UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA DE COMPUTAÇÃO

ALLAN PALMERIO VIEIRA
RICARDO FANTIN DA COSTA

CONTAGEM VOLUMÉTRICA DE VEÍCULOS EM VIAS URBANAS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2014

ALLAN PALMERIO VIEIRA
RICARDO FANTIN DA COSTA

CONTAGEM VOLUMÉTRICA DE VEÍCULOS EM VIAS URBANAS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Bogdan Tomoyuki Nassu

Orientadora: Leyza Elmeri Baldo Dorini

CURITIBA

2014

AGRADECIMENTOS

Gostaríamos de agradecer primeiramente ao nosso colega Marlon Subtil Marçal, que participou de forma ativa da concepção e de grande parte do desenvolvimento deste trabalho, integrando a equipe durante o maior tempo dedicado a este, mas infelizmente saindo dela pouco antes de seu término.

Nossos mais sinceros agradecimentos à professora orientadora Leyza, sem a qual seria impossível enfrentar um problema de tal magnitude em um trabalho de conclusão de curso. Todo o apoio e conselho nos momentos de resultados tanto bons quanto insatisfatórios nos guiaram e incentivaram para conseguirmos alcançar os objetivos que decidimos desafiar. É também uma grande inspiração para os entusiastas da área de processamento de imagens.

Durante o desenvolvimento recebemos o reforço de um segundo orientador, o professor Bogdan, cuja criatividade e experiência imprimiram significativo aumento no ritmo de desenvolvimento do trabalho. Sua ajuda nos proporcionou grande aprendizado e motivação, e por isso somos muito gratos.

Por fim, não poderíamos deixar de agradecer a professora Keiko Verônica Ono Fonseca, que com imensa boa vontade e a abordagem sempre didática de encorajar a discussão de problemas levados pelos alunos, proporcionou contato com profissionais da área de engenharia de tráfego em visita técnica, fator relevante para o desenvolvimento deste trabalho.

“A arte nunca está acabada, apenas abandonada.”

Leonardo da Vinci

RESUMO

VIEIRA, Allan; COSTA, Ricardo. CONTAGEM VOLUMÉTRICA DE VEÍCULOS EM VIAS URBANAS. 56 f. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

O planejamento de vias urbanas deve levar em conta, entre outros fatores, o fluxo de veículos na área analisada, informação cada vez mais difícil de se conseguir devido ao aumento expressivo da frota de automóveis. Algumas formas de se alcançar tal informação incluem a alocação de pessoas para contagem ou disposição de sensores nas vias. O objetivo deste trabalho é prover uma abordagem computacional automática de contagem de veículos através do processamento de vídeos obtidos a partir da filmagem de vias urbanas.

O trabalho consistiu na aplicação de uma série de algoritmos de visão computacional sobre vídeos, de modo a realizar a segmentação e a contagem de veículos. Várias técnicas de processamento são utilizadas, cada qual com objetivo específico de extrair alguma informação ou realçar alguma característica da imagem (quadro do vídeo). A abordagem utiliza-se da técnica de subtração de fundo para detecção de objetos em movimento e, após aplicação de filtros morfológicos, a imagem de cada quadro do vídeo, nesse ponto representada por diversas manchas(blobs), é analisada para descobrir quais manchas representam carros, e, se positivo, quantos carros representam. A análise da mancha é feita de duas maneiras, uma analisando a forma e sua área e outra acompanhando a velocidade dos pontos de interesse presentes na imagem.

O resultado do projeto é bastante promissor. Nos 274 carros dos vídeos, o programa deixou de contar um carro e contou outros dezessete a mais, resultando em um erro total de 6% nos vídeos analisados na abordagem baseada na área. Já a abordagem baseada nas velocidades dos pontos de interesse o erro total foi de 18%.

Palavras-chave: Contagem Volumétrica. Análise de Contorno. Fluxo Ótico Esparso. Clusterização.

ABSTRACT

VIEIRA, Allan; COSTA, Ricardo. VEHICLE COUNT ON URBAN ROADS. 56 f. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

The design of urban roads must take into account, among other things, the quantity of vehicles driven on the roads, an increasingly difficult information to obtain due to the growth of the vehicles's population. Some of the ways of getting such information include the assignment of people to do the counting or the placement of sensors on the road. The purpose of this work is to provide an automatic computational approach for counting vehicles through video processing (obtained from urban roads recordings).

This work consists on applying a series of computer vision algorithms over the videos in order to segment and count the vehicles. Many processing techniques are used, each with its own objective of extrating some information or enhacing some feature of the image (video frame). The proposed approach utilizes the background subtraction technique to detect moving objects and, after applying morphological filters, each image of the video, represented by blobs at this point, is analyzed so it can be known which blobs represent cars and, if so, how many. The blob analysis is performed in two ways, one by checking its shape and area and another by tracking the speed of the points of interest of the image.

The result of the project is promising. Out of the 274 vehicles in the video, the area approach misses a single one and counts 17 extras, representing a 6% total error, yet the speed approach gets it wrong 18% of the time. A calibrating phase is necessary so that the approach can work with videos that were recorded from different angles.

Keywords: Vehicle Counting. Contours Analysis. Sparse Optical Flow. Cluster.

LISTA DE FIGURAS

FIGURA 1 – Exemplo de aplicação da dilatação 14
FIGURA 2 – Exemplo de aplicação da erosão 14
FIGURA 3 – Exemplo de aplicação do fechamento. 15
FIGURA 4 – Exemplo de pontos bons e ruins a serem seguidos. 18
FIGURA 5 – Fluxo Ótico de Lucas e Kanade em uma dimensão 20
FIGURA 6 – Pirâmide de imagem. 22
FIGURA 7 – Exemplo de execução do k-means. 23
FIGURA 8 – Exemplo da execução do algoritmo do Fecho Convexo. 24
FIGURA 9 – Sistema de Visão Computacional 26
FIGURA 10 – Exemplo de limitação de região de interesse. 28
FIGURA 11 – Diferença de Imagens com módulo e limiar. 28
FIGURA 12 – Exemplo de junção de blobs devido à operação de fechamento. 29
FIGURA 13 – Situação de "buraco" entre dois blobs 30
FIGURA 14 – Exemplo de utilização de máscaras de dilatação e erosão perpendiculares entre si. 30
FIGURA 15 – Dois veículos representados em único blob. 31
FIGURA 16 – Dois blobs representando um único veículo. 32
FIGURA 17 – Oclusão e não oclusão dentro do polígono convexo. 32
FIGURA 18 – Fecho Convexo contorno externo e contorno em árvore. 33
FIGURA 19 – Aplicação do fecho convexo em blob. 34
FIGURA 20 – Buraco em blob. 34
FIGURA 21 – Blobs a serem segmentados de um frame. 35
FIGURA 22 – Exemplo de Labeling. 35
FIGURA 23 – Comparação de velocidades de carros muito próximos. 37
FIGURA 24 – Resultado errado do fluxo ótico. 37
FIGURA 25 – Linha de referência de contagem. 39
FIGURA 26 – Exemplo de contagem com linha de referência. 39
FIGURA 27 – Exemplo de erro de intersecção da linha de referência com meio do quadrado. 40
FIGURA 28 – Região de Contagem. 40
FIGURA 29 – Contagem correta em situação de carro separado em dois blobs. 41
FIGURA 30 – Aplicação de máscara redondas de raio 3. 42
FIGURA 31 – Aplicação de máscara elíptica de eixo maior 5 e eixo menor 3. 43
FIGURA 32 – Aplicação de máscara elíptica de eixo maior 5 e eixo menor 3 seguido de erosão. 43
FIGURA 33 – Diferentes limiares para fragmentação. 44
FIGURA 34 – Rotulação aplicado em diferentes áreas. 44
FIGURA 35 – Modelo de desenvolvimento em espiral. 52
FIGURA 36 – Desenvolvimento baseado em componentes. 53
FIGURA 37 – Componentes prontos do OpenCV. 54
FIGURA 38 – Diagrama de Classes do software. 55

LISTA DE QUADROS

QUADRO 1 – Quadro com o resultado da aplicação do software em vários vídeos. Abordagem baseada nas áreas 45
QUADRO 2 – Quadro com o resultado da aplicação do software em vários vídeos. Abordagem baseada nas velocidades 45

SUMÁRIO

1 INTRODUÇÃO 10
1.1 PROPÓSITO 11
1.2 OBJETIVOS 11
1.2.1 Objetivo Geral 11
1.2.2 Objetivos Específicos 11
2 FUNDAMENTAÇÃO TEÓRICA 12
2.1 IMAGEM DIGITAL 12
2.2 SUBTRAÇÃO DE FUNDO 13
2.3 MORFOLOGIA MATEMÁTICA 13
2.4 ROTULAÇÃO DE COMPONENTES CONEXOS 15
2.5 GOOD FEATURES TO TRACK (BONS PONTOS A SEREM SEGUIDOS) 16
2.6 OPTICAL FLOW (FLUXO ÓTICO) - MÉTODO DE LUCAS E KANADE 18
2.7 AGRUPAMENTO DOS PONTOS 22
2.8 FECHO CONVEXO 24
3 DESENVOLVIMENTO 25
3.1 FLUXO GERAL DO SOFTWARE 25
3.2 DETERMINAÇÃO DA REGIÃO DE INTERESSE 27
3.3 PROCESSAMENTO INICIAL 27
3.4 BONS PONTOS A SEREM SEGUIDOS E FLUXO ÓTICO DE LUCAS E KANADE 30
3.5 AGRUPAMENTO PELA FORMA (FECHO CONVEXO) 31
3.6 ROTULAÇÃO DE SEGMENTAÇÃO 34
3.7 AGRUPAMENTO PELA VELOCIDADE 36
3.8 REGIÃO DE CONTAGEM 38
4 RESULTADOS 42
4.1 DILATAÇÃO E EROSÃO 42
4.2 FECHO CONVEXO 42
4.3 ROTULAÇÃO 43
4.4 RESULTADOS DE CONTAGEM 43
5 CONSIDERAÇÕES FINAIS 46
5.1 TRABALHOS FUTUROS 46
REFERÊNCIAS 48
Apêndice A – METODOLOGIA DE DESENVOLVIMENTO 50
A.1 PROJETO DE SOFTWARE 50
A.1.1 Requisitos Funcionais 50
A.1.2 Requisitos Não Funcionais 51
A.2 DESENVOLVIMENTO EM ESPIRAL 51
A.3 ENGENHARIA DE COMPONENTES 52
A.4 PROJETO DE SOFTWARE 53
A.5 DIAGRAMA DE CLASSES 54
Anexo A – LICENÇA DO OPENCV (BSD) 56

1 INTRODUÇÃO

O trânsito é um lugar de importantes interações sociais, as quais ficam cada vez mais complexas devido, dentre outros fatores, ao aumento do número de pessoas e da facilidade para se adquirir veículos. De forma a acompanhar esta mudança, é fundamental possuir o conhecimento de informações tal como o fluxo de veículos, para possibilitar que o posicionamento de itens tais como semáforos, placas de informação, definição de sentido e faixas de pedestres seja corretamente planejado. A utilização de ferramentas computacionais pode ser muito útil para tal fim, permitindo a realização de tarefas tais como estimativa de velocidade e classificação de veículos em uma via.

Este trabalho de conclusão de curso busca auxiliar a captação de informações de fluxo por meio da contagem de veículos em vias urbanas. O projeto desenvolvido analisa o conteúdo de vídeos captados a partir de câmeras fixas, gerando como resultado o número de veículos que trafegam na região de interesse em dado intervalo de tempo.

A literatura acerca da problemática de contagem de veículos é vasta, existindo diferentes abordagens, cada qual geralmente acompanhada por pontos fortes e pontos fracos. Os trabalhos (CUNHA, 2013) e (RUAS; BENSO, 2006) comparam diversas técnicas de segmentação, concluindo que uma combinação das técnicas de média móvel e moda móvel é a melhor técnica. Uma abordagem original aprimorada de segmentação é apresentada em (VARGAS et al., 2010). O trabalho de (COIFMAN et al., 1998) define pontos de interesse na imagem a serem seguidos e define as faixas de trânsito manualmente para atacar o problema.

O projeto desenvolvido baseia-se em diferença de frames, realce de características por erosão e dilatação, definição de direção e sentido por fluxo ótico e separação de blobs por fecho convexo. A contagem é realizada por meio da análise de uma região projetada em cada quadro, devidamente modelada para o tipo de vídeo analisado.

Os resultados de contagem obtiveram acerto de 94% em uma abordagem baseada na análise da área de blobs e 82% em outra baseada na velocidade de pontos de interesse, com baixa quantidade de falsos-positivos e também de falsos-negativos.

Este documento é organizado em cinco capítulos. Neste primeiro, o objetivo do trabalho e seu propósito são apresentados. O segundo capítulo trata da fundamentação teórica, onde são explicados os algoritmos mais relevantes para o entendimento do trabalho aqui documentado. O Capítulo 3 explicita como se deu o processo de desenvolvimento do projeto, incluindo detalhes de implementação e ajuste dos algoritmos utilizados. No quarto capítulo são mostrados e discutidos os resultados obtidos nas etapas mais relevantes do desenvolvimento, bem como o resultado final da contagem de veículos em oito vídeos de um cruzamento de ruas. No Capítulo 5, por fim, as considerações sobre o projeto e seu processo de desenvolvimento são apresentadas, bem como sugestões de abordagens a serem exploradas.

1.1 PROPÓSITO

A área de engenharia de tráfego proporciona um vasto conjunto de possibilidades para a aplicação de conceitos que envolvem o engenheiro de computação. Este ramo proporciona a exploração de conceitos que podem levar a inovações que vão além das do escopo do projeto detalhado neste documento. Além disso, a elaboração de um sistema com aplicabilidade direta em projetos de engenharia, mais ainda em projetos de considerável impacto social, é um fator motivante para aquele que está a ingressar na carreira de engenheiro.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Prover um sistema de contagem de veículos baseado em técnicas de processamento de imagens que funcione de maneira prática, automatizada e robusta (baixo erro em comparação a outros métodos) através de técnicas de processamento de imagens.

1.2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um sistema robusto de detecção de veículos.
- Desenvolver um sistema de contagem de veículos que seguem o mesmo sentido.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são detalhados a teoria e os algoritmos utilizados no desenvolvimento deste trabalho.

2.1 IMAGEM DIGITAL

Uma imagem digital pode ser descrita como uma matriz bidimensional de valores inteiros. O espaço e a intensidade dos componentes das cores são discretizados para fins de armazenamento. É comum representar a intensidade como um inteiro de 8 bits (1 byte) que permite valores entre 0 e 255, onde valores próximos de 0 são escuros e valores próximos a 255 são claros. Essa dimensão de valores é geralmente a precisão permitida pela câmera (SHAPIRO; STOCKMAN, 2001) e também é conveniente para ser processada pelo computador.

Uma imagem colorida é uma matriz na qual cada posição tem um vetor com três valores. Matematicamente escreve-se:

$$M_{x,y} = [v1_{x,y}, v2_{x,y}, v3_{x,y}] \tag{1}$$

com

$$0 \leq x < \text{largura} \tag{2}$$

$$0 \leq y < \text{altura} \tag{3}$$

$$0 \leq v1, v2, v3 \leq 255. \tag{4}$$

Usando a notação tricromática *Red*, *Green*, *Blue* (RGB - Vermelho, Verde e Azul) com os limites estabelecidos na Equação 4, é possível a codificação de dezesseis milhões de cores, quantidade muito superior à capacidade do olho humano de distinguir tonalidades (SHAPIRO; STOCKMAN, 2001).

Um outro formato conveniente para aplicações de algoritmos de visão computacional é o sistema de cores Hue-Saturation-Intensity (HSI, Tonalidade-Saturação-Intensidade), que

separa a informação da cor em matiz, saturação e intensidade. Essa representação da cor pode ser calculada a partir dos valores RGB e possui diversas serventias como, por exemplo, na detecção de sombras, que modificam todos os componentes da representação RGB mas variam apenas o componente I da representação HSI.

2.2 SUBTRAÇÃO DE FUNDO

Em um vídeo capturado por uma câmera estática, o *fundo* é a região das imagens na qual não ocorrem eventos significativos. O conceito de fundo não tem uma definição rigorosa, e pode variar dependendo da aplicação. Regiões estáticas ou com objetos movimentando-se de forma periódica podem ser consideradas como fundo. Por outro lado, em uma rua de grande movimentação, o fluxo médio dos carros talvez deva ser considerado como fundo (BRADSKI; KAEHLER, 2008), isto ocorre especialmente no problema de encontrar veículos que estacionam em uma via e possivelmente precisam de algum socorro.

Duas imagens representadas por matrizes de números podem ser subtraídas resultando em uma nova imagem. Tal operação, conhecida como diferença de quadros, é especialmente útil para se descobrir em quais pontos duas imagens são diferentes. Em uma situação de movimentação de objetos, por exemplo, ao se considerar uma fonte de aquisição estática, a diferença de quadros indica os pontos em movimento.

Os modelos de subtração de fundo apresentados nesta seção funcionam relativamente bem, mas assumem que cada ponto da imagem é independente dos demais. Para considerar os pontos próximos de cada coordenada mais recursos computacionais seriam necessários. Devido a esse custo extra, em geral modelos mais complexos, como análise da textura, são evitados e a eliminação de falsos positivos é feita através de operações de processamento de imagens como as operações da morfologia matemática definidas na seção a seguir (BRADSKI; KAEHLER, 2008).

2.3 MORFOLOGIA MATEMÁTICA

A Morfologia Matemática envolve técnicas úteis para descrever e representar o formato de uma região (ANDRADE, 2011). As operações fundamentais da morfologia matemática são a erosão e a dilatação e permitem alterar o formato de imagens binárias.

Dilatar uma imagem binária A a partir de uma segunda imagem B resulta em aplicar a união de diversas imagens B transladadas em todos os pontos da imagem A. Uma definição

rigorosa, baseada na teoria dos conjuntos, é feita em (DORINI, 2009). A Figura 1 exemplifica a aplicação da dilatação em uma imagem binária. Ao final da dilatação uma imagem maior, mais "grossa" é obtida.

As diversas imagens B transladadas por toda a imagem A do parágrafo anterior são definidas como elementos estruturantes. Para extrair características da forma e estrutura, o elemento estruturante modela a imagem na qual se está operando.

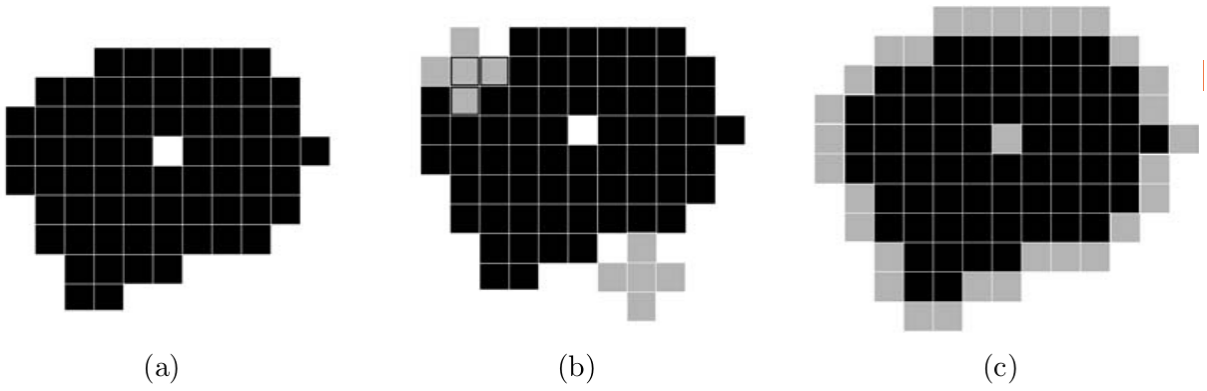

Figura 1: Aplicação da dilatação sobre uma imagem. Em (a) é exibida a imagem original. Em (b) o elemento estrutural é apresentado em diversas posições. Em (c) a imagem resultante é mostrada (onde os pontos em cinza são os novos pontos criados).

Fonte: (DORINI, 2009).

A erosão, por sua vez, remove de uma imagem binária diversos elementos estruturantes transladados nas posições de fundo da primeira imagem. A Figura 2 demonstra o resultado da erosão.

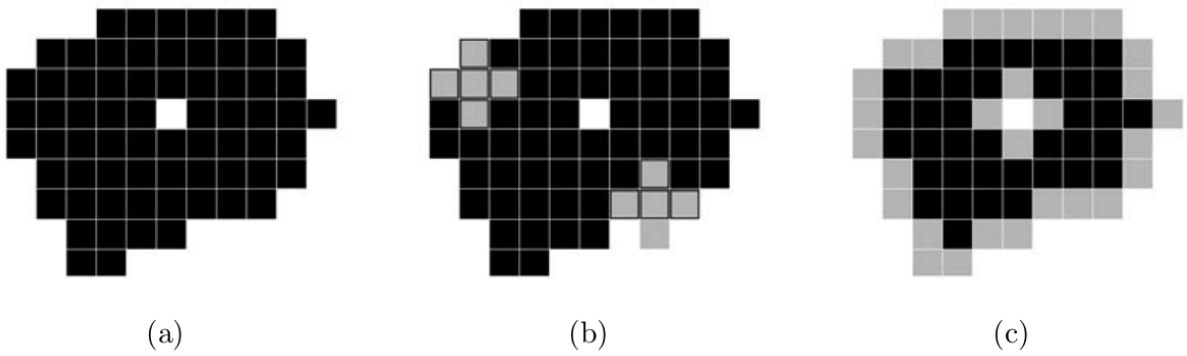

Figura 2: Aplicação da erosão sobre uma imagem. Em (a) é exibida a imagem original. Em (b) o elemento estruturante é apresentado em diversas posições. Em (c) a imagem resultante é mostrada (onde os pontos em cinza são os pontos retirados).

Fonte: (DORINI, 2009).

Aplicar uma dilatação seguida de uma erosão é chamada de operação de fechamento, geralmente o fechamento funde regiões próximas, elimina pequenos orifícios e preenche irregularidades no contorno (FILHO; NETO, 1999). A Figura 3 exemplifica a aplicação do fechamento.

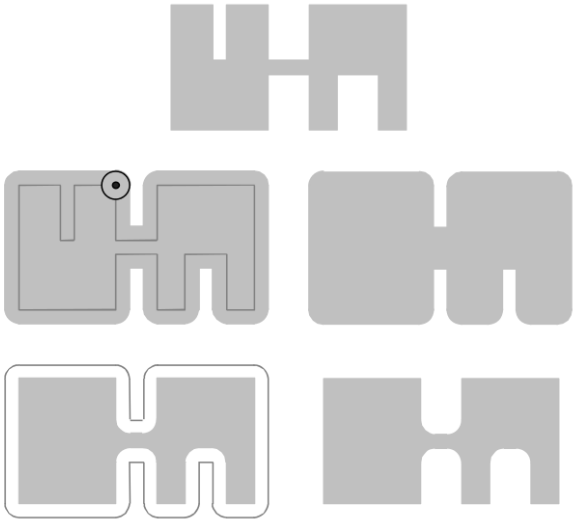

Figura 3: Exemplo da aplicação do fechamento em uma forma. A forma mais acima é o formato original, na linha central é aplicado a dilatação e na linha mais de baixo é aplicada a erosão sobre a imagem dilatada.

Fonte: Adaptado de (FILHO; NETO, 1999).

2.4 ROTULAÇÃO DE COMPONENTES CONEXOS

A rotulação de componentes conexos de uma imagem binária consiste em atribuir um identificador para cada ponto da imagem de tal forma que pontos conectados possuam o mesmo identificador. Existem diversas implementações deste algoritmo, baseadas em abordagens tais como percorrer a imagem através de uma busca em profundidade ou processar apenas duas linhas da imagem por vez utilizando a estrutura de dados denominada *Union-find* (SHAPIRO; STOCKMAN, 2001). Ambas possuem a mesma ordem de complexidade, mas aquela baseada na busca em profundidade é mais simples de ser implementada e pode ser adaptada mais facilmente para percorrer todos os pontos de uma região não retangular. Existem ainda técnicas baseadas em processamento paralelo (SHAPIRO; STOCKMAN, 2001).

A implementação da rotulação exige espaço de memória suficiente para registrar um identificador para cada pixel. As implementações comuns da rotulação visam processar diversos blobs de uma só vez, mas algumas vezes é necessário contar o número de elementos pertencentes à apenas um blob específico, ou buscar uma característica dos pontos de um mesmo

blob. Nestas situações, a versão do algoritmo baseada na busca em profundidade se torna eficaz. Processar os blob's assim que são encontrados torna possível, por exemplo, não reservar um espaço de memória para indicar o rótulo de cada ponto.

2.5 GOOD FEATURES TO TRACK (BONS PONTOS A SEREM SEGUIDOS)

O título desta seção é inspirado no artigo com o mesmo nome e também na função pronta do OpenCV `goodFeaturesToTrack()` que seleciona bons pontos a serem seguidos em imagens. A função de seguir os pontos está explicada na seção a seguir. Aqui será explicado detalhadamente o processo de selecionar os pontos para serem seguidos.

O processo construído de detecção de pontos a serem seguidos começa com a definição de Moravec (HARRIS; STEPHENS, 1988). O detector de vértices de Moravec percorre a imagem e para cada janela determina a mudança média de intensidade ao deslizar um pouco a região em várias direções. Ao ponto central de cada região analisada atribui-se um valor. Temos três situações: análise sobre uma região lisa, sobre uma borda e sobre um vértice. Uma região lisa resulta em pequenas mudanças. Sobre uma borda a mudança de intensidade será grande em uma direção, mas pequena em outra. Por fim, sobre um vértice ou um ponto isolado, haverá uma grande variação na intensidade em diversas direções. Matematicamente isto significa:

$$E_{x,y} = \sum_{u,v} w_{u,v} \left| I_{x+u,y+v} - I_{x,y} \right|^2 \tag{5}$$

Onde x e y são as coordenadas de um pixel específico da imagem e u e v são as coordenadas relativas de uma janela no qual as intensidades são comparadas. A letra I simboliza a intensidade e w é um peso dado para cada elemento da janela de comparação sendo 1 deslocando o centro para um dos quatro vizinhos e 0 nas demais posições.

Existem três melhorias sugeridas em (HARRIS; STEPHENS, 1988):

1. Considerar a variação da intensidade em todas as direções através da fórmula:

$$E_{x,y} = A \left[I_{x-1,y}, I_{x,y}, I_{x+1,y} \right]^2 + 2C \left[I_{x-1,y} I_{x,y-1}, I_{x,y} I_{x,y}, I_{x+1,y} I_{x,y+1} \right] + B \left[I_{x,y-1}, I_{x,y}, I_{x,y+1} \right]^{2T} \tag{6}$$

onde

$$A = X^2 \otimes w \tag{7}$$

$$B = Y^2 \otimes w \tag{8}$$

$$C = (XY) \otimes w \tag{9}$$

$$X = I \otimes (-1,0,1) \approx \delta I/\delta X \tag{10}$$

$$Y = I \otimes (-1,0,1)^T \approx \delta I/\delta Y \tag{11}$$

2. Utilizar uma janela circular, como por exemplo uma gaussiana, para evitar ruídos devido ao grande valor dado a elementos longes do ponto sendo analisado. Na notação usada:

$$w_{u,v} = e^{-(u^2+v^2)/2\sigma^2} \tag{12}$$

3. Usar medidas que relacionem bordas em direções distintas evita considerar uma borda como região de interesse. O artigo de mesmo título desta seção (SHI; TOMASI, 1994) propõem calcular o valor do gradiente de segunda ordem em X, Y e na diagonal e calcular o menor autovalor da matriz $\begin{bmatrix} A & C \\ C & B \end{bmatrix}$, esta solução é melhor pois não necessita de um parâmetro a mais para classificar quando um ponto é bom.

É importante entender o significado do menor autovalor na matriz criada. Desenvolvendo a definição de autovalor encontra-se:

$$det\left(\begin{bmatrix} A-\lambda & C \\ C & B-\lambda \end{bmatrix}\right) = 0 \tag{13}$$

$$(A-\lambda)(B-\lambda) = CC \tag{14}$$

$$(A-\lambda)(B-\lambda) = C^2 \tag{15}$$

$$\lambda^2 - \lambda(A+B) + A^2B^2 - C^2 = 0 \tag{16}$$

$$\lambda_{menor} = \frac{A+B-\sqrt{(A+B)^2 - 4(A^2B^2 - C^2)}}{2} \tag{17}$$

Na Equação 17 primeiro deve-se verificar se a raiz quadrada tem valor no domínio dos reais. Para isto o valor de C deve ser grande, segundo (SHI; TOMASI, 1994) nos pontos desejados essa raiz é calculável. Essa equação representa uma pontuação e é mais alta quanto forem altos os valores do gradiente em X e Y.

Segundo (BRADSKI; KAEHLER, 2008), uma vantagem de usar a derivada de segunda ordem é que ela não responde a gradientes uniformes. A figura 4 exemplifica pontos bons e ruins para serem seguidos em um carro.

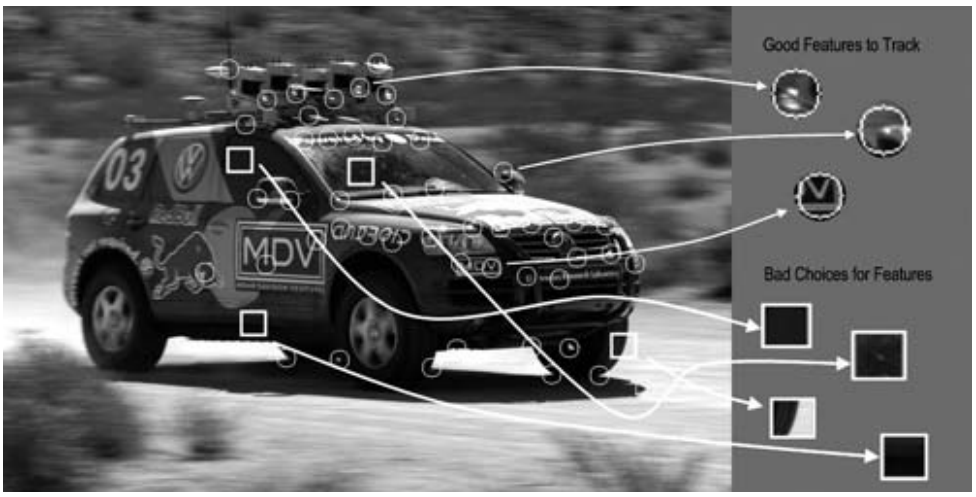

Figura 4: Os bons pontos a serem seguidos são vértices e apresentam mudança de gradiente em várias direções. Já os pontos ruins de serem seguidos são regiões sem gradiente ou com gradiente em apenas uma direção.

Fonte: (BRADSKI; KAEHLER, 2008).

2.6 OPTICAL FLOW (FLUXO ÓTICO) - MÉTODO DE LUCAS E KANADE

Encontrar o movimento (direção, sentido e velocidade) de objetos em imagens de vídeos permite manter um histórico das posições dos pixels, de forma a encontrar suas trajetórias e contar quantos objetos passaram por determinada região.

As técnicas de fluxo ótico esparso permitem encontrar o movimento de regiões específicas considerando apenas sua vizinhança na imagem. Desta forma o processo tende a ser mais rápido comparado à abordagem do fluxo ótico denso, que define o movimento de todas as regiões da imagem e analisa a imagem inteira. A movimentação de um objeto com cor homogênea muda apenas os pixels de sua borda. Este fato dificulta encontrar o movimento de tais pixels. Para evitar estas regiões, usa-se a abordagem para selecionar apenas bons pontos a serem seguidos.

O método de Lucas e Kanade é uma técnica de fluxo ótico esparso que analisa apenas as regiões muito próximas ao pixel, junto a uma técnica que analisa a imagem em uma resolução menor que, ao ter sua resolução aumentada sucessivamente, encontra deslocamentos maiores, a cada passo refinando o resultado e tornando este procedimento importante e efetivo (BRADSKI; KAEHLER, 2008) para rastrear pontos.

O método original de Lucas e Kanade (LUCAS; KANADE, 1981) assume que:

1. o brilho do ponto seguido se mantém constante;

2. os oito pontos vizinhos ao ponto seguido se movem junto;

3. haverá pequenas movimentações entre os quadros consecutivos dos vídeos.

Essas hipóteses são aceitáveis. A primeira delas, brilho constante, sugere que o ponto continue parecido no decorrer do tempo. Em forma de equação isto é escrito como:

$$f(x,t) \equiv I(x(t),t) = I(x(t+dt),t+dt) \tag{18}$$

Assumir uma vizinhança movimentando-se junto ao ponto implica em uma diferença constante de intensidade entre os pontos. Mesmo que a imagem seja rotacionada e transladada a diferença dos pontos poderá ser usada para determinar para onde o ponto foi. O uso desta propriedade será descrito mais detalhadamente adiante.

A última presunção espera que haja pouca movimentação da imagem. Ainda que posteriormente seja aplicada uma técnica para reduzir essa limitação, a busca procura o ponto desejado nas proximidades por um tipo de Newton-Rapson e se a estimativa inicial for muito errada (devido a uma movimentação grande na imagem, por exemplo) o algoritmo pode convergir para um ponto errado. Pode-se observar na Figura 5 o processo similar as aproximações sucessivas do método de Newton-Rapson.

A Figura 5 apresenta, sem perda de generalidade em imagens unidimensionais, uma região de uma imagem. Para encontrar o ponto desejado, primeiro verifica a intensidade na ultima localização conhecida do ponto procurado na imagem nova e calcula através da velha qual posição que o ponto procurado deve estar.

Na Figura 5a deseja-se determinar quanto o ponto seguido da imagem foi deslocado. A solução de Lucas e Kanade baseia-se em aproximações lineares progressivas, a estimativa de movimentação em cada passo é feita a partir da fórmula

$$F'(x) \approx \frac{F(x+h)-F(x)}{h} = \frac{G(x)-F(x)}{h} \tag{19}$$

então

$$h \approx \frac{G(x)-F(x)}{F'(x)} \tag{20}$$

Na equação 20 o $F(x)$ é a intensidade do ponto na imagem prévia, $G(x)$ é o valor da intensidade na imagem analisada na mesma coordenada do ponto na imagem prévia e $F'(x)$ é

calculado pela fórmula

$$F'(x) \approx \frac{F(x+\delta x) - F(x)}{\delta x} \tag{21}$$

Onde δx é apropriadamente pequeno, um ponto por exemplo.

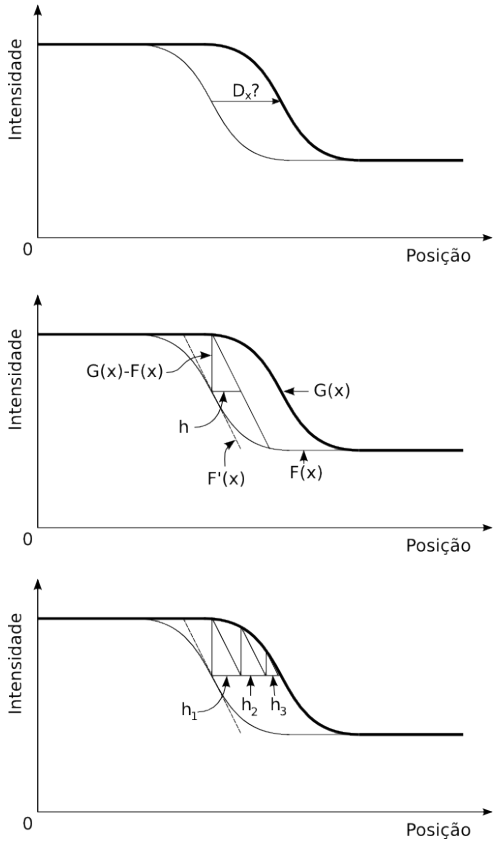

Figura 5: Representação gráfica do Fluxo Ótico de Lucas e Kanade em uma dimensão. A representação gráfica é muito similar à representação gráfica do Newton-Rapson.

Fonte: Autoria Própria.

Para generalizar para mais dimensões é conveniente escrever a Equação 20 como em 22 e sua forma iterativa como em 23 e 24. Nestas equações w(x) representa um peso e é definido em 28.

$$h = \sum_x \frac{w(x)[G(x) - F(x)]}{F'(x) \sum_x w(x)} \tag{22}$$

$$h_0 = 0 \tag{23}$$

$$h_{k+1} = h_k + \sum_x \frac{w(x)[G(x) - F(x+h_k)]}{F'(x+h_k)\sum_x w(x)} \tag{24}$$

Por analogia à Equação 22, pode-se escrever a equação da movimentação em duas dimensões como um termo indicando a direção multiplicado pela diferença da função no momento atual e no momento anterior dividido pela derivada da função no momento anterior. Em duas dimensões deve-se observar que a função w(x) de peso é substituída pela derivada. A fórmula é escrita em 25 e sua forma iterativa em 26 e 27.

$$h = \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x (F'(x))^2} \tag{25}$$

$$h_0 = 0 \tag{26}$$

$$h_{k+1} = h_k + \frac{\sum_x w(x)F'(x+h_k)[G(x) - F(x+h_k)]}{\sum_x w(x)(F'(x+h_k))^2} \tag{27}$$

$$w(x) = \frac{1}{|G(x) - F'(x)|} \tag{28}$$

Uma propriedade interessante do fluxo ótico de Lucas e Kanade é que ele é uma generalização da medida de diferença de brilho do algoritmo de achar bons pontos a serem seguidos, a Equação 29 tem uma relação direta com a Equação 5. Fica claro a importância das duas primeiras hipóteses assumidas pelo algoritmo, com o brilho constante e a vizinhança constante o valor da intensidade definida na Equação 29 deve-se manter alto e continuar em destaque na região.

$$I = \sqrt{\sum_{u,v} (F(x+u, y+v) - F(x,y))^2} \tag{29}$$

Uma busca extensiva usando a Equação 29 nas proximidades da região que o ponto foi encontrado permitiria obter resultado semelhante ao fluxo ótico. Por isso a performance do método deve ser analisada e comparada, pois se houver demora para convergir, a necessidade de muitos cálculos que não existem na busca extensiva, como as derivadas, possivelmente tornariam o método não eficiente. Quanto maior for a região de busca, maior o processamento da busca extensiva em relação à área procurada e maior o processamento da busca iterativa em relação a linha que liga aos pontos intermediários.

A terceira hipótese do método supõem pouca movimentação do ponto, mas isto depende da velocidade dos objetos na cena e da frequência da aquisição das imagens pela câmera. Uma solução é processar a área em uma imagem reduzida e aumentar o tamanho da imagem aos poucos, desta forma as buscas iniciais podem encontrar grandes movimentações. Essa técnica é denominada pirâmide de imagem e está ilustrada na Figura 6.

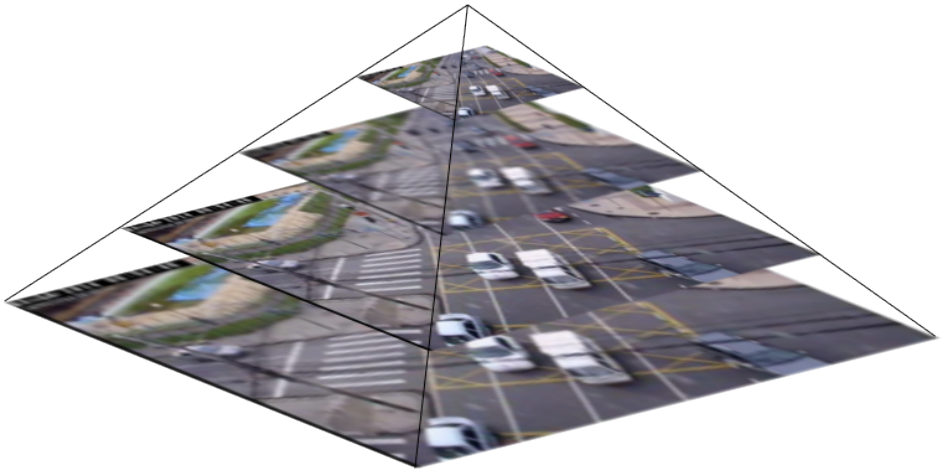

Figura 6: Aplicação da pirâmide de imagem em um quadro do vídeo dos carros. Na parte superior há a imagem reduzida quatro vezes, seguida da imagem reduzida pela metade, uma reduzida 75% e finalmente a imagem no tamanho original. As imagens são processadas da menor para a maior.

Fonte: Autoria Própria.

2.7 AGRUPAMENTO DOS PONTOS

Os pontos seguidos pela técnica de fluxo ótico tem um histórico formado determinando o percurso feito, a velocidade média em qualquer período de tempo. Além destas características a última posição do ponto analisado e a cor da imagem neste ponto são armazenados. Com todos estes dados é possível agrupar os pontos com características parecidas e caracterizar um veículo.

Será apresentada a técnica *K-Means* (K-médias), por ser uma das técnicas de agrupamento mais usadas (BRADSKI; KAEHLER, 2008), além de uma modificação do algoritmo clássico voltado para contar carros. Este procedimento tenta agrupar os elementos estipulando k centros e distribuindo os pontos nestes centros. O pseudo-código do algoritmo é listado a seguir:

1. Recebe a lista dos dados e o número de grupos k (escolhido pelo usuário, ou determinado pelo método proposto em).

2. Aleatoriamente define os centros dos grupos.
3. Associa cada ponto com o centro de agrupamento mais próximo.
4. Atualiza a posição dos centros levando em consideração o centroide de todos os elementos associados aos respectivos centros.
5. Retorna ao passo 3 até que os centros parrem de se mover.

A Figura 7 exemplifica a execução do algoritmo com pontos bidimensionais em uma situação em que converge em quatro iterações.

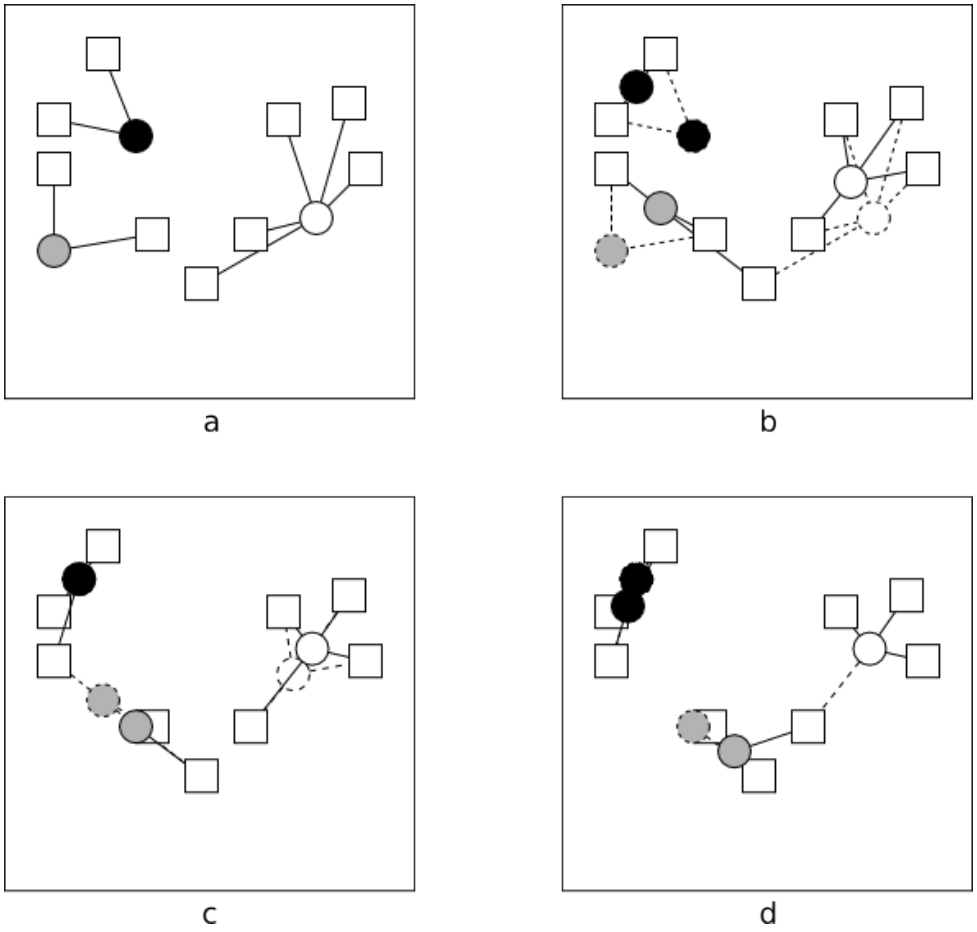

Figura 7: Execução do algoritmo k-means em um ambiente com três centros (representados por circulos) e nove pontos (representados por quadrados). a) são atribuídos os pontos aos centros mais próximos. b,c) novas posições dos centros e novos agrupamentos. d) os agrupamentos convergiram, em novas iterações os agrupamentos continuarão constantes.

Fonte: Autoria Própria.

A realocação gradativa dos centros no algoritmo do K-Means minimiza a Equação 30. Esta equação é a soma de todas as distâncias ($d(p,C_i)$) dos pontos (p) aos seus respectivos

centros (C_i) e determina o erro (E). O erro é uma forma, entre muitas (HALKIDI et al., 2001), de validar a qualidade de um agrupamento. Outra forma inclui medir a distância do ponto mais distante ao seu respectivo centro de agrupamento.

$$E = \sum_{i=1}^{C} \sum_{p \in C_i} d(p, C_i) \tag{30}$$

2.8 FECHO CONVEXO

A técnica do Fecho Convexo define um polígono que envolve todo um blob. É uma técnica útil para se encontrar a forma de um objeto (blob) na imagem. Uma das formas de analisar os blob's é através das área defeitos (regiões compreendidas entre o polígono e o objeto).

O algoritmo para formar o Fecho Convexo é ilustrado na Figura 8. O algoritmo começa adicionando os dois pontos mais distantes do blob a uma lista que formará o polígono. Em seguida novos pontos são adicionados a essa lista até que o blob esteja completamente dentro do polígono. A escolha do próximo ponto a adicionar é feita encontrando-se o ponto do blob mais distante do polígono formato até o momento. A inserção destes pontos mais distantes formam um polígono convexo, como desejado.

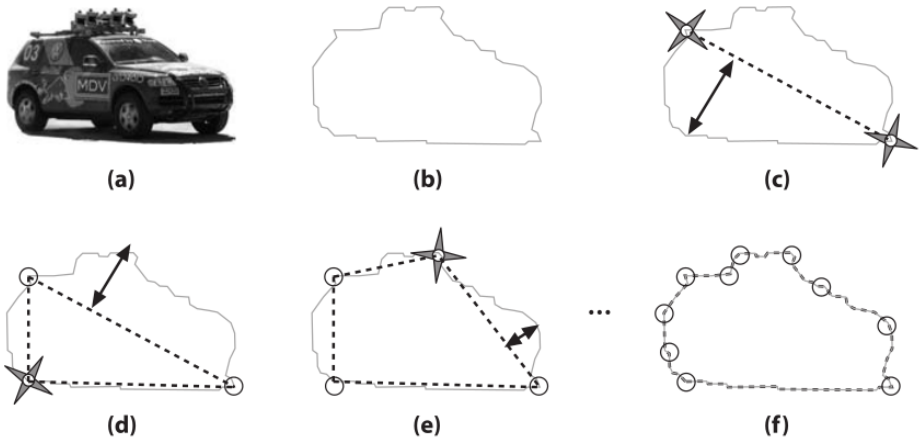

Figura 8: Exemplo da execução do algoritmo do Fecho Convexo em um carro. Em (a) esta a imagem original e em (b) somente seus limites. Em (c) são definidos as posições dos dois pontos mais distantes. Em (d) até (f) é adicionado o ponto mais distante ao polígono calculado até que todo o blob esteja dentro do Fecho.

Fonte: (BRADSKI; KAEHLER, 2008).

3 DESENVOLVIMENTO

Esse capítulo relata o processo de desenvolvimento do projeto, incluindo a forma de obtenção os vídeos, tecnologia utilizada e parametrização dos algoritmos utilizados.

3.1 FLUXO GERAL DO SOFTWARE

Sistemas de Visão Computacional são processos capazes "de adquirir, processar e interpretar imagens correspondentes a imagens reais" (FILHO; NETO, 1999). A Figura 9 mostra as etapas em que foram divididas as principais funções da abordagem proposta.

A primeira etapa do processo consiste na aquisição das imagens por meio de uma câmera digital e representa todo processo de obtenção de dados. O escopo do trabalho determina que tais imagens serão obtidas a partir de câmeras fixas posicionadas em lugares suficientemente altos de modo a obter imagens de todas as faixas e uma extensão suficientemente grande para analisar diversos quadros antes de confirmar a identificação do veículo. Além disso, com uma câmera fixa, o sistema não terá que segmentar a pista automaticamente, pois como ela estará sempre na mesma posição relativa à câmera, isso poderá ser feito manualmente.

Conseguiu-se junto a empresa Urbanização de Curitiba S/A (URBS) mais de cinco horas de gravações distribuídas em diversas câmeras de trânsito e que atendem aos requisitos do projeto. O contato para a obtenção dos vídeos se deu em uma visita técnica, organizada pela professora Keiko Verônica Ono Fonseca, ao Centro de Controle de Operações da URBS, onde foi também observado o processo de gravação os vídeos. O sistema de câmeras da URBS utiliza um formato proprietário para armazenar os vídeos e fornece um programa de conversão que converte para qualquer formato com CODEC disponível no computador. O formato de vídeo original tem extensão .dat e é transformado para o formato MPEG4.

A segunda etapa tem o objetivo de determinar as regiões de interesse e salientar algumas características da imagem. Visando reduzir o tempo de execução, primeiro é selecionada uma região da imagem que permita a completa resolução do problema apenas com

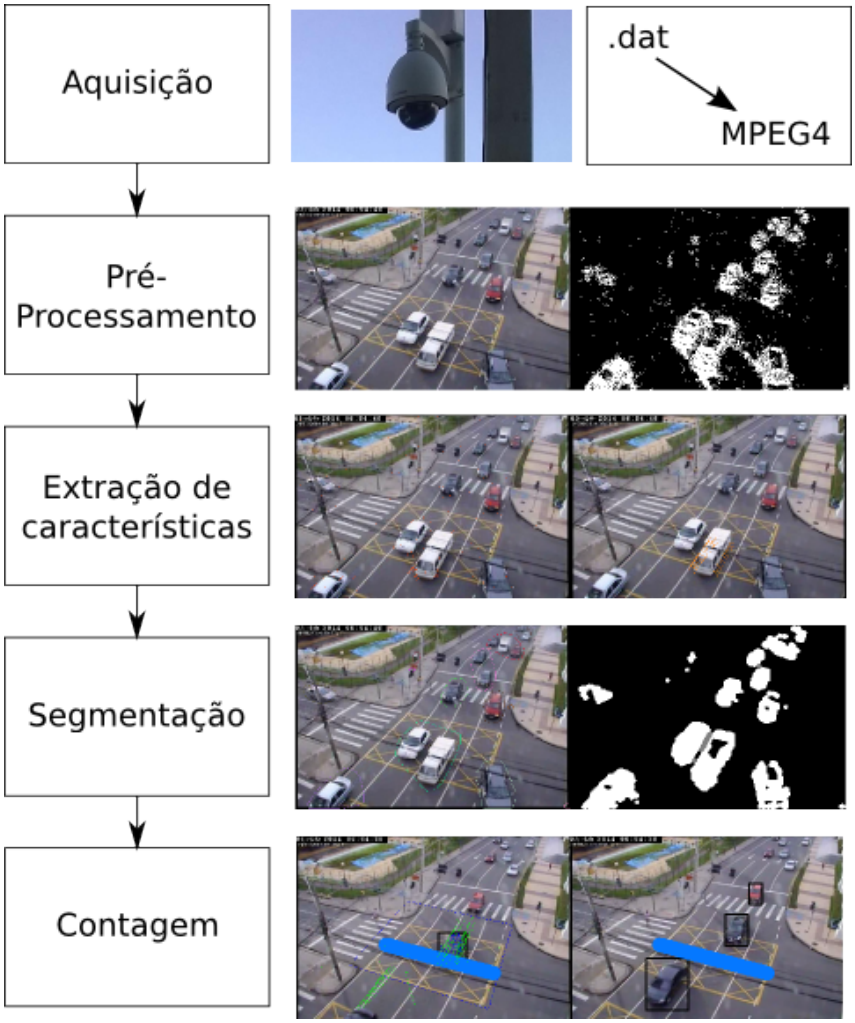

Figura 9: Divisão das funções a serem processadas pelo software contador de carros.
Fonte: Autoria Própria.

as informações nela contida. Uma segunda etapa consiste em converter a imagem para preto e branco, pois tratar um único inteiro representando a tonalidade do cinza de uma imagem é mais rápido que tratar três inteiros representando a combinação das cores vermelho, verde e azul. Além disso, diversos algoritmos são definidos apenas para imagens binárias ou em níveis de cinza.

Há uma primeira segmentação de veículos realizada junto com a etapa anterior baseada na diferença entre imagens consecutivas. Este processo divide os pontos da imagem como em movimento ou parados. Assim como na etapa anterior, esse processo auxilia na execução mais rápida dos algoritmos na sequência. Os pontos em movimento formando um componente conexo representam potencialmente um ou mais veículos.

Após a subtração de imagens, na terceira etapa, deseja-se realçar algumas

características da mesma através das operações morfológicas de dilatação e erosão. Tais operações tem a função de juntar blobs que pertencem ao mesmo veículo, de modo a facilitar a segmentação em passo futuro.

Uma vez realizada as operações morfológicas, tem-se a etapa de definição de sentido de fluxo, através da técnica de fluxo ótico e depois separação de blobs, que tem o objetivo de fragmentar os blobs quando estes representam mais de um veículo. Em seguida aplica-se operação para identificar os componentes conexos, chamada rotulação (*labeling*). Esse dado informa também o número de pixels em cada grupo e as coordenadas do retângulo que envolve todo o blob.

Por fim, a contagem dos veículos é feita sempre que os retângulos que os representam ultrapassem uma região de contagem. O sistema apresenta a imagem com os veículos separados e informa a contagem durante a execução do vídeo, a exibição do vídeo permite detectar quando os veículos foram corretamente contados, para fins de avaliação.

Todo o trabalho foi desenvolvido em C++ com o uso da biblioteca OpenCV, uma biblioteca de visão computacional e aprendizado de máquina. As funções disponíveis em tal biblioteca usam as vantagens de processamento paralelo e permitem até mesmo usar os recursos programáveis da placa de vídeo quando disponíveis, além de implementar diversas rotinas de processamento de imagens, o que facilita o teste de algoritmos no projeto.

3.2 DETERMINAÇÃO DA REGIÃO DE INTERESSE

A *Region of Interest* (ROI - Região de Interesse) de uma imagem determina qual área deve ser analisada para que seja possível adquirir todas as informações desejadas. A vantagem de analisar apenas uma região de interesse é a velocidade de execução do programa e a possibilidade de se evitar a detecção de objetos que não correspondam a veículos. O tamanho da região de interesse deve abranger todo o carro. Considerando a perspectiva da câmera, deve-se incluir bordas grandes o suficiente para conter toda a imagem do veículos maiores (caminhões e ônibus, por exemplo) que possam trafegar por ali. A Figura 10 ilustra o resultado das imagens analisadas com e sem a determinação de uma região de interesse.

3.3 PROCESSAMENTO INICIAL

O processamento inicial do trabalho trata-se da aplicação da diferença de quadros seguidos de limiarização e da aplicação de operações de morfologia matemática. O objetivo

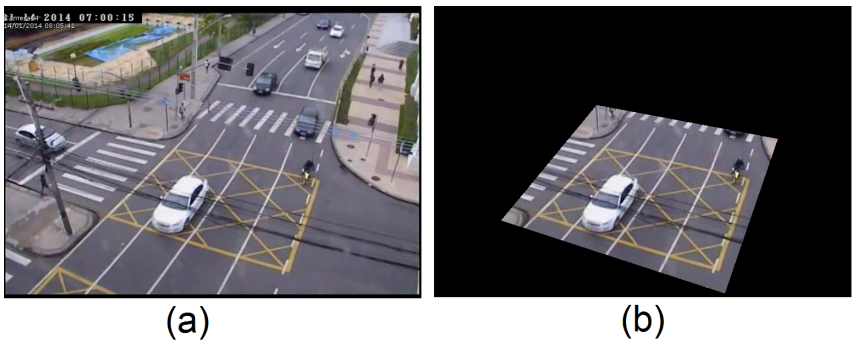

Figura 10: Exemplo de imagem com e sem definição de ROI.

Fonte: Autoria Própria.

é prover para etapa de extração de características e segmentação imagens mais fáceis de serem processadas no contexto de contagem de veículos. De modo a fornecer tais imagens, algumas características devem ser observadas: os blobs que representem o mesmo veículo estarem unidos e os blobs que representem dois veículos ou mais estarem separados.

O módulo diferença de quadros é utilizado para que seja segmentado qualquer objeto que apresente movimento, e é aplicado em conjunto com um limiar para que se tenha uma imagem binária com pixels que representem o objeto veículo. A Figura 11 mostra o resultado da subtração com a operação módulo e um limiar e a Equação 31 resume matematicamente essa operação de subtração. O limiar foi definido a partir de resultados experimentais visando obter todo o veículo, sem sua sombra - o resultado para diferentes valores de limiares é apresentado na Seção 4.

Figura 11: Diferença em módulo entre as imagens seguidas de um limiar.

Fonte: Autoria Própria.

$$I_{y,x,t} = \begin{cases} 255 & ,se|I_{y,x,t} - I_{y,x,t-1}| \geq \text{limiar} \\ 0 & ,se|I_{y,x,t} - I_{y,x,t-1}| < \text{limiar} \end{cases} \tag{31}$$

A etapa seguinte de processamento inicial é a etapa de aplicação das operações de morfologia matemática. Pode-se observar que após a subtração de quadros muitos pixels estão isolados e os blobs não são muito consistentes. A operação de dilatação seguida de erosão gera a operação morfológica de fechamento, muito útil para para deixar os blobs mais densos.

Pode-se observar da Figura 12 que, em algumas situações, a operação de fechamento também faz com que o blob de dois veículos se juntem em um único blob. Tal situação é discutida em mais detalhes na Seção 3.5, mas ainda nesta etapa é possível de se minimizar esse problema.

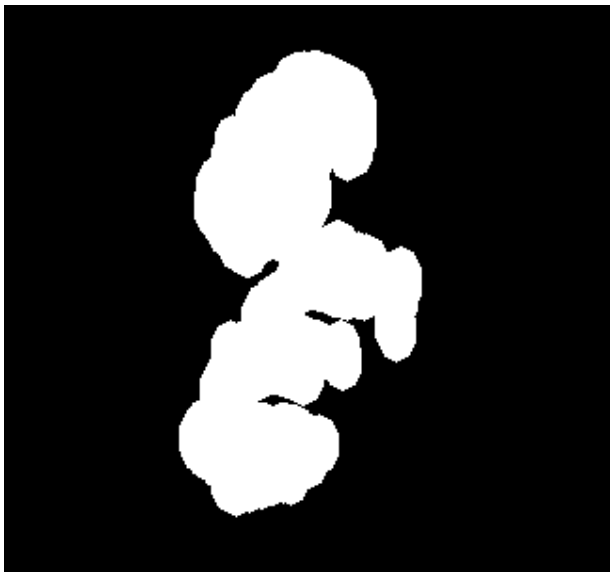

Figura 12: Exemplo de junção de blobs devido à operação de fechamento.
Fonte: Autoria Própria.

O objetivo das operações de morfologia matemática é unir os pixels de um mesmo veículo. Pode-se observar que os blobs gerados muitas vezes representam a parte dianteira e a parte traseira de um veículo, deixando um "buraco" entre esses dois blobs, conforme ilustrado na Figura 13 . A partir disso podemos definir que uma boa operação morfológica seria uma operação que "esticasse" os blobs no sentido de locomoção do automóvel. Qualquer operação morfológica, porém, acarreta em eventuais junções indesejadas de blobs. Nesse caso, ao se "esticar" os blobs dianteiro-traseiro para se conectarem, inevitavelmente também seriam conectadas as parte dianteiras de um carro com a traseira de outro. Assim sendo, a operação de fechamento é realizada com dilatação no sentido de locomoção dos veículos e erosão no sentido perpendicular.

Os elementos estruturantes perpendiculares mostraram-se um bom modelo para a aplicação, conforme pode-se observar na Figura 14, pois a dilatação agrupa componentes

Figura 13: Situação de "buraco" entre dois blobs.
Fonte: Autoria Própria.

desconexos distântes verticalmente e a erosão separa horizontalmente os blobs. No Capítulo 4, contudo, pode-se observar a aplicação de outras máscaras nas operações de morfologia.

Figura 14: Exemplo de utilização de máscaras de dilatação e erosão perpendiculares entre si.
Fonte: Autoria Própria.

3.4 BONS PONTOS A SEREM SEGUIDOS E FLUXO ÓTICO DE LUCAS E KANADE

Em regiões consideradas em movimento pelo processamento inicial, pontos a serem seguidos são procurados. Os pontos recebem uma pontuação, como definido na Equação 17, e apenas os pontos com valores maiores que 1% da maior pontuação encontrada e que possuam uma distância mínima de 10 pontos entre si são usados. Na sequência, os pontos são acompanhados na imagem pelo fluxo ótico de Lucas e Kanade. Todos os pontos possuem um histórico com todas as suas posições passadas.

Todos os pontos com suas coordenadas atuais, suas velocidades instantâneas, suas velocidades médias e a mediana do módulo das duas dez últimas velocidades são passados para

o agrupamento de pontos. Todos esses dados são passados para a etapa posterior de desagrupar pela velocidade veículos do mesmo blob. O agrupamento pela forma precisa saber a direção predominante do movimento na imagem.

3.5 AGRUPAMENTO PELA FORMA (FECHO CONVEXO)

Após o processo de subtração de imagens e das operações de dilatação, é comum que dois carros sejam representados em um único blob. Essa situação é indesejada, pois o restante do algoritmo de detecção depende dessa etapa para a correta segmentação. O efeito é direto no resultado final, causando erro na contagem – dois veículos próximos seriam considerados como sendo apenas um.

A determinação do número de veículos representados em um único blob não é uma tarefa simples. Ao criar uma regra para separar um blob em dois, deve-se verificar se tal regra também não separa um único veículo. Tal situação geraria um problema oposto ao anteriormente apresentado – enquanto na primeira situação dois carros são representados em um único blob, contando menos carros, na segunda um único carro seria representado em dois blobs, contando mais carros. As Figuras 15 e 16 ilustram as situações mencionadas.

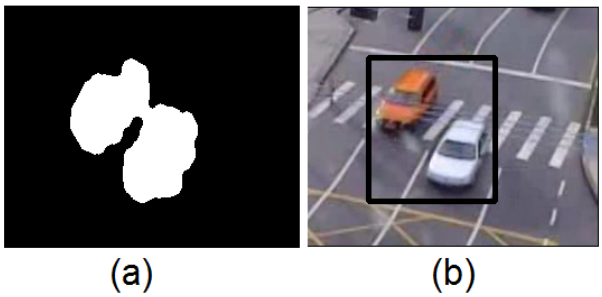

Figura 15: Demonstração de situação onde dois veículos são representados em um único blob e o problema de segmentá-los como um único veículo.

Fonte: Autoria Própria.

Além da regra para determinar a quantidade de carros que estão sendo representados em um certo blob, é necessário definir qual parte do blob pertence a qual veículo. A abordagem de Ahmadzadeh (HEIDARI; AHMADZADEH, 2013), que utiliza-se do algoritmo de fecho convexo, foi utilizado para realizar tal separação.

Heidari e Ahmadzadeh (2013) desenvolveu uma abordagem para identificar oclusões na imagem de modo a fragmentar os blobs em dois, a qual considera que quando não há oclusão a área do blob relativa a área do polígono gerado pelo fecho convexo é grande. Quando há

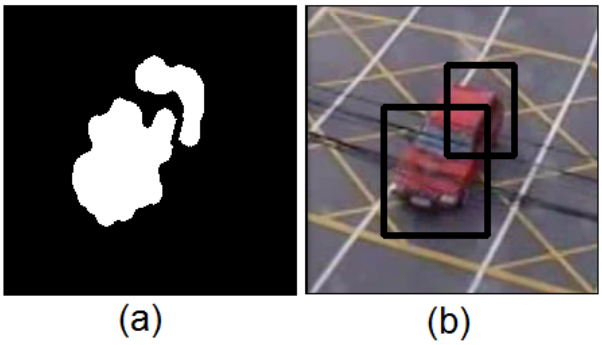

Figura 16: Demonstração de situação onde um único veículos é representado em dois blobs e o problema de segmentá-lo como dois veículos.

Fonte: Autoria Própria.

oclusão, por sua vez, existem grandes "vãos" na segmentação, conforme pode-se observar na Figura 17.

Figura 17: Demonstração de casos de não oclusão e oclusão, respectivamente. A área do blob (parte branca) é maior relativo ao polígono na situação de não oclusão.

Fonte: Autoria Própria.

No OpenCV, a função de fecho convexo retorna um vetor de pontos que representa cada um dos blobs. Esse vetor, ao ter seus pontos conectados, define o menor polígono convexo capaz de envolver determinado blob. O OpenCV permite, ainda, definir o modo de contorno a ser aplicado. O único modo que interessa no contexto deste trabalho é o de contorno externo, pois retorna os ponto do polígono externo ao blob (ao contrário do modo de contorno em árvore, que retorna também os polígono internos de cada blob).

Conhecendo o blob e seu respectivo polígono convexo, pode-se calcular a relação mencionada em (HEIDARI; AHMADZADEH, 2013), que é a área do blob dividida pela área do polígono convexo. Após testes, chegou-se ao valor de 0.85 como limiar de corte, ou seja, caso a área do blob seja menor do que 85% da área do polígono, será realizada a operação de separação em duas regiões. Esse valor de 0.85 foi testado empiricamente e o resultado com outros valores pode ser observado no Capítulo 4. A Figura 17 exemplifica as situações onde o

Figura 18: Diferença de fecho convexo externo e em árvore, respectivamente.
Fonte: Autoria Própria.

blob deve ou não ser fragmentado. Pode-se observar que o polígono que envolve o blob no caso de oclusão agrega também uma considerável área que não pertence ao blob (área cinza). O caso de não oclusão mostra uma situação em que o blob não deve ser fragmentado devido ao fato de a área que não pertence ao blob (denominada área de defeito, pintada de cinza) ser pequena em relação ao polígono.

Nos casos em que a separação se faz necessária, é preciso definir o ponto de fragmentação. Na presença de oclusão é possível explorar a forma dos polígono convexo, a qual apresenta duas áreas de defeito especialmente grandes em relação as outras. Isso acontece porque geralmente a oclusão acarreta na junção de duas partes pequenas (relativas ao tamanho total do blob) dos veículos, ao mesmo tempo em que o fecho convexo define um polígono, que,por ser convexo, cobre todo o conjunto de pontos do blob. Neste trabalho os carros muito próximos são também tratados como oclusão, pois apresentam as mesmas características de formato de blob. Para realizar o "corte" do blob, elimina-se os pontos que pertencem a linha de conexão entre as duas maiores áreas de defeito. Tal linha de conexão é um reta entre os pontos, pertencentes as áreas de defeito, mais distantes das linhas do polígono convexo.

O cálculo para se encontrar a área do polígono convexo é direto, uma vez que o ponto de todos os vértices são conhecidos, pelo método da soma dos determinantes. Para a determinação da área do blob, por sua vez, o cálculo é mais complexo. Tal complexidade vem do fato de existirem alguns "buracos" nos blobs, pois, não sendo áreas de defeito, devem ser considerados como partes blob. A Figura 20 demonstra tal situação.

O OpenCV implementa uma função pronta para fazer a fragmentação do blob conforme Ahmadzadeh, porém não possui função para determinação da área do blob (necessária

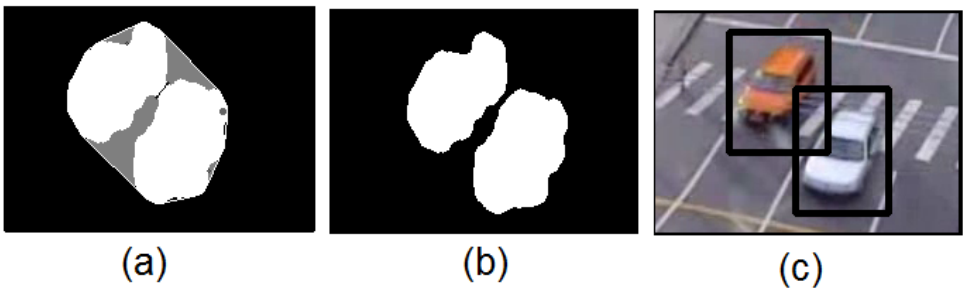

Figura 19: Linha de conexão entre os pontos mais distantes do polígono nas duas maiores áreas de defeito, blob fragmentado em dois e segmentação dos dois veículos presentes, respectivamente.

Fonte: Autoria Própria.

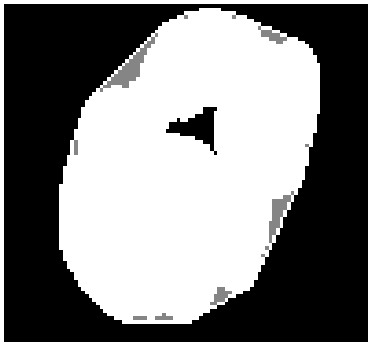

Figura 20: Buraco, em preto, do blob, que deve ser considerado como área do mesmo, sendo diferente das áreas de defeito, em cinza.

Fonte: Autoria Própria.

para decidir se tal fragmentação deve ser realizada). Assim, foi desenvolvida uma função para determinar a área total de defeito e encontrar os pontos mais distantes das retas do polígono. Tal função utiliza-se do mesmo princípio da função de rotulação, retornando, além da área de cada defeito, o ponto mais distante das linhas do polígono. No final, os dois pontos das duas maiores áreas, mais distantes do polígono são conectados de forma a formar a linha de corte do blob. Essa linha é desenhada diretamente na imagem binária contendo os blobs, conforme podemos notar na Figura 19.

3.6 ROTULAÇÃO DE SEGMENTAÇÃO

Uma vez realizado todo o processamento sob a imagem, tem-se a etapa de segmentação e definição do que considerar como veículo. Ao fim das etapas anteriores temos uma imagem com muitos blobs, conforme mostra a Figura 21.

O objetivo da rotulação de segmentação é conseguir a informações do número total de

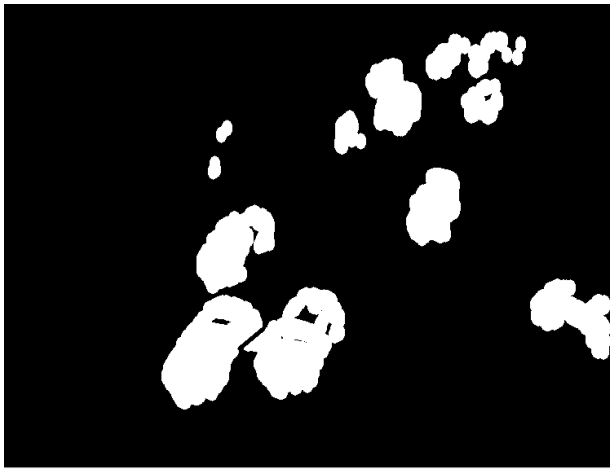

Figura 21: Resultado após a aplicação dos processamentos, faltando apenas segmentação.

Fonte: Autoria Própria.

blobs na imagem, de modo a ser um primeiro indicador do número total de veículos na mesma. Para realizar tal feito é aplicado o algoritmo *Connected Components Labeling*, explicado na Seção 2.4, de forma que cada blob passa a ser representado por um número. Na prática, a rotulação de segmentação retorna uma imagem conforme ilustra a Figura 22.

Figura 22: Resultado após a aplicação da função labeling. Os números representam o valor de cada pixel.

Fonte: Autoria Própria.

É a partir da aplicação da rotulação de segmentação que quadrados são definidos em volta de cada blob. Tal processamento é necessário para que se possa realizar a contagem do número de veículos, explicado em detalhes na Seção 3.8. A definição da posição e tamanho de cada quadrado é definida a partir das posições mínimas e máximas, tanto no eixo horizontal como vertical, de cada blob.

A maioria dos blobs, como pode-se observar na Figura 21, assemelham-se ao formato de carros. Nota-se, contudo, que há alguns blobs especialmente pequenos, que muito provavelmente representam outros objetos. A função *labeling* (rotulação) também calcula a

área total de cada blob, de forma que se pode definir uma regra para quando desenhar ou não o quadrado, ou seja, quando considerar ou não que determinado blob é um veículo. Na seção de resultados é apresentado o resultado da aplicação da função para diferentes tamanhos mínimos de contagem.

3.7 AGRUPAMENTO PELA VELOCIDADE

Os dados obtidos pelos bons pontos a serem seguidos e a função de seguir pontos pelo fluxo ótico são usados nesta etapa para separar veículos no mesmo blob. Duas estratégias de agrupamento foram inicialmente testadas:

A primeira consistia em agrupar os pontos em clusters com um distância máxima entre qualquer ponto e seu ponto mais próximo pertencente ao agrupamento. A velocidade instantânea e a média também foram integradas ao sistema removendo ligações entre pontos com diferença de velocidades muito grandes. Uma outra variação incluiu ao cálculo da distância uma terceira dimensão igual a velocidade. Este método foi programado com complexidade $O(n\_pontos^2)$.

A segunda tentativa, ao contrário da primeira, baseou-se em um método consolidado na literatura, o k-means. Um dos parâmetros necessários para a execução deste algoritmo é o número de clusters que devem se dividir os pontos, no entanto, este é o número de veículos na região, que é exatamente o que estamos procurando. A largura reservada para cada veículo na rua é fixa, portanto concluiu-se que blobs com tamanho maior que a distância entre duas faixas de trânsito – mesmo considerando a perspectiva da imagem – devem ser divididos em dois grupos distintos. Se após a divisão algum dos agrupamentos continuar maior que a distância estipulada, repetia-se o k-means. Uma modificação foi reduzir uma das coordenadas dos pontos para o cálculo da distância entre os pontos baseado se a movimentação predominantemente é vertical ou horizontal na região. O artigo (ZOU et al., 2010) sugere considerar isoladamente os pontos que estão no mesmo componente conexo e depois realizar a divisão pelo k-means, e assim foi feito.

A representação gráfica das velocidades e o retângulo que engloba o blob foram fundamentais para a elaboração da melhor estratégia de agrupamentos desenvolvida. Antes de apresentar a solução observe graficamente o problema. A Figura 23 mostra dois carros agrupados em um único blob, mas que possuem o módulo das velocidades de seus pontos de interesse divididas claramente nos dois veículos. Uma situação especial ocorre quando o fluxo ótico de Lucas e Kanade converge errado e acaba encontrando a posição final do ponto em um

local errado. A Figura 24 apresenta a movimentação de um ponto muito diferente de todas as outras da imagem. Foi testado o uso da média das dez últimas velocidades para reduzir essa discrepância, mas a mediana eliminou esse ruído de forma mais eficiente.

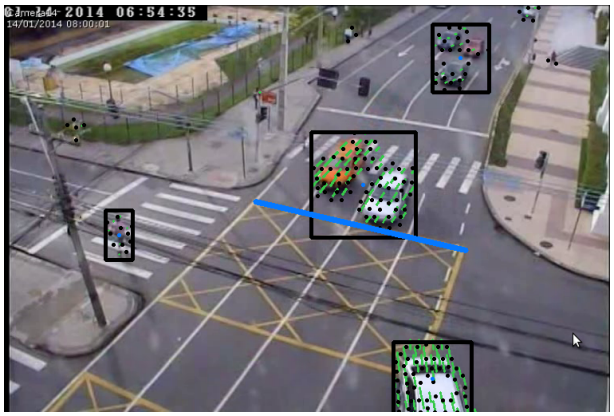

Figura 23: As linhas saindo dos carros representam a posição prévia do ponto, esse deslocamento instantâneo é a velocidade. Os dois carros ao centro, embora muito próximos fisicamente, possuem velocidades significativamente diferentes.

Fonte: Autoria Própria.

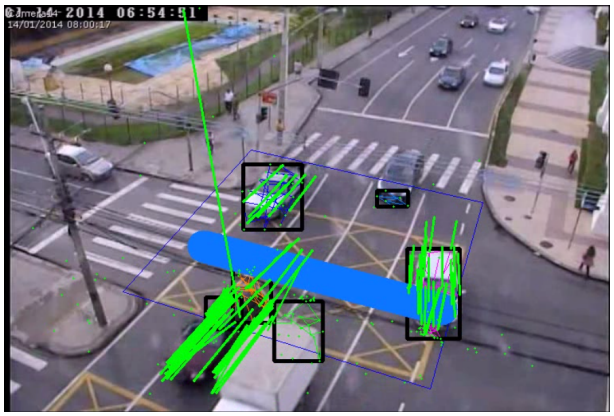

Figura 24: As linhas saindo dos veículos são suas posições a dez quadros atrás. Observe que uma das linhas aponta para muito longe e portante tem seu valor em módulo grande.

Fonte: Autoria Própria.

A fim de determinar os grupos de velocidades diferentes e, consequentemente, quantos veículos existem em cada agrupamento realizado pela rotulação, usou-se o *k-means* com o módulo das velocidades como dados de entrada. Inicialmente deve-se testar se há apenas uma velocidade predominante, excluindo velocidades zero de pontos recém encontrados, e consequentemente apenas um carro. Nestas condições, a distância máxima entre o valor mais distante de seu respectivo centro também deve ser limitada. Se alguma destas condições não for satisfeita, é testado um agrupamento com dois veículos e são aplicadas as mesmas restrições.

Se dois agrupamentos de velocidades não satisfizeram as condições são testados agrupamentos de três veículos. Esse processo continua até determinar o número de velocidades predominantes existentes no agrupamento.

É importante notar os parâmetros da utilização da função do OpenCV do k-means. São rodadas diversas vezes o algoritmo com centros iniciais distinto a fim de encontrar um erro reduzido. O número máximo de iterações em cada execução do algoritmo é limitado.

Decidido o número de veículos em um blob, repete-se o k-means com pontos com três dimensões, duas são as coordenadas do ponto e uma terceira é a mediana do módulo das últimas dez velocidades do ponto. Essa execução divide cada veículo em um blob distinto e seu resultado é passado para a contagem.

3.8 REGIÃO DE CONTAGEM

Conforme explicado na Seção 2.4, após o processo de segmentação, é realizado o processo de contagem. A abordagem definida foi a partir da análise dos quadrados gerados na etapa de rotulação.

Considerando a existência de vários retângulos em um vídeo, e supondo que eles mudem de posição sempre na mesma direção, uma maneira de se realizar a contagem é a partir da definição de uma linha, com largura de um pixel, e contar quantas vezes o meio dos quadrados tem intersecção com tal linha. Como o retângulo se move uniformemente, teríamos a certeza de que o meio de cada quadrado somente teria intersecção com a linha uma única vez. O princípio aplicado é como se segue: dada um linha de referência de espessura de um pixel colocada para realizar a contagem, para cada quadrado previamente definido (e que mudaria de posição conforme o blob), seu meio só teria intersecção com linha de referência uma única vez. As Figuras 25 e 26 exemplificam tal linha e sua situação de contagem.

Tal método, apesar de conseguir realizar a contagem de alguns veículos, é bem limitado quanto a robustez. Os blobs gerados em cada quadro tem seu tamanho e forma bem diferentes, mesmo quando são referentes a um mesmo veículo. Dessa forma, conforme situação ilustrada na Figura 27, é bastante comum o meio do quadrado não ter intersecção.

Como isso acontece com frequência, foi definida uma região de contagem ao invés de uma linha, mostrada na Figura 28. Nesse caso, porém, a premissa de que o meio do quadrado de cada blob só intersecta com a região uma única vez não é válida, de forma que algumas adaptações devem ser realizadas. Como a área de contagem é grande, seria natural que em alguns casos um mesmo carro fosse contado mais de uma vez (já que o meio do quadrado teria

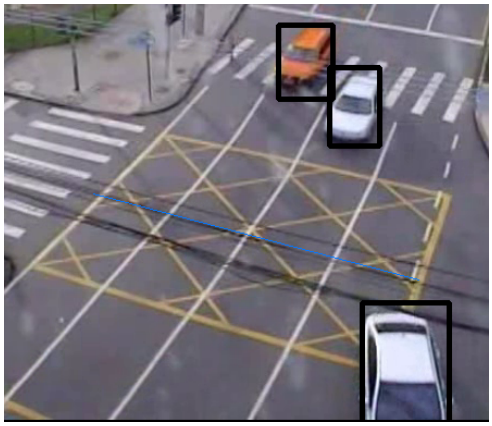

Figura 25: Linha de referência para contagem de veículos (em azul).

Fonte: Autoria Própria.

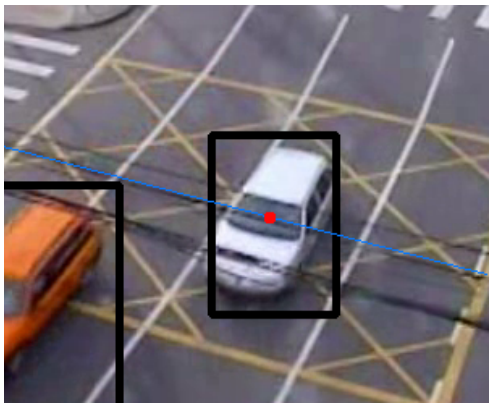

Figura 26: Exemplo de situação de contagem. Observa-se que o meio do quadrado (em vermelho) intersecta com a linha de referência (azul).

Fonte: Autoria Própria.

intersecção com a área de contagem em vários quadros).

Após observação do comportamento dos quadrados dentro da região de contagem, pode-se salientar duas características: o quadrado de um veículo demora mais do que 10 quadros para passar pela região de contagem e o meio dos quadrados tem uma distância mínima de 50 pixels entre si (nos vídeos sobre os quais o algoritmo foi calibrado). Assim sendo, tais características foram consideradas para evitar a múltiplas contagens de um mesmo veículo. Tal contagem é feita, então, quando acontece a intersecção entre o meio do quadrado e a região de contagem, mas nos 10 próximos quadros nenhum quadrado cujo meio esteja a uma distância menor do que 50 pixels do quadrado já considerado será contado.

Justamente por evitar o problema de contagens múltiplas, a regra mencionada também ameniza o problema de um carro ser representado por dois blobs, como mostrado na Figura

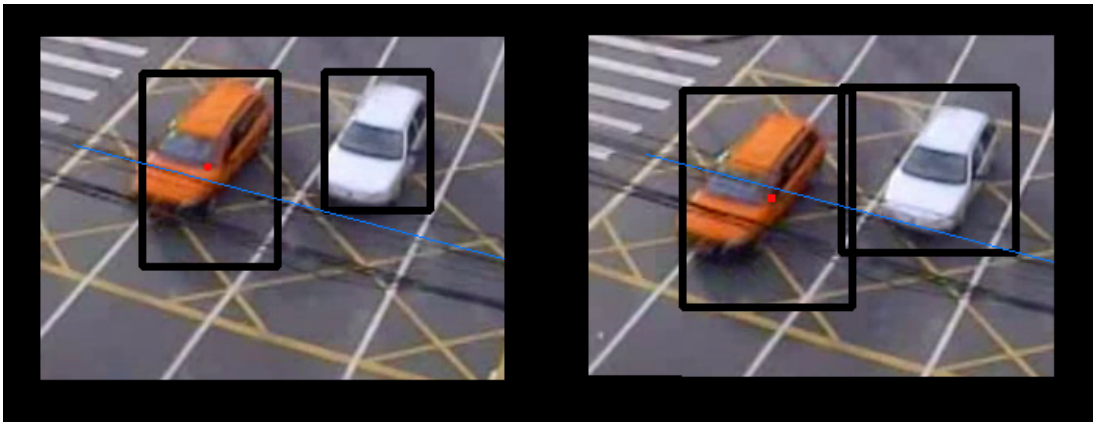

Figura 27: Exemplo de situação em que o meio do quadrado encontra-se antes da linha de referência. No quadro seguinte, o quadrado passou pela linha de referência. Como o meio do quadrado não intersectou com a linha, o veículo não seria contado.

Fonte: Autoria Própria.

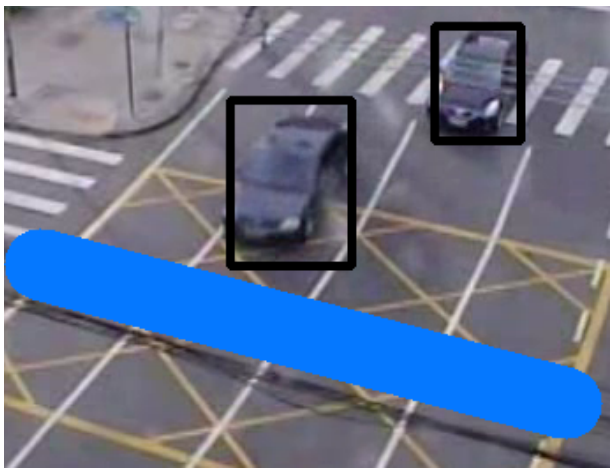

Figura 28: Região de contagem. Diferentemente da linha de referência, que tinha somente 1 pixel de espessura, a região de contagem tem espessura de 40 pixels.

Fonte: Autoria Própria.

16. Nessa situação, a rotulação define cada parte do carro como sendo dois veículos distintos. Porém, como a distância de seus meios é bem menor em comparação ao caso de dois carros, e como estão se deslocando na mesma direção, a regra, não permitindo a contagem na mesma região quando intersecções ocorrem até os 10 quadros seguintes, conta os dois blobs menores como apenas um carro, ou seja, não ocorre o problema de múltiplas contagens. A Figura 29 ilustra tal situação.

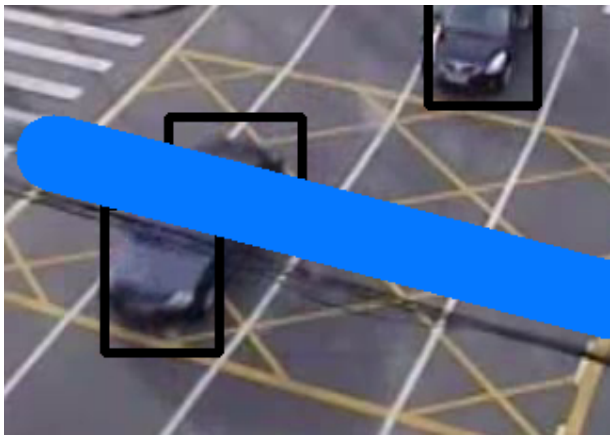

Figura 29: Exemplo de situação onde um carro é representado em dois blobs. Mesmo isso ocorrendo, a região de contagem consegue definir que os dois quadrados pertencem ao mesmo veículo

Fonte: Autoria Própria.

4 RESULTADOS

O desempenho do sistema foi medido a cada etapa e em algumas situações existiram algoritmos testados que apresentaram performance inferior ao método final proposto. Os resultados seguem a ordem de execução do software.

4.1 DILATAÇÃO E EROSÃO

O processamento de dilatações e erosões interferem bastante do resultado final. Conforme o tamanho e o formato da máscara de aplicação, diferentes resultados podem ser observados. As Figuras 30, 31 e 32 explicitam o resultado de uma certa imagem após diferentes máscaras serem aplicadas na imagem.

Figura 30: 1, 2 e 3 Aplicações, respectivamente, de máscara redonda de raio 3.
Fonte: Autoria Própria.

4.2 FECHO CONVEXO

A decisão de fragmentação dos blobs é feita a partir da porcentagem da área de erro em relação à área do polígono convexo. Diferentes porcentagens, geram diferentes segmentações, conforme se pode observar na Figura 33.

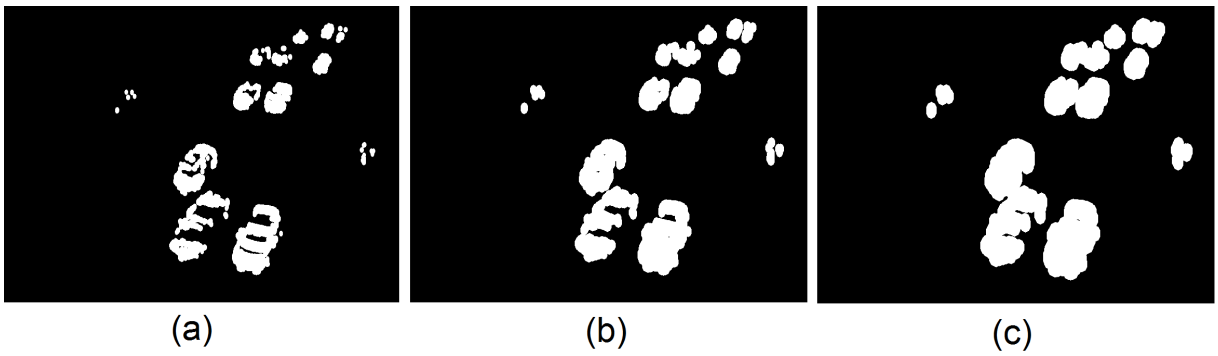

Figura 31: 1, 2 e 3 Aplicações, respectivamente, de máscara elíptica de eixos 5 e 3, sendo o eixo maior aplicado no sentido da movimentação dos blobs.

Fonte: Autoria Própria.

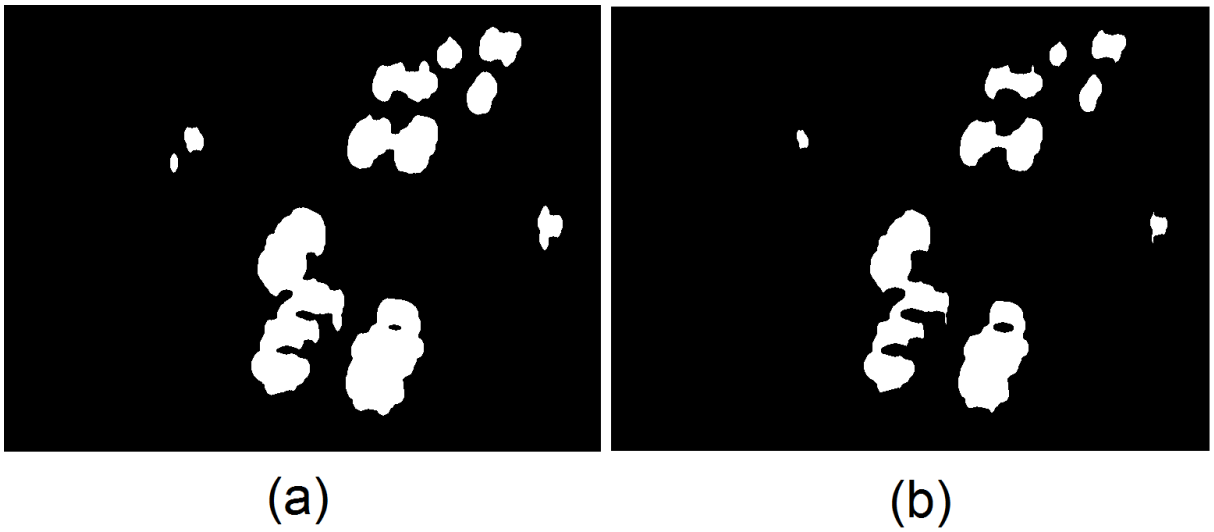

Figura 32: 3 Aplicações de máscara elíptica de eixos 5 e 3 seguido, respectivamente, de 1 e 2 erosões elípticas, sendo o eixo maior destas aplicada no sentido paralelo de movimentação dos blobs.

Fonte: Autoria Própria.

4.3 ROTULAÇÃO

Diferentes áreas de segmentação consideradas geram diferentes quantidades de objetos detectados. A Figura 34 demostra a aplicação da rotulação para diferentes áreas.

4.4 RESULTADOS DE CONTAGEM

O quadro a seguir apresenta o resultado da aplicação do algoritmo em 8 vídeos. A porcentagem de acerto ficou próxima dos 94% para os vídeos testados.

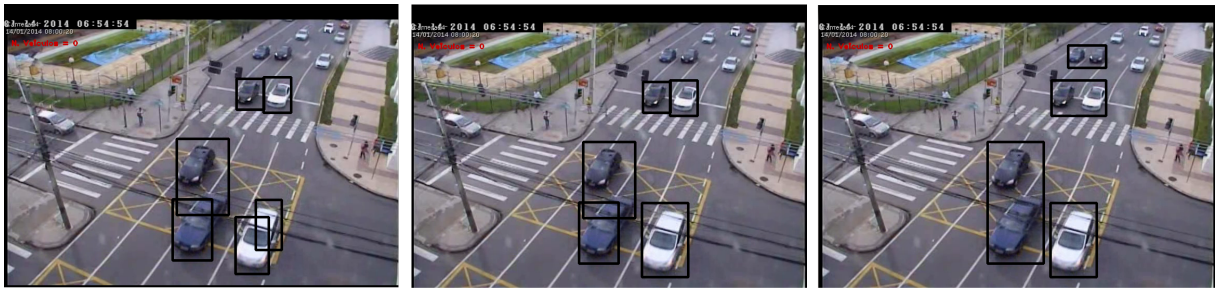

Figura 33: Resultado de limiares nas porcentagens de 5, 15 e 50%, respectivamente. Observa-se que o limiar que melhor separa os carros é o de 15%

Fonte: Autoria Própria.

Figura 34: Resultado da rotulação para diferentes áreas. Observa-se que o tamanho 2000, terceiro, mostra-se como mais adequado.

Fonte: Autoria Própria.

Quadro 1: Quadro com o resultado da aplicação do software em vários vídeos. Abordagem baseada nas áreas.

Número do Vídeo	Falsos Negativos	Falsos Positivos	Total de Erros	Contagem Software	Contagem Real	Índice de Erro	Índice de Acerto
1	0	1	1	46	45	2,22%	97,78%
2	0	3	3	50	47	6,38%	93,62%
3	0	4	4	49	45	8,89%	91,11%
4	0	4	4	56	52	7,69%	92,31%
5	1	2	3	32	31	9,68%	90,32%
6	0	0	0	9	9	0,00%	100,00%
7	0	1	1	32	31	3,23%	96,77%
8	0	1	1	42	41	2,44%	97,56%
Total	1	16	17	289	274	6,20%	93,80%

Fonte: Autoria própria.

Quadro 2: Quadro com o resultado da aplicação do software em vários vídeos. Abordagem baseada nas velocidades.

Número do Vídeo	Falsos Negativos	Falsos Positivos	Total de Erros	Contagem Software	Contagem Real	Índice de Erro	Índice de Acerto
1	2	1	3	44	45	6,67%	93,33%
2	2	1	3	46	47	6,38%	93,62%
3	1	6	7	50	45	15,56%	84,44%
4	1	5	6	56	52	11,54%	88,46%
5	2	3	5	32	31	16,13%	83,87%
6	0	0	0	9	9	00,00%	100,00%
7	4	2	6	29	31	19,35%	80,65%
8	1	1	2	41	41	4,88%	95,12%
Total	13	19	32	307	301	10,63%	89,37%

Fonte: Autoria própria.

5 CONSIDERAÇÕES FINAIS

A finalidade do projeto foi atingida, em uma rua com tráfego pequeno de ônibus e caminhões a contagem é feita com alto grau de exatidão. A interpretação de caminhões é especialmente difícil, o processamento inicial acaba respondendo pouco a áreas homogêneas e possivelmente o veículo é interpretado como dois automóveis menores já na fase da diferença de quadros seguida das operações morfológicas. O acerto da abordagem baseada na análise da área foi de 94,35% e são contados os veículos de quatro pistas, seriam necessárias duas pessoas para contar o fluxo em uma rua tão larga. Aparentemente a análise das velocidades pode ser vantageosa quando as condições de iluminação atrapalhem a abordagem baseada totalmente em blobs.

Várias ruas urbanas possuem carros em uma quantidade muito superior à veículos maiores como ônibus e caminhões. Em alguns horários e vias é até proibido o tráfego de caminhões. Desta forma contar um veículo grande como dois menores acaba não representando um erro expressivo.

5.1 TRABALHOS FUTUROS

Para a execução do programa é necessário configurar alguns parâmetros específicos de cada vídeo. Os parâmetros que devem ser regulados são: limiar da diferença de frames para separar veículos de sombras, tamanho elemento morfológico e região de interesse ou linha de contagem dos veículos. A contagem poderia ser realizada quando o veículo deixa o quadro, o tamanho do elemento morfológico deve ser estimado a partir da velocidade e possívelmente outros fatores relevantes para a subtração de fundo, a área de interesse precisaria processar o vídeo um tempo e registrar os locais de movimentação dos veículos. A variação das condições climáticas poderia modificar significativamente a iluminação e exigir o reajuste do valor do limiar estimado frequentemente. Ajustar automaticamente os valores é um trabalho futuro.

Os veículos são tratados como meros grupos de pontos brancos ou pontos com suas respectivas velocidades. Características como a cor, a forma de um veículo, o formato e a

posição do para-brisa permitem outras abordagens de classificação de um veículo. O artigo (PANG et al., 2007) realiza a segmentação considerando a diferença de frames no canal de tonalidade e utiliza esta informação em conjunto com a diferença de frames na intensidade. O uso da tonalidade encontra problemas ao tentar segmentar veículos pretos sobre o asfalto preto, mas em alguns agrupamentos a cor poderia ser decisiva para a separação de carros agrupados por uma etapa anterior.

No processamento inicial, os pontos são considerados separadamente, porém as abordagens de subtração de fundo baseadas em textura consideram uma região em volta de um centro antes de classificar o ponto como fundo estático ou objeto em movimento. Algumas vezes os carros ficam com o seu teto classificado como fundo devido a ser uma região homogênea e o veículo acaba sendo dividido em dois, (CUNHA, 2013) lista várias abordagens e seus resultados para segmentar carros, mas não permite concluir exatamente como o blob fica após o processo. Testar tais métodos como a parte inicial do programa potencialmente pode melhorar o resultado.

REFERÊNCIAS

ANDRADE, M. L. S. C. de. **Aplicação do método Level Set para Segmentação e Classificação de Padrões e Medidas de Escoamento Bifásico Gás-Líquido**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2011.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. California, Estados Unidos da América: O'REILLY, 2008.

COIFMAN, B. et al. **A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance**. 1998.

CUNHA, A. L. B. N. da. **Sistema Automática para Obtenção de Parâmetros de Tráfego Veicular A Partir de Imagens de Vídeo Usando OpenCV**. Tese (Doutorado) — Universidade de São Paulo, 2013.

DORINI, L. E. B. **Transformações de Imagens Baseadas em Morfologia Matemática**. Tese (Doutorado) — Universidade Estadual de Campinas, 2009.

FILHO, O. M.; NETO, H. V. **Processamento Digital de Imagens**. Rio de Janeiro, Brazil: Editora Brasport, 1999.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. **Intelligent Information Systems**, p. 107–145, 2001.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. p. 147–152, 1988.

HEIDARI, V.; AHMADZADEH, M. R. Method for vehicle classification and resolving vehicle occlusion in traffic images. 2013.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. p. 674–679, 1981.

OPENCV. **OpenCV**. 2014. Disponível em: <opencv.org>. Acesso em: 14 de julho de 2014.

PANG, C. C. C.; LAM, W. W. L.; YUNG, N. H. C. A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images. 2007.

RUAS, G. I. S.; BENSO, V. A. P. **Estudo e Implementação de um Sistema para Monitoramento de Fluxo de Veículos e Cruzamentos Urbanos Através de Vídeo Digital**. 52 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal do Paraná, Curitiba, Paraná, 2006.

SHAPIRO, L.; STOCKMAN, G. **Computer Vision**. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130307963.

SHI, J.; TOMASI, C. Good features to track. p. 147–152, 1994.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. [S.l.]: Editora Pearson Addison-Wesley, 2007.

VARGAS, M. et al. An enhanced background estimation algorithm for vehicle detection in urban traffic scenes. **IEEE T. Vehicular Technology**, v. 59, n. 8, p. 3694–3709, 2010.

ZOU, Y. et al. A moving vehicle segmentation method based on clustering of feature points for tracking at urban intersection. p. 120–123, 2010.

APÊNDICE A – METODOLOGIA DE DESENVOLVIMENTO

Neste capítulo são apresentados e discutidos os métodos de engenharia de software usados no desenvolvimento do programa de computador que contará carros, o qual é o produto final deste trabalho. O método de desenvolvimento visa facilitar a produção de um software de alta qualidade dentro dos recursos disponíveis.

Um bom software deve ser de fácil manutenção e evolução, deve evitar danos físicos ou econômicos em caso de falha do sistema, deve ser eficiente tanto em uso de memória quanto de processamento além de ter um tempo de resposta compatível compatível com a necessidade, deve ser usável pelo operador possuindo interface e documentação adequados. Para conseguir todas estas qualidades o trabalho deve usar os processos de software mais adequados a tarefa. Os processos que julgamos mais compatíveis com o trabalho em questão são o processo em espiral e a engenharia de componentes.

Antes de apresentar o processo em espiral e a engenharia de componentes, analisamos os requisitos funcionais e não funcionais do sistema, assim como realizamos as etapas iniciais do projeto de software e suas aplicações que definiram quais processos de software seguimos. Então o processo em espiral e a engenharia de componentes são discutidos. Na sequência fizemos uma estimativa do tempo necessário e, para finalizar o capítulo, um cronograma.

A.1 PROJETO DE SOFTWARE

Esta etapa visa esclarecer o que programa deve fazer, e desta forma deixar mais claro as tarefas que devem ser executadas, a ordem de suas execução e a sua possível paralelização.

A.1.1 REQUISITOS FUNCIONAIS

RF01: O software deve permitir verificar as etapas intermediárias nas imagens.

RF02: O software deve ser configurável para exibir quando necessário o resultado da diferença de frames, dos pontos sendo seguidos, seus respectivos históricos, o polígono convexo

de cada blob e/ou o centro de cada blob.

RF03: A representação nos vídeos deve permitir inferir a interpretação que o processo está tendo sobre os vídeos.

A.1.2 REQUISITOS NÃO FUNCIONAIS

RNF01: O software deve poder exibir na tela o resultado.

RNF02: O software deve permitir salvar o vídeo do processamento de um arquivo.

RNF03: O software deve permitir desabilitar parte do processo para aumentar a velocidade de execução.

A.2 DESENVOLVIMENTO EM ESPIRAL

O software irá evoluir aos poucos de forma a responder melhor ao problema de contar carros, no entanto não sabemos a priori o que fazer ou o que estaremos buscando melhorar em cada passo desta evolução. O trabalho envolve pesquisa constante em todo o processo de desenvolvimento o que reforça a escolha de um processo de software que permita modificar os requisitos de software muitas vezes e com isto modificar todo a organização do trabalho.

O modelo de desenvolvimento de software escolhido é o em espiral por ter todas essas características discutidas. Tal modelo baseia-se em repetir quantas vezes forem necessários os seguintes quatro passos: Definição de Objetivos, Avaliação e Redução de Riscos, Desenvolvimento e Validação, e Planejamento.

1. Definição de Objetivos: Os objetivos desta fase são definidos. Junto com os objetivos uma organização geral da fase é feita, são elencados os possível riscos e um plano de atividades é elaborado. Dependendo das conclusões tiradas nesta etapa, pode-se elaborar estratégias alternativas podem ser tomadas.

2. Avaliação e Redução de Riscos: Para cada risco de projeto identificado uma análise é realizada e possíveis precauções para redução dos riscos são tomadas. Uma precaução comum foi o uso de protótipos para verificar os resultados reais das técnicas aplicadas.

3. Desenvolvimento e Validação: Baseado nos riscos e objetivos levantados, escolhe-se um método de desenvolvimento que permita concluir os objetivos minimizando os riscos. No item 2.3 (Engenharia de Componentes) será detalhado o principal meio de produção de

código. Métricas devem ser definidas para estimar a qualidade do software e conduzir sua validação.

4.Planejamento: Todo o processo é revisado e uma decisão é tomada para o prosseguimento a mais um loop da espiral ou se termina o trabalho.

A Figura 35 ilustra a iteração dos passos do modelo em espiral.

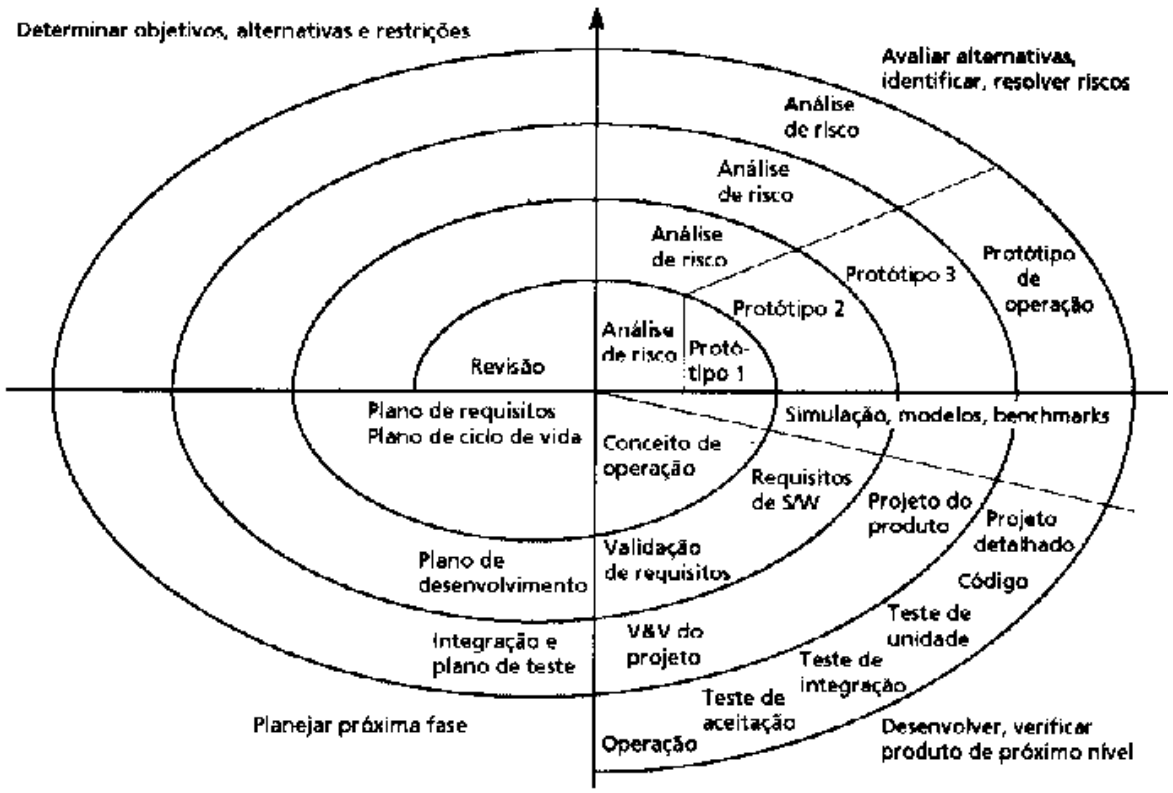

Figura 35: Modelo de desenvolvimento em espiral.
Fonte: (SOMMERVILLE, 2007)

A.3 ENGENHARIA DE COMPONENTES

Na maioria dos software há algum reuso de código, há uma evolução ou uma definição melhor do código, e existe – de modo amplo – uma série de passos a serem seguidos. Neste trabalho o reuso de funções prontas é predominantemente maior que os dois outros devido ao uso da biblioteca OpenCV que inclui funções prontas para a todos os algoritmos clássicos usados.

A Engenharia de Software Baseada em Componentes (do inglês CBSE – Component-Based Software Engineering) define uma metodologia para reaproveitar códigos de de uma

grande base de componentes reutilizáveis. O modelo genérico do processo é mostrado na Figura 36.

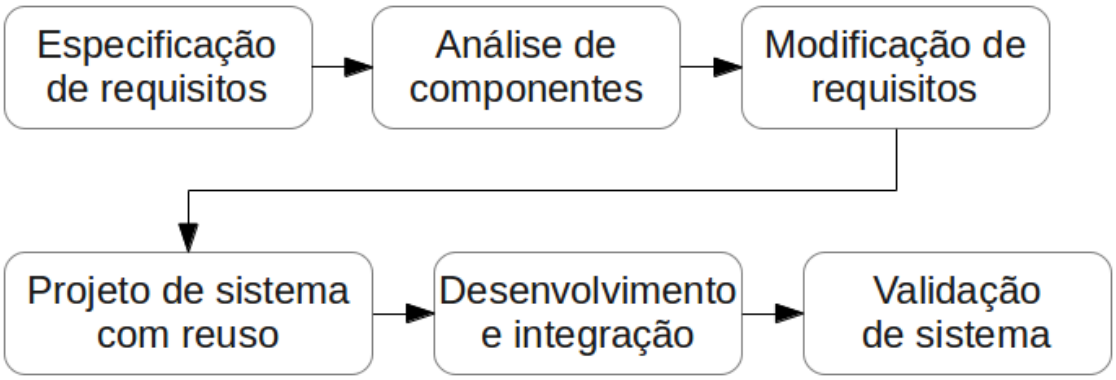

Figura 36: Modelo genérico do processo de desenvolvimento baseado em componentes. Fonte: Baseado na figura 4.3 do livro (SOMMERVILLE, 2007)

Como mostrado na figura 36, o processo de desenvolvimento baseado em componentes contém as seguintes etapas: especificação de requisitos, análise de componentes, modificação de requisitos, projeto de sistema com reuso, desenvolvimento e integração, e validação de sistema. Na etapa de especificação de requisitos é levantado os requisitos do sistema detalhadamente através da análise do problema. Depois de definidos os requisitos é feita busca por componentes, na fase de análise de componentes, para implementar tal especificação. Na terceira etapa é necessário modificar alguns requisitos para refletir os componentes disponíveis. O projeto de sistema com reuso consiste em projetar uma camada de comunicação do software com os componentes encontrados. Depois de todo esse planejamento, desenvolve-se o software e realiza a integração na quinta etapa. E por fim, a validação do software consiste em verificar se o sistema implementa tudo que estava especificado a fazer.

Os componentes encontrados e escolhidos da biblioteca OpenCV são mostrados na Figura 37.

No Capítulo 3 (Desenvolvimento) todas as tarefas da engenharia baseada em componentes é retomada e os resultados e modificações necessárias são detalhadas.

A.4 PROJETO DE SOFTWARE

O software foi inicialmente feito no paradigma estruturado, mas logo ficou claro que para uma melhor organização do software o paradigma orientado a objetos seria melhor. Para

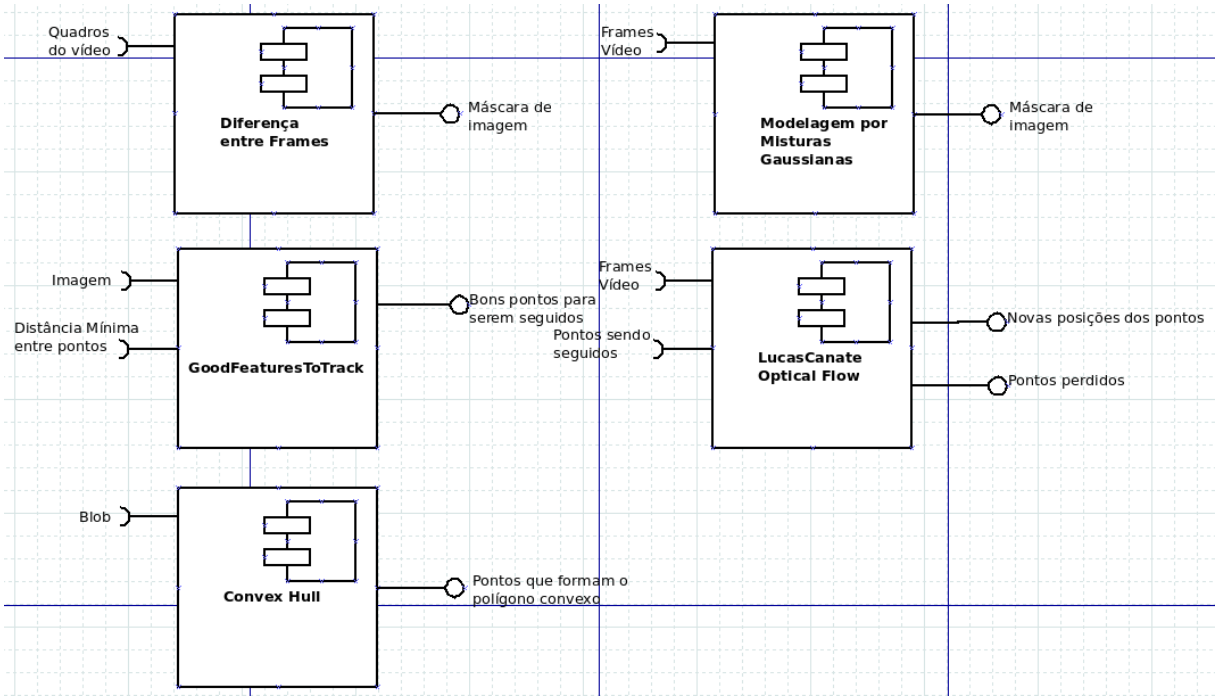

Figura 37: Componentes escolhidos do OpenCV e suas interfaces com seus dados de entrada e saída.

Fonte: Autoria própria.

fins de planejamento foi realizado um diagrama de classes inicial e depois incrementado, a versão final é exibida na próxima seção. O software, devido a grande quantidade de dados de entrada – vídeos –, acaba tendo processos demorados, portanto acaba sendo imprescindível a execução em paralelo dos itens da interface gráfica.

A.5 DIAGRAMA DE CLASSES

A figura 38 mostra o diagrama de classes do software.

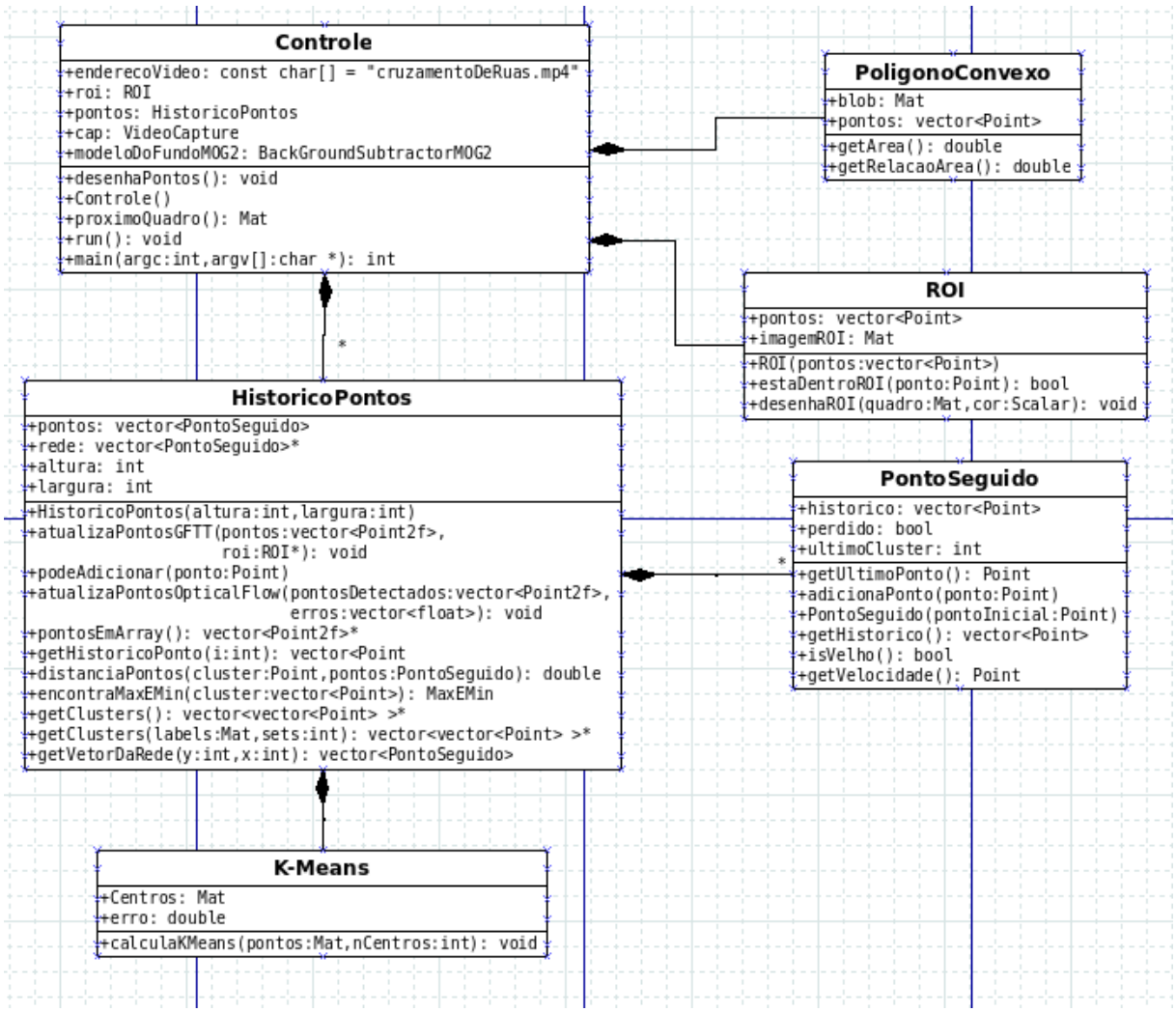

Figura 38: Diagrama de classes do software.

Fonte: Autoria própria.

ANEXO A – LICENÇA DO OPENCV (BSD)

Esta é a licença da biblioteca OpenCV, tal documento foi obtido em (OPENCV, 2014).

License Agreement
For Open Source Computer Vision Library

Copyright (C) 2000-2008, Intel Corporation, all rights reserved.
Copyright (C) 2008-2010, Willow Garage Inc., all rights reserved.
Third party copyrights are property of their respective owners.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistribution's [sic] of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.

* Redistribution's [sic] in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* The name of the copyright holders may not be used to endorse or promote products
  derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and
any express or implied warranties, including, but not limited to, the implied
warranties of merchantability and fitness for a particular purpose are disclaimed.
In no event shall the Intel Corporation or contributors be liable for any direct,
indirect, incidental, special, exemplary, or consequential damages
(including, but not limited to, procurement of substitute goods or services;
loss of use, data, or profits; or business interruption) however caused
and on any theory of liability, whether in contract, strict liability,
or tort (including negligence or otherwise) arising in any way out of
the use of this software, even if advised of the possibility of such damage.