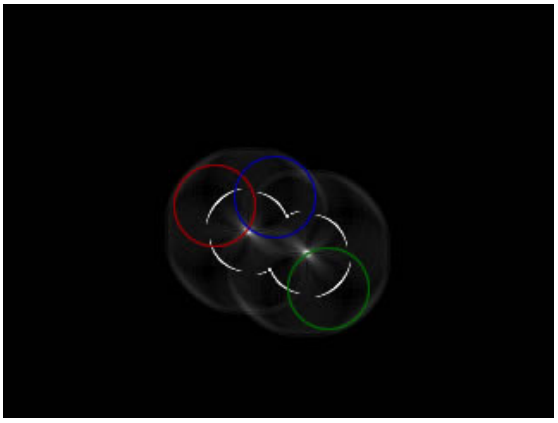
Rebecca Dsouza | Roll No 56 | Batch D | Group 2

Circle Detection in OCR by Hough Transform

Introduction

The Hough transform in its simplest form is a method to detect straight lines but it can also be used to detect circles or ellipses. The algorithm assumes that the edge is detected and it is robust against noise or missing points.



Mathematics

A circle can be described completely with three pieces of information: the center (a, b) and the radius. (The center consists of two parts, hence a total of three)

```
x = a + Rcosθ
y = b + Rsinθ
```

When the θ varies from 0 to 360, a complete circle of radius R is generated.

So with the Circle Hough Transform, we expect to find triplets of (x, y, R) that are highly probably circles in the image. That is, we want to find three parameters. Thus, the parameter space is 3D.

If different values of R are tried, every point in the xy space will be equivalent to a circle in the ab space (R isn't a parameter, we already know it). This is because on rearranging the equations, we get:

```
a = x1 - Rcosθ
b = y1 - Rsinθ
```

for a particular point (x1, y1). And θ sweeps from 0 to 360 degrees.

Algorithmic Steps

- Load an image
- Detect edges and generate a binary image
- For every 'edge' pixel, generate a circle in the ab space

- For every point on the circle in the ab space, cast 'votes' in the accumulator cells
- The cells with greater number of votes are the centers

Uploading Image from Desktop

```
from google.colab import files
uploaded = files.upload()
```

- **ip.png**(image/png) - 6341 bytes, last modified: 10/28/2019 - 100% done

```
Saving ip.png to ip.png
```

Importing Numpy, Scikit Image and Open-CV

```
import numpy as np

from skimage import color
from skimage.transform import hough_circle, hough_circle_peaks
from skimage.feature import canny
from skimage.draw import circle_perimeter

from cv2 import imread
from google.colab.patches import cv2_imshow
```

Reading Image in Binary Form

```
image = imread('ip.png', 0)
cv2_imshow(image)
```

Applying Canny Edge Detection to Binary Image

```
edges = canny(image, sigma=3, low_threshold=10, high_threshold=50)
cv2_imshow(np.where(edges.astype(int)==1, 255, edges.astype(int)))
```

Computing Hough Circles on Edge Detected Image

```
hough_radii = np.arange(20, 35, 2)
hough_res = hough_circle(edges, hough_radii)
accums, cx, cy, radii = hough_circle_peaks(hough_res, hough_radii,total_num_peaks=5)
```

Plotting Hough Circles on Image

```
image = color.gray2rgb(image)
for center_y, center_x, radius in zip(cy, cx, radii):
    circy, circx = circle_perimeter(center_y, center_x, radius, shape=image.shape)
    image[circy, circx] = (57, 255, 20)

cv2_imshow(image)
```