PSI3472 Concepção e Implementação de Sistemas Eletrônicos Inteligentes
Segundo Semestre de 2018 1º exercício-programa

Nas praças de pedágio das rodovias do Brasil, o valor do pedágio depende do tipo de veículo (moto, carro, utilitário, caminhão, ônibus, etc.). No caso de veículos grandes, o valor também depende do número de eixos rodantes (em contato com o solo) e do número de eixos suspensos.

O vídeo:

`http://www.lps.usp.br/hae/psi3472/ep1-2017/vid3.avi`

mostra a movimentação dos veículos numa praça de pedágio. Este vídeo tem duração de quase 10 minutos. Escolhemos alguns trechos desse vídeo em:

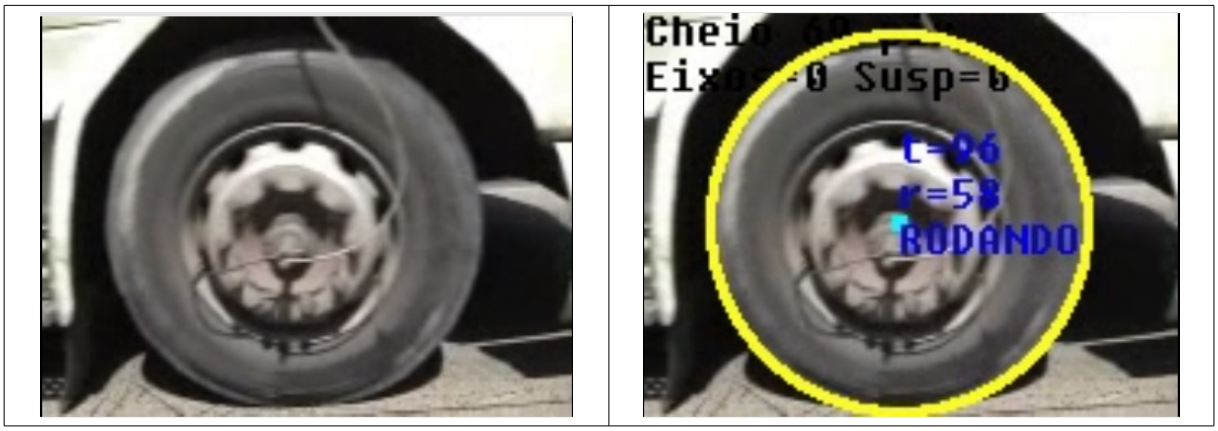
`http://www.lps.usp.br/hae/psi3472/ep1-2017/vid4.avi`

que tem duração de aproximadamente 1 minuto.

O objetivo final do projeto é identificar o tipo de veículo e contar o número de eixos rodantes e número de eixos suspensos (se o veículo for grande), como nos vídeos:

`http://www.lps.usp.br/hae/psi3472/ep1-2017/vid3-ped.avi`
`http://www.lps.usp.br/hae/psi3472/ep1-2017/vid4-ped.avi`

O objetivo deste exercício é fazer um pedaço desse projeto. Vamos apenas detectar os pneus dos veículos dos veículos grandes, como caminhões e ônibus. Não há necessidade de detectar os pneus dos veículos pequenos (pois o valor do pedágio destes veículos não depende do número de eixos).



Faça um programa detpneu que lê o vídeo vid4.avi e gera um outro vídeo pneu4.avi onde as rodas são detectadas e marcadas com círculos amarelos. Depois que o seu programa funcionar para vid4.avi, execute-o para vid3.avi para verificar se ele funciona também no vídeo longo.

Nota: Você pode usar a técnica que achar melhor para resolver este problema. Porém, a transformada de Hough pode ser uma boa solução. Neste caso, escreva a sua própria função Hough para círculos, sem usar as funções prontas do OpenCV ou de outras bibliotecas (nem pegar alguma função pronta na internet). Quem testar a função HoughCircles do OpenCV, provavelmente irá verificar que ela não funciona muito bem e que é necessário escrever a sua própria função de qualquer forma.

Neste projeto, vamos fazer os seguintes sub-programas que ajudarão resolver o problema final:

Aula4-Ex3 vale 2) Faça um programa "extrai" que lê um vídeo e extrai um quadro especificado como uma imagem. Vamos usar os quadros extraídos para testar o resto do processamento, pois é mais fácil desenvolver programa para processar uma imagem do que um vídeo.

```
>extrai vid4.avi 20 quad20.png
```

Extrai o quadro 20 do vídeo vid4.avi e grava como quad20.png

Aula4-Ex4 vale 3) Faça um programa gradiente que recebe um quadro (quad.png), calcula o gradiente e o grava como uma imagem complexa (grad.img). Deve permitir aplicar um filtro gaussiano antes de calcular o gradiente.

```
>gradiente quad.png grad.img 2
```

Calcula gradiente do quad.png e grava como imagem complexa quad.img, passando antes um filtro gaussiano de desvio-padrão 2 pixels.

Teste o seu programa para triang.png, triangc.png, fillcircle.png e fillcirclec.png.

Nota: Na biblioteca Cekeikon, o comando imp(imgx,"nome.img") grava uma imagem complexa (Mat_<CPX>). Os comandos "kcek mostrax" e "kcek campox" mostram uma imagem complexa extensão .img. Nota: Se der nome de arquivo "win" no comando de impressão, a imagem é mostrada na tela, em vez de ser impressa. A função:

    mostra(imgx);

mostra imagem complexa imgx no sistema HSI (magnitude=Intensidade, ângulo=Hue). A função:

    Mat_<COR> imgc = campox(imgx, fator, espaco)

converte uma imagem complexa imgx como "campo com flechas" (fator é fator de multiplicação de magnitude, espaco = distancia em pixels entre dois números complexos).

Nota: Como a imagem de entrada quad.png é uma imagem colorida, você pode converter a imagem colorida em três imagens em níveis de cinza, calcular o gradiente de cada componente de cor, e escolher (para cada pixel) o gradiente de maior magnitude.

Exemplo de uso de números complexos.

```
Mat_<CPX> imagem onde cada pixel é um número complexo.
CPX é o mesmo que complex<float>
CPX x(3,2);
CPX y; y=CPX(2,7);
float m=abs(x);
float a=arg(x);
float i=x.imag();
float r=x.real();
```

Exercícios da aula 5

Os exercícios da aula 5 podem ser entregues até a próxima segunda-feira, 27/08/2018, 9:20. Os exercícios desta aula valem peso 2 (isto é, valem tanto quanto os exercícios de duas aulas).

Aula5-Ex1 vale 3) Escreva hougrad que usa direção e magnitude do gradiente para calcular a transformada de Hough para detecção de círculos. Deve gravar as imagens do espaço de Hough. Para testar se o programa está fucionando, rode:

```
>hougrad circle1.jpg c1 54 70 2
```

Esse comando calcula a transformada de Hough para detecção de círculos usando módulo e ângulo do gradiente na imagem circle1.png, e grava as imagens de c1-054.png a c1-070.png pulando de dois em dois (c1-054.png, c1-056.png, ... c1-070.png). Você deve obter as imagens de saída semelhantes às mostradas na apostila Hough, seção "Transformada de Hough para detectar círculos usando módulo e direção de gradiente".

Aula5-Ex2 vale 3) Escreva gradcir que detecta o pneu de caminhão/ônibus num quadro do vídeo, pintando-o de amarelo.

```
>gradcir circle1.jpg c1-hg.png 54 70 2
```

O comando acima detecta o pneu da imagem circle1.jpg, pinta-o de amarelo e grava como c1-hg.png. Procura círculos com raios 54 a 70 pixels, com passo 2 (testa raios 54, 56, ..., 70). Verifique se o programa está funcionando, detectando os pneus nas imagens circle1.jpg, circle2.jpg e circle3.jpg.

Aula5-Ex3 vale 4) Escreva detpneu que detecta pneus nos vídeos vid3.avi e vid4.avi

```
>detpneu vid4.avi pneu4.avi
```