Isabel Pugliese, Rui Martins, Sofia Pereira

Sistema de deteção de veículos na mesma faixa

Tópicos Avançados de Processamento Digital de Imagem

Orientador: André Teixeira Bento Damas Mora, Professor doutor,
Universidade Nova de Lisboa

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Junho, 2021

Índice

Lista de Figuras

Listagens

Capítulo 1

Introdução

Os veículos inteligentes têm sido um grande tópico para os investigadores durante a última década. As tecnologias de sensores em veículos inteligentes têm-se desenvolvido dramaticamente, incluindo radares, lidar, e visão por computador. Particularmente, tecnologias de processamento de imagem têm sido amplamente utilizadas nos últimos anos, e as câmaras são agora mais baratas, mais pequenas e de maior qualidade do que antes. Além disso, um forte desenvolvimento de GPU's e hardware permitem abordagens de visão por computador para a detecção de veículos e faixas de rodagem em tempo real [1].

Muitos investigadores estudaram e propuseram as estratégias para seguir veículos e faixas, mas a maior parte das obras têm sido feito separadamente. Para a detecção da faixa de rodagem, há muitas características que podem extrair os pixeis de marcação da faixa de rodagem de uma imagem, como elemento de borda, elemento de cor, elemento de Haar, e transformada de hough. Entre as abordagens para a deteção de veículos, há também muitos métodos com base em características, tais como características Haar e redes neuronais artificiais, Histograma de gradientes orientados e classificadores em cascata. Estes algoritmos propostos, que são poderosos e robustos, funcionam em tempo real com alta precisão. No entanto, apenas detecção de estradas ou de veículos não é suficiente para apoiar o sistema de assistência à condução, avisando sistema, ou sistema de mudança de faixa [1].

Dito isto, o objectivo deste projecto é desenvolver uma aplicação, que a partir de um vídeo e utilizando a GPU para processamento de imagem, encontra faixas e veículos, e marca a vermelho os veículos que estão na nossa faixa e a verde os

outros.

O sistema a implementar terá as seguintes especificações:

- O sistema deve utilizar um vídeo gravado por uma câmara colocada no interior do carro;
- O processo de deteção das faixas de rodagem deve fazer uso da transformação de Hough, implementado em OpenCL;
- O processo de detecção de veículos deve ser baseado nos Classificadores em Cascata (biblioteca OpenCV), utilizando um classificador previamente treinado para detectar carros;
- A interface do sistema deve mostrar o vídeo, marcar as faixas (linha azul) e os veículos encontrados (caixa vermelha ou verde), exemplificado na figura 1.1;
- Os veículos devem ser marcados com um rectângulo em vermelho se estiverem entre as nossas dois linhas da via ou em verde noutros casos.

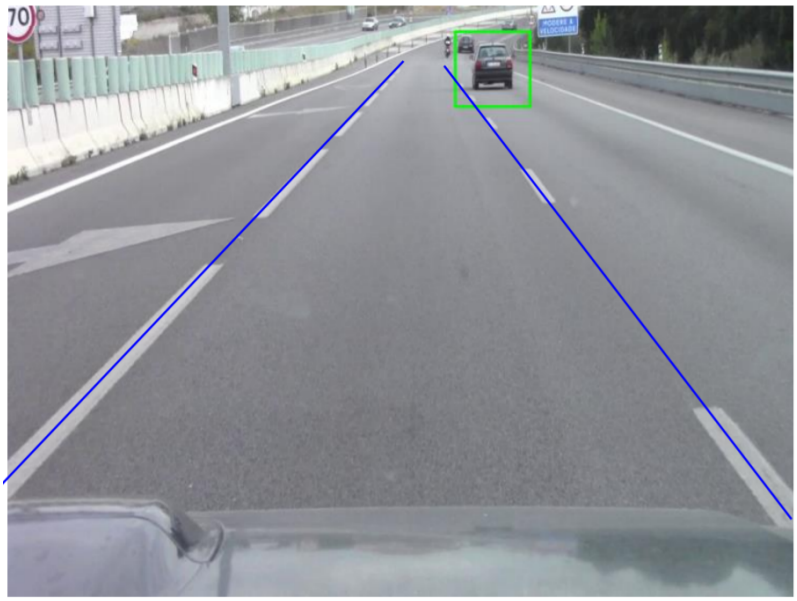

Figura 1.1: Exemplo da deteção de um veículo e das faixas

Capítulo 2

Desenvolvimento do sistema

O seguinte diagrama representa a arquitetura do sistema implementado.

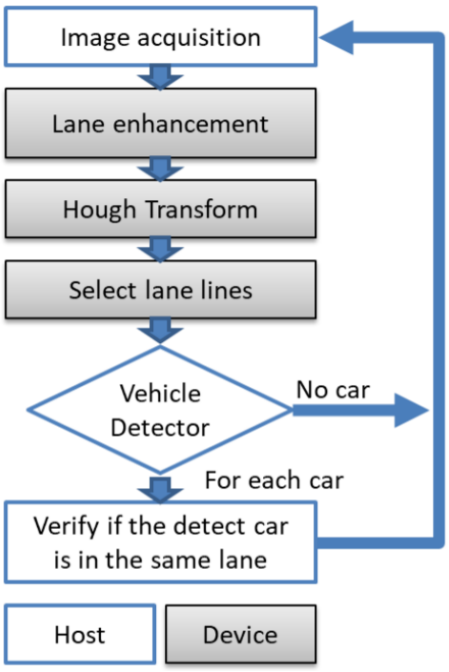

Figura 2.1: Arquitetura do sistema

Uma das especificações do sistema é a utilização da transformada de Hough, que deve ser implementada em OpenCL com o intuito de aproveitar a capacidade computacional da GPU e executar o algoritmo num tempo muito menor. A transformada de Hough é uma técnica muito útil para identificar formas simples como linhas e círculos numa imagem. Uma imagem simples pode ser representada

por uma equação simples com apenas alguns parâmetros. As linhas precisam de dois parâmetros (inclinação e intercepção); os círculos precisam de três (x_centro, y_centro, raio). A transformada de Hough envolve primeiro a identificação das arestas e depois a identificação da forma [2] [3].

2.1 Algoritmo de deteção das faixas de rodagem

2.1.1 aquisição da frame

Cada frame do vídeo é analisada. Primeiramente a frame é redimensionada e convertida para o espaço de cor RGBA. Adquire-se o formato da frame (height, width, padding) e atribui-se memória para o output onde o resultado pode ser copiado (list.2.1).

Listagem 2.1: Leitura da frame

```
1  cap = cv.VideoCapture(pathname + filename)
2
3  while (True):
4      # Capture frame-by-frame
5      ret, frame = cap.read()
6
7      # resizing for faster detection
8      frame = cv.resize(frame, (840, 440))
9      frame = cv.cvtColor(frame, cv.COLOR_BGR2RGBA)
10     imgOut = np.copy(frame)
11     imgOut_1 = np.copy(frame)
12
13     height = imgIn.shape[0]
14     width = imgIn.shape[1]
15     padding = imgIn.strides[0] - imgIn.shape[1] * imgIn.strides[1]
16     arrayIn = np.zeros(180*180)
17     arrayIn_1 = np.zeros(180 * 180)
```

2.1.2 Realce das faixas de rodagem

Para realçamento das faixas de rodagem, executa-se a função grey_image2D, implementada em OpenCL, que converte a frame para uma escala de cinzentos (list.2.2). Seguidamente aplica-se uma binarização também implementada em OpenCL (list.2.3). Note-se que na list.2.3 a linha 7 define o treshold da binarização.

Listagem 2.2: Grey Image Kernel

```
kernelName_2 = hough.grey_image2D

kernelName_2.set_arg(0, imgInBuffer)
kernelName_2.set_arg(1, imgOutBuffer)
kernelName_2.set_arg(2, np.int32(width))  # Width
kernelName_2.set_arg(3, np.int32(height))  # Height
```

Listagem 2.3: Binary Treshold Kernel

```
kernelName_3 = hough.binary_threshold_image2D

kernelName_3.set_arg(0, imgOutBuffer)
kernelName_3.set_arg(1, imgOutBuffer_1)
kernelName_3.set_arg(2, np.int32(width))  # Width
kernelName_3.set_arg(3, np.int32(height))  # Height
kernelName_3.set_arg(4, np.int32(180))      #Treshold
```

A figura 2.2 mostra o resultado das operações efetuadas à frame.

Figura 2.2: Frame binarizada

2.1.3 Deteção da faixa de rodagem

Para a deteção da faixa de rodagem do veículo, aplica-se a tranformada de Hough, implementada em OpenCL. A linha 7 (list.2.4) define o treshold da transformada.

Após a transformada, calculam-se os rho's e theta's das faixas de rodagem e desenham-se as linhas a azul.

Listagem 2.4: Hough Kernel

```
kernelName = hough.line_seg_image2D

kernelName.set_arg(0, imgOutBuffer)
kernelName.set_arg(1, np.int32(imgIn.shape[1]))  # Width
```

```
kernelName.set_arg(2, np.int32(imgIn.shape[0]))   # Height
kernelName.set_arg(3, np.int32(padding))          # Padding
kernelName.set_arg(4, np.int32(255))              # Threshold
kernelName.set_arg(5, memBuffer)
kernelName.set_arg(6, memBuffer_1)
```

A figura 2.3 mostra o resultado das operações efetuadas neste processo.

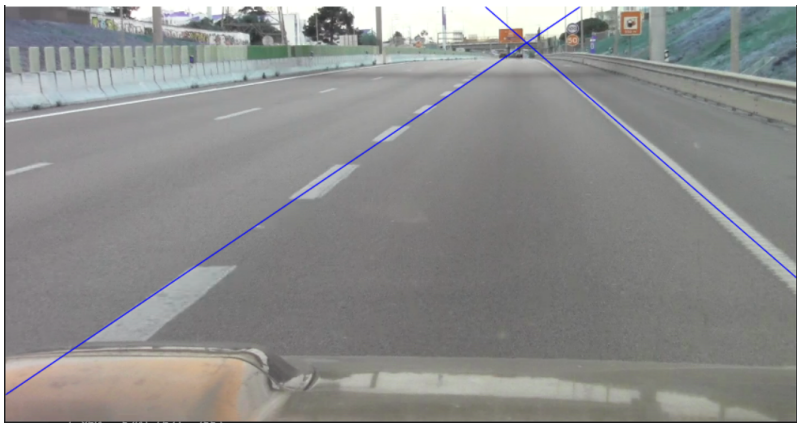

Figura 2.3: Deteção das faixas de rodagem

2.2 Algoritmo de deteção de veículos

A detecção de objectos utilizando classificadores em cascata Haar baseados em características é um método eficaz de detecção de objectos. É uma abordagem baseada em machine learning onde uma função em cascata é treinada a partir de muitas imagens positivas e negativas. É depois utilizada para detectar objectos noutras imagens [4]. Este algoritmo foi implementado em OpenCV.

OpenCV fornece um método de treino ou modelos pré-treinados, que podem ser lidos usando o método CascadeClassifier.

O algoritmo utiliza modelos em cascata Haar pré-treinados para detectar veículos numa imagem. Primeiro, é criado um CascadeClassifier e o ficheiro XML necessário é carregado (list.2.5).

Listagem 2.5: Classificadores em Cascata

```
args = parser.parse_args()
car_1_cascade_name = args.car_1_cascade
car_2_cascade_name = args.car_2_cascade
car_3_cascade_name = args.car_3_cascade
car_1_cascade = cv.CascadeClassifier()  # import xml
car_2_cascade = cv.CascadeClassifier()  # import xml
car_3_cascade = cv.CascadeClassifier()  # import xml
```

```
8
9  # Load the cascades
10 if not car_1_cascade.load(cv.samples.findFile(car_1_cascade_name)):
11     print('--(!)Error loading car_1 cascade')
12     exit(0)
13
14 if not car_2_cascade.load(cv.samples.findFile(car_2_cascade_name)):
15     print('--(!)Error loading car_2 cascade')
16     exit(0)
17
18 if not car_3_cascade.load(cv.samples.findFile(car_3_cascade_name)):
19     print('--(!)Error loading car_3 cascade')
20     exit(0)
```

Depois, a detecção é feita utilizando a função detectAndDisplay, que devolve rectângulos de limite para os carros detectados (list.2.6).

Listagem 2.6: Função de deteção de veículos

```
1 car_det.detectAndDisplay(frame, m, b, m_1, b_1)
```

Esta função recebe como parametros de entrada a frame analisada e os limites da faixa de rodagem, de modo calcular se o veículo detetado se encontra na mesma faixa que a nossa. A figura 2.4 mostra o resultado obtido.

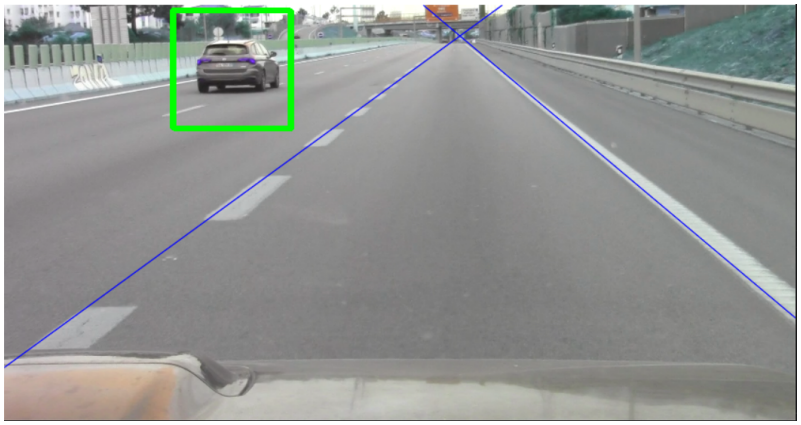

Figura 2.4: Deteção de carro

Capítulo 3

Conclusão

Bibliografia

[1] V.-Q. Nguyen, C. Seo, H. Kim e K. Boo. "A study on detection method of vehicle based on lane detection for a driver assistance system using a camera on highway". Em: *2017 11th Asian Control Conference (ASCC)*. 2017, pp. 424–429. DOI: 10.1109/ASCC.2017.8287207.

[2] *Hough Line Transform*. URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[3] G.-J. Van den Braak, C. Nugteren, B. Mesman e H. Corporaal. "Fast hough transform on GPUs: Exploration of algorithm trade-offs". Em: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer. 2011, pp. 611–622.

[4] *Casscade Classifiers*. URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.