PSI-5796: Processamento e Análise de Imagens e Vídeos
Segundo período de 2016 1º exercício-programa Prof. Hae
Data de entrega: 31/07/2016 (domingo) até 24:00 horas

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: Este EP é igual ao primeiro EP de PSI2651 de 2016.

São dadas 16 imagens qr??.jpg, onde em cada imagem aparece um único QR-code. Os QR-codes aparecem aproximadamente de pé, com pequenas distorções em perspectiva. Essas imagens estão disponíveis em:

http://www.lps.usp.br/hae/psi5796/ep1-2016/qr.zip

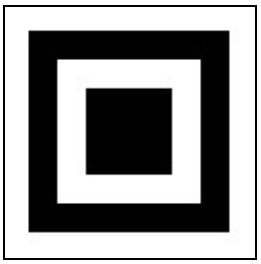
qr00.jpg


qr01.jpg

Faça um programa C/C++ ep1.cpp que lê uma imagem qr??.jpg e gera uma imagem qr??.png que localiza as três marcas quadradas que permitem localizar o qr-code. O seguinte comando deve ler qr00.jpg e gerar qr00.png:

```
diretorio>EP1 qr00.jpg qr00.png
```

qr00.png


padrao.png

Se quiser, pode utilizar a imagem-modelo a ser buscado:

http://www.lps.usp.br/hae/psi5796/ep1-2016/padrao.png

Suponha que este arquivo está no seu diretório default.

Opcional: Além disso, são dadas 5 imagens op??.jpg, onde em cada imagem aparece um QR-code e partes quebradas de outros QR-codes. Essas imagens estão disponíveis em:

http://www.lps.usp.br/hae/psi5796/ep1-2016/op.zip

op00.jpg

op04.jpg

Opcionalmente, faça um programa C/C++ op1.cpp que lê uma das imagens op??.jpg e gera uma imagem op??.png que localiza as três marcas quadradas que aparecem no QR-code inteiro, desprezando as marcas quadradas dos QR-codes parciais. Por exemplo:

```
diretorio>OP1 op00.jpg op00.png
```

op00.png

Nota 1: Em Cekeikon, as seguintes funções podem ser úteis:

- `void reta(Mat_<COR>& b, int l1, int c1, int l2, int c2, COR cor=COR(0,0,0), int largura=1);`
- `void retang(Mat_<COR>& a, int l1, int c1, int l2, int c2, COR cor=COR(0,0,0), int largura=1);`
- `void putTxt(Mat_<COR>& ap, int la, int ca, string st, COR fore=COR(0,0,0), int ampl=1, bool transp=true, COR back=COR(255,255,255));`

Essas funções respectivamente traça uma reta, desenha um retângulo, e imprime um texto. As funções semelhantes do OpenCV são:

- `void line(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int lineType=8, int shift=0);`
- `void rectangle(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int lineType=8, int shift=0);`
- `void putText( Mat& img, const string& text, Point org, int fontFace, double fontScale, Scalar color, int thickness=1, int lineType=8, bool bottomLeftOrigin=false );`

Nota 2: Para achar um padrão com tolerância a mudança de escala, deve procurá-lo depois de redimensioná-lo para várias escalas. Para redimensionar uma imagem, pode usar a função do OpenCV resize:

```
void resize(InputArray src, OutputArray dst, Size dsize, double fx=0, double fy=0, int interpolation=INTER_LINEAR )
```

Obs. 1: Pode usar (se quiser) a biblioteca Cekeikon/OpenCV.

Obs. 2: Entregue o programa-fonte (ep1.cpp) e um documento PDF (relatorio.pdf) ou DOC (relatorio.doc) com os comentários descrevendo o funcionamento do programa. O envio do relatório é obrigatório (veja o anexo).

(a) Se você fez o programa no ambiente usado na classe (cekeikon/opencv): não é necessário enviar o programa executável. Se você fez o programa usando só OpenCV em Linux ou Mac, também não é necessário enviar o programa executável (desde que você não utilize nenhuma função exclusiva desses sistemas).

(b) Se você usou um ambiente diferente: É necessário descrever como gerar o executável a partir do código fonte. É também necessário enviar 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. 3: Compacte todos os arquivos como **`SeuNome_Sobrenome.ZIP`** e envie um email colocando como assunto **"PSI5796 EP1"** para o endereço abaixo:

- **`hae@lps.usp.br`**

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

Anexo: Tópicos exigidos no relatório dos Exercícios Programas

1. Descrição do problema / objetivos

Descreva o enunciado do problema a ser resolvido.

2. Ambiente de desenvolvimento utilizado

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

3. Operação

Como o usuário deve executar o programa? Quais os argumentos para execução? É necessário a presença de alguns arquivos de entrada para que o programa funcione? Há parâmetros a serem configurados? Quais arquivos de saída são gerados?

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

Descreva quais algoritmos e técnicas foram usados para resolver o problema. Use imagens intermediárias para ilustrar o processo. Não copie e cole o código fonte, a não ser que o mesmo seja relevante para a explicação.

5. Resultados Obtidos

Descreva os resultados obtidos. Qual é o tempo de processamento? Qual é a taxa de erros? O problema foi resolvido de forma robusta? Quais as limitações encontradas? Quais as sugestões de melhorias?

6. Referências

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