CSE 4310: Introduction to Computer Vision
Summer 2023
Program #2: Show me the money

In this assignment, you will implement a counting program for a coin sorting machine using OpenCV. The goal is to design a basic perception pipeline using some of the OpenCV functions that we have discussed in class. This application is a real-world use case for computer vision, and as such, you will likely encounter challenges with lighting, image consistency, and other practical considerations.

The program should take a single command line argument, which will contain the file path for the input image (assumed to be a 3 channel color PNG). The program will only operate on a single input image when executed (i.e., manipulating a different image will require a restart).

Once the program is loaded, the image will be processed and a result will be displayed in the console and image window(s). There will be no runtime user interaction, only an immediate display of results. The program should continue to display the image windows until a key is pressed, after which it will terminate normally.

For this assignment, you will be locating and counting the total monetary value of United States coins (penny, nickel, dime, and quarter) in 5 test images, which you will acquire yourself. The test images may be captured with the camera (cell phone is OK), lighting source, and background of your choice, but they may not be manipulated with image editing software prior to being used in your program (you may change the resolution or file format, but you may not annotate the image or remove the background artificially to make processing easier). You must also acquire the images yourself (don't share images with others or copy from the internet).

In each of your 5 test images, you must place a minimum of 5 coins with a total monetary value greater than $0.50 on a flat surface. Each image must have a different total monetary value and contain at least one of each coin (penny, nickel, dime, and quarter). Additionally, the locations of coins should change in each image, and your algorithm must not attempt to identify a coin based on its position.

You may use whatever surface and lighting conditions that you wish. The "easiest" setup would probably be to place a single sheet of white or black paper on a table, arrange the coins on top, and take cell phone pictures from 12" or so away. Using the same camera position in each image will ensure a consistent scale and angle, which will allow size to be used in your comparisons.

When your program is executed, the target file will be loaded according to a single command line argument. The image will be loaded and processed, and both the input and result images must be displayed in separate windows. In the result image, an ellipse should be drawn around each coin. Penny ellipses should be red, nickels yellow, dimes blue, and quarters green. Each ellipse should fit the corresponding coin as tightly as possible. The console should also show the count for each coin and the total monetary value.

In the example case below, the left image is provided as input. Once processing is completed, the right image is displayed in a separate window, and the console should print the following…

```
Penny - 1
Nickel - 1
Dime - 2
Quarter – 2
Total - $0.76
```

Your program should run on the class development environment using OpenCV. If you implement your program using C++ (highly recommended), you must include a working CMakeLists.txt file with your source code. The program should be able to be compiled and executed by running the following set of commands in the program directory if C++ is used:

```
cmake .
make
./program2 <PATH TO FILE>
```

For example…

```
cmake .
Make
./program2 test.png
```

You may choose to use python for the assignment, though this is not recommended. Your program should then be able to run in the class development environment with the following set of commands in the program directory:

```
python program2 <PATH TO FILE>
```

For example…

```
python program2 test.png
```

You may develop your application on your own system / OS, but it must run properly on the class development environment at the time it is evaluated. Submit your source code, test images, and all other necessary files to blackboard in a single zip file by the deadline. Late submissions will incur a penalty of 10 points per day after the deadline.

Points will be assigned as follows:

1. Program identifies and draws penny successfully – 15 points
2. Program identifies and draws nickel successfully – 15 points
3. Program identifies and draws dime successfully – 15 points
4. Program identifies and draws quarter successfully – 15 points
5. Program does not generate false positives – 20 points
6. Program displays counts and total monetary amount correctly – 20 points

Partial credit may be given, at the discretion of the grader, for items which are not fully functional or contain bugs. You may be asked to demo your program to the grader if errors occur during the initial run. Write your code as cleanly as possible with proper formatting and comments in order to bolster your case for partial credit on non-functional features. It is highly recommended to follow a standard coding style, such as ANSI C++, Google C++, etc. in your program (HINT: The Code::Blocks IDE provided with the development environment has a built in formatter under the plugins menu).