Deteção de linhas para condução autónoma com OpenCV

André Ribeiro Almeida 88960

Introdução

A condução autónoma é a capacidade de movimentação correta de um veículo com pouca ou nenhuma intervenção por parte humana. A chegada deste feito é um marco que é extremamente antecipado pelos apreciadores de tecnologia. Contudo, e principalmente devido a questões de segurança do condutor, a criação deste tipo de tecnologia é um processo demorado e complexo. Com este projeto, tenciona-se fornecer uma primeira visão de como a etapa de processamento de imagem é realizada. Este projeto foi realizado no âmbito da unidade curricular de Computação Visual do 4º ano do curso Mestrado Integrado em Engenharia de Computadores e Telemática.

Organização do trabalho

O processamento de imagem está dividido em múltiplas etapas de execução. Primeiramente, a imagem é transposta para escala de cinzentos, uma vez que a cor não é particularmente importante para o objetivo do trabalho. Seguidamente, é realizada uma operação de desfoque na imagem, com o intuito de reduzir eventuais ruídos ou outras disrupções similares. Adiante neste trabalho será mencionado o exemplo do asfalto, que pode ser disruptivo na deteção das linhas. Em terceiro lugar, é realizada uma operação de *Canny Edge Detection*, por forma a detetar as fronteiras das linhas da estrada com o pavimento. Tal facilita a realização da última etapa de processamento. De seguida, para reduzir a imagem ao seu essencial - a própria estrada - é feito uma "máscara", cortando-se da imagem o que é supérfluo e desnecessário. Finalmente, a transformada de Hough, uma operação disponível nas bibliotecas de *Opencv*. Este último passo deteta segmentos de reta na imagem, sendo útil na obtenção do ângulo para o carro seguir na direção certa.

Ao longo deste trabalho, para cada etapa do processo, será mostrada a imagem resultante do mesmo. No processo da transformada de Hough, será desenhada, em cima da imagem original, as linhas que este detetou, no local onde foram detetadas.

A imagem base onde as operações serão realizadas é a seguinte:

Pré-processamento

Grayscale

Primeiramente, a imagem da estrada é convertida de RGB para escala de cinzentos, com o seguinte comando:

```
lane_image = cv2.imread(utils.get_abs_path(img_path))
gray = cv2.cvtColor(lane_image, cv2.COLOR_RGB2GRAY)
```

Resultando, então, na seguinte imagem:

Blurring

Por forma a realizar a etapa de *blurring*, existem diversas opções:

- Average
- Gaussian
- Median

Para qualquer destes métodos, é fornecida a opção de ajustar a intensidade:

```
blur = (blur_level, blur_level)
blur = cv2.GaussianBlur(
    src=gray, ksize=blur, sigmaX=0
)
```

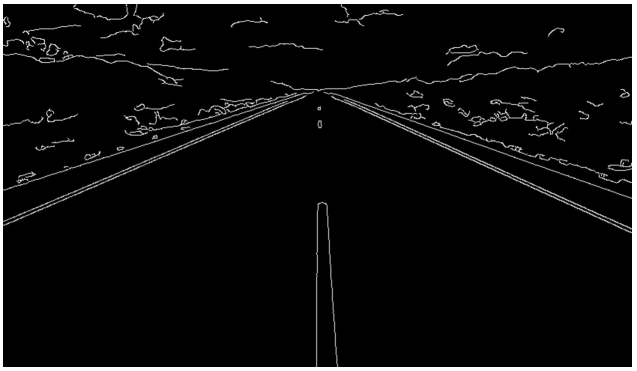
Sendo que *blur_level* é o tamanho do kernel de processamento que produz o desfoque: quanto maior o tamanho do kernel, entre os possíveis valores, maior o desfoque.
O valor default foi definido a 5 uma vez que, recorrendo a verificações empíricas, se mostrou eficaz para a maioria dos casos.

Canny Edge Detection

O procedimento *Canny Edge Detection* é efetuado com o intuito de detetar as linhas na estrada. Uma vez que estas linhas possuem elevado contraste, são facilmente detetadas:

```
canny_image = cv2.Canny(
                    image=blur,     threshold1=thresholds[0],
threshold2=thresholds[1]
)
```

Os *threshold* default são 50 e 150 para o *threshold1* e *threshold2*, respetivamente no procedimento por histerese. Contudo, outros valores próximos na ordem das dezenas são também aceitáveis, sendo possível ajustar estes valores se necessário.



Masking

Por forma a reduzir o espaço de análise da última etapa (Transformada de Hough) foi definida uma região de interesse da imagem para ser processada. Para tal, foi definido um conjunto de pontos que formam um polígono. Neste trabalho, para simplificar o processo, usou-se um triângulo, mas outros polígonos podem ser usados. Seguidamente, foi realizado um masking da seguinte forma:

```
polygon = [(0, 0), (0, .4), (.5, .78), (.99, .4), (.99, 0)]
…
polygon = np.array(polygon, np.int32)
mask = np.zeros_like(image)
cv2.fillPoly(mask, [polygon], 255)
cropped_image = cv2.bitwise_and(image, mask)
```

Nota: os valores dos pontos do triângulo são definidos em fração dos limites da imagem.

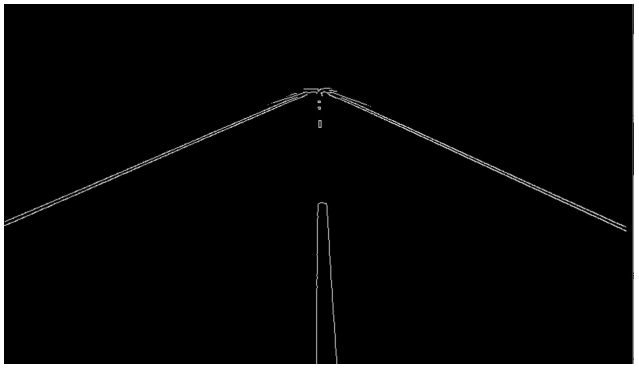
Exemplos:

- (0.5, 0.5) seria um ponto no centro;
- (1.0, 1.0) seria um ponto no pixel do canto superior esquerdo da imagem;
- (1.0, 0.0) seria um ponto no pixel do canto inferior esquerdo da imagem.

A "máscara" default é definida manualmente. Por forma a averiguar a melhor configuração possível consoante o ambiente, é necessário testar a câmara do carro. Para o caso mais genérico, ou seja, para definir uma máscara dinâmica conforme o ambiente, um estudo mais aprofundado é requerido.

Foi gerada então, neste trabalho, esta máscara:

Utilizando essa máscara, foi gerada a seguinte imagem, reduzida apenas aos seus pontos de interesse:



Mapping

Hough Transform

Com o objetivo de detetar as linhas, é usado o método das Transformadas de Hough. Na condução autónoma, idealmente, serve para:

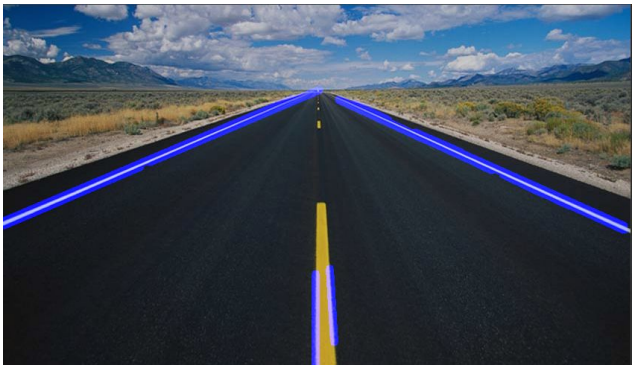
- Fazer o "desenho" das retas por estimativa das linhas da estrada;
- Realizar o cálculo do declive dessas mesmas retas;
- A partir do declive, calcular o ângulo de viragem.

Com base nas funções fornecidas pelo *Opencv*, existem 2 opções:

- usar a função simples de *hough;*
- usar a função probabilística de hough.

A primeira função é útil na questão da condução autónoma, em que o necessário é apenas o declive das mesmas para obter o ângulo de viragem. Contudo, neste trabalho, foi escolhida a função probabilística, uma vez que é mais eficiente em questões de esforço computacional.

Utilizando esse método foi possível chegar ao seguinte resultado:



Se considerado necessário, é possível ajustar a máscara para detetar mais ou menos do que o que está já detetado na estrada.

Resultados

No final do processamento, é possível obter linhas para qualquer estrada (salvo em algumas exceções, em que é necessário ajustar as configurações):

Figura 1.

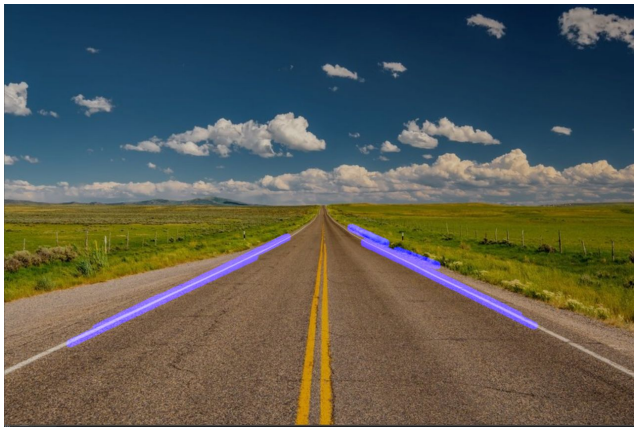


Figura 2.

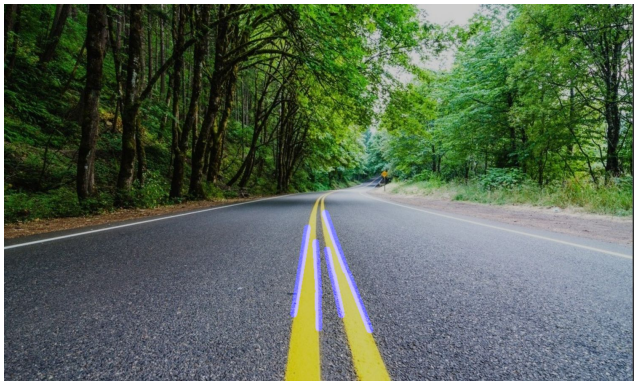


Figura 3.

Notas finais

Face ao largo espetro de estradas possíveis, é difícil encontrar os parâmetros certos de processamento de imagem "universais", que sirvam para tudo. Face a este problema, é fornecida uma configuração mais rápida e simples dos mesmos ao nível do código, de forma a ser mais rápido o estudo das combinações mais eficazes.

```
im_path = "road_photos/road7.jpg"
lane_detector(im_path,
              blur_level=5,
              blur_funct=cv2.GaussianBlur,
              thresholds=(85, 150),
                  polygon=[(0, 0), (0, .4), (.5, .78), (.99, .4), (.99, 0)]
              )
```

Exemplificando, a imagem 3 dos resultados foi processada com blur_level de 7, visto que num primeiro processamento com blur_level de 5 a gravilha do pavimento da estrada levou a serem desenhadas linhas onde não deviam existir:

Um pequeno extra foi adicionado, que imprime na consola o valor do ângulo final que o carro deve virar, baseado nos declives das linhas. Exemplificando, 0 graus significa seguir em frente, +90 significa virar à esquerda e -90 significa virar à direita.

Conclusão

Fazendo uso de ferramentas de *Opencv* para processamento de imagem, nomeadamente *Grayscale*, desfocamento (*blur*), *Canny Edge Detection*, masking e Hough Transform é possível realizar a deteção de linhas de estrada para condução autónoma. Não obstante, é necessária alguma configuração manual, consoante o ambiente onde se encontra o veículo. O ajuste dos parâmetros do processamento de imagem pode ser realizado em trabalhos futuros.

Referências

1. https://www.learnpython.org/en/Multiple_Function_Arguments
2. https://medium.com/@yogeshojha/self-driving-cars-beginners-guide-to-computer-vision-finding-simple-lane-lines-using-python-a4977015e232
3. https://docs.opencv.org/master/d6/d10/tutorial_py_houghlines.html