga_1	640×480	00:05:43	29
4	carlos_luz_pampulha_vga_2	640×480	00:05:19	29
5	carlos_luz_pampulha_hd_2_resized	640×360	00:06:19	29

Tanto o desenvolvimento do *software*, quanto os testes, foram feitos em um *Laptop* Dell Vostro V131, Processador Intel Core I3-2330M (2.20GHZ, dual core 4 Threads, 3MB L3 cache), 8GB de memória RAM, em ambiente Linux Fedora 18 (Spherical Cow).

5.2 Resultados e discussão

A Tabela 5.2 mostra o resultado da contagem dos eventos da matriz de confusão e os índices de desempenho calculados a partir deles, incluindo o índice Kappa (K).

Pode-se observar que nas amostras 1 e 2, que são imagens da Av. Carlos Luz no sentido Centro, a qualidade ficou entre muito bom e excelente, segundo a escala

definida na Tabela 4.2; já nas amostras 3, 4 e 5, que se referem à mesma via no sentido Pampulha, a qualidade ficou entre bom e muito bom. Essa variação na qualidade da classificação entre as vias acontece principalmente pelas características do tráfego de cada sentido. Analisando os vídeos é possível identificar vários veículos de grande porte trafegando no sentido Pampulha, muitos saindo da Fábrica da Coca-Cola ali localizada

Tabela 5.2: Resultado obtido da contagem nas amostras de teste, índices de desempenho e índice Kappa (K).

Nº	VP	VN	FP	FN	E	VPN	P	R	FM	A	K	Qual.
1	92	35	2	6	0.95	0.85	0.98	0.94	0.96	0.94	0.86	Excelente
2	76	34	6	10	0.85	0.77	0.93	0.88	0.90	0.87	0.71	Muito bom
3	107	19	1	26	0.95	0.42	0.99	0.80	0.89	0.82	0.49	Bom
4	87	28	8	22	0.78	0.56	0.92	0.80	0.85	0.79	0.51	Bom
5	94	37	2	15	0.95	0.71	0.98	0.86	0.92	0.89	0.73	Muito bom



Figura 5.2: Tipo de tráfego encontrado nas vias. (a) Veículos muito grandes causam problemas como oclusão e vibrações na passarela. (b) Quando a cor de um veículo é muito próxima da cor do *background*, problemas na detecção do objeto podem acontecer.



Figura 5.3: Problemas encontrados no processamento de imagens. (a) A operação de subtração de *background* fica comprometida quando veículos com área muito grande aparecem na cena. (b) Veículos próximos são detectados como um único objeto, gerando eventos falso negativo FN.

Capítulo 6

Considerações Finais

O desempenho do algoritmo para as amostras em HD foi baixo, inviabilizando a execução em tempo real. No ambiente de testes utilizado, descrito na Seção 5.1, apenas os vídeos com resolução 640×360 poderiam ser utilizados para processamento em tempo real. Com um poder de processamento maior, provavelmente seria possível realizar a contagem em tempo real nos vídeos com resolução 640×480 .

Embora já existam diversos métodos para a realização de contagem volumétrica de veículos, como descrito no Capítulo 2, geralmente eles são muito caros e causam transtornos na via durante sua instalação e manutenção. O trabalho em questão propõe um método não-invasivo, de baixo custo e simples implementação, capaz de obter medições de boa precisão.

6.1 Limitações

Vários aspectos de alteração dinâmica da cena interferiram no processamento digital. O algoritmo de subtração de *background*, por exemplo, apresentou problemas quando veículos de grande porte aparecem nas imagens, como mostra a Figura 5.3a. Além disso, movimentações na câmeras causadas por trepidações da passarela quando veículos de grande porte transitavam na via, também geraram ruídos na subtração de *background*, comprometendo as etapas subsequentes.

Como não foi desenvolvido um método para identificar se o objeto em movimento é um veículo ou não, alguns elementos indesejáveis foram considerados na contagem. Ao mesmo tempo, devido à proximidade, a detecção de *blobs* falhou e alguns veículos não foram contabilizados, como mostrado na Figura 5.3b.

6.2 Trabalhos futuros

Como possíveis trabalhos futuros, podem ser apontados:

- Minimizar o prejuízo causado pela movimentação da câmera pelo uso de pontos fiduciais;
- Definição de uma região de interesse baseada na área com maior movimentação nas imagens;
- Desenvolvimento de um método para contagem de veículos no período noturno ou em locais com baixa iluminação;
- Criação de técnicas para classificação de veículos quanto ao tamanho;
- Contagem de veículos em cruzamentos, com deslocamentos em várias direções e sentidos;
- Estimação do volume do tráfego em vias congestionadas.

Referências Bibliográficas

- Almeida, F. A. M. (2010). Classificação Automática de Veículos pelo Perfil Magnético através de Técnicas de Aprendizagem de Máquina. Dissertação de mestrado, Universidade Estadual do Ceará.
- Behrisch, M.; Bieker, L.; Erdmann, J. & Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility: An Overview. Em *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pp. 63–68, Barcelona, Spain.
- Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- Bradski, G. & Kaehler, A. (2008). *Learning OpenCV*. O'Reilly.
- CITILABS (2013). Cube Dynasim 4. <http://www.citilabs.com/cube-dynasim-4>.
- cplusplus.com (2013). The C++ Resources Network. <http://wwwcplusplus.com/>.
- DNIT (2011). Plano Nacional de Contagem de Trânsito. <http://www.dnit.gov.br/rodovias/operacoes-rodoviarias/postos-de-contagem/plano-nacional-de-contagem-de-transito>.
- Feitosa, F. C. C. (2012). Um estudo prático para contagem volumétrica automática de veículos usando Visão Computacional. Dissertação de mestrado, Universidade Federal do Goiás.
- FENABRAVE (2013). Federação Nacional da Distribuição de Veículos Automotores. <http://www3.fenabrade.org.br:8082/plus/>.
- GIMP (2013). The GNU Image Manipulation Program. <http://www.gimp.org/>.
- Goldner, L. G. (2009). Engenharia de Tráfego 1º Módulo. Relatório técnico, Universidade Federal de Santa Catarina.
- Gonzalez, R. C. & Woods, R. E. (2000). *Processamento de Imagens Digitais*. Edgard Blücher Ltda.

- Intel (2013). Intel Corporation. <http://www.intel.com/>.
- Invent Vision (2013). Aplicações. <http://www.inventvision.com.br/>.
- Johnson, S. (2006). *Stephen Johnson on Digital Photography*. O'Reilly.
- Laganière, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing Ltd.
- Lancaster, H. (1973). *An introduction to medical statistics*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley. ISBN 9780471512509.
- Landis, J. R. & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pp. 159--174.
- Magalhães, H. A. (2008). *Análise em Alta Resolução de Perfis Magnéticos de Sensores a Laço Indutivo para Classificação de Veículos Automotores*. Doutorado, Universidade Federal de Minas Gerais - UFMG.
- Martinsky, O. (2007). Algorithmic and Mathematical Principles of Automatic Number Plate Recognition Systems. Dissertação de mestrado, BRNO University of Technology.
- Morris, T. (2004). *Computer Vision and Image Processing*. Palgrave Macmillan.
- of Transportation, M. D.; Administration, U. S. F. H.; of Transportation, U. S. D.; Guidestar, M. & Group, S. C. (1997). *Field Test of Monitoring of Urban Vehicle Operations Using Non-intrusive Technologies*. Federal Highway Administration.
- OpenCV Development Team (2013). OpenCV 2.4.5.0 documentation. <http://docs.opencv.org/>.
- Powers, D. M. W. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Relatório técnico SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia.
- PTV (2013). Vissim. <http://vision-traffic.ptvgroup.com/en-uk/products/ptv-vissim/>.
- Rosenfield, G. H. (1986). A coefficient of agreement as a measure of thematic classification accuracy. *Photogrammetric Engineering and Remote Sensing (PE&RS)*, 52(2):223--227.

- Samsung (2013). OMNIA W GT-I8350. <http://www.samsung.com/sg/consumer/mobile-devices/smartphone/windows-os/GT-I8350HKAXSP-spec>.
- Schouten, T. (2003). Inhoud college Image Processing - Chapter 2: Fundamental Aspects. <http://www.cs.ru.nl/~ths/rt2/col/h2/2fundENG.html>.
- Shapiro, L. G. & Stockman, G. C. (2001). *Computer Vision*. Prentice Hall.
- Song, X. & Nevatia, R. (2005). A Model-based Vehicle Segmentation Method for Tracking. Em *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference*, volume 2, pp. 1124–1131, Beijing, China.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- Tancredi, P. R. (2012). Monitoramento do Acesso de Veículos de Carga em Vias Urbanas. Dissertação de mestrado, Universidade de São Paulo.
- TRL Software (2013). ARCADY. https://www.trlsoftware.co.uk/products/junction_signal_design/arcady.
- Zivkovic, Z. (2004). Improved adaptive Gaussian mixture model for background subtraction. Em *International Conference Pattern Recognition*, UK.
- Zivkovic, Z. & van der Heijden, F. (2006). Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction. Em *Pattern Recognition Letters*, volume 27, pp. 773–780.