PSI3471 - Sistemas eletrônicos inteligentes

EP1

Tiago Azevedo Amano 5974802

16 de Junho de 2019

Conteúdo

---

1 Enunciado

O objetivo deste exercício é fazer um programa C/C++ que, dada uma imagem de moedas brasileiras atuais sobre fundo de papel azul, conta quantas moedas tem na imagem e localiza espacialmente cada moeda. Opcionalmente, o programa pode identificar o valor de cada moeda e calcular o valor total das moedas na imagem.

A forma de chamar o programa deve ser obrigatoriamente:

```
linux$ ep1 moeda1.jpg saida1.png
windows> ep1 moeda1.jpg saida1.png
```

O programa deve emitir um aviso amigável se o usuário entrar argumentos inválidos (número de parâmetros incorreto, arquivo inexistente, etc).

2 Resolução

- Primeiramente, reduzi a imagem em 5 vezes (de 3264x2448 para 652x489) para melhorar o desempenho em tempo de execução e no reconhecimento das moedas.

Figura 1: Imagem moeda4.jpg reduzida de 5 vezes

- Converti a imagem de BGR para HSV pois o parâmetro *Hue* tem uma separação razoável entre as moedas e o fundo azul.
- Adaptei o algoritmo apresentado em aula pelo prof. Hae para pintar usando fila os pixels do fundo da imagem de branco (tomei como pixels de fundo todos os que tinham valor de *Hue* maiores que 30)
- Pintei de preto as moedas utlizando componentes conexos, i.e. varri a imagem procurando pixels que não fossem do fundo (i.e. pretos) ou já pintados (i.e. brancos). Quando encontrado, o pixel é parte de um componente conexo, e então pinto a partir desse pixel todos os pixels deste componente conexo, fazendo isso para todos os pixels da imagem. Mostrado na Figura 2

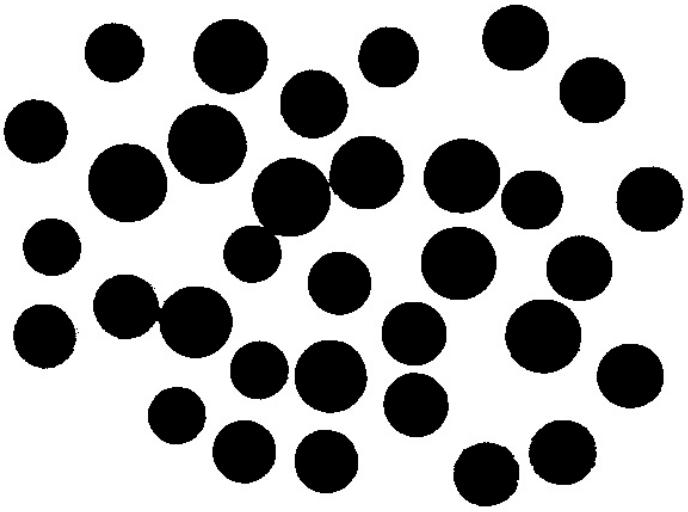

Figura 2: Imagem moeda4.jpg com moedas pintadas de preto e fundo de branco

- Converti a imagem de HSV de volta para BGR.
- Criei 8 modelos de círculo, pretos por dentro e brancos por fora com diferentes escalas separadas por progressão aritmética.
- Procurei todos os modelos de diferentes escalas na imagem, gerando uma imagem com os valores de correlação em ponto flutuante para cada modelo.
- Converti as 8 imagens com os valores de correlação em ponto flutuante para escala de cinza para poder trabalhar mais facilmente, dado que o programa de edição/visualização de imagem que utilizei me dá os valores das cores pixels de 0 a 255 em RGB.
- Criei uma imagem em BGR consolidando as 8 imagens em uma só com os valores de correlação como o primeiro parâmetro, um valor de 0 a 7 representando a escala do modelo com o maior valor de correlação em um dado pixel como o segundo parâmetro e o valor 0 como o terceiro parâmetro. Mostrado na Figura 4.

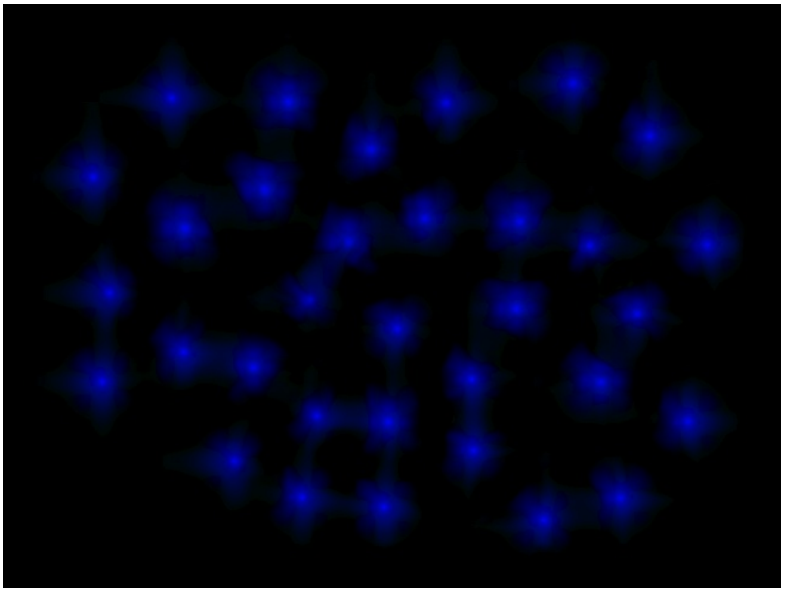

Figura 3: Valores de correlação e escala do modelo com maior correlação para a imagem moeda4.jpg

- Criei uma lista com todos os pixels candidatos a centro, i.e. os pixels que tem valor de correlação maior que um limiar (no caso, o limiar é 190) e que são maiores ou iguais aos pixels vizinhos.

- Eliminei as falsas detecções eliminando os pixels-candidatos que estão muito próximos entre si (usei como parâmetro a distância euclidiana menor que 15 pixels).

  Para isso, crio uma cópia da lista e testo cada item da lista original com a segunda lista e toda vez que a distância entre dois pixels-candidatos da lista era menor que 15, coloco o com maior correlação no fim da lista ou uma média dos pixels, da correlação e da escala caso tiverem mesmo valor de correlação para testar novamente caso haja mais pixels-candidatos próximos. Caso a distância euclidiana for maior que 15, coloco o pixel numa lista separada para os centros garantidos.

- Conto o número de elementos da lista de centros para saber o número de moedas, escrevo na imagem reduzida o número de moedas que está nela e desenho na imagem reduzida circulos das moedas com tamanhos do modelo com maior número de correlação para cada centro da lista centrados nesses centros e um ponto no centro finalmente volto a imagem reduzida para a resolução da imagem original.

Figura 4: Imagem com a contagem e localização das moedas para a imagem moeda4.jpg

3 Ambiente de desenvolvimento

Desenvolvi o programa no linux, utilizando as bibliotecas cekeikon5.6 e queue. Para compilar o programa, pode utilizar o comando ***compila*** com o parâmetro ***-cek***:

```
$ compila ep1.cpp -cek
```

4 Operação

Para executar o programa, deve chamar no terminal ou no prompt de comando:

```
$ ep1 imagem-de-entrada.jpg imagem-de-saida.jpg
```

Onde a imagem de entrada deve ser uma imagem com pelo menos uma moeda de 1 real e os formatos das imagens de entrada e saída devem ser os mesmos. São gerados, além da imagem-de-saída.jpg, as imagens intermediárias apresentadas neste relatório, como a imagem de entrada reduzida, a imagem reduzida com fundo branco e moedas em preto e a imagem com os valores de correlação e escala.

5 Resultados

Os tempos de execução, tirando média de 10 execuções usando WSL (bash ubuntu no windows):

- Para imagem moeda1.jpg: 1.036s
- Para imagem moeda2.jpg: 1.011s
- Para imagem moeda3.jpg: 1.104s
- Para imagem moeda4.jpg: 1.091s

Média para todas as execuções: 1.060s

Em todas as execuções do programa foi contada corretamente o número de moedas, com pequeno erro na localização, porém satisfatório