GABRIEL RAMIRES DE BRITTO

DESENVOLVIMENTO DE ALGORITMO PARA "TRACKING" DE VEÍCULOS

São Paulo
2011

GABRIEL RAMIRES DE BRITTO

DESENVOLVIMENTO DE ALGORITMO PARA "TRACKING" DE VEÍCULOS

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Sistemas Eletrônicos

Área de Concentração:
Análise e Processamento de Imagens

Orientador:
Prof. Livre-Docente Hae Yong Kim

São Paulo
2011

RESUMO

Este trabalho consiste em desenvolver um algoritmo para realizar "tracking" de veículos. A palavra "tracking" é um jargão em processamento de imagens que designa o ato de identificar a trajetória de um objeto numa seqüência de imagens. Dado um arquivo de vídeo que mostra um fluxo de veículos, este trabalho pretende identificar a trajetória de cada veículo, simultaneamente, gerando um relatório no final do processo com informações sobre cada trajetória. Entre essas informações estão posição e velocidade de cada veículo quadro a quadro do vídeo. Uma das possíveis aplicações que podem surgir a partir deste tipo de algoritmo é a monitoração do tráfego de veículos por "software". Outra possível aplicação é a detecção de infrações de trânsito automaticamente.

Palavras-chave: Análise de imagens. Processamento de imagens. "Tracking". Monitoramento de veículos. Subtração de fundo. Reconhecimento de padrões.

ABSTRACT

The aim of this project is the development of an algorithm to track vehicles. The tracking process consists in analyzing the trajectory of moving objects on a video file. Based on footage of traffic, this project intends to extract information from the trajectory of each vehicle seen on video, such as position and velocity on each frame. The extracted information is reported on a text file after the last frame of the video has been processed. One possible application of such an algorithm is traffic monitoring. Another possible application is the detection of traffic violation.

Keywords: Image Analysis. Image processing. Tracking. Traffic monitoring. Background subtraction. Template matching.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

Contexto

O cotidiano das grandes cidades apresenta inúmeros problemas de conforto e segurança para seus habitantes, tanto no Brasil como nos demais países. Entre os problemas mais preocupantes estão aqueles relacionados ao trânsito de veículos. Infrações de trânsito, congestionamentos, e outras complicações envolvendo automóveis, são cada vez mais comuns, e exigem soluções para se garantir o bom funcionamento do trânsito nas cidades. Segundo estatísticas do DETRAN do Rio Grande do Sul [1], o número de infrações de trânsito vem crescendo no estado com o passar dos anos, situação que se repete nos demais estados brasileiros.

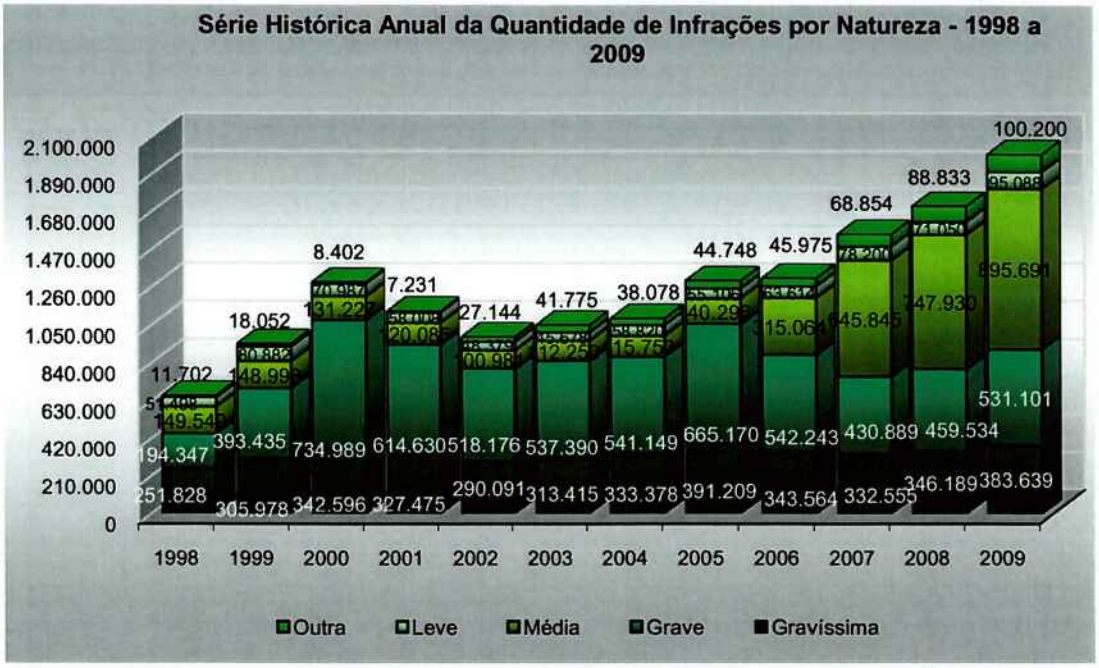


Figura 1 - Evolução das infrações de trânsito no Rio Grande do Sul

Boa parte dos problemas relacionados ao trânsito nas cidades brasileiras deve-se ao aumento do fluxo de veículos nos últimos anos. Segundo o Anuário 2009 da FENABRAVE (Federação Nacional da Distribuição de Veículos Automotores) [2], a venda de automóveis e comerciais leves no Brasil cresceu continuamente entre 2003 e 2009. O aumento das vendas promove um aumento da frota circulante, que, segundo dados do mesmo anuário, atingiu 34.032.121 unidades em 2009.

	País	2009	2008		2007		2006		2005		2004		2003		2002	
1°	Estados Unidos	10.421.142	13.221.150	1°	16.121.776	1°	16.525.716	1°	16.961.710	1°	16.874.137	1°	16.663.452	1°	16.814.276	1°
2°	China	9.848.086	6.492.553	2°	6.072.015	2°	4.263.864	3°	5.696.301	2°	2.489.470	5°	2.882.650	4°	3.434.338	3°
3°	Japão	4.577.288	5.060.639	3°	5.270.101	3°	5.579.593	2°	3.131.456	4°	3.456.063	3°	5.713.624	2°	1.246.603	11°
4°	Alemanha	3.982.467	3.318.310	4°	3.374.740	4°	3.669.837	4°	3.523.330	3°	5.698.021	2°	2.149.456	7°	768.063	14°
5°	Brasil	3.009.191	2.671.554	6°	2.327.192	9°	1.881.574	9°	1.617.785	9°	1.474.097	10°	1.356.989	9°	1.390.144	10°
6°	França	2.642.657	2.510.555	7°	2.526.611	8°	2.440.581	7°	1.298.342	11°	1.534.604	9°	1.345.223	10°	1.600.245	7°
7°	Itália	2.336.758	2.381.667	9°	2.727.884	6°	2.565.203	6°	2.456.659	7°	1.218.585	11°	1.593.479	8°	1.400.303	9°
8°	Inglaterra	2.181.387	2.421.256	8°	2.741.743	5°	2.672.026	5°	2.762.639	5°	2.896.853	4°	3.414.555	3°	5.721.215	2°
9°	India	1.967.472	1.665.795	10°	1.499.755	12°	1.309.970	12°	1.108.237	14°	1.093.310	12°	846.963	15°	800.900	13°
10°	Russia	1.465.925	2.923.540	5°	2.556.784	7°	1.871.041	10°	1.896.182	8°	2.422.147	7°	898.381	13°	1.562.533	8°
11°	Canadá	1.459.735	1.637.440	11°	1.653.364	11°	1.614.763	11°	1.583.395	10°	1.829.329	8°	2.390.680	6°	2.579.757	5°
12°	Coreia	1.408.575	1.170.640	13°	1.212.357	13°	1.152.970	14°	1.126.711	13°	1.064.924	13°	1.270.248	11°	1.703.037	6°
13°	Espanha	1.060.338	1.324.627	12°	1.884.354	10°	1.895.713	8°	2.487.854	6°	2.488.926	6°	2.459.195	5°	2.829.848	4°
14°	Austrália	908.047	974.831	15°	1.011.157	15°	928.821	15°	953.013	15°	923.285	15°	883.946	14°	995.617	12°
15°	México	722.463	1.008.719	14°	1.074.410	14°	1.157.509	13°	1.126.950	12°	1.041.922	14°	972.233	12°	708.137	15°
16°	Turquia	555.057	492.259	20°	594.379	18°	622.102	18°	717.491	16°	429.009	20°	565.772	16°	336.349	19°
17°	Tailândia	531.686	597.084	17°	608.477	17°	659.543	17°	677.132	17°	596.727	17°	508.052	18°	160.946	20°
18°	Bélgica	527.512	600.691	16°	590.268	19°	584.350	19°	540.068	19°	541.683	19°	508.845	17°	392.863	18°
19°	Holanda	436.843	582.427	18°	583.610	20°	547.797	20°	533.864	20°	570.511	18°	364.623	19°	515.147	17°
20°	África do Sul	376.409	498.507	19°	639.114	16°	669.269	16°	575.640	18°	696.107	16°	352.143	20°	591.385	16°

Fonte:
JATO do Brasil Informações Automotivas Ltda

Figura 2 - Evolução das vendas de automóveis e comerciais leves

Com o aumento da frota de automóveis que circulam diariamente, torna-se mais difícil gerenciar os problemas relacionados ao trânsito, principalmente quando á necessária a ação humana direta, como a detecção de certas infrações de trânsito. Além disso, o aumento do fluxo de veículos torna o trânsito menos seguro para motoristas e pedestres, uma vez que os acidentes passam a ser mais freqüentes. Nesse contexto, soluções de engenharia para melhorar as condições do trânsito devem sempre ser pesquisadas.

Tendo em vista a dificuldade de se controlar, monitorar e tomar providências quando necessário num fluxo cada vez maior de automóveis, este projeto propõe um algoritmo para monitoramento de veículos, baseado em processamento de vídeo e algoritmos de "tracking". Aqui, entende-se por "tracking" o processo de se identificar a trajetória de veículos numa sequência de quadros de um vídeo. Esse processo permite extrair informações como posição e velocidade de cada veículo, o que possibilita uma séria de aplicações úteis para o controle do trânsito. Uma dessas aplicações é a detecção de infrações de trânsito, como excesso de velocidade, conversões proibidas, tráfego em acostamento, entre outras.

Objetivos

O objetivo deste projeto é, dado um arquivo de vídeo que contenha um fluxo de veículos, gerar um relatório na forma de um arquivo de texto que descreva, quadro a quadro, a trajetória de cada veículo. Neste relatório devem constar a posição e velocidade de cada veículo em cada quadro do vídeo. A Figura 3 representa o relatório descrito.

RELATÓRIO

Veículo 1:

N° Quadro	Posição	Velocidade
1	x1, y1	v1
2	x2, y2	v2
3	x3, y3	v3
4	x4, y4	v4
5	x5, y5	v5

Veículo 2:

N° Quadro	Posição	Velocidade
1	x1, y1	v1
2	x2, y2	v2
3	x3, y3	v3
4	x4, y4	v4
5	x5, y5	v5

. . .

Figura 3 - Representação do relatório desejado na saída do algoritmo

1 TÉCNICAS PARA SE REALIZAR "TRACKING"

Para prosseguir neste projeto, é necessário fazer algumas considerações sobre as técnicas existentes para se realizar "tracking", além de algumas técnicas de processamento de imagem auxiliares, necessárias para adequar os quadros do vídeo às exigências do processo de "tracking".

O processo de "tracking" em vídeos é amplamente estudado, e existem inúmeros algoritmos para implementar esse processo. A escolha do algoritmo depende da aplicação e das características do vídeo a ser processado.

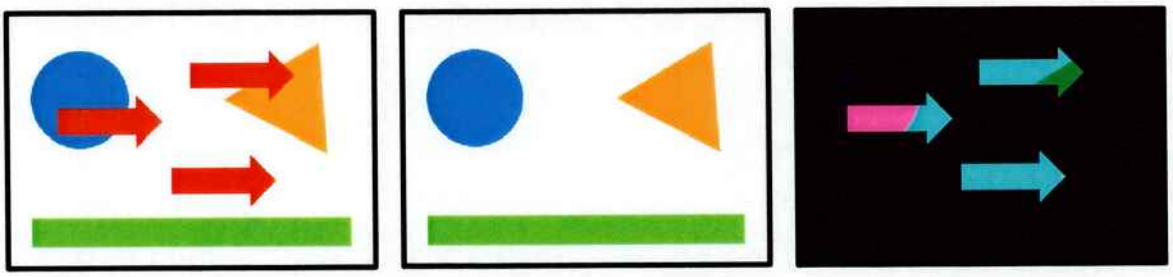
Para se realizar "tracking" de veículos deve-se identificar, em quadros consecutivos, um conjunto de píxeis que caracterize o veículo. O sucesso da aplicação depende da escolha adequada desse conjunto de píxeis [3], que deve ser facilmente reconhecido em quadros distintos do vídeo. Um método utilizado para se identificar o mesmo objeto em quadros distintos do vídeo é a detecção de cantos [4]. Alguns algoritmos de "tracking" utilizam a técnica da subtração de fundo ("background subtraction") [4] para separar os elementos em movimento dos elementos estáticos do vídeo. A imagem resultante, contendo apenas os elementos em movimento, é então processada de acordo com a aplicação.

Muitas vezes deseja-se realizar "tracking" de objetos sem um conhecimento prévio da forma ou comportamento desses objetos. Nesses casos, pontos específicos da imagem são selecionados e analisados de quadro para quadro. A cada um desses pontos é atribuído um vetor velocidade, baseado nas diferenças entre os quadros. Essa técnica é conhecida como "optical flow" [4]. Um importante método de "optical flow" é o algoritmo de Lucas-Kanade [4] [5], ou algoritmo LK. Outros métodos existentes são os algoritmos conhecidos por "Mean-Shift Tacking" e "Camshift Tracking" [4], sendo o último deles uma variação do primeiro. Esses algoritmos serão brevemente discutidos neste trabalho.

1.1 Subtração de Fundo

A subtração de fundo é um método utilizado para se separar objetos de interesse do restante da imagem. Desta forma, elementos da cena que não se deseja analisar podem ser removidos, tornando o processamento mais rápido ou até mais preciso. No caso deste projeto, a subtração de fundo seria adequada para separar os veículos dos elementos estáticos do vídeo. Isso resultaria num segundo vídeo, contendo apenas os veículos, que seria então submetido ao processo de "tracking".

O processo envolvido na subtração de fundo consiste em comparar cada quadro do vídeo com um quadro de referência. Este quadro deve conter apenas os elementos estáticos da cena, ou seja, o "fundo" da imagem. Quando um píxel do quadro analisado é muito diferente do píxel correspondente no quadro de referência, considera-se que esse píxel pertence a um objeto em movimento. A Figura 4 demonstra um exemplo de subtração de fundo.



(a) Quadro do vídeo **(b) Imagem de fundo** **(c) Resultado da subtração de fundo**

Figura 4 - Representação da técnica de subtração de fundo

Se for considerado o sistema de cores RGB ("red", "green", "blue"), a subtração de fundo é calculada da seguinte forma:

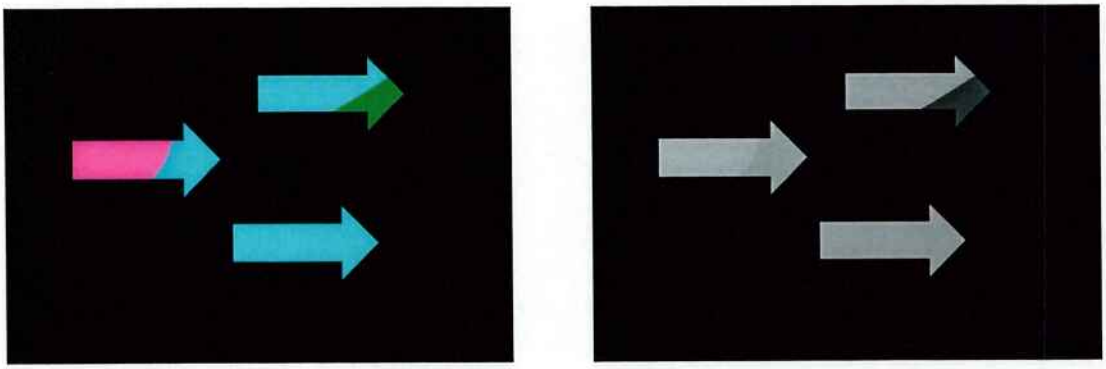
$$R_{i,j}(subtração) = abs\left(R_{i,j}(quadro) - R_{i,j}(imagem\ de\ fundo)\right) \quad (1)$$

$$G_{i,j}(subtração) = abs\left(G_{i,j}(quadro) - G_{i,j}(imagem\ de\ fundo)\right) \quad (2)$$

$$B_{i,j}(subtração) = abs\left(B_{i,j}(quadro) - B_{i,j}(imagem\ de\ fundo)\right) \quad (3)$$

Nas equações (1), (2) e (3), $R_{i,j}$, $G_{i,j}$ e $B_{i,j}$ são as componentes da escala RGB do píxel com coordenadas (i,j) das imagens utilizadas no processo.

Dependendo da aplicação, pode ser conveniente converter a imagem resultante da subtração de fundo para uma escala de tons de cinza, como mostra a Figura 5.



Resultado da subtração de fundo em cores **Resultado da subtração de fundo em tons de cinza**

Figura 5 - Mudança na escala de cores da subtração de fundo

A mudança para tons de cinza pode ser utilizada quando o que interessa é o formato do objeto analisado, e não sua cor.

As bibliotecas OpenCV e Proeikon, que são utilizadas pelo programa desenvolvido neste projeto, possuem formas eficientes de lidar com a manipulação de imagens, em diferentes escalas de cor.

1.2 Redimensionamento não Uniforme ("Warp")

O processo conhecido como "warp" consiste em deformar uma imagem, a partir de uma matriz de transformação. Essa matriz pode ser 2x3, caracterizando uma transformação afim, ou 3x3, caracterizando uma transformação de perspectiva. O segundo caso corresponde a uma transformação não linear.

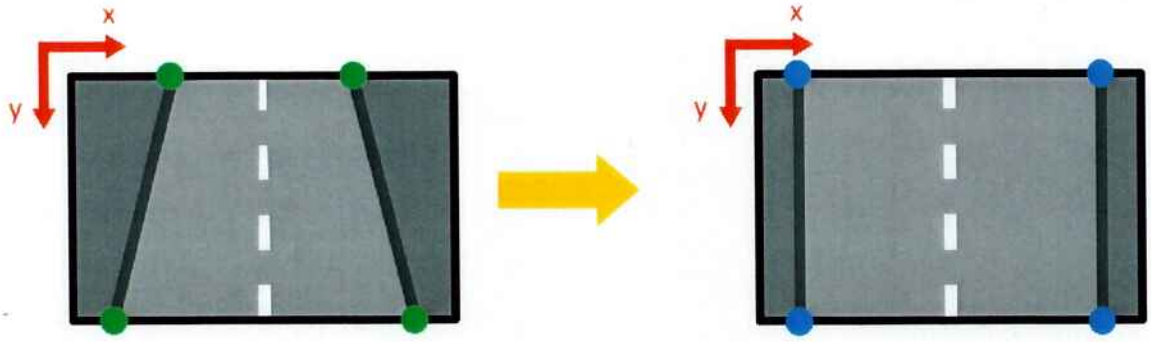
Para se obter a matriz de transformação é necessário selecionar pontos chave na imagem original, e determinar suas respectivas coordenadas na imagem transformada. A equação (4) demonstra uma transformação de perspectiva.

$$\begin{pmatrix} t_i \cdot x'_i \\ t_i \cdot y'_i \\ t_i \end{pmatrix} = M \cdot \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}, \qquad i = 0,1,2,3 \tag{4}$$

Na equação (4), x_i e y_i são as coordenadas originais dos pontos chave, x'_i e y'_i são as coordenadas transformadas, M é a matriz de transformação. Os valores t_i são necessários pois a transformação de perspectiva não é linear.

Esse procedimento é útil, por exemplo, para se corrigir distorções de perspectiva presentes nas imagens que se deseja processar. A Figura 6 esquematiza a aplicação do processo de "warp". Os pontos verdes representam as

coordenadas selecionadas na imagem original. Os pontos azuis representam as respectivas coordenadas transformadas. Os demais pontos da imagem são transformados segundo uma matriz que mapeia os pontos verdes nos pontos azuis.



(a) Imagem original **(b) Imagem transformada**

Figura 6 - Exemplificação do processo de "warp"

A Figura 7 demonstra o resultado da aplicação do processo de "warp" numa imagem real, adequada ao escopo deste projeto.

(a) **(b)**

Figura 7 - Processo de "warp" aplicado numa imagem real

A biblioteca OpenCV oferece diversas funções e recursos para se realizar "warp" [4].

1.3 Reconhecimento de Padrões ("Template Matching")

A técnica conhecida como "template matching" consiste em identificar numa imagem formas previamente conhecidas. Essas formas são representadas através de imagens chamadas de "templates". A Figura 8 exemplifica o processo de "template matching".

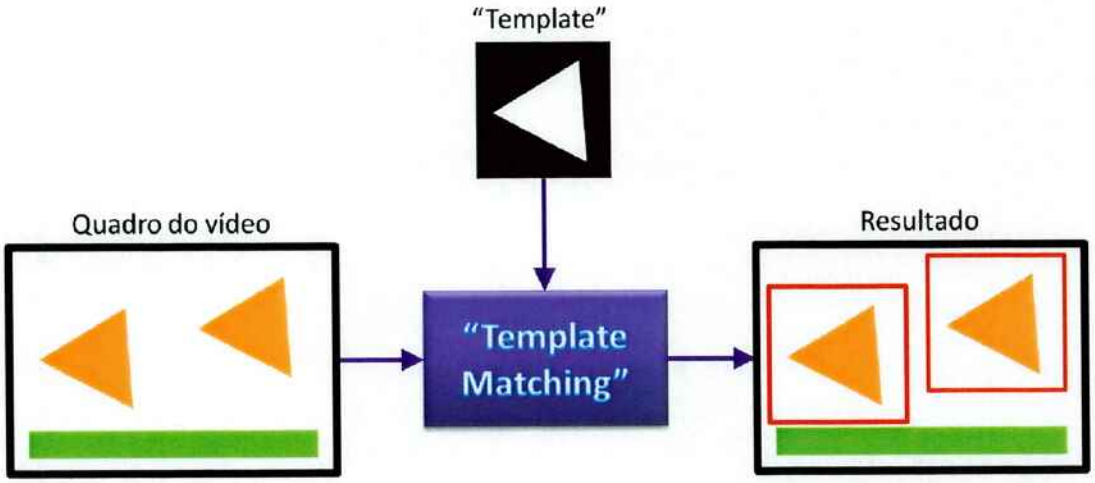
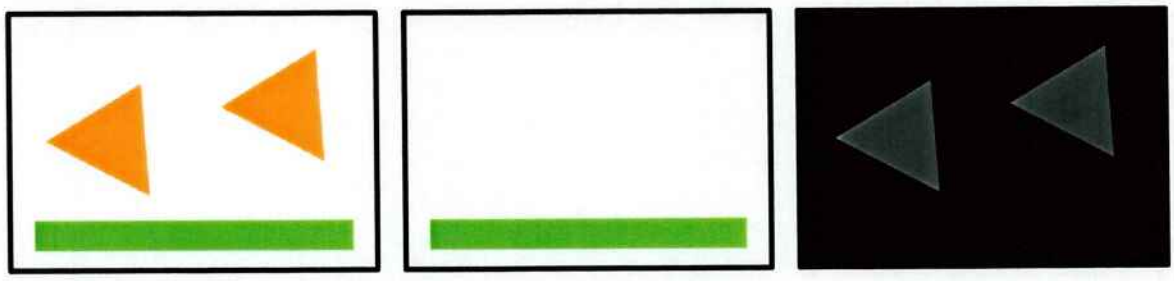


Figura 8 - Esquematização do processo de "template matching"

Cada ocorrência da forma representada pelo "template" deve ser localizada pelo processo.

Considerem-se o quadro e a imagem de fundo da Figura 9. Antes de realizar o "template matching", é necessário realizar a subtração de fundo. A imagem resultante é então convertida para uma escala de tons de cinza, como mostra a Figura 9(c).



(a) Exemplo de quadro **(b) Imagem de fundo correspondente** **(c) Resultado da subtração de fundo**

Figura 9 - Representação da subtração de fundo

Considere-se agora o "template" da Figura 10. O processo de "template matching" começa sobrepondo-se o "template" à imagem em todas as posições possíveis, como é mostrado na Figura 11.

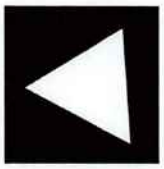
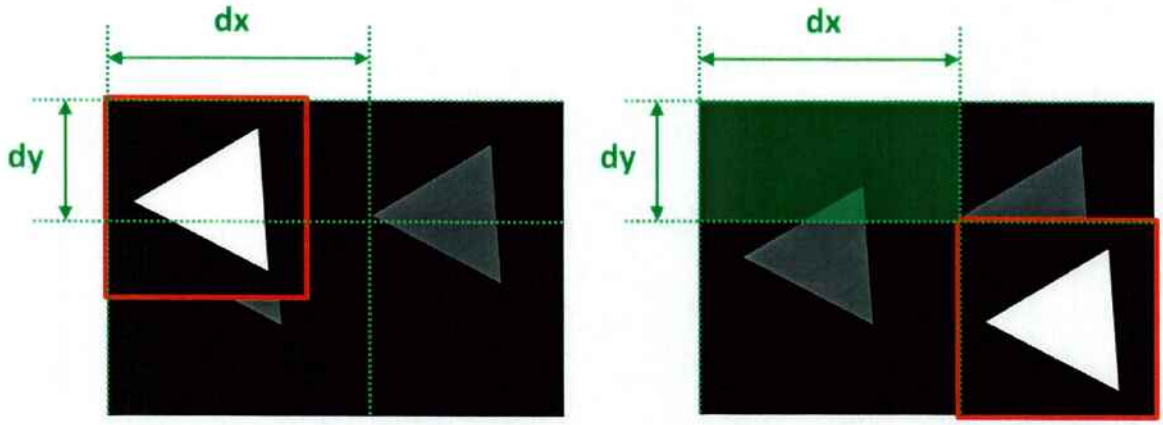


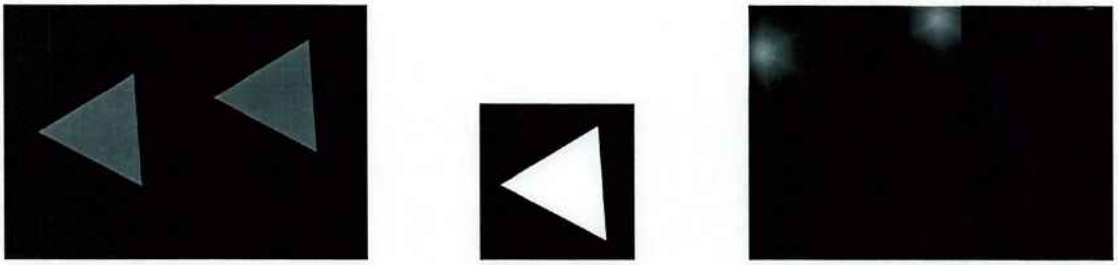
Figura 10 - "Template" para ser utilizado sobre o resultado da subtração de fundo



(a) Primeira posição possível **(b) Última posição possível**

Figura 11 - Sobrepondo o "template" ao resultado da subtração de fundo em todas as posições possíveis

Para cada posição em que o "template" é sobreposto ao resultado da subtração de fundo, calcula-se a correlação entre os píxeis do "template" e os píxeis da área sobreposta. A bibliotaca OpenCV, assim como a biblioteca Proeikon, possuem funções específicas para lidar com esse tipo de cálculo. O resultado, como mostrado na Figura 12, é uma terceira imagem, em tons de cinza, que mostra a correlação em cada posição. Os píxeis mais claros indicam que naquela posição o valor da correlação é elevado. Os píxeis mais escuros indicam que naquela posição a correlação é baixa.



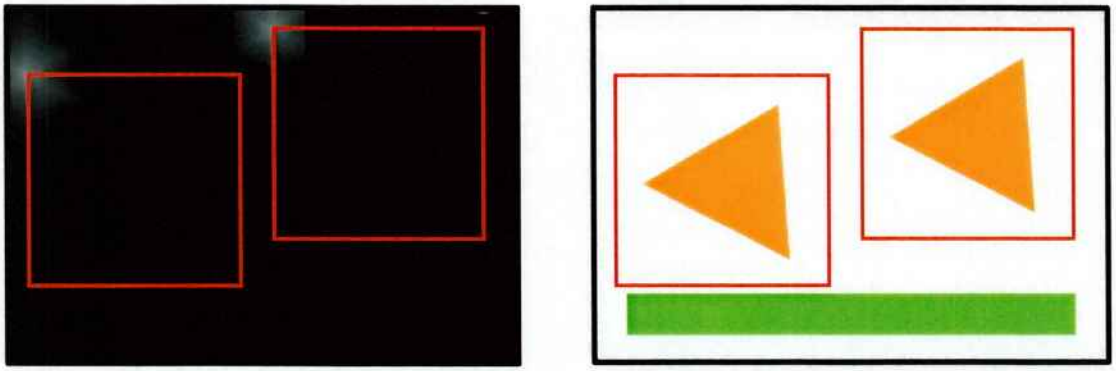
(a) Resultado da subtração de fundo

(b) "Template"

(c) Resultado do processo de "template matching"

Figura 12 - Entradas e saída do processo de "template matching"

Terminado o processo de "template matching", basta localizar os máximos locais da imagem resultante para encontrar as posições dos objetos procurados. O resultado é mostrado na Figura 13.



(a) Máximos locais na imagem de correlação

(b) Resultado sobreposto à imagem original

Figura 13 - Localizando os máximos locais e, consequentemente, as posições dos objetos procurados

Se o "template" utilizado aproxima bem o formato das formas procuradas na imagem, o resultado é satisfatório.

1.4 Detecção de Cantos

Para se realizar "tracking" de objetos num vídeo é necessário escolher pontos ou regiões da imagem que identifiquem e diferenciem esses objetos dos demais elementos da cena. Uma técnica muito utilizada para identificar objetos na imagem é a detecção de cantos [4]. Neste contexto, "canto" corresponde a uma região da imagem que possui uma grande variação de cor em duas direções ortogonais. Ao se localizar cantos de um objeto localiza-se o próprio objeto, sendo possível determinar seu movimento entre dois quadros consecutivos. Na Figura 14 encontra-se uma representação do que se considera "canto" em uma imagem.

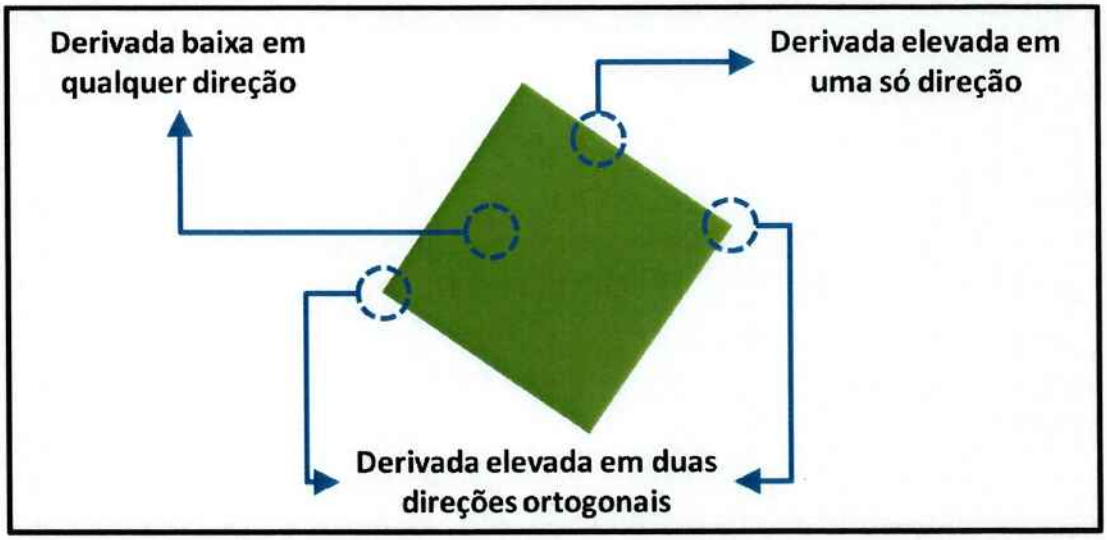


Figura 14 - Análise da derivada da intensidade em diferentes regiões da imagem

Nas regiões de "canto", o valor da derivada da intensidade dos píxeis deve ser elevado em duas direções ortogonais. Se for elevado em apenas uma direção, então o píxel pertence a uma aresta. Em regiões homogêneas da imagem, a derivada é baixa em qualquer direção.

Segundo uma definição matemática mais precisa, "cantos" são os pontos da imagem para os quais a matriz de autocorrelação abaixo possui dois autovalores elevados [4].

$$M(i,j) = \begin{bmatrix} \sum_{-K \le i,j \le K} w_{i,j} I_x^2(x+i, y+j) & \sum_{-K \le i,j \le K} w_{i,j} I_x(x+i, y+j) I_y(x+i, y+j) \\ \sum_{-K \le i,j \le K} w_{i,j} I_x(x+i, y+j) I_y(x+i, y+j) & \sum_{-K \le i,j \le K} w_{i,j} I_y^2(x+i, y+j) \end{bmatrix} \quad (5)$$

Na equação (5), I representa a matriz de intensidades dos píxeis, I_x e I_y são as derivadas da matriz I nos sentidos x e y respectivamente, e $w_{i,j}$ são pesos

atribuídos a cada ponto da imagem para se obter o formato da janela desejada. Se todos os $w_{i,j}$ forem iguais, a janela é retangular.

A detecção de cantos por si só não é um método de "tracking", mas uma ferramenta para se determinar quais pontos da imagem são adequados para se realizar "tracking".

A seguir estão brevemente descritos alguns métodos para se realizar "tracking", que podem ser bastante úteis para este projeto.

1.3 O Algoritmo LK

O alroritmo de Lucas-Kanade assume três hipóteses a respeito do vídeo utilizado no processo de "tracking":

1) Píxeis distintos, em quadros distintos, que representam o mesmo ponto de um objeto em movimento, possuem intensidades equivalentes. Em outras palavras, os objetos mantêm a mesma aparência ao longo do vídeo, não havendo mudança de brilho, contraste ou cor.
2) Entre dois quadros consecutivos do vídeo, os movimentos dos objetos são relativamente pequenos, de forma que as diferenças entre esses mesmos quadros são sutis. Em outras palavras, a freqüência de captura da câmera é muito superior à velocidade dos objetos.
3) Pontos vizinhos na cena pertencem à mesma superfície (mesmo objeto), possuem movimentos similares e são projetados em pontos vizinhos no plano da imagem.

Após selecionar píxeis adequados para o processo, anteriormente referidos como "cantos", o método de Lucas-Kanade realiza "tracking" através do cálculo da velocidade desses píxeis no quadro atual, para que sua posição no próximo quadro possa ser estimada. Para isso, duas informações precisam ser encontradas: a variação espacial e a variação temporal das intensidades dos píxeis, entre o quadro atual e o quadro anterior. Com essas informações, a velocidade de cada píxel, decomposta nas componentes u e v, pode ser calculada pela solução da equação (6).

$$I_x u + I_y v + I_t = 0$$

$$\begin{aligned} I_x &= derivada\ espacial\ em\ x & \qquad u &= velocidade\ em\ x \\ I_y &= derivada\ espacial\ em\ y & \qquad v &= velocidade\ em\ y \\ I_t &= derivada\ temporal \end{aligned} \tag{6}$$

A equação (6), por si só, não é suficiente para se encontrar os valores das componentes u e v. Para tal, é necessário montar um sistema com as equações referentes aos píxeis vizinhos àquele que está sendo analisado, uma vez que, pela

terceira hipótese, píxeis vizinhos têm movimento similar, e, portanto, possuem a mesma velocidade. Se o número de equações for muito grande, ou seja, se for utilizada uma janela muito grande ao redor do píxel analisado, corre-se o risco de incorporar píxeis que não pertencem ao objeto em movimento. Desta forma, deve-se resolver o sistema de equações a seguir, em que os píxeis p_i correspondem ao píxel analisado e os píxeis vizinhos.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{bmatrix} \tag{7}$$

O sistema anterior pode ser resolvido se os píxeis considerados correspondem a uma região de "canto" da imagem.

1.4 Os métodos "Mean-Shift Tracking" e "Camshift Tracking"

O "Mean-Shift Tracking" é uma técnica genérica para análise de dados, não sendo restrito à análise de imagens ou vídeos. Seu princípio é encontrar o máximo valor local da densidade de probabilidade de alguma variável. Isso é feito analisando-se um subconjunto de valores determinado por uma janela em torno de algum ponto de partida. Seleciona-se o maior valor dentro desta janela. Na próxima iteração, desloca-se a janela para o ponto determinado anteriormente e o processo é repetido. O máximo local terá sido encontrado quando a janela parar de se deslocar entre iterações, ou se deslocar menos do que um limiar determinado.

No contexto de "tracking", o conjunto de valores analisados seria uma matriz associada à imagem, em que cada valor é à probabilidade do píxel correspondente ser um "canto". Realizando-se o processo de "Mean-Shift Tracking" para um único quadro, localiza-se uma região de canto da imagem. Repetindo-se o processo para o próximo quadro, a partir da última posição da janela, localiza-se o mesmo canto no próximo quadro. Desta forma, os cantos têm sua trajetória acompanhada, que corresponde à trajetória do objeto à que pertencem.

A seguir, na Figura 15, está uma representação do funcionamento do algoritmo de "Mean-Shift Tracking". O diâmetro dos círculos indica a probabilidade do centro desse círculo ser um "canto".

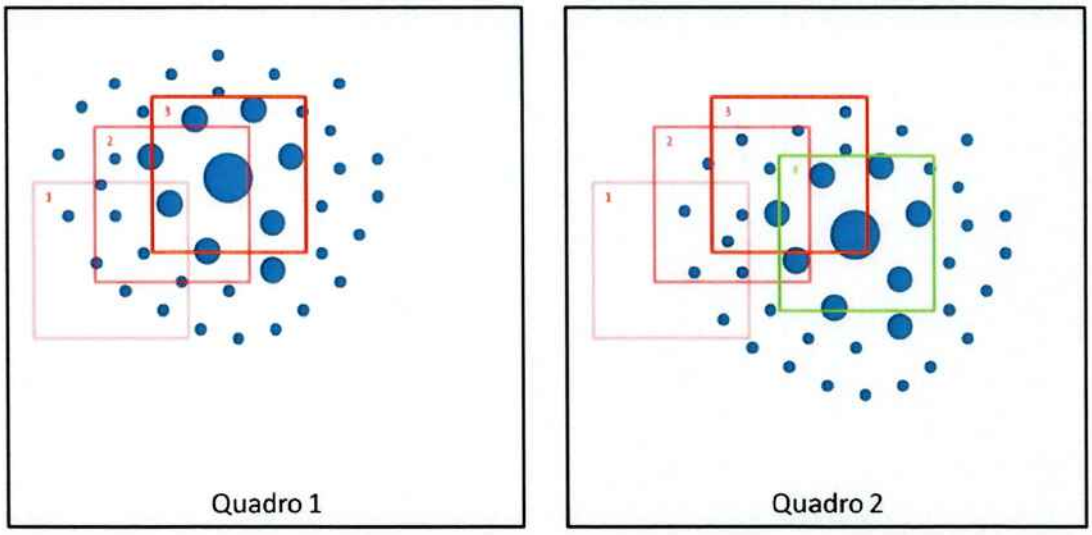


Figura 15 - Funcionamento do "Mean-Shift Tracking"

O "Camshift Tracking" é uma variação do método anterior. Neste método o tamanho da janela utilizada é ajustado automaticamente conforme os objetos rastreados se afastam ou se aproximam da câmera.

Os métodos "Mean-Shift Tracking" e "Camshift Tracking" não são utilizados neste projeto.

2 METODOLOGIA

Neste capítulo os detalhes do projeto serão abordados. O sistema proposto neste projeto recebe como entrada um vídeo contendo um fluxo de veículos. Esse vídeo é processado e cada veículo tem sua trajetória acompanhada pelo sistema. A saída do sistema é um relatório contendo informações relevantes sobre a trajetória de cada veículo identificado.

Figura 16 - Descrição simplificada do sistema

Os vídeos utilizados devem possuir algumas características específicas. Entre elas:

a) O campo de visão deve ser abrangente, possibilitando a visualização dos veículos por um percurso relativamente longo;
b) O ângulo de visão deve ser de tal forma que, para um tráfego intenso, não deve haver sobreposição total entre os veículos;
c) Se possível, o vídeo não deve conter obstáculos entre o ângulo de visão e os veículos, dificultando a identificação dos mesmos pelo "software".

A Figura 17, obtida a partir de um vídeo fornecido pelo professor Hae Yong Kim, orientador deste projeto, representa um quadro de um vídeo que atende às condições descritas anteriormente.

Figura 17 - Exemplo de vídeo a ser processado pelo "software" de "tracking"

2.1 Recursos Utilizados

Este projeto utiliza a linguagem de programação C++. As técnicas de processamento de imagens são aplicadas com o auxílio das bibliotecas OpenCV e Proeikon, sendo a última desenvolvida pelo professor Hae Yong Kim, orientador deste projeto.

Os vídeos utilizados para realização dos testes foram fornecidos pelo professor Hae Yong Kim.

2.2 Visão Geral do Algoritmo

O algoritmo que se deseja implementar pode ser separado em etapas. Elas estão representadas a seguir, na Figura 18.

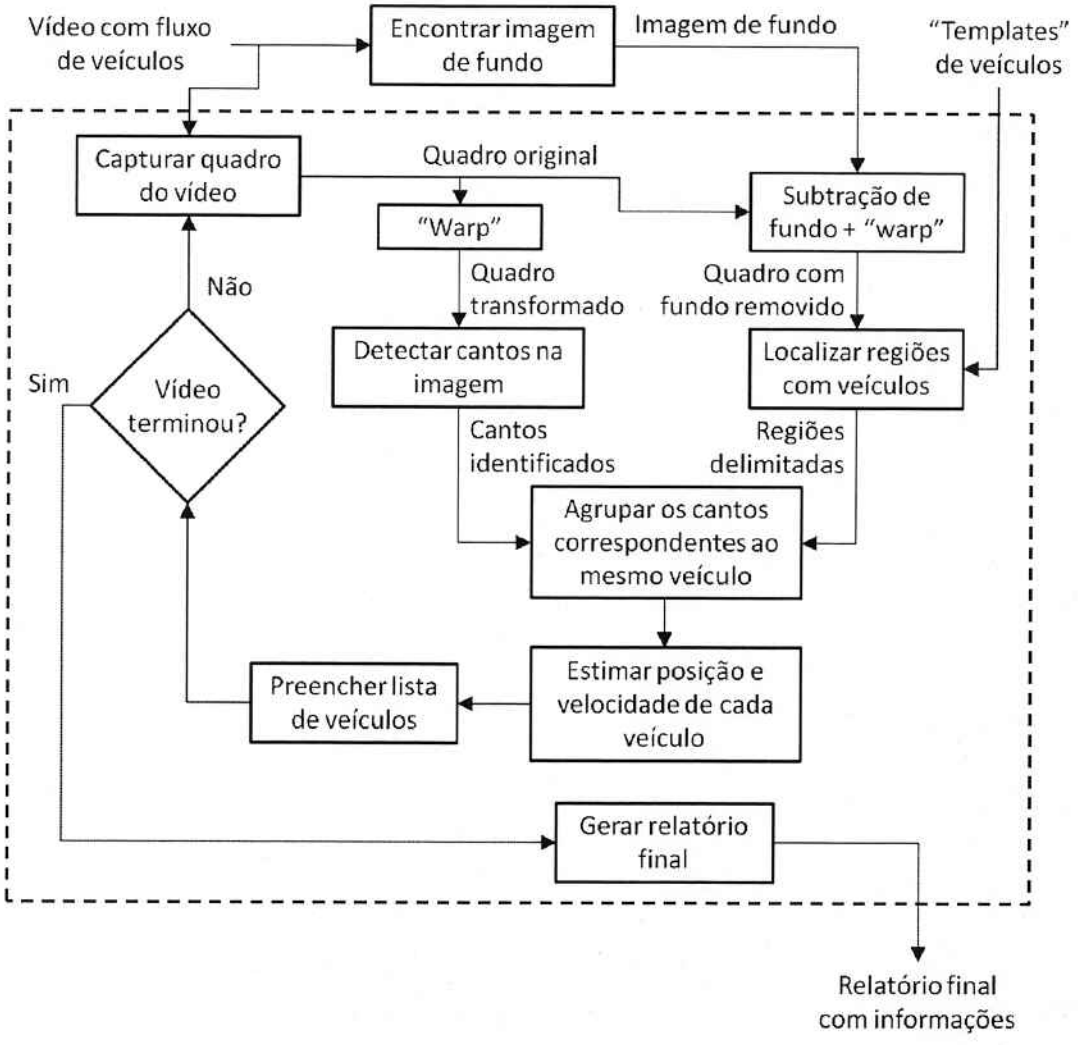


Figura 18 - Diagrama do algoritmo

Nos próximos parágrafos está uma breve descrição do funcionamento do algoritmo da Figura 18.

A primeira etapa no desenvolvimento do algoritmo é a obtenção da imagem de fundo a partir de um trecho do vídeo analisado. Esse processo será detalhado mais adiante.

Com a imagem de fundo gerada, o programa principal, delimitado pela linha tracejada da Figura 18, pode iniciar. A etapa de subtração de fundo permite obter uma imagem que contém apenas os veículos, sem os elementos estáticos da cena. Esta imagem é então submetida ao processo "warp", para remover distorções de perspectiva na imagem, deixando-a mais adequada para passar pelo processo de "template matching".

O processo de "template matching" encontra as posições em que os "templates" disponíveis melhor se sobrepõem à imagem. Ao término deste processo, as posições dos veículos presentes no quadro são conhecidas.

O quadro original é também submetido ao processo de "warp". A imagem resultante é analisada a procura de "cantos". Os cantos encontrados no quadro atual são equivalentes a cantos do quadro anterior do vídeo, o que permite estimar a velocidade de cada canto.

Combinando o resultado do "template matching" com as velocidades dos "cantos" encontrados, é possível estimar a velocidade e a próxima posição de cada veículo. Esses valores são guardados para serem consultados no próximo quadro.

O programa deve decidir se cada veículo encontrado no quadro atual já estava em quadros anteriores do vídeo. Isso é feito comparando-se a posição atual do veículo encontrado com todas as próximas posições calculadas no quadro anterior para todos os veículos. Caso não haja um veículo cuja próxima posição, calculada anteriormente, seja próxima da posição atual, o veículo encontrado no quadro atual é um novo veículo.

As informações sobre cada veículo e suas respectivas trajetórias são armazenadas numa lista ligada, que é preenchida no decorrer do processamento.

Quando não há mais quadros para processar, a lista ligada é percorrida e as informações contidas nela são registradas num arquivo de texto.

Este é apenas um breve resumo do funcionamento do algoritmo. Detalhes sobre cada etapa serão apresentados mais adiante.

Este documento apresenta o funcionamento do algoritmo desenvolvido sem entrar em detalhes sobre a sintaxe e comandos específicos utilizados. O código fonte e informações sobre os comandos podem ser encontrados no seguinte site:

http://www.lps.usp.br/~hae/projform/2010_gabriel_ramires

2.3 Implementando a Subtração de Fundo

2.3.1 Corrigindo o Contraste do Vídeo

Para que a subtração de fundo seja implementada com sucesso, é necessário que o vídeo (ou trecho de vídeo) considerado, possua os mesmos níveis de brilho e contraste em todos os quadros. Dependendo do vídeo utilizado, essa condição pode não ser verificada. A Figura 19 abaixo demonstra o problema, para um dos vídeos fornecidos pelo professor Hae Yong Kim, em que o quadro de número 200 é consideravelmente "mais escuro" que o quadro de número 1.

(a) Quadro n° 1 **(b) Quadro n° 200**

Figura 19 - Diferenças nos níveis de contraste

Para corrigir este problema foi utilizado um modelo de correção linear do contraste. A correção de brilho não será considerada.

Tomando como referência o primeiro quadro do vídeo, o modelo de correção de contraste deve mapear as cores dos demais quadros nas cores "corretas" em relação ao primeiro quadro. A Figura 20 abaixo demonstra esse modelo.

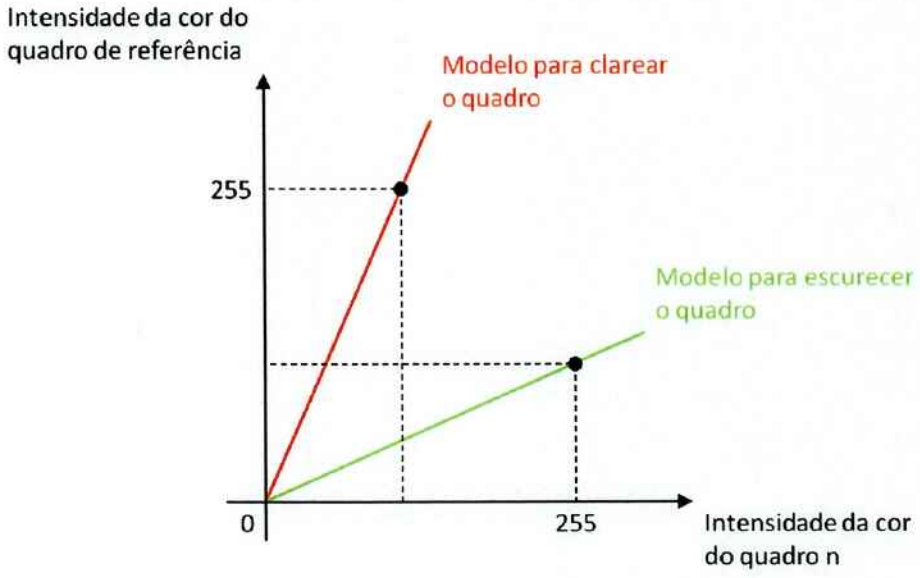


Figura 20 - Modelo de correção de contraste

Quando o quadro atual do vídeo possui tons mais "claros" do que o quadro de referência, deve-se "escurecer" esse quadro. Quando o quadro atual do vídeo possui tons mais "escuros" do que o quadro de referência, deve-se "clarear" esse quadro.

Para determinar o modelo descrito anteriormente, basta calcular o coeficiente angular adequado da reta que mapeia as intensidades das cores do quadro atual nas intensidades das cores do quadro de referência.

O cálculo desse coeficiente angular requer que uma determinada região do quadro de referência esteja presente em todos os demais quadros do vídeo, nunca obstruída pelos elementos que se movem durante o vídeo. É nessa região que as intensidades das cores do quadro atual são comparadas com as intensidades das cores do quadro de referência, possibilitando o cálculo do coeficiente.

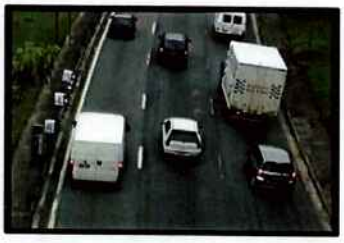
A aplicação do modelo para os quadros de n° 1 e 200, mostrados anteriormente, fornece o resultado da Figura 21.

(a) Quadro n° 1

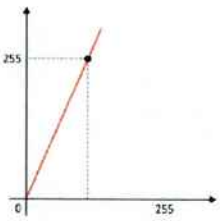
(b) Quadro n° 200

Figura 21 - Resultado da correção de contraste

Comparando as regiões comuns aos dois quadros é possível perceber que o contraste foi corrigido. Houve também uma saturação da intensidade das cores claras devido a essa correção. A Figura 22 resume o procedimento.



(a) Quadro n° 200 antes da correção de contraste



(b) Modelo aplicado

(c) Quadro n° 200 após a correção de contraste

Figura 22 - Resumo da correção aplicada

2.3.2 Recuperando a Imagem de Fundo

Para implementar a subtração de fundo é necessário conhecer a imagem de referência, ou seja, a imagem que contém apenas os elementos que se deseja ignorar. Dependendo do vídeo a ser processado, talvez um dos quadros possa ser utilizado como imagem de fundo. Em situações mais genéricas, em que o tráfego é tão intenso ao longo do vídeo que não é possível identificar um quadro que sirva como imagem de fundo, é interessante possuir um método para recuperar essa imagem.

O método utilizado neste trabalho para recuperar a imagem de fundo assume a seguinte hipótese sobre o vídeo utilizado: cada píxel do vídeo passa mais tempo sobre um ponto da imagem de fundo do que sobre um veículo. Para os vídeos em que essa hipótese não se aplica, o método que será mostrado aqui não funciona.

Se esta hipótese for válida, então é possível recuperar a imagem de fundo calculando a mediana das intensidades das cores de cada píxel ao longo do vídeo. As Figuras 23 a 38 ilustram o funcionamento do método. Cada imagem representa um quadro do vídeo. Os retângulos vermelhos representam veículos, e cada quadrado tracejado representa um píxel. O quadrado destacado em amarelo representa o píxel analisado.

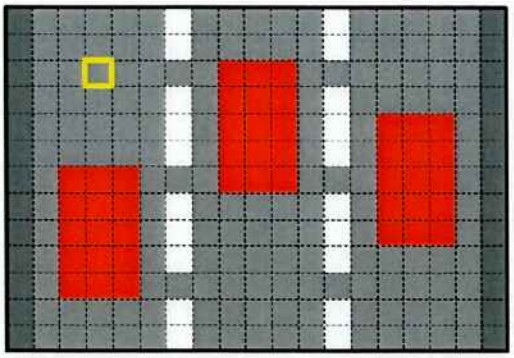


Figura 23 - Quadro 1 para a recuperação da imagem de fundo

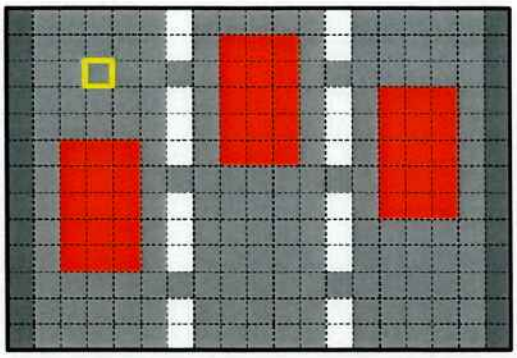


Figura 24 - Quadro 2 para a recuperação da imagem de fundo

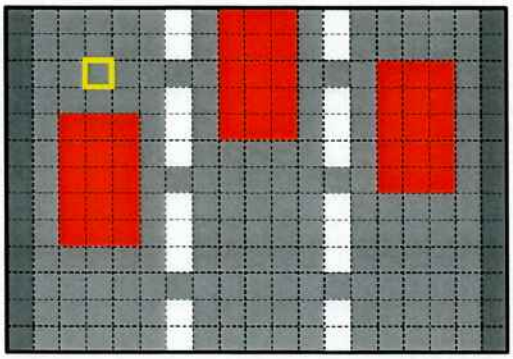


Figura 25 - Quadro 3 para a recuperação da imagem de fundo

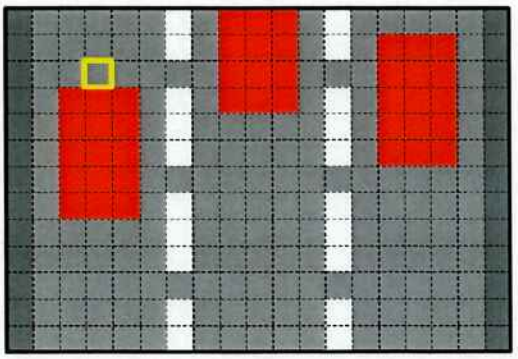


Figura 26 - Quadro 4 para a recuperação da imagem de fundo

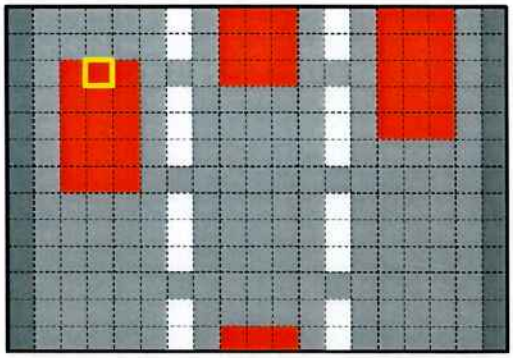


Figura 27 - Quadro 5 para a recuperação da imagem de fundo

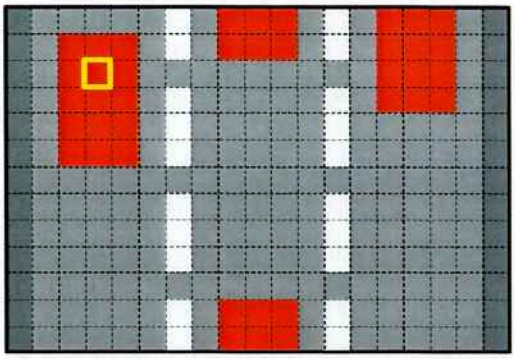


Figura 28 - Quadro 6 para a recuperação da imagem de fundo

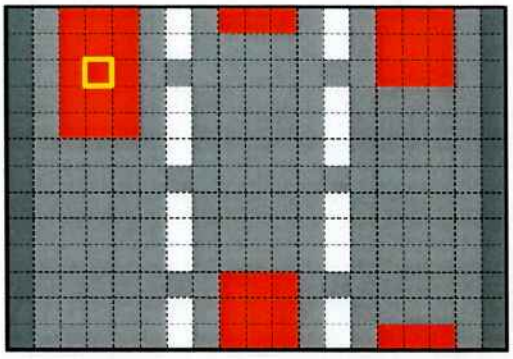


Figura 29 - Quadro 7 para a recuperação da imagem de fundo

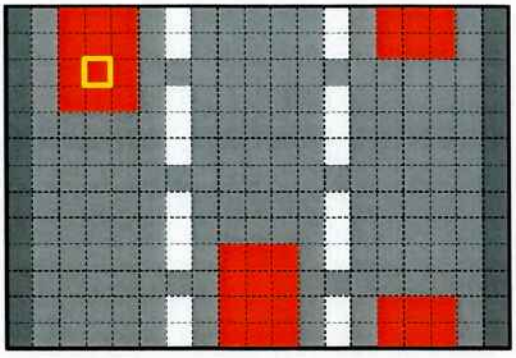


Figura 30 - Quadro 8 para a recuperação da imagem de fundo

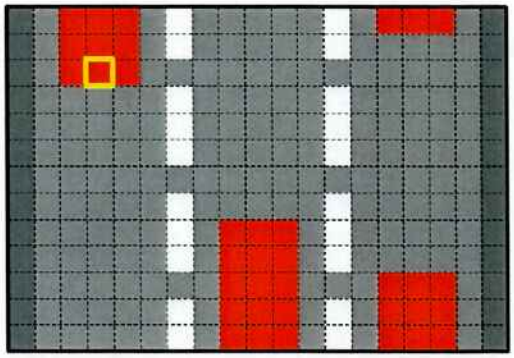


Figura 31 - Quadro 9 para a recuperação da imagem de fundo

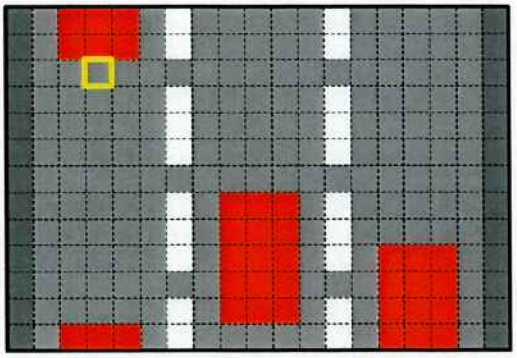


Figura 32 - Quadro 10 para a recuperação da imagem de fundo

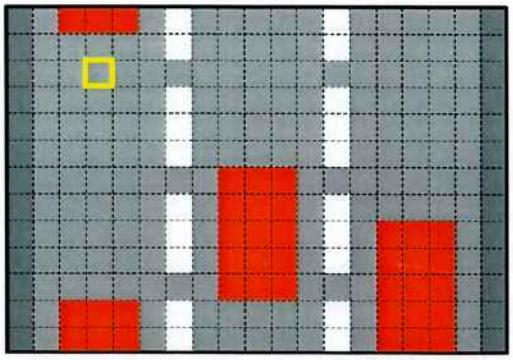


Figura 33 - Quadro 11 para a recuperação da imagem de fundo

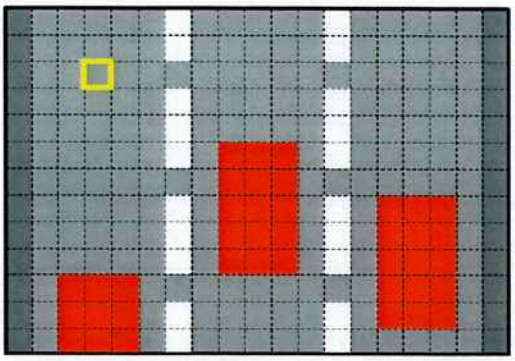


Figura 34 - Quadro 12 para a recuperação da imagem de fundo

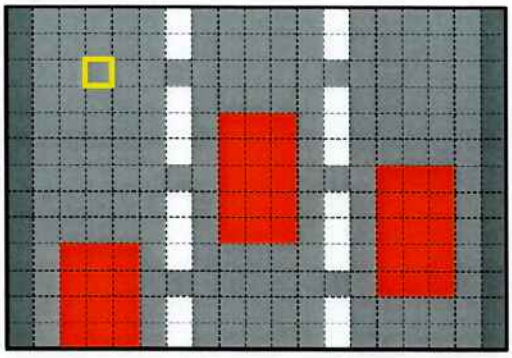


Figura 35 - Quadro 13 para a recuperação da imagem de fundo

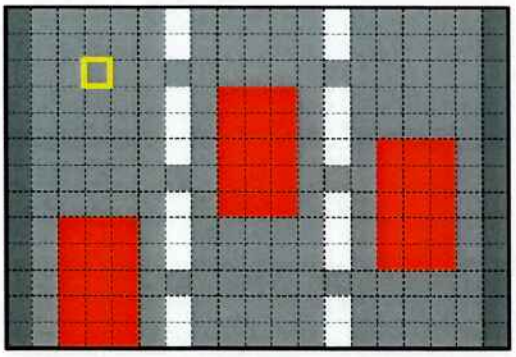


Figura 36 - Quadro 14 para a recuperação da imagem de fundo

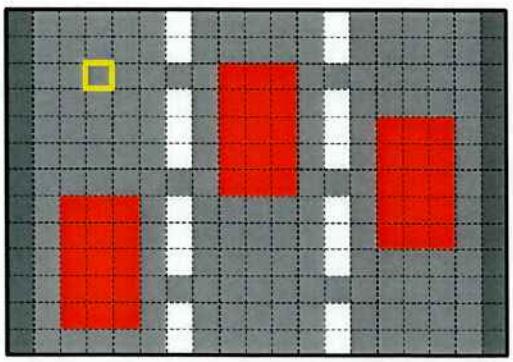


Figura 37 - Quadro 15 para a recuperação da imagem de fundo

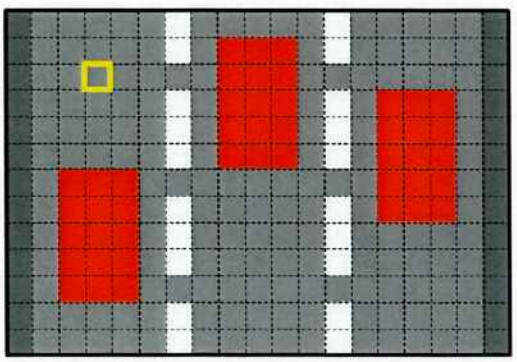


Figura 38 - Quadro 16 para a recuperação da imagem de fundo

Se a sequência de cores do píxel destacado for armazenada num vetor, o resultado é aquele representado na Figura 39.

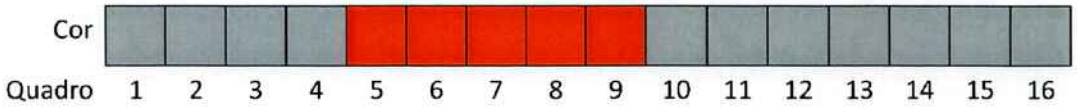


Figura 39 - Evolução das cores do píxel analisado

Para descobrir qual a cor do píxel destacado na imagem de fundo é preciso adotar algum critério para ordenar o vetor. Decompondo-se as cores em suas coordenadas RGB ("Red", "Green", "Blue"), por exemplo, pode-se ordenar o vetor de acordo com as intensidades em cada coordenada. Para a coordenada R, a ordenação do vetor gera o resultado mostrado na Figura 40.

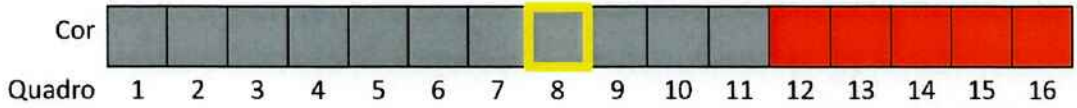


Figura 40 - Ordenação do vetor segundo a componente R da cor do píxel

Desta forma, a componente R da cor do píxel destacado na imagem de fundo é a componente R da mediana do vetor da Figura 40.

Repetindo-se o procedimento para as coordenadas G e B, descobre-se a cor do píxel na imagem de fundo. Ao repetir todo o procedimento para cada píxel da imagem, recupera-se a imagem de fundo do vídeo, representada na Figura 41.

Novamente, para este método funcionar é necessário que a hipótese inicial seja verdadeira.

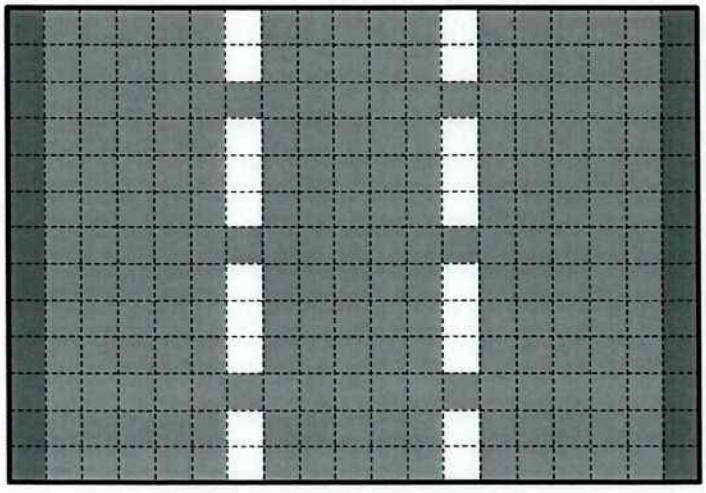


Figura 41 - Imagem de fundo para os quadros das Figuras 23 a 38

Aplicando-se o procedimento descrito em trechos do vídeo considerado, foram obtidos os resultados apresentados na Figura 42 para a reconstrução da imagem de fundo.

(a) Trecho de vídeo entre os quadros de n° 1 e 1000

(b) Resultado para o trecho de vídeo entre os quadros de n° 1 e 5000

(c) Resultado para o trecho de vídeo entre os quadros de n° 10000 e 11000

Figura 42 - Resultados da recuperação da imagem de fundo para diferentes trechos do vídeo

Analisando-se os resultados percebe-se que a imagem de fundo obtida para o trecho entre os quadros de n° 10000 e 11000 é bastante satisfatória.

2.3.3 Aplicando a Subtração de Fundo

Com os níveis de contraste corrigidos e a imagem de fundo recuperada, é possível agora implementar a subtração de fundo. O processo consiste em, para cada píxel do quadro, subtrair as coordenadas RGB da imagem original e da imagem de fundo. O valor absoluto dessa subtração forma as coordenadas RGB do píxel na imagem final. O resultado deste procedimento está representado na Figura 43 a seguir.

(a) Quadro n° 1

(b) Quadro n° 100

(c) Quadro n° 200

Figura 43 - Resultados da subtração de fundo para diferentes quadros

2.4 Delimitando os Veículos

2.4.1 Alterando o Espaço de Coordenadas

Para realizar "tracking" de veículos é preciso conhecer o espaço de coordenadas no qual os veículos se movem. Nos vídeos utilizados neste trabalho, como já pôde ser observado, o ângulo de observação é tal que há uma distorção de perspectiva na trajetória dos veículos. A Figura 44 mostra esse problema.

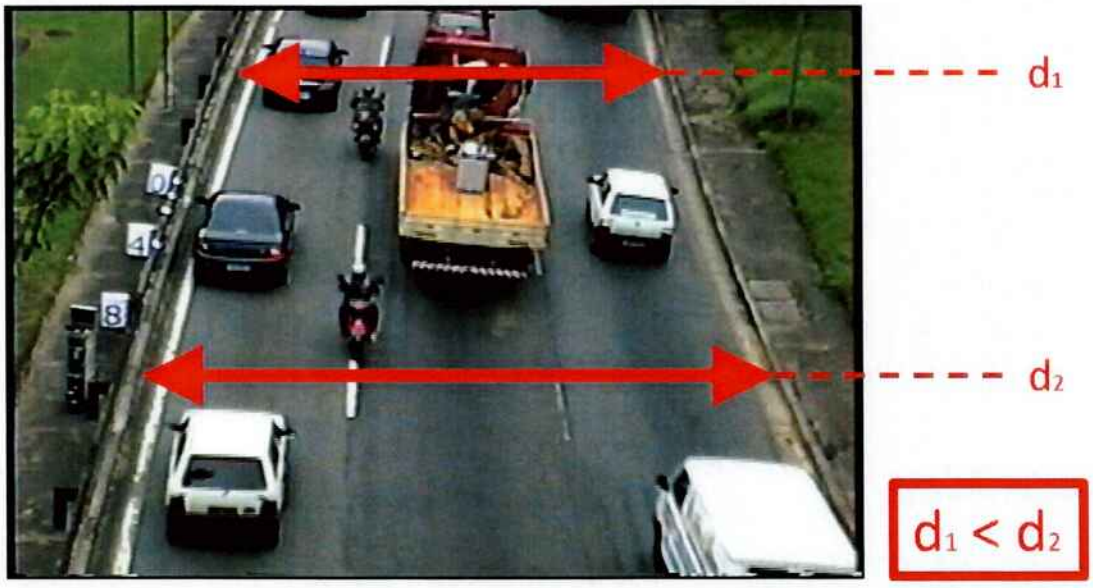


Figura 44 - Distorção de perspectiva nos quadros do vídeo

Essa distorção é inconveniente para se tratar matematicamente as trajetórias dos veículos. Seria mais adequado mapear as coordenadas das trajetórias nas coordenadas x e y da imagem, realizando uma transformação em cada quadro do vídeo semelhante à exemplificada na Figura 45.

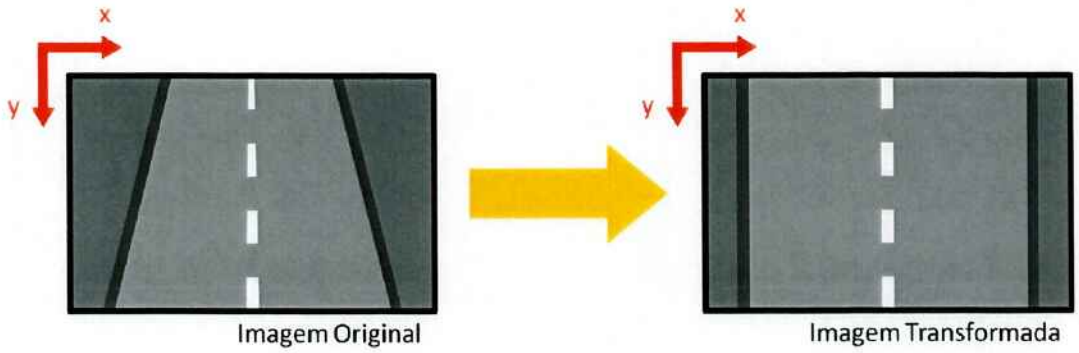


Figura 45 - Transformação de coordenadas desejada

Esse tipo de transformação já está implementado em OpenCV, que é a biblioteca de processamento de imagens utilizada neste trabalho. Utilizando as rotinas de OpenCV para realizar a transformação no vídeo utilizado até então, obtém-se o resultado da Figura 46.

(a) Quadro n° 50

(b) Quadro n° 100

(c) Quadro n° 150

Figura 46 - Aplicação da transformação de coordenadas utilizando rotinas de OpenCV

2.4.2 Procurando por Padrões ("Template Matching")

Para o algoritmo de "tracking" funcionar com múltiplos veículos, é preciso individualizá-los em cada imagem do vídeo. Em outras palavras, é preciso identificar com precisão a região da imagem que envolve cada veículo. As imagens resultantes da subtração de fundo, após passarem pela transformação de coordenadas descrita anteriormente, devem ser varridas a procura de regiões da imagem que se assemelham a veículos. Para fazer isso é necessário utilizar uma imagem auxiliar que representa - ou aproxima - o padrão de um veículo. A Figura 47 demonstra o resultado da subtração de fundo combinado à transformação de coordenadas. A Figura 48 demonstra exemplos de "templates" para representar o formato dos veículos no vídeo utilizado.

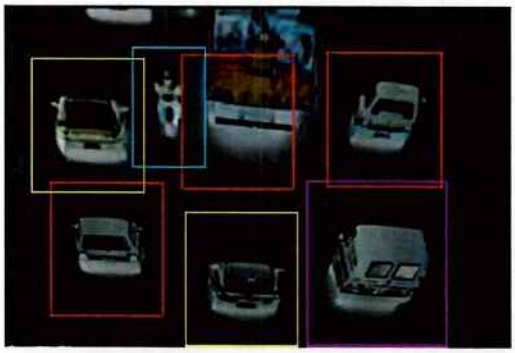
Figura 47 - Quadro n° 50 do vídeo, após subtração de fundo e "warp"

Figura 48 - "Templates" que aproximam o tamanho e forma dos veículos

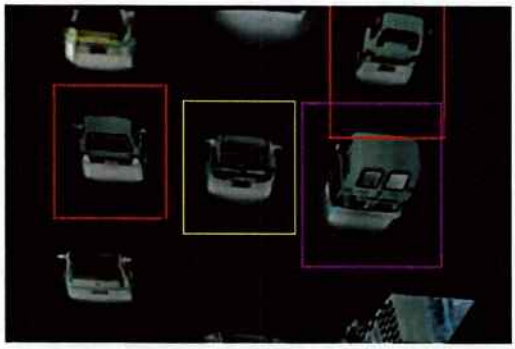
A próxima etapa é percorrer o quadro do vídeo, sobrepondo os "templates" ao mesmo. Para cada posição do "template" sobre o quadro, calcula-se o coeficiente de correlação entre a intensidade dos píxeis nas duas imagens. Se o "template" for adequado, o valor desse coeficiente de correlação deve ser mais elevado nas posições em que o "template" alinha-se aos veículos.

Os valores do coeficiente de correlação podem ser armazenados numa nova imagem em tons de cinza, em que os tons mais claros indicam valores altos, e os tons mais escuros indicam valores baixos. Essa imagem será chamada de Imagem de Correlação.

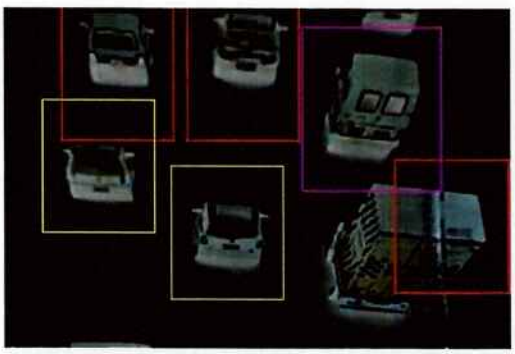
Percorrendo a Imagem de Correlação e determinando a posição dos máximos locais, determina-se a posição dos veículos. A Figura 49 mostra o resultado desse processo para alguns quadros do vídeo.



(a) Quadro n° 50



(b) Quadro n° 100



(c) Quadro n° 150

Figura 49 - Resultados do processo de "template matching" para alguns quadros do vídeo

2.4.3 Resultados da Delimitação

A delimitação de veículos não foi plenamente satisfatória. Carros e motocicletas foram identificados com relativo sucesso. A variedade de formas e tamanhos dos caminhões dificulta a utilização de "templates" para identificar essa categoria de veículo. Sendo assim, os caminhões foram desprezados neste trabalho.

Outro problema identificado foi a sobreposição de veículos. O algoritmo não identifica os veículos que sobrepõem ou são sobrepostos por outros veículos.

A conclusão que se tira desta etapa é que o processo de "template matching" não é robusto o suficiente para delimitar os veículos com a precisão desejada. Apesar disso, optou-se por dar sequência ao projeto a fim de não comprometer as etapas seguintes.

2.5 Realizando "Tracking" de Pontos (Fluxo Óptico)

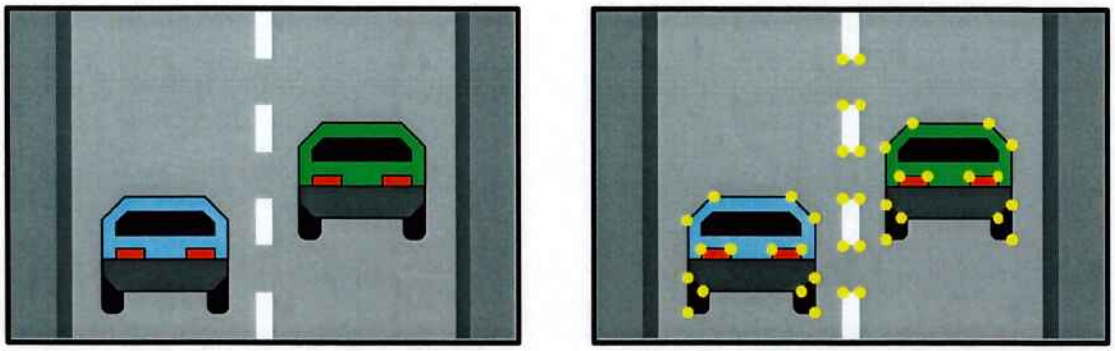
Antes de poder realizar "tracking" de veículos é necessário conseguir realizar "tracking" de pontos. Se for possível encontrar em cada quadro da imagem um conjunto de píxeis adequados para realizar "tracking" (píxeis de "canto", como explicado em capítulos anteriores), basta agrupar os píxeis correspondentes ao mesmo veículo para realizar "tracking" desse veículo. A idéia é utilizar o método de delimitação de veículos, mostrado anteriormente, para agrupar os píxeis pertencentes ao mesmo veículo. Nesta seção apenas o "tracking" de pontos será tratado.

A biblioteca de processamento de imagens OpenCV, utilizada neste trabalho, apresenta rotinas já implementadas para realizar "tracking" de pontos.

Neste trabalho foi utilizada uma implementação do algoritmo de Lucas-Kanade [4] [5], em OpenCV, para realizar "tracking" de cantos selecionados dos quadros do vídeo.

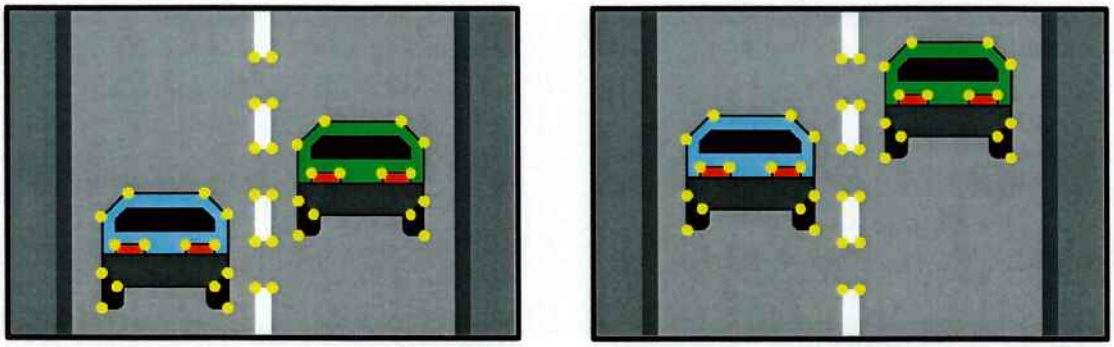
O processo começa ao selecionar píxeis do quadro que correspondem a cantos, como esquematizado na Figura 50. Esses cantos são armazenados num vetor.

Em seguida, o quadro seguinte do vídeo é analisado, e os píxeis equivalentes aos cantos do primeiro quadro são localizados e armazenados num novo vetor. Isso estabelece uma correspondência entre os cantos do quadro atual e os cantos do quadro anterior, como mostra a Figura 51. Desta forma, é possível determinar o deslocamento dos cantos selecionados no primeiro quadro.



(a) Quadro do vídeo (b) Cantos selecionados

Figura 50 - Selecionando cantos na imagem



(a) Quadro n (b) Quadro n+1

Figura 51 - Fazendo correspondência entre cantos de dois quadros consecutivos

2.6 Realizando "Tracking" de Veículos

A partir dos resultados dos processos de "template matching" e fluxo óptico, é possível acompanhar as trajetórias dos veículos do vídeo.

Para cada veículo encontrado num quadro do vídeo, analisam-se os cantos que estão dentro da região delimitada pelo "template" utilizado. Isso está representado na Figura 52.

Os deslocamentos dos cantos dentro do "template", entre o quadro anterior e o quadro atual, são armazenados num vetor. Esse vetor é então organizado em ordem crescente, segundo um algoritmo de ordenação. A mediana desse vetor é então tomada como o deslocamento do veículo entre o quadro anterior e o quadro atual. Esse deslocamento é utilizado para estimar a próxima posição do veículo. Esse procedimento assume que a frequência de repetição dos quadros do vídeo é alta o suficiente para que o deslocamento real do veículo entre dois quadros consecutivos seja pequeno.

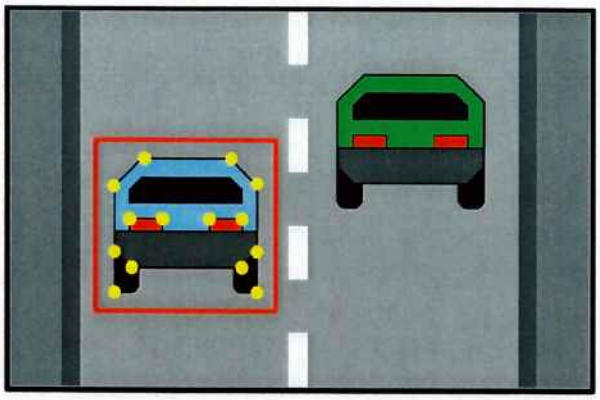


Figura 52 - Analisando somente os cantos dentro do "template"

A estimativa da próxima posição é armazenada para cada veículo do quadro atual.

Sabendo a posição que cada veículo ocupará no próximo quadro, é possível acompanhar suas trajetórias, registrando cada posição e velocidade numa estrutura de dados. Neste projeto foi utilizada uma lista ligada para armazenar as trajetórias de cada veículo. Uma representação dessa lista ligada está na Figura 53.

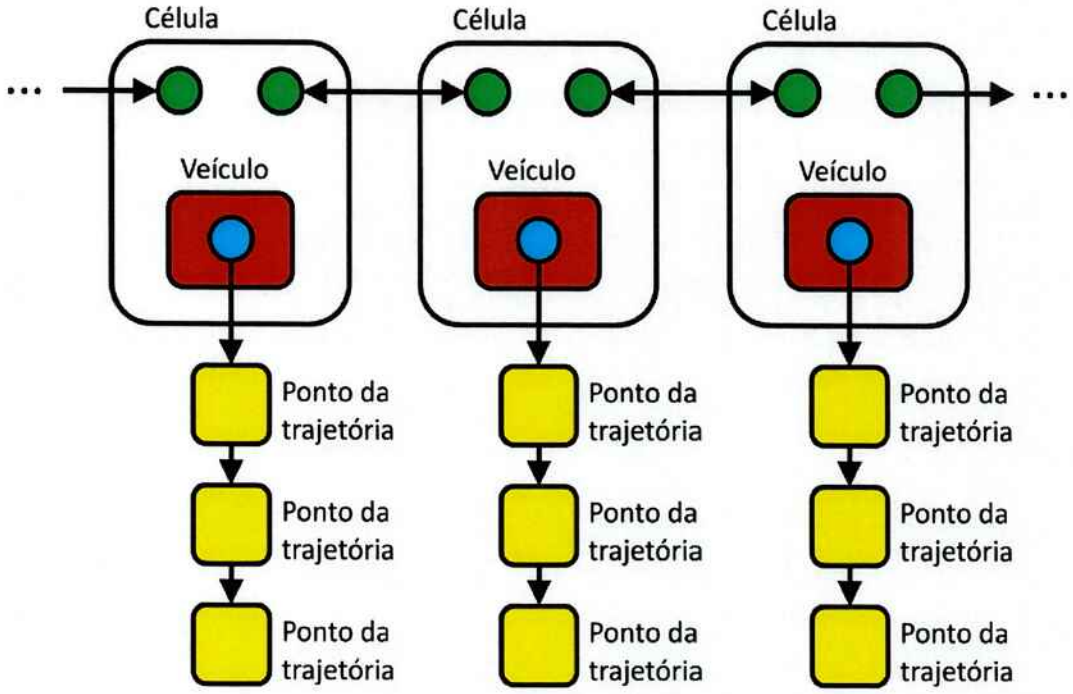


Figura 53 - Lista ligada com as informações sobre as trajetórias de cada veículo

A cada veículo encontrado pelo processo de "template matching" deve-se decidir se este é um novo veículo ou um veículo que estava anteriormente em outra posição. Para fazer isso, basta comparar a posição atual do veículo com as posições

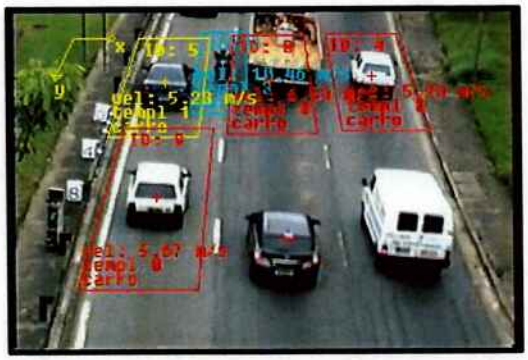
estimadas de todos os veículos do quadro anterior. Se houver um veículo cuja posição estimada anteriormente for próxima da posição atual, então não foi encontrado um novo veículo, mas sim um novo ponto da trajetória de um veículo já existente. Caso contrário, um novo veículo deve ser adicionado à lista.

O procedimento descrito nos parágrafos anteriores é suficiente para registrar as posições de todos os veículos quadro a quadro do vídeo. Para estimar a velocidade de cada veículo comparam-se as posições dos mesmos a cada 10 quadros. Esse intervalo foi escolhido arbitrariamente. A velocidade média do veículo é então calculada dividindo-se o deslocamento espacial nesses 10 quadros pelo tempo decorrido, que é obtido a partir da frequência de repetição dos quadros (30 quadros por segundo para o vídeo utilizado neste projeto). Desta forma, a velocidade de cada veículo é atualizada a cada 10 quadros.

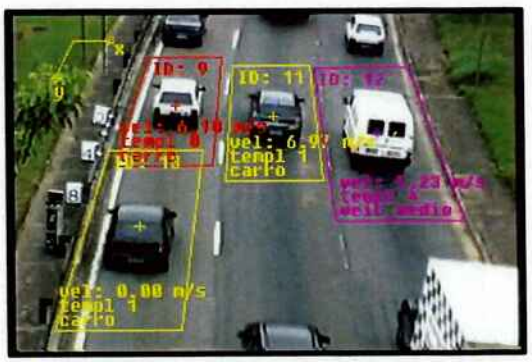
$$velocidade\ média = \frac{((posição\ no\ quadro\ n) - (posição\ no\ quadro\ n - 10))}{tempo\ decorrido\ entre\ 10\ quadros} \quad (8)$$

Seguindo o raciocínio anterior, preenche-se a lista ligada até o último quadro do vídeo. No final do processo, basta percorrer as células da lista ligada para extrair as trajetórias de cada veículo.

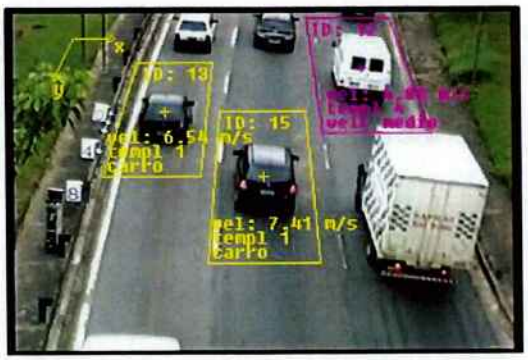
O resultado final, além de ser registrado no arquivo de texto como determinado nos objetivos, foi representado no próprio vídeo, como mostra a Figura 54.

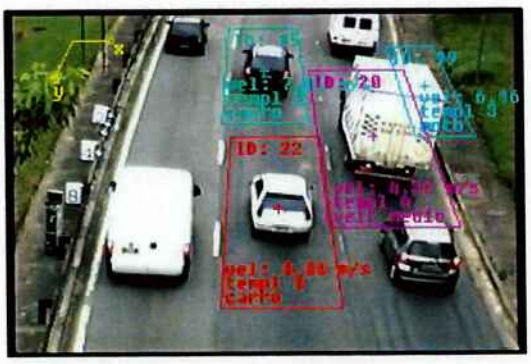


(a) Quadro n° 50

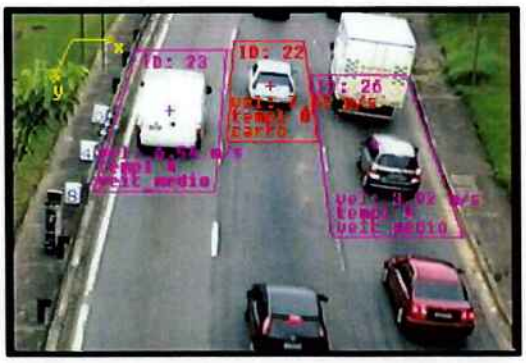


(b) Quadro n° 100

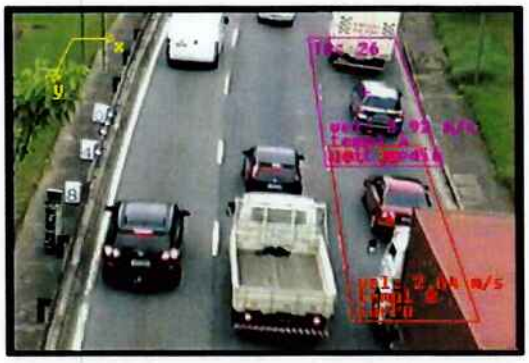




(c) Quadro n° 150



(d) Quadro n° 200



(e) Quadro n° 250

(f) Quadro n° 300

Figura 54 - Resultado final do programa para alguns quadros do vídeo

Os resultados mostram o reconhecimento de alguns veículos em alguns quadros do vídeo. As regiões delimitadas ao redor dos veículos correspondem aos "templates" utilizados. O número na parte superior de cada veículo é um parâmetro que identifica o veículo dentro da lista ligada.

Dentro das limitações evidenciadas pelo processo de "template matching" os resultados obtidos foram satisfatórios. A maior parte dos veículos foi identificada com sucesso. Caminhões não foram levados em consideração neste projeto, apenas carros, veículos de porte médio e motocicletas. A sobreposição de veículos, mencionada anteriormente, impediu a correta identificação de alguns veículos.

3 O ALGORITMO FINAL

Todas as etapas descritas no decorrer deste trabalho foram separadas em dois programas distintos. O primeiro deles serve para gerar a imagem de fundo que será utilizada na subtração de fundo. O segundo programa é o programa principal, e concentra todas as etapas dentro da região tracejada da Figura 18.

Informações detalhadas sobre os dois programas, bem como os códigos fonte e os vídeos e imagens utilizados neste projeto podem ser encontrados no seguinte site:

http://www.lps.usp.br/~hae/projform/2010_gabriel_ramires

4 CONCLUSÕES

Os resultados esperados para este projeto foram parcialmente alcançados. No decorrer da implementação as fragilidades dos métodos e algoritmos escolhidos foram evidenciadas, levando a alterações no escopo do projeto.

A técnica de subtração de fundo necessita de uma imagem de fundo previamente selecionada. Isso exige um pré-processamento do vídeo para recuperar a imagem de fundo. Além disso, a técnica utilizada para recuperar a imagem de fundo exige que todo o vídeo seja carregado na memória do computador, o que dificulta o uso de vídeos de alta resolução.

Quanto à técnica de "template matching", a detecção de caminhões foi ignorada devido à grande variedade de formas e tamanhos dos caminhões. Além disso, a sobreposição de veículos impede a correta detecção da posição de cada veículo pelos "templates".

A etapa de "tracking" de pontos conta com rotinas já programadas de OpenCV, e não ofereceu grandes dificuldades.

A etapa de "tracking" de veículos depende do resultado do processo de "template matching", e apresentou resultados satisfatórios dentro das limitações já mencionas do "template matching".

A situação atual deste projeto não é suficiente para implementar um sistema de monitoramento de trânsito, visto todas as deficiências já mencionadas. Porém, este projeto serve como ponto de partida para projetos futuros no mesmo tema, que se utilizem de técnicas mais sofisticadas para individualizar veículos.

5 REFERÊNCIAS

[1] Acessoria Técnica da Presidência do DETRAN/RS – Estatística e planejamento. Disponível em: http://www.detran.rs.gov.br. Acesso em: abril de 2010.

[2] Acessoria Econômica Interna da FENABRAVE. Anuário do setor de distribuição de veículos automotores no Brasil, 2009. São Paulo, abril de 2010.

[3] Shi, Jianbo; Tomasi, Carlo. Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition, Seattle, junho de 1994.

[4] Bradski, Gary; Kaebler, Adrian. Learning OpenCV, computer vision with the OpenCV library. 2. ed. Sebastopol, CA: Editora O'Reilly, Setembro de 2008.

[5] Bouguet, Jean-Yves. Pyramidal implementation of the Lucas Kanade feature tracker, description of the algorithm. Local: Intel, Micropocessor Research Labs.

6 BIBLIOGRAFIA

Bradski, Gary; Kaebler, Adrian. Learning OpenCV, computer vision with the OpenCV library. 2. ed. Sebastopol, CA: Editora O'Reilly, Setembro de 2008.

Han, Mei; Xu, Wei; Tao, Hai; Gong, Yihong. An algorithm for multiple object trajectory tracking. Santa Cruz, CA: University of California, 2004.