

Chapter 8: Coin detection

Summarization of all previous exercises

If we put the algorithms from the previous exercises together we can create a simple coin detection.

Goal:

- Detect Euro coins in an image
- Count the total value of all coins in the image



Input image

coin detection →



Input image + overlay of
detected coins

Image processing

1. Preparation of the input image
 - a. Grayscaleing
 - b. Brightness Adjustment
 - c. Contrast Adjustment
 - d. Blur image (reduction of details to improve the result of the edge detection)
 1. Convolution of 2D blur kernel (separated into two 1D kernels)
2. Edge detection
 - a. Thresholding
 - b. Erosion
 - c. Subtraction
3. Circle detection (Hough transformation)
4. Coin value estimation (based on size and color)

From circles to (Euro) coins

Problems:

1. Not all circles in the image may be coins
2. The size of the coins in the image is unknown
(in case of a camera image the size depends on the distance between the camera and the coin)
3. Some (euro) coins have a similar size

Solutions:

1. Calibration of the coin with a reference coin (e.g. 1 Euro coin) and usage of this size as reference

→ radius as first criterion

2. Usage of the color of the coins ring and core
 - Possible colors in case of Euro coins: bronze, silver, gold

→ ring and core color as second criterion

Note:

The colors of the input image are highly dependent on the lightning situation and, therefore, a **color calibration** is required.

Code example

Coins

```
// 1 Euro Cent  
coin = new CoinPrototype();  
coin->value = 0.01;  
coin->diameter = 16.25;  
coin->colorCore = CoinColor::Bronze;  
coin->colorRing = CoinColor::Bronze;  
coinList.push_back(*coin);
```

```
// 50 Euro Cent  
coin = new CoinPrototype();  
coin->value = 0.50;  
coin->diameter = 24.25;  
coin->colorCore = CoinColor::Gold;  
coin->colorRing = CoinColor::Gold;  
coinList.push_back(*coin);
```

```
// 1 Euro  
coin = new CoinPrototype();  
coin->value = 1.00;  
coin->diameter = 23.25;  
coin->colorCore = CoinColor::Silver;  
coin->colorRing = CoinColor::Gold;  
coinList.push_back(*coin);
```

Well known size (in mm) of a Euro coin

Colors (ring and core)

Calibration example

Program output

```
Start calibration ...
Calibration coin: radius=26          x=74          y=87
detectable coins (sorted by radius):
    coin      value=0.01 EUR      radius=18 px
    coin      value=0.02 EUR      radius=21 px
    coin      value=0.1 EUR       radius=22 px
    coin      value=0.05 EUR      radius=24 px
    coin      value=0.2 EUR       radius=25 px
    coin      value=1 EUR         radius=26 px
    coin      value=0.5 EUR       radius=27 px
    coin      value=2 EUR         radius=29 px
Reference coin found and accepted. New search radius for coins: 16 ... 31 px
```

Explanation

- (1) An image with only one coin (1 Euro reference coin) is used. The circle detection finds a circle with a radius of 26 px
- (2) The size (in mm) of Euro coins is well known and we can calculate the sizes (in px) for all detectable coins
- (3) The circle detection (Hough transformation) will use the new range for the radii

Demonstration & Code explanation

Important files

- main.cpp
- Segmentation.h
- Segmentation.cpp
- Coin.h
- Coin.cpp