


Brazos Fitch Pattern Recognition Project "Coin Classifier"

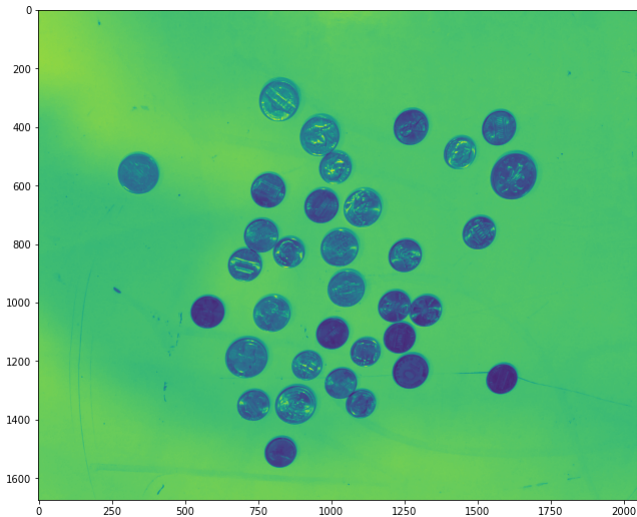
This will be the beginning code which will eventually inform a final code. The final code will take all these techniques and make it a much more efficient function. Here is an attempt to do segmentation using openCV.

using this video as a guide : <https://www.youtube.com/watch?v=-o9jNEbR5P8>

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline


img = cv2.imread('attemptcoins1.jpg')
img_orig = img.copy()
img_crop = img.copy()
#img = np.full((1674,2044,3), 12, dtype = np.uint8)
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.rcParams["figure.figsize"] = (16,9)
plt.imshow(img,cmap = 'gray')
plt.imshow(img)
```

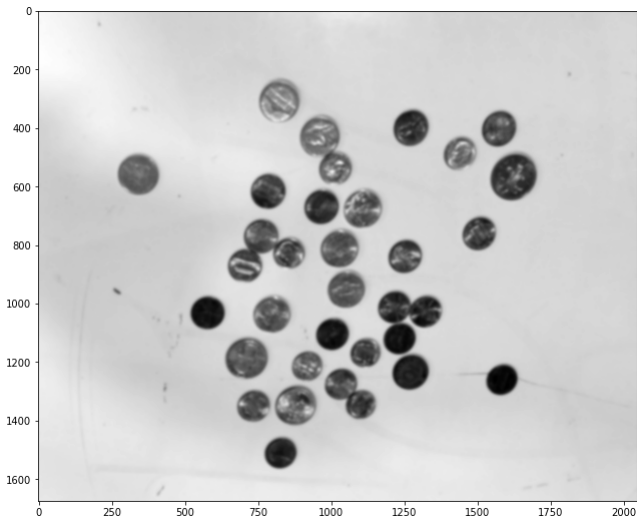
 <matplotlib.image.AxesImage at 0x7f102926fcf8>



blur image as to not find circles within the coin

```
plt.rcParams["figure.figsize"] = (16,9)
img = cv2.GaussianBlur(img, (21,21), cv2.BORDER_DEFAULT)
plt.imshow(img, cmap = 'gray')
```

 <matplotlib.image.AxesImage at 0x7f1029021da0>



```
all_circs = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 0.9, 120, param1 = 50, param2 = 30, minRadius = 60, maxRadius = 125)
all_circs_rounded = np.uint(np.around(all_circs))
```

Double-click (or enter) to edit

```
print(all_circs_rounded)
print(all_circs_rounded.shape)
print('I have found ' + str(all_circs_rounded.shape[1]) + ' coins.')
```

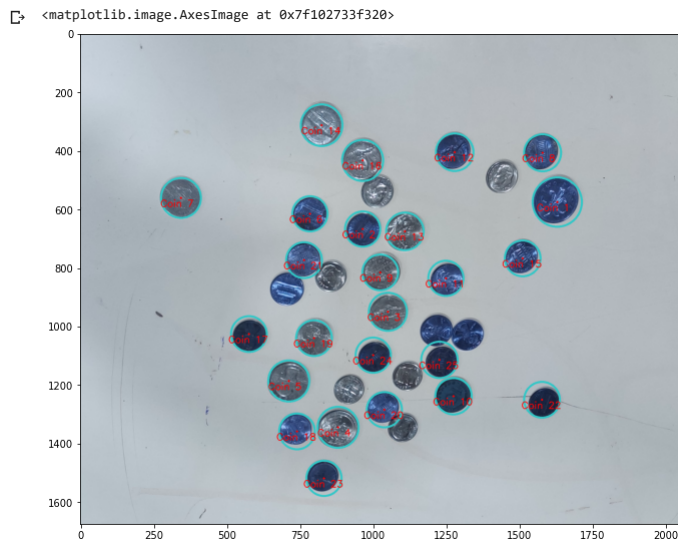
```
### One option for feature extraction is the third column below is radius or size
### So there could be a comparison done
```

```
### This matrix represents x,y, and radius. which can be extracted as a feature
```

```
[[[1628 576 84]
 [ 964 668 60]
 [1050 950 65]
 [ 880 1344 70]
 [ 712 1186 70]
 [ 784 616 60]
 [ 344 562 68]
 [1578 408 63]
 [1024 816 61]
 [1274 1238 62]
 [1248 836 60]
 [1278 406 64]
 [1108 676 64]
 [ 824 314 71]
 [1510 768 61]
 [ 964 434 70]
 [ 576 1026 61]
 [ 740 1358 61]
 [ 798 1040 61]
 [1038 1284 60]
 [ 764 774 61]
 [1576 1250 61]
 [ 832 1518 60]
 [1000 1098 60]
 [1226 1114 62]]]
(1, 25, 3)
I have found 25 coins.
```

```
count = 1
for i in all_circls_rounded[0, :]:
    cv2.circle(img_orig, (i[0],i[1]),i[2],(50,200,200),5)
    cv2.circle(img_orig,(i[0],i[1]),2,(255,0,0),3)
    cv2.putText(img_orig,'Coin ' + str(count), (int(i[0]-70),int(i[1]+30)), cv2.FONT_HERSHEY_SIMPLEX, 1.1, (255,0,0),2)
    count += 1
```

```
plt.rcParams["figure.figsize"] = (16,9)
plt.imshow(img_orig)
```



Ok, Cool. So I have identified some circle/coins. Not to bad, also pretty bad. It's likely because of the distance between circles parameter I previously clarified.

Side Note: The Gray image is kind of odd, sort of blue. Not sure why, it could be my screen.

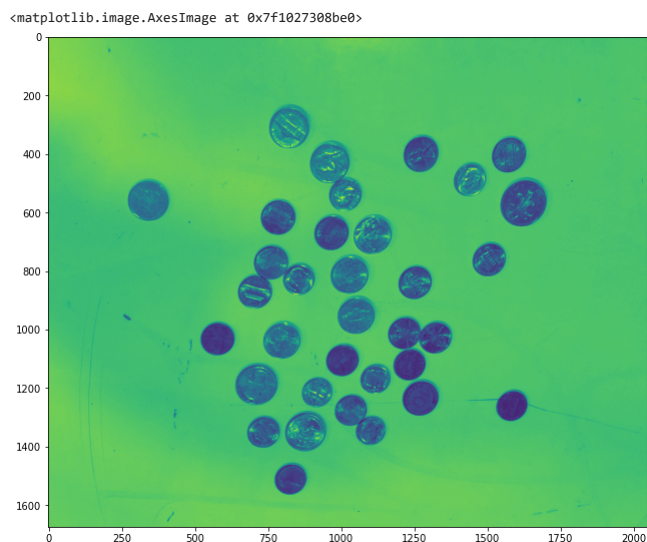
Another Side Note: One of the parameters is minimum radius, which must have been too big because none of the dimes were identified.

The next step (to get started) is to see about cropping out the individual coins and making them into their own photos

Above I have the matrix which is position and radius. So if I crop the image using that information then it should be easy to create a function to do so.

```
#img_crop = np.full((1674,2044,3), 12, dtype = np.uint8)
img_crop = cv2.cvtColor(img_crop, cv2.COLOR_RGB2GRAY)
plt.rcParams["figure.figsize"] = (16,9)
plt.imshow(img_crop,cmap = 'gray')

plt.imshow(img_crop)
```



```
### Attempt to crop the 1st coin listed in the counted coins
### I understand that there are missing coins from the list
```

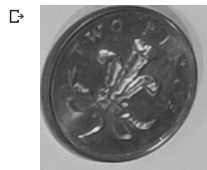
```
# The crop function is done from the top left corner
```

```
from PIL import Image
img_1 = Image.open("attemptcoins1.jpg")
img_1 = img_1.convert('L')
x1 = ((all_circs_rounded[0][0])[0]) - ((all_circs_rounded[0][0])[2])
x2 = ((all_circs_rounded[0][0])[0]) + ((all_circs_rounded[0][0])[2])
y2 = ((all_circs_rounded[0][0])[1]) + ((all_circs_rounded[0][0])[2])
y1 = ((all_circs_rounded[0][0])[1]) - ((all_circs_rounded[0][0])[2])
```

```
img_new = img_1.crop(box = (x1,y1,x2,y2))
```

```
#plt.rcParams["figure.figsize"] = (16,9)
#plt.imshow(img_new,cmap = 'gray')
#img_new
```

```
img_new
```



```
all_circs_rounded
```

```
array([[1628, 576, 84],
       [ 964, 668, 60],
       [1050, 950, 65],
       [ 880, 1344, 70],
       [ 712, 1186, 70],
       [ 784, 616, 60],
       [ 344, 562, 68],
       [1578, 408, 63],
       [1024, 816, 61],
       [1274, 1238, 62],
       [1248, 836, 60],
       [1278, 406, 64],
       [1108, 676, 64],
       [ 824, 314, 71],
       [1510, 768, 61],
       [ 964, 434, 70],
       [ 576, 1026, 61],
       [ 740, 1358, 61],
       [ 798, 1040, 61],
       [1038, 1284, 60],
       [ 764, 774, 61],
       [1576, 1250, 61],
       [ 832, 1518, 60],
       [1000, 1098, 60],
       [1226, 1114, 62]], dtype=uint64)
```

Some notes:

It seems that from here we could classify the coins based on their radius, and the radius could be the sole defining feature. This does not include any machine learning or pattern recognition, and could have trouble distinguishing between U.S. and Non-U.S. Currency, as well as coins close in size like that of pennies, and dimes.

This Part of the project is supposed to be about preparing the image, or image processing. Since we would like to use a classifier to determine what type of coin this is-- the next steps looks like this: Find a way to keep align all the coins in a direction which reduces the work done by the classifier (and improves accuracy!!)

Once the images are processed, the features then can be pixels, similar to the OCR project. This means that all images need to be centered, and cut to the correct size, and then put into a large dataset.

Our training data sets can also be generated this way.

img_new



Alright, Some more notes:

I would like to start to play with more image processing, and I would like to do this with openCV library. So I want to recrop the photo if possible, and I would like to rotate the photo.

Update: If I use PIL to do the cropping, then it makes the image easier to work with.

I'd like to patch this up into a smooth process which receives a photos and returns a numpy array, which can then be used as the test or training data.

img_new



```
### New Function, Make Big Images with Many Coin into Many Small Images with Big Coin
#### This function will accept a jpg and then crop it it into Smaller images of individual coins

from PIL import Image
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

def coincutter(image):
    # The first step for non training data is to read in the image and convert it to grayscale
    img = cv2.imread(image)
    img_orig = img.copy()
    #img = np.full((1674,2044,3), 12, dtype = np.uint8)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.GaussianBlur(img, (21,21), cv2.BORDER_DEFAULT) ### Here we blur the images to help find the right circles

    # This uses PIL to create a new image that we can crop soon
    img_1 = Image.open(image)
    img_1 = img_1.convert('L')

    # Then we find the circles in the image using HoughCircles, and receive location and radius of all circles
    all_circls = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 0.92, 100, param1 = 37, param2 = 20, minRadius = 40, maxRadius = 250)
    all_circls_rounded = np.uint(np.around(all_circls))
    X_test = np.arange(all_circls_rounded.shape[1]*4096).reshape(all_circls_rounded.shape[1],4096)
    X_test[:,0] = 0

    # This is for showin the whole photo with the circles identified
    count = 1
    for i in all_circls_rounded[0, :]:
        cv2.circle(img_orig, (i[0],i[1]),i[2],(50,200,200),5)
        cv2.circle(img_orig,(i[0],i[1]),2,(255,0,0),3)
        cv2.putText(img_orig,'Coin ' + str(count), (int(i[0]-70),int(i[1]+30)), cv2.FONT_HERSHEY_SIMPLEX, 1.1, (255,0,0),2)
        count += 1
    plt.rcParams["figure.figsize"] = (16,9)
    plt.imshow(img_orig)

    # Next we need to cycle through the list and crop out all the of the individual coins
    for i in range(0,all_circls_rounded.shape[1]):

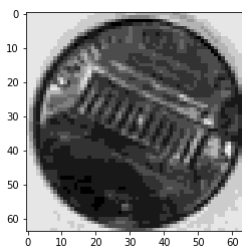
        x1 = ((all_circls_rounded[0][i])[0])-(all_circls_rounded[0][i])[2])
        x2 = ((all_circls_rounded[0][i])[0])+(all_circls_rounded[0][i])[2])
        y2 = ((all_circls_rounded[0][i])[1])+(all_circls_rounded[0][i])[2])
        y1 = ((all_circls_rounded[0][i])[1])-(all_circls_rounded[0][i])[2])

        img_new = img_1.crop(box = (x1,y1,x2,y2))
        img_new = img_new.resize((64,64))
        X0 = np.array(img_new)
        X0 = X0/255*15
        X0 = 15 - X0
        X0 = X0.reshape(1,4096)
        X_test[i] = X0

    # return the array
    return(X_test)

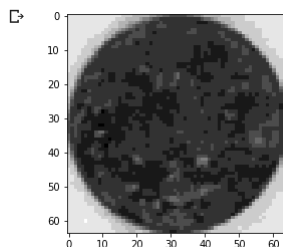
X_test = coincutter("attemptcoins1.jpg")
test = X_test[0].reshape((64,64))
plt.imshow(test,cmap = plt.cm.binary)
plt.show()
```





```
X_test = coineutter("attemptcoins2.jpg")
```

```
test = X_test[0].reshape((64,64))
plt.imshow(test,cmap = plt.cm.binary)
plt.show()
```



Alright, some more notes:

Now I have packaged it into a nice function. You pass in the name of the jpg and you are returned with a data array, prime for a classifier. The next thing I need to do is make sure its getting all the coins, this just means adjusting some parameters in the function.

```
print(X_test.shape)
```

```
(12, 4096)
```

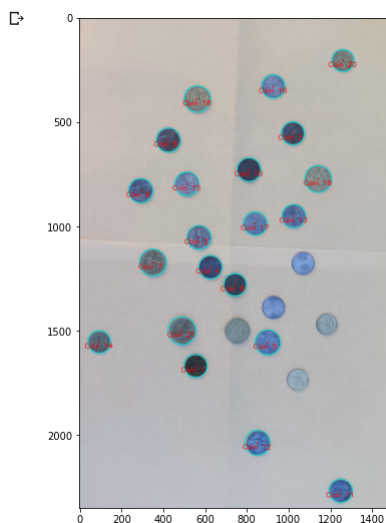
```
### This above shows me that current function only identifies 12 out of 26 coins.
### To fix this i must adjust the parameters of the hough circles
```

```
print(X_test.shape)
```

```
(21, 4096)
```

```
### 21 is even closer to 26, it would be helpful to know which circles
### arent being classified.
```

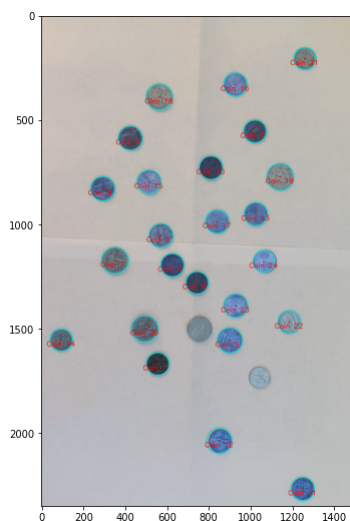
```
X_test = coineutter("attemptcoins2.jpg")
```



```
### going to play with the parameters againn
### all_circls = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 0.92, 100,
### param1 = 50, param2 = 15, minRadius = 40, maxRadius = 250)
```

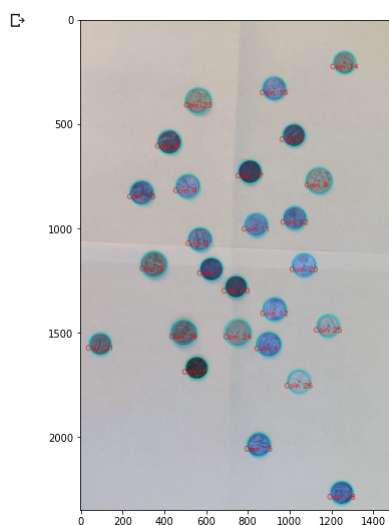
```
X_test = coineutter("attemptcoins2.jpg")
```

```
(21, 4096)
```



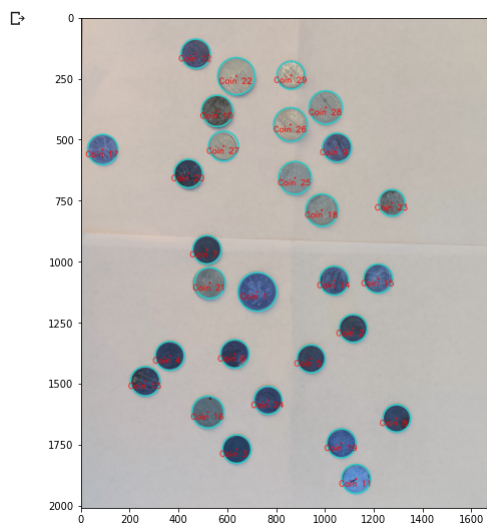
```
### Still need a little more out of this
### all_circls = cv2.HoughCircles(img, cv2.HOUGH_GRADIENT, 0.92, 100,
### param1 = 37, param2 = 20, minRadius = 40, maxRadius = 250)
```

```
X_test = coineutter("attemptcoins2.jpg")
```



```
### Great now that I have all these coins noticed, I can test a couple more groups
```

```
X_test = coineutter("attemptcoins3.jpg")
```



```
X_test = coineutter("attemptcoins4.jpg")
```



