PSI3471 - Fundamentos de Sistemas Eletrônicos Inteligentes

Exercício programa 1 - Identificação de placas de trânsito

Aluna: Beatriz Soares Passanezi - 10336167

Professor: Hae Yong Kim

Sumário

1. Introdução

O exercício programa consiste em identificar placas de "proibido virar", para direita ou para esquerda, em uma amostra de 44 imagens do banco de dados http://inf-server.inf.uth.gr/~gpotamianos/traffic_sign_database.html. Alguns exemplos de imagens selecionadas são:


O objetivo final do exercício programa é identificar somente a placa de "proibido virar" e destacá-la. Exemplos de imagens com a placa destacada são:


2. Técnica

A técnica usada para resolver o problema consistiu nos seguintes passos:

1) Definir um template de placa que será usado para encontrar as placas na imagem
    - A imagem usada como template tem o nome *template_1.jpg* e será enviada junto com o código fonte do programa

2) Transformar a imagem colorida em uma imagem em níveis de cinza destacando somente tons de vermelho.

Para realizar essa tarefa, primeiro converti a imagem de RGB para HSV, usando a função *cv2.cvtColor*. Em seguida, defini intervalos da cor vermelha no padrão HSV e

apliquei a função *cv2.inRange* com os parâmetros apropriados para obter a máscara da imagem somente onde ela se encaixa no intervalos definidos para cor vermelha.

Essa etapa é realizada na função ***acha_vermelho*** do exercício programa.

3) Encontrar o tamanho de template que melhor se encaixa na imagem a ser analisada

No conjunto de imagens que foi passado, há placas de diversos tamanho. Assim, é necessário redimensionar o template para diferentes tamanhos para encontrar qual tamanho se encaixa melhor na imagem que está sendo analisada.

Para fazer isso, redimensionei o template em diferentes tamanhos usando a função *cv2.resize* e procurei qual possuía o ponto de maior correlação após realizar o casamento de template com a imagem pela função *cv2.matchTemplate*. O método usado para o casamento de template foi TM_CCOEFF_NORMED.

Essa etapa é realizada na função ***acha_tamanho*** no exercício programa.

4) Desenhar o círculo na imagem

Após possuir o ponto de máxima correlação e qual o tamanho de template ideal, desenhei um círculo verde ao redor desse ponto usando a função *cv2.circle*.

Essa etapa é realizada na função ***desenha_circulo*** no exercício programa.

5) Unir as funções usadas

Após realizar cada etapa e separá-las em funções, basta unir as etapas em uma função que encapsula todos os passos. Essa função é a função ***acha_placa*** no exercício programa.

3. Ambiente de desenvolvimento

Para desenvolver o programa, escolhi usar a biblioteca OpenCV em Python, por ter mais familiaridade com seu uso e com a linguagem. Além disso, usei também as bibliotecas *sys* e *numpy*, responsáveis respectivamente por ler a entrada do programa pela linha de comando e fazer operações com matrizes e vetores.

Por ser desenvolvido na linguagem Python, o arquivo não precisa ser compilado. Para rodar o programa desenvolvido, basta instalar as bibliotecas usadas. Isso pode ser feito pelos comandos:

```
pip install opencv-python
```

```
pip install numpy
```

4. Operação

O programa recebe o *path* da imagem de entrada, encontra e destaca a placa e, em seguida, salva a imagem com a placa destacada em um arquivo especificado pelo usuário.

Para rodar o programa, basta usar o comando:

```
python ep1.py <imagem_entrada> <imagem_saida>
```

Nota-se que o programa desenvolvido recebe dois argumentos ao ser chamado, o primeiro é o *path* da imagem de entrada relativo à pasta atual e o segundo é o nome do arquivo onde a imagem de saída será salva (note que para o segundo argumento não se passa um *path* completo, mas somente o nome do arquivo de saída. Assim, não é possível salvar a imagem dentro de outra pasta).

Alguns exemplos de chamada do programa seriam:

```
python ep1.py proibido_virar/00.jpg output00.jpg
```

```
python ep1.py 01.jpg output01.jpg
```

5. Resultados obtidos

O tempo de processamento de cada imagem é rápido, menos de 10s por imagem. Isso ocorre pois, mesmo usando a linguagem Python, a maior parte do código é realizada usando a biblioteca OpenCV, que é otimizada em C++.

Os resultados obtidos foram satisfatórios e o programa desenvolvido foi capaz de identificar as placas em todas imagens em que foi testado, mesmo imagens em que a placa estava escondida ou em que havia outras placas parecidas. Alguns exemplos de saída do programa são:


6. Referências

[1] https://pypi.org/project/opencv-python/
[2] https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html
[3] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html
[4] https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html
[5] https://realpython.com/python-opencv-color-spaces/
[6] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html