PSI-5796: Processamento e Análise de Imagens e Vídeos
Segundo período de 2014 2º exercício-programa Prof. Hae
Data de entrega: 14/09/2014 (domingo) até 24:00 horas

Este enunciado é uma versão preliminar. Pode sofrer alterações no futuro próximo.

O banco de dados de imagens "MIT CBCL pedestrian database[1]" contém aproximadamente 1000 imagens coloridas 128x64 de pedestres vistos de frente ou de trás (figura 1). Peguei aleatoriamente 10 dessas imagens e as "escondi" em 5 imagens-hóspedes (sem mudar a escala nem fazer rotação), como na figura 2a. Cada imagem-hóspede contém duas imagens-escondidas. O objetivo deste EP é fazer um programa C/C++ que localize essas imagens-escondidas, como na figura 2b.

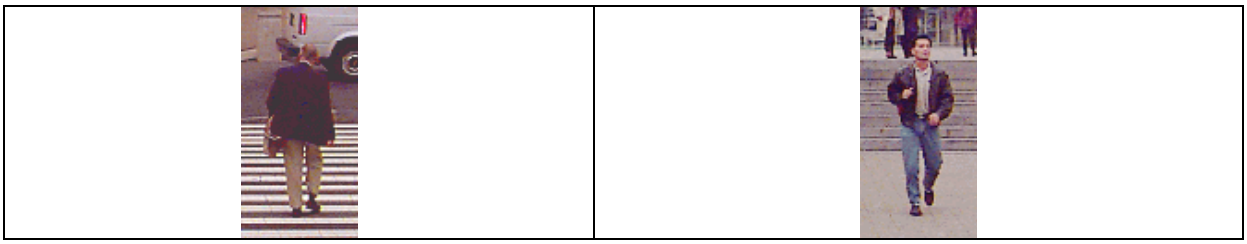

Figura 1: Duas imagens de pedestres do "MIT CBCL pedestrian database".

(a) (b)

Figura 2: (a) Uma imagem hóspede com duas imagens de pedestres "escondidas". (b) Imagem com pedestres localizadas.

Nota: Evidentemente, há uma transição abrupta de cor da imagem-hóspede para a imagem-escondida 128x64. Você NÃO pode usar este fato para localizar as imagens-escondidas.

Nota: Escondi as imagens-pedestres 128x64 nas imagens-hóspedes. Porém, disponibilizei imagens-pedestres 120x56 (após remover 4 linhas/colunas das bordas das imagens originais do banco de dados). Isto é para deixar mais evidente que você não pode usar a transição de cor da imagem-hóspede para imagem-escondida.

[1] http://cbcl.mit.edu/software-datasets/PedestrianData.html

As imagens a serem usadas no desenvolvimento do sistema são:

1) **ap???.ppm**: Amostras Positivas de treinamento. São as 913 imagens 120x56 de pedestres que devem ser usadas para o treinamento (as 10 imagens escondidas não aparecem aqui).
2) **an?.jpg**: Amostras Negativas de treinamento. São 5 imagens 960x1280 (que não contêm pedestres) a serem usadas no treinamento. Estas imagens possuem características visuais mais ou menos similares às imagens-hóspedes.
3) **q?.jpg**: Imagens de busca (query). São as 5 imagens-hóspedes 960x1280 com as 10 imagens-escondidas.

Além dessas, também disponibilizei as seguintes imagens:

4) **qp???.ppm**: A parte central 120x56 das 10 imagens-escondidas inseridas nas imagens-hóspedes. As imagens-escondidas originais são 128x64. As imagens disponibilziadas não contêm 4 linhas/colunas mais externas.
5) **qn?.jpg**: As 5 imagens-hóspedes 960x1280 antes de inserir as imagens-escondidas.

Todas essas imagens estão disponíveis em:

- **http://www.lps.usp.br/~hae/psi5796/ep2-2014/ep2.zip**

Para facilitar a correção dos EPs (para não ter que passar um monte de argumentos), vamos convencionar descompactar obrigatoriamente essas imagens no diretório **c:\ep2**.

Você deve fazer três programas: **treina.cpp, testa.cpp** e **localiza.cpp**.

1) O programa **treina.cpp** utiliza as imagens de treinamento positivas **c:\ep2\ap???.jpg** e negativas **c:\ep2\an?.jpg** para criar um critério que decide se uma image-janela 120x56 é de pedestre ou não. Cada imagem **ap???.ppm** resultará numa única imagem-janela positiva. Cada imagem **an?.jpg** resultará em (960-120+1)*(1280-56+1) = 1.030.225 imagens-janelas negativas obtidas percorrendo a janela 120x56 pela imagem **an?.jpg**. Use essas imagens-janelas de treinamento para criar um critério que decide se uma imagem 120x56 é de pedestre ou não. Salve esse critério como **c:\ep2\criterio.xml**.

2) O programa **testa.cpp** aplica o critério de decisão **c:\ep2\criterio.xml** para todas as janelas 120x56 das imagens **ap???.ppm, an?.jpg, qp???.ppm** e **qn?.jpg**, informando a quantidade de janelas positivas (pedestres) e negativas (não-pedestres) em cada imagem:

```
c:\diret>testa
```

deve gerar saída como:

```
an1.jpg.   pos=0 neg=1030225
ap002.ppm. pos=1 neg=0
...
```

3) O programa **localiza.cpp** usa o critério de decisão **criterio.xml** para localizar os dois pedestres em cada **c:\ep2\q?.jpg** e grava as saídas como **c:\ep2\p?.jpg**. Este programa pode usar o fato de que cada imagem-teste contém exatamente 2 pedestres.

Nota: A seguinte função do Cekeikon pode ser usada para achar a lista de arquivos que satisfaz um certo "wildcard":

```
void vsWildCard(string nome, vector<string>& vs)
```

Por exemplo, a sequência de comandos:

```
vector<string> vs;
vsWildCard("c:\\ep2\\an?.jpg",vs);
vsWildCard("c:\\ep2\\ap???.ppm",vs);
```

fará com que o vetor de strings **vs** contenha a lista de todas as amostras que devem ser usadas no treino.

Obs. 1: Cada dia de atraso acarreta uma perda de 1 ponto no exercício.

Obs. 2: Este EP deve ser resolvido individualmente. EPs iguais receberão nota zero.

Obs. 3: Pode usar (se quiser) a biblioteca Cekeikon4/OpenCV248.

Obs. 4: Entregue os programas-fontes (**treina.cpp, testa.cpp e localiza.cpp**) e o criterio.xml (para que o monitor não precise rodar todo o treino). Entregue também um documento **coment.pdf** ou **coment.doc** descrevendo em português o funcionamento do seu programa. Informe nesse documento quantos falsos positivos e falsos negativos o seu programa cometeu e o tempo gasto no treino, no teste e na localização. O envio dos comentários é obrigatório.

(a) Se você fez o programa no ambiente usado na classe (cekeikon/opencv): não é necessário enviar o programa executável.

(b) Se você usou um ambiente diferente: É necessário descrever como gerar o executável a partir do código fonte. É também necessário enviar o executável. Mude a extensão **.EXE** para **.EEE** pois há servidores que não aceitam enviar/receber **.EXE**. Um programa **.EXE** pode necessitar de vários arquivos **.DLL** para funcionar - envie todos os **.DLLs** necessários.

Obs. 5: Compacte todos os arquivos como **SeuNome_Sobrenome.ZIP** e envie um email colocando como assunto **"PSI5796 EP2"** para os endereços abaixo:

- **hae@lps.usp.br**
- **walter.mayortoro@usp.br** (monitor)

Para evitar confusão, envie um único email. Se você enviar dois ou mais emails, considerarei somente o último email enviado, ***descartando*** os anteriores.