EP – Placas de Trânsito

Aluno: Renan Ikeda Fernandes

NUSP: 10439892

Problema

Este relatório descreve o método utilizado para a solução do problema posto em forma de EP na disciplina PSI3471, em que consiste na criação de um algoritmo que identifica placas de trânsito de “proibido virar” em uma imagem e devolve a mesma figura com a placa destacada.

Método

Para resolver o problema proposto foi utilizado C++ como linguagem base e a biblioteca Cekeikon, desenvolvida pelo professor Hae, que utiliza como base o OpenCV. A partir disso foi utilizado as imagens de “proibido virar” disponibilizadas na disciplina e aplicada uma rotina que destaca cores vermelhas na imagem, de forma a devolver a mesma figura nas cores preto e branco para que possa localizar o tom vermelho da placa na imagem e sua silhueta.

7Para isso se utilizou uma imagem base em preto e pintou de branco somente onde na imagem original tinha uma distância Euclidiana menor que um limiar do tom de vermelho escolhido [1]. Após uma série de testes heurísticos foi escolhido a cor vermelha base como (10, 10, 160) e a distância máxima de 100. O resultado segue na Figura 1 abaixo.

Figura 1: Imagem 00.jpg antes e depois do destacamento de vermelho.

Em seguida se extraiu a placa de uma imagem base para que sirva de *template* para a localização nas outras figuras, neste caso se utilizou a imagem 00.jpg após o destacamento de cor vermelha e em seguida foi recortada a placa. Ilustrado na Figura 2 abaixo.

Figura 2: Template utilizado a partir da Figura 1.

Para a localização do *template* na imagem foi utilizado o *Template Matching* a partir da função da biblioteca Ceikeikon matchTemplateSame [2], utilizando Coeficiente de Correlação Normalizado, pelo fato dele ser invariante a brilho e contraste e se obteve resultados melhores comparado a técnica de Correlação Cruzada.

Para garantir que a proporção da placa na imagem seja diferente do *template* usado se aplicou a função resize de OpenCV [3] para que fosse possível gerar tamanhos variados de *template*. Dessa forma o pseudo código que descreve a forma que se fez o *Template Matching* no problema segue abaixo.

Tome x e y as dimensões do template aumentadas em uma razão p.

Tome as matrizes C e ARG da mesma dimensão da imagem de entrada;

Para todos valores de k:

Use x e y a nova dimensão do template multiplicada por uma razão q elevada a k;

Faça o resize do template pelos valores de x e y;

Faça o Template Matching da imagem base após o destaque de vermelho;

Encontre o maior valor a imagem resultante e armazene seu valor na matriz C e seu k na matriz ARG;

Dessa forma o código acima armazena na matriz C todos os valores máximos de todos os *Matchings* de diversos tamanhos, ou seja, o maior valor de C apresenta a maior correlação de todos os templates e a matriz ARG armazena qual é a dimensão do *template* usado. Após alguns testes heurísticos se chegou no valor do aumento do template de 1.05^5, na razão q de 0.95 da diminuição geométrica e o valor de $k = 30$ templates. Segue abaixo a saída do *Template Matching* a partir da imagem 00,jpg após o destaque de vermelho.

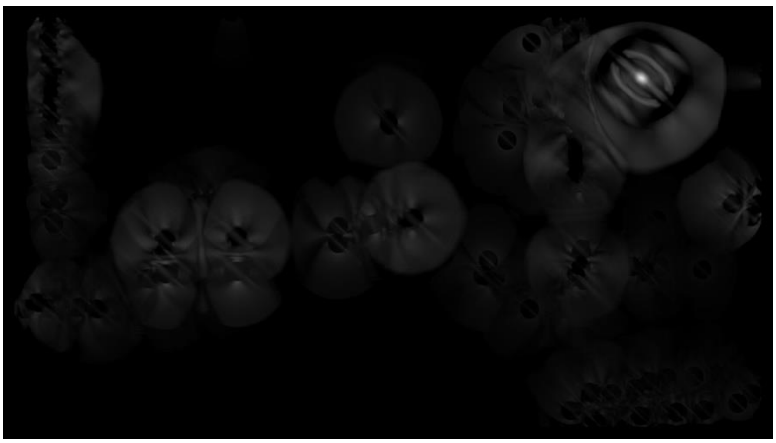

Figura 3: Saída do Template Matching.

Nela é claro que o ponto de máximo se localiza no canto superior direito, sendo ele o centro da placa, o que é coerente com a Figura 1.

Com isso é foi possível localizar a placa na imagem original, para isso bastou encontrar o máximo na matriz C obtida anteriormente e localizar o valor de k na matriz ARG, assim foi destacado com um quadrado o centro da placa na imagem original com a cor verde e utilizando o fator k foi traçado um quadrado em volta da placa. Como ilustrado na Figura 4.

Figura 4: Saída da detecção.

Operação

Para a execução do programa é preciso utilizar o template disponibilizado na pasta, em seguida é necessário compilar o programa EP1 pelo cmd no diretório do arquivo, utilizando o cekeikon na sintaxe: compila EP1 -cek. Em seguida para executar é necessário entrar com 3 argumentos, o nome do programa o local e nome da imagem de entrada e o local e nome da imagem gerada na saída, na seguinte sintaxe: EP1 C:\...\00.jpg C:\...\Saída.jpg. Não há a necessidade de parâmetros, só a garantia que a imagem de *template* (Temp1cinza1.jpg) esteja no diretório do programa.

Resultados

Utilizando a metodologia e os parâmetros descritos houve a identificação de todas as 44 placas dadas como base, delas 43 tiveram uma localização praticamente perfeita da placa na figura enquanto somente uma não houve um casamento perfeito, que foi o caso da imagem 32.jpg mostrada abaixo.

Figura 4: Resultado da imagem 32.jpg

Nesse caso fica claro que o *template* utilizado não tinha as mesmas proporções da placa na imagem, mas o que era esperado já que neste caso a placa parece estar achatada, em forma retangular, diferente do template que é quadrado, isso também resultou em uma baixa correlação, sendo seu máximo em torno de 0.42. Apesar disso seu centro foi localizado corretamente.

Para a detecção de uma imagem somente o tempo médio para a computação do programa é em torno de 5 a 7 segundos, boa parte desse tempo se deve aos diversos redimensionamentos e *matchings* realizados na detecção.

Referências

[1] Slides de aula, Conceitos básicos, GCC, Cekeikon, OpenCV, sistemas de cores (RGB, HSI, CieLab).

[2] Slides de aula, Casamento de modelo (template matching).

[3] Slides de aula, Transformações geométricas.