PSI-5796: Processamento e Análise de Imagens e Vídeos
Segundo período de 2015 2º exercício-programa Prof. Hae
Data de entrega: 13/09/2014 (domingo) até 24:00 horas

O objetivo deste EP é fazer um programa C/C++ que localize pessoas em vistas frontais ou posteriores, como na figura 1.

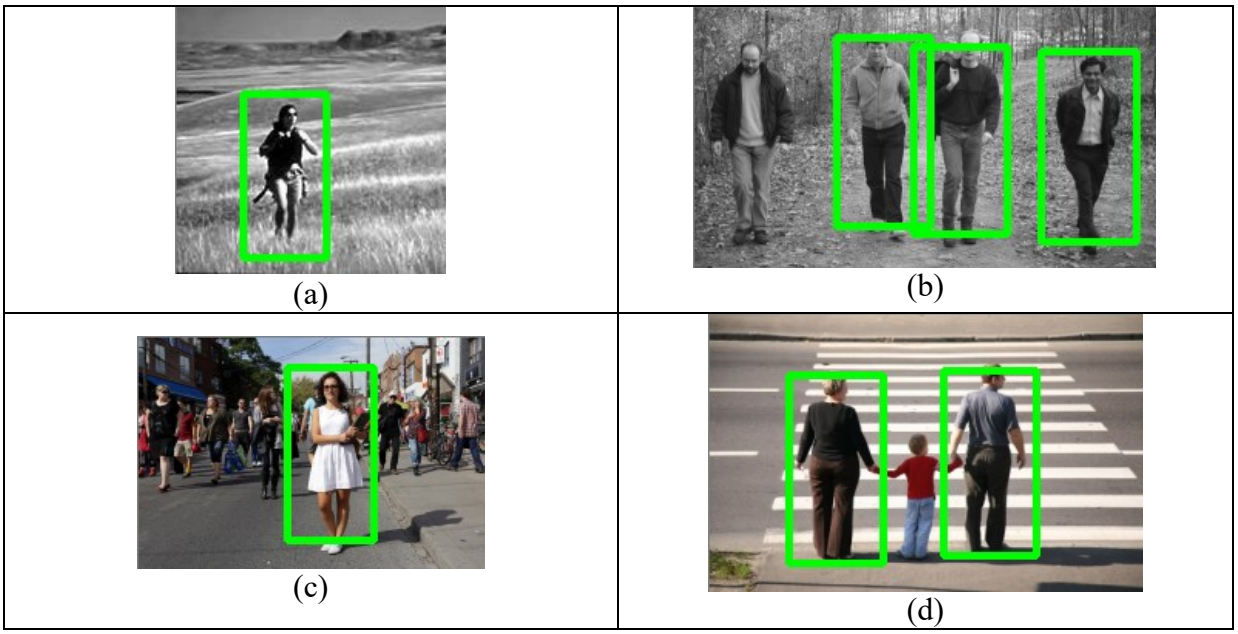

Figura 1: Vistas frontais ou posteriores de pessoas, localizadas pelo programa "cpp-example-peopledetect.exe" do OpenCV. Note que há falsos negativos, como na imagem (b).

Para isso, são dadas as imagens abaixo, compactadas em "imagens.zip":
1) Amostras de imagens positivas ap???.ppm da base de imagens "MIT CBCL pedestrian database[1]". Esta base contém 924 imagens coloridas 128x64 de pedestres (figura 2). Você pode usar estas imagens como exemplos positivos para treinar o seu sistema de aprendizagem de máquina.

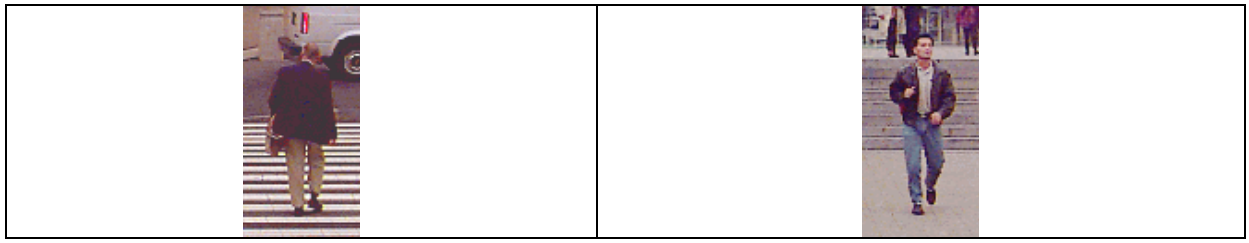

Figura 2: Duas imagens de pedestres do "MIT CBCL pedestrian database".

2) 13 amostras de imagens negativas an??.???, onde não há pedestres. Percorrendo estas imagens com janela móvel 128x64, você vai obter muitos exemplos negativos.

3) 23 imagens testes qx??.???, tanto coloridas como em níveis de cinza. As imagens em níveis de cinza foram obtidas de "USC Pedestrian Detection Test Set[2]". As imagens coloridas foram obtidas de diversos sites.

[1] http://cbcl.mit.edu/software-datasets/PedestrianData.html
[2] http://iris.usc.edu/Vision-Users/OldUsers/bowu/DatasetWebpage/dataset.html

Todas as imagens estão disponíveis em:

- **http://www.lps.usp.br/~hae/psi5796/ep2-2015/imagens.zip**

Para facilitar a correção (diminuir o número de parâmetros dos programas), vamos convencionar que todas as imagens devem ser descompactadas obrigatoriamente no diretório **c:\ep2\imagens** ou **~/ep2/imagens**. As saídas devem ser colocadas obrigatoriamente no diretório **c:\ep2\saidas** ou **~/ep2/saidas**.

Você deve fazer dois programas: **treina.cpp** e **localiza.cpp**, que lêem as imagens e gravam as saídas nos diretórios especificados acima.

1) O programa **treina.cpp** utiliza as imagens de treinamento positivas **c:\ep2\imagens\ap???.ppm** e negativas **c:\ep2\imagens\an??.???** para criar um critério que decide se uma imagem-janela 128x64 é de pessoa ou não. Cada imagem **ap???.ppm** resultará numa única imagem-janela positiva. Cada imagem **an??.???** resultará em várias imagens-janelas negativas obtidas percorrendo a janela 128x64 pela imagen. Use essas imagens-janelas de treinamento para criar um critério que decide se uma imagem 128x64 é de pedestre ou não. Salve esse critério como **c:\ep2\saidas\criterio.xml**.

2) O programa **localiza.cpp** lê e usa o critério de decisão **c:\ep2\saidas\criterio.xml** para localizar os pedestres em cada imagem **c:\ep2\imagens\qx??.???** e grava as saídas como **c:\ep2\saidas\qp??.ppm**.

Nota: O programa cpp-example-peopledetect.exe cometeu vários erros (falsos negativos). Provavelmente, isto acontece por estar procurando pedestres em imagens de baixa resolução.

Nota: É normal que o seu programa cometa alguns erros (falsos positivos ou negativos). Não se preocupe se o seu programa não acertar 100%.

Nota: Você pode acrescentar mais imagens de treinamento para diminuir as taxas de erro. Neste caso, deixe isto explícito no seu relatório e entregue estas imagens.

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\\imagens\\an??.???",vs);
vsWildCard("c:\\ep2\\imagens\\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: No OpenCV, há o programa cpp-example-peopledetect.cpp que faz o que este enunciado pede. Você não pode usar esse programa (todo ou trechos).

Obs. 4: Pode usar (se quiser) a biblioteca Cekeikon/OpenCV.

Obs. 5: Entregue os programas-fontes (**`treina.cpp e localiza.cpp`**) e o conteúdo do diretório **`c:\ep2\saidas`** (isto é, o arquivo **`criterio.xml`** e as imagens processadas **`qp??.ppm`**). Entregue também um documento **`relatorio.pdf`** ou **`relatorio.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 e na localização e o modelo do seu computador. O envio do relatório é obrigatório.

(a) Se você fez o programa no ambiente usado na classe (windows/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.

Obs. 6: 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`**
- **`gerson.faria@gmail.com`**

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

O monitor da disciplina, Gerson Faria, escreveu o seguinte documento sobre os itens que deve conter o "relatorio.doc". Sugiro a todos que sigam essas instruções.

Tópicos mínimos exigidos no relatório dos Exercícios Programas

Descrição do problema / objetivos

Descreva claramente o enunciado do problema a ser resolvido. Isso é importante para você se assegurar de que está resolvendo o problema pedido.

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

Descreva de forma clara quais algoritmos e técnicas foram necessários para resolver o problema. Utilize o nome adequado, se existente (e.g. filtragem Gaussiana, classificador SVM, algoritmo Sift etc.). Use elementos gráficos como imagens e diagramas se necessário. Não copie e cole código fonte, a não ser que o mesmo seja de fato relevante, mas se o fizer, comente-o. No relatório, o comentário é mais importante do que o código. Prefira o uso de pseudocódigo.

Ambiente de desenvolvimento utilizado

Em qual plataforma a solução foi desenvolvida? Em qual plataforma a solução será utilizada? Como o usuário pode compilar o programa? Quais bibliotecas foram utilizadas?

Operação

Como o usuário deve executar o programa? Quais os argumentos para execução? Há parâmetros necessários a serem configurados? Quais arquivos de entrada são necessários? Quais arquivos de saída são gerados?

Resultados Obtidos

Descreva os resultados obtidos. O problema foi resolvido de forma satisfatória / robusta? Quais as limitações encontradas? Quais as sugestões de melhorias?

Referências

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