PSI-3471: Fundamentos de Sistemas Eletrônicos Inteligentes
Primeiro semestre de 2020 Exercício-programa Prof. Hae
Data de entrega: ~~08/06/2020~~ ~~15/06/2020~~ 21/06/2020 (domingo) até 24:00 horas

Obs: Este EP deve ser resolvido individualmente.

O banco de dados abaixo contém imagens de placas de trânsito, algumas delas com sujeira:
http://inf-server.inf.uth.gr/~gpotamianos/traffic_sign_database.html
Desse banco de dados, peguei 44 imagens com a placa "proibido virar" (à esquerda ou à direita), renomeei-as de 00.jpg até 43.jpg, e deixei em:
http://www.lps.usp.br/hae/psi3471/ep1-2020/proibido_virar.zip


37.jpg


42.jpg

Faça um programa C/C++ ep1.cpp (ou Python3 ep1.py) que lê uma dessas 44 imagens e localiza a placa "proibido virar". Você pode usar o fato de que em cada imagem aparece uma e somente uma placa "proibido virar". Note que este conhecimento facilita muito a resolução do problema. A forma de chamar o programa deve ser "ep1 entrada.ext saida.ext", por exemplo:

```
linux$ ep1 ../37.jpg ~/diretorio/e37.png
windows> ep1 ..\37.jpg c:\diretorio\e37.png
```

Se chamar o programa com parâmetros incorretos, o programa deve imprimir alguma mensagem amigável:

```
diretorio> ep1
ep1: Detecta placa "proibido virar"
sintaxe: ep1 ent.ext sai.ext
Erro: Numero de argumentos invalido
```

Se o arquivo a ser processado não existe, o programa também deve imprimir alguma mensagem amigável:

```
diretorio> ep1 99.jpg 99.png
Erro: Arquivo 99.jpg nao existe
```

e37.png

e42.png

Para resolver este problema, você pode usar a qualquer técnica. Só não use transformada de Hough para círculos, pois esta técnica não será vista neste curso (será vista no próximo) e só consegue detectar círculos (se a placa fosse quadrada ou triangular, precisaria usar transformada Hough para formas genéricas, que é muito diferente da Hough para círculos).

Você pode escolher até 8 imagens (das 44) para servirem como “amostras de treinamento” ou “modelos” a serem procuradas. Você pode editar as imagens escolhidas para criar as imagens-amostras de treinamento, por exemplo, para especificar o que é “cor vermelho” e/ou para especificar o “modelo” a ser procurado. Estas imagens devem estar no diretório *default*, podem ser lidos por seu programa e devem ser entregues juntos com o seu programa. Neste caso, deixe explícito no relatório quais foram as imagens escolhidas para servirem de “amostras” e como foram editadas. Para padronizar, nomeie estas imagens como am*.* (use obrigatoriamente o prefixo “am”).

Deixei as 44 saídas geradas pelo meu programa em:
http://www.lps.usp.br/hae/psi3471/ep1-2020/saida_professor.zip

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

Obs. 2: Entregue o programa-fonte (ep1.cpp ou ep1.py), os modelos e outros dados necessários (am*.*) para a execução do seu programa, e um documento PDF (relatorio.pdf) 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 (C++/Cekeikon/OpenCV ou Python/OpenCV), basta entregar o programa-fonte para poder corrigir o seu programa.

(b) Se você quiser fazer usar alguma biblioteca diferente ou programar em outro ambiente, converse antes com o professor.

Obs. 3: Compacte todos os arquivos como nome_sobrenome.zip e envie o arquivo através de edisciplinas. Dentro do prazo, você pode substituir o arquivo anterior por um novo. Só o último arquivo entregue será corrigido.

Para quem for fazer este exercício como recuperação (2022):

1) Entregue também um vídeo de no máximo três minutos explicando o funcionamento dos seus programas e os resultados obtidos. No vídeo deve aparecer em algum momento o seu rosto e um documento seu. É necessário que o vídeo contenha áudio, pois é muito difícil entender um vídeo sem a explicação falada. O vídeo não pode ter sido acelerado para diminuir a duração, pois dificulta entender a fala. Enviem um link para o vídeo (youtube, google drive, etc) para não entupir sistema de email. Assegure que todos os professores (Hae e Magno - hae.kim@usp.br, magno.silva@usp.br) tenham acesso ao seu vídeo.

2) A entrega deve ser feita pelo email: hae.kim@usp.br.

Dicas para resolver este exercício:

As explicações a seguir normalmente são dadas em sala de aula. Devido à pandemia, estou deixando-as escritas.

Descrevo abaixo uma possível solução para este exercício. Você pode usar técnica que quiser, não é obrigado a seguir os passos que listo abaixo. No enunciado, só estou pedindo para não usar transformada de Hough para círculos, pois esta técnica não irá funcionar se quiser detectar qualquer forma que não seja circular.

Passo 1) Transforme a imagem colorida em níveis de cinzas, destacando as regiões de cor vermelha. Aplique esta transformação nas 44 imagens e verifique que não perdeu nenhuma região vermelha das placas.

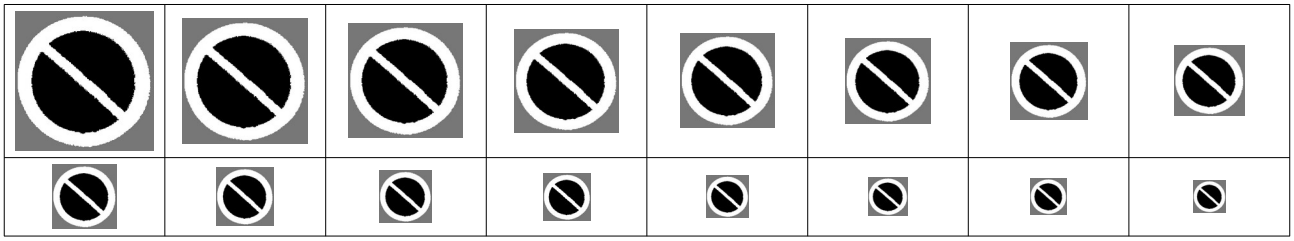
Passo 2) Escolha uma das 44 imagens e extraia a placa da imagem para servir de modelo. Edite-a para que possa servir de modelo de “template matching”.

(a) Placa original.

(b) Imagem editada para servir de modelo. Cinza representa pixels “indiferentes”.

(c) Outra imagem editada para servir de modelo.

Passo 3) Redimensione o modelo para várias escalas em progressão geométrica, tomando cuidado para que todas as placas das imagens estejam dentro do intervalo de escala dos modelos. Isto é, a menor placa das imagens não pode ser menor que o menor modelo, e a maior placa das imagens não pode ser maior que o maior modelo. Pense por que estou recomendando que use progressão geométrica - o que aconteceria se usasse progressão aritmética?



Nota: Para redimensionar o tamanho do modelo, você pode usar a função *resize* do OpenCV. Cuidado: Não é o membro *resize* da classe *Mat* do OpenCV (este membro faz outra coisa). A sintaxe é:

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

Python: cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation ] ] ] ]) → dst
```

Exemplo: Para redimensionar uma imagem A para 100×100 pixels:

```
C++:
Mat_<GRY> A,B; resize(A, B, Size(100,100));
Mat_<FLT> A,B; resize(A, B, Size(100,100));

Python:
B=cv2.resize(A, (100,100) )
```

Passo 4) Decida se você vai fazer casamento de modelo CC, NCC ou uma combinação das duas. O que vou descrever abaixo funciona melhor com casamento de modelo no modo “same”, onde a imagens de entrada e saída têm o mesmo tamanho e a correlação é marcada no centro do modelo. Se você tem n modelos, efetue n casamentos de modelo. O casamento do modelo T_i com a imagem a analisar A irá resultar numa matriz de correlação C_i, i =0, 1, ..., n-1. Construa duas matrizes:

$$C = \mathrm{MAX}_{i=0}^{n-1} C_i \qquad A = \mathrm{ARGMAX}_{i=0}^{n-1} C_i$$

onde MAX calcula o maior valor (pixel a pixel) das n matrizes e ARGMAX calcula o índice do maior valor (pixel a pixel) das n matrizes.

Após calcular C e A, o pixel $C(l,c)$ que tiver o maior valor dentro da matriz C é o centro da placa e $A(l,c)$ é o índice de escala (representa o tamanho) da placa.

Anexo: Relatórios dos exercícios programas

O mais importante numa comunicação escrita é que o leitor entenda, sem esforço e inequivocamente, o que o escritor quis dizer. O texto ficar "bonito" é um aspecto secundário. Se uma (pseudo) regra de escrita dificultar o entendimento do leitor, essa regra está indo contra a finalidade primária da comunicação. No site do governo americano [1], há regras denominadas de “plain language” para que comunicações governamentais sejam escritas de forma clara. As ideias por trás dessas regras podem ser usadas em outros domínios, como na escrita científica. Resumo abaixo algumas dessas ideias.

(1) Escreva para a sua audiência. No caso do relatório, a sua audiência será o professor ou o monitor que irá corrigir o seu exercício. Você deve enfocar na informação que o seu leitor quer conhecer. Não precisa escrever informações que são inúteis ou óbvias para o seu leitor.

(2) Organize a informação. Você é livre para organizar o relatório como achar melhor, porém sempre procurando facilitar o entendimento do leitor. Seja breve. Quebre o texto em seções com títulos claros. Use sentenças curtas. Elimine as frases e palavras que podem ser retiradas sem prejudicar o entendimento. Use sentenças em ordem direta (sujeito-verbo-predicado).

(3) Use palavras simples. Use o tempo verbal o mais simples possível. Evite cadeia longa de nomes, substituindo-os por verbos (em vez de “desenvolvimento de procedimento de proteção de segurança de trabalhadores de minas subterrâneas” escreva “desenvolvendo procedimentos para proteger a segurança dos trabalhadores em minas subterrâneas”). Minimize o uso de abreviações (para que o leitor não tenha que decorá-las). Use sempre o mesmo termo para se referir à mesma realidade (pode confundir o leitor se usar termos diferentes para se referir a uma mesma coisa). O relatório não é obra literária, não tem problema repetir várias vezes a mesma palavra.

(4) Use voz ativa. Deixe claro quem fez o quê. 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?). O site diz: “Passive voice obscures who is responsible for what and is one of the biggest problems with government writing.”

(5) Use exemplos, diagramas, tabelas, figuras e listas. Ajudam bastante o entendimento.

O relatório deve conter pelo menos as seguintes informações:

Identificação

Nomes dos integrantes do grupo, números USP, nome da disciplina, etc.

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 a sua ideia, deixando claro que a ideia foi sua. 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

1 https://plainlanguage.gov/guidelines/

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.