PSI3472 Concepção e Implementação de Sistemas Eletrônicos Inteligentes
Segundo Semestre de 2017 1º exercício-programa

Data de entrega: até 17/setembro, domingo 24:00 horas.

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`

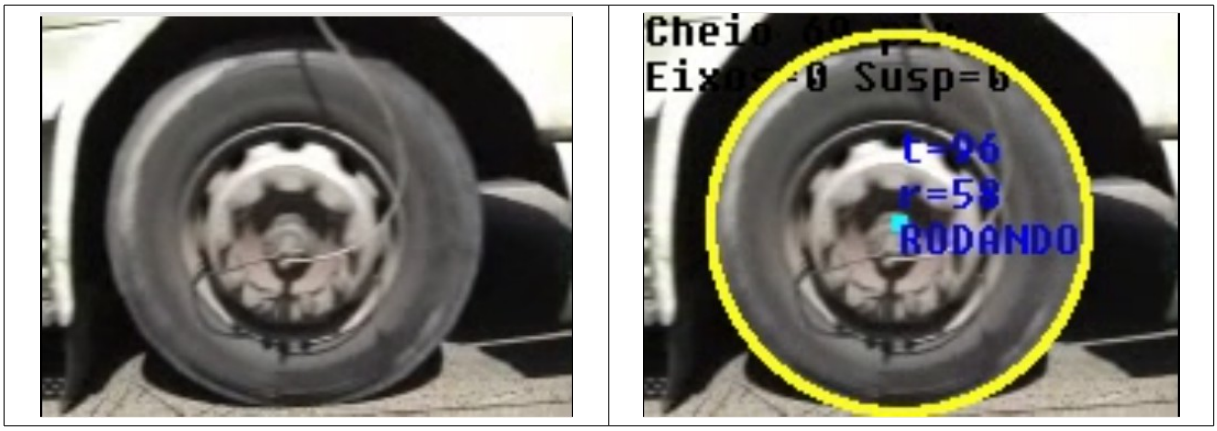
que tem duração de aproximadamente 1 minuto.

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. Você pode escolher entre:

1) Detectar os pneus somente dos veículos grandes, como nos vídeos acima, ou
2) Detectar os pneus de todos os veículos (inclusive carros e motos).



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:

1) Faça um programa "extrai" que lê o vídeo vid4.avi e extrai dez quadros distribuídos ao longo do vídeo: quad0.png, quad1.png, ..., quad9.png. Vamos usar os quadros extraídos para testar o resto do processamento.

```
>extrai vid4.avi quad  => gera quad0.png ... quad9.png
```

opcional 2) Faça um programa canny que recebe um quadro de vid4.avi (quadi.png) e detecta as arestas usando o algoritmo de Canny (cannyi.png). Use para isso a função pronta do OpenCV.

```
>canny quad4.png canny4.png
```

opcional 3) Faça um programa houghb que recebe cannyi.png e detecta círculo (se existir), gerando houghbi.png. Pode pegar trechos dos exemplos da apostila.

```
>houghb canny4.png houghb4.png
```

---

4) Faça um programa gradiente que recebe um quadro (quadi.png), calcula o gradiente e o grava como uma imagem complexa (gradi.img). Permita aplicar o filtro gaussiano antes de calcular o gradiente.

```
>gradiente quad4.png grad4.img 2

(2 é o desvio do filtro gaussiano)
```

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 quadi.png é uma imagem colorida, você pode (opcionalmente) 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.

opcional 5) Escreva o programa houmag que usa o magnitude do gradiente para detectar círculos. Houmag deve receber quadi.png e gerar uma série de imagens Houmag???.png que mostra o espaço da transformada de Hough.

```
>houmag quad4.png houmag 18 20 => gera houmag018.png houmag019.png houmag020.png
```

opcional 6) Escreva o programa magcir que usa o magnitude do gradiente para detectar círculos. Magcir deve receber quadi.png, detectar os círculos, pintar os círculos de amarelo, e gravar como Magciri.png.

```
>magcir quad4.png magcir4.png 18 20
```

---

7) Escreva hougrad que usa direção e magnitude do gradiente para calcular a transformada de Hough. Deve gravar as imagens do espaço de Hough como Hougrad???.png.

```
>hougrad quad4.png hougrad 18 20 => gera hougrad018.png hougrad019.png hougrad020.png
```

---

8) Escreva gradcir que detecta os círculos em imagens, pintando-os de amarelo.

```
>gradcir quad4.png houcir4.png 18 20
```

9) Escreva detpneu que detecta pneus nos vídeos vid3.avi e vid4.avi

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

Você deve entregar os programas que você fez, e um relatório ("diário de bordo") descrevendo os passos que seguiu.

Compacte todos os arquivos (programas e o relatório) usando ZIP com os nomes do grupo (nome1_sobrenome1_nome2_sobrenome2.zip) e envie via edisciplinas (Moodle).

Anexo: Relatórios dos exercícios programas

A primeira regra para uma comunicação clara é escrever para a sua audiência (http://www.plainlanguage.gov/howto/guidelines/FederalPLGuidelines/think.cfm). No caso do relatório, a sua audiência será o professor ou o monitor que irá corrigir o seu exercício. Assim, você não precisa escrever informações que são óbvias para professor/monitor. Por outro lado, precisa escrever aqueles dados que ajudarão professor/monitor a avaliar corretamente a sua solução. Seja breve. Passse as informações necessárias usando menos palavras/páginas possível.

Identificação

Seu nome, número USP, nome da disciplina, etc.

Para cada um dos sub-programas, escreva:

Breve enunciado do problema

Apesar do enunciado do problema ser conhecido ao professor/monitor, descreva brevemente o problema que está resolvendo. Isto tornará o documento compreensível para alguma pessoa que não tem o enunciado do EP à mão.

Técnica(s) utilizada(s) para resolver o problema

Descreva quais técnicas você usou para resolver o problema. Se você mesmo inventou a técnica, descreva em português a sua ideia, deixando claro que a ideia foi sua. Se você utilizar oração com sujeito indeterminado ou na voz passiva, o leitor pode não entender quem foi o responsável (Ex: "Criou-se um novo algoritmo" - Quem criou? Você? Ou algum autor da literatura científica? http://www.plainlanguage.gov/howto/guidelines/FederalPLGuidelines/writeActive.cfm). Se você utilizou alguma técnica já conhecida, utilize o nome próprio da técnica (por exemplo, filtragem Gaussiana, algoritmo SIFT, etc.) juntamente com alguma referência bibliográfica onde a técnica está descrita. Use elementos gráficos como imagens intermediárias e diagramas, pois ajudam muito a compreensão. Não "copie-e-cole" código-fonte, a não ser que seja relevante. Use preferencialmente o pseudo-código.

Ambiente de desenvolvimento utilizado

Em qual plataforma você desenvolveu o programa? Como o professor/monitor pode compilar o programa? Você utilizou que bibliotecas?

Operação

Como o professor/monitor pode executar o programa? Que argumentos são necessários para a execução do programa? Há parâmetros que devem ser configurados? Quais arquivos de entrada são necessários? Quais arquivos de saída são gerados?

Resultados Obtidos

Descreva os resultados obtidos. Qual é o tempo de processamento típico? O problema foi resolvido de forma satisfatória?

Referências

Descreva o material externo utilizado, como livros/artigos consultados, websites visitados, etc.