COS756 - Processamento de Imagens
Leonardo Neves
Medidor de velocidade para monitoramento de veículos

Introdução

Este relatório tem como objetivo descrever o desenvolvimento do de uma aplicação para medição de velocidade de veículos em tempo real, desenvolvida como projeto final da material COS756, Processamento de Imagens. Serão discutidas as duas abordagens implementadas e os passos intermediários que levaram ao resultado. Grande parte da implementação se baseou no artigo "Speed Detection Camera System using Image Processing Techniques on Video Streams"[1]

Desenvolvimento

O projeto foi desenvolvido em linguagem Python utilizando-se da biblioteca Numpy.
Foram postas em prática duas abordagens, optando-se pela segunda como solução final do projeto por este apresentar um resultado mais confiável e de computação mais rápida.

a. Lucas-Kanade

A primeira abordagem foi implementar o algoritmo de fluxo ótico Lucas Kanade. Utilizando-se do algoritmo de Harrison para detecção de pontos úteis na imagem, já convertida para escala de cinza, e aplicando-se o filtro de sobel para calcular as matrizes necessárias para o fluxo, o algoritmo era capaz de calcular o vetor de velocidade dos pontos e buscar o ponto no frame seguinte.
Por outro lado, o tempo de processamento da implementação grande, comparando-se com a segunda abordagem, sendo perceptivel no video um certo atraso na reprodução, além de que, quando a imagem não se mostrava bem controlada, o algoritmo divergia, provavelmente por algum problema na implementação.

b. Monitoramento do Centróide

Como segunda abordagem, foi seguido o procedimento descrito em [1]. Primeiramente, a imagem foi tratada e convertida em escalas de cinza, comparada ao seu estado em t - 1 e t-2, além de comparada ao fundo da imagem. Além disso, também foi aplicada a formula para o calculo de sombra na imagem. Podemos ver os ganhos obtidos por cada uma dos processamentos nas imagens 1 à 4. A imagem 5 mostra o resultado obtido.

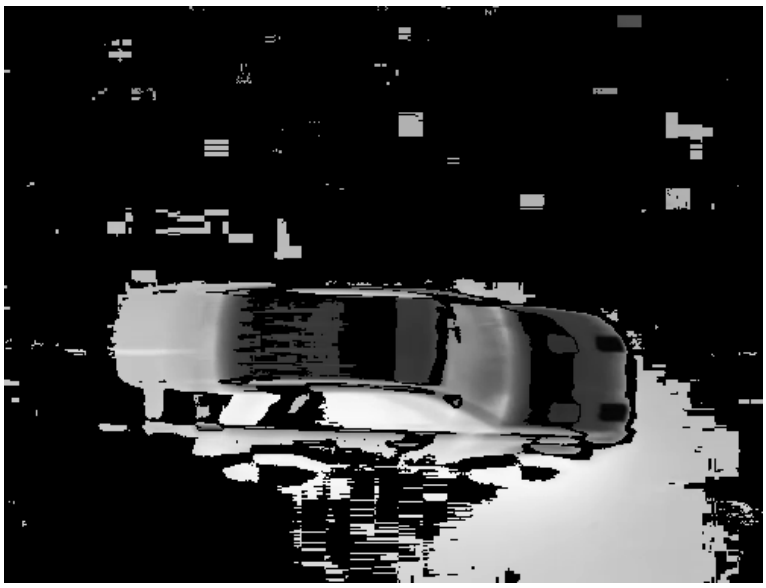

Imagem 1: $I_t - I_{t-1}$

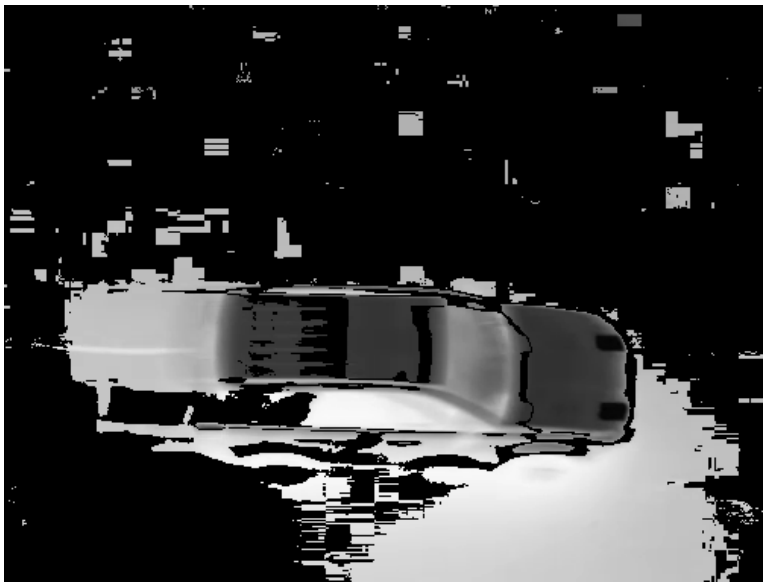

Imagem 2: $I_t - I_{t-2}$

Imagem 3: I_t – Background

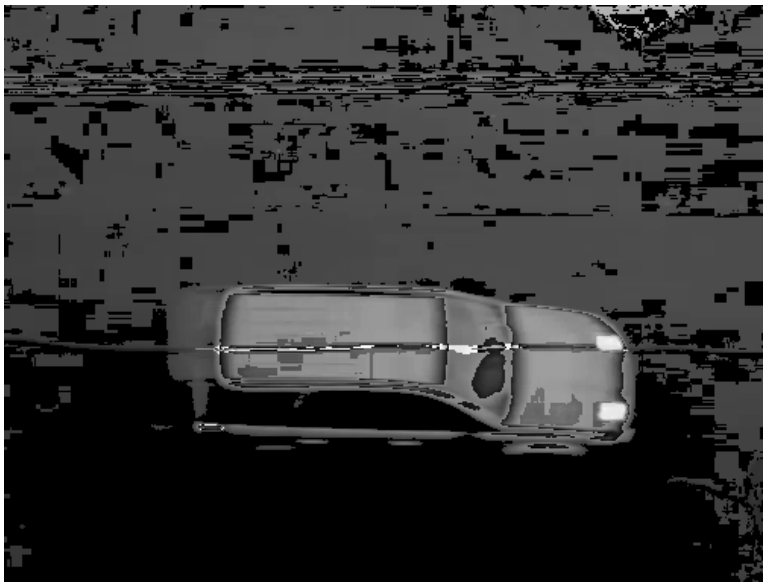

Imagem 4: I_t/Background

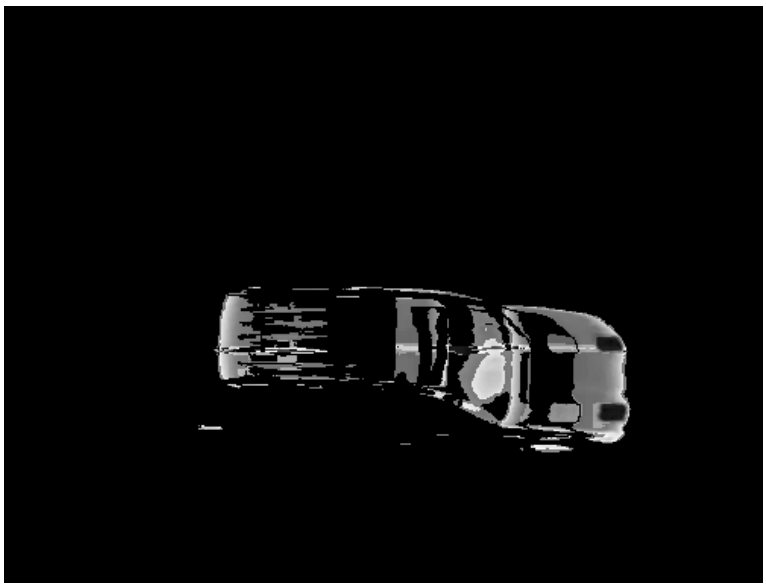

Imagem 5: Resultado

Através das imagens, é possível separar um veículo em movimento do fundo da imagem e de outros objetos estáticos. Com o veículo separado do resto dos outros objetos , é possível aproxima-lo à área que ele ocupa, como um retângulo, apresentado na imagem 6.

Imagem 6: Área do veículo

Para se calcular a área da imagem, pegamos as coordenadas de P_0 como ($X_{min,}$ Y_{min}) e P_1 como ($X_{máx}$,$Y_{máx}$). No caso de existirem mais de um objeto na imagem, através de um limiar do tamanho do objeto a ser definido no código, este divide a imagem em 4 quadrantes e, recursivamente, calcula os objetos de cada quadrante. Dessa forma,

podemos ver, nas imagens 7 e 8, que o algoritmo separa os objetos e monitora cada um deles.

Imagem 7: Imagem com dois veículos

Imagem 8: Áreas Distintas

Para acompanhar o objeto em frames consecutivos, calcula-se o centroide do retângulo e, através da fórmula da distância euclidiana, atualiza-se o centroide para o do novo centroide mais próximo.
Monitorando-se os veículos, sabemos quando um novo veículo entra em cena e sai, calculando assim o número de frames. Definindo a distância percorrida, a velocidade de cada veículo é calculada facilmente com o tempo por frame multiplicado pelo número de frames.

Resultados

Após implementar o algoritmo e otimizar o código com o perfilador kernpref para Python, o tempo de computação para cada frame foi de aproximadamente 40ms. O cálculo se mostrou preciso e a identificação de veículos também, como aparece na imagem 9. A área determinada para o cálculo aparece em destaque, com a velocidade calculada para p último veículo a atravessar a área no canto superior esquerdo da figura.

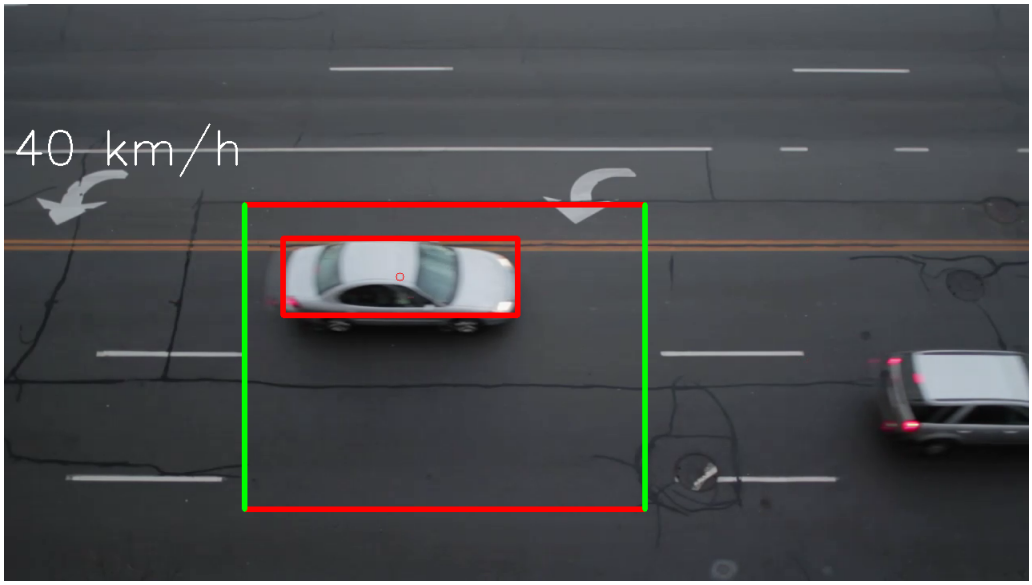

Imagem 9: Detector de velocidade em funcionamento

Conclusão

Ao longo do projeto foram estudados e implementados algoritmos de detecção de pontos de interesse como o algoritmo de Harris, cálculo de fluxo ótico como o de Lucas-Kanade e, por mais que mais elegantes, estas soluções não foram levadas adiante pelo custo computacional e complexidade desnecessários para resolver o problema. A abordagem utilizada, portanto, foi muito mais simples e se mostrou bastante eficaz para solucionar o problema de monitorar e calcular a velocidade de um veículo em certa figura. Os resultados, se adaptados e agregados a um sistema de detecção e placas, poderia ser efetivamente utilizado em sistemas de vigilância e controle de trânsito.

O código para o resultado final deste projeto se encontra em https://github.com/lrmneves/speedtracker

Referências

[1] Osman Ibrahim, Hazem ElGendy, and Ahmed M. ElShafee, "Speed Detection Camera System using Image Processing Techniques on Video Streams," *International Journal of Computer and Electrical Engineering* vol. 3, no. 6, pp. 771-778, 2011.