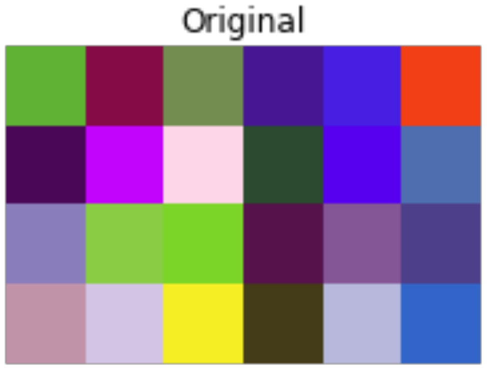
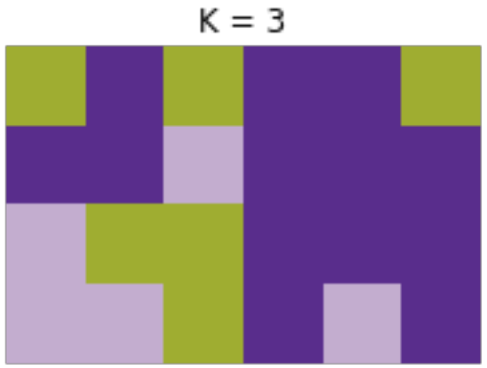
CS 4476 Project 4

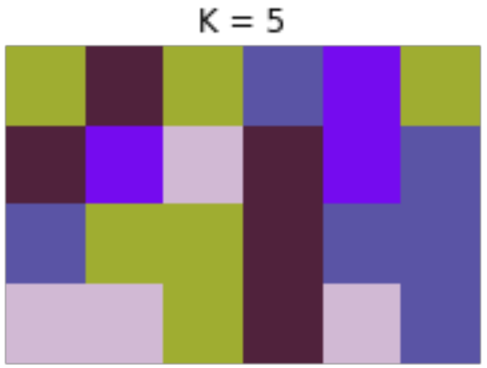
Heejoo Jin
hjin77@gatech.edu
hjin77
903304385

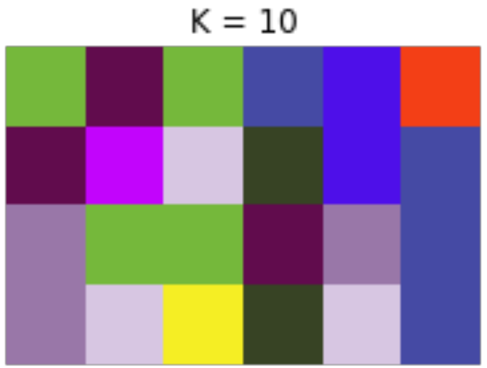
1.1 Color Quantization of RGB Images

< insert visualizations of the quantized RGB image at k = 3, 5, 10 here as noted proj4.ipynb>





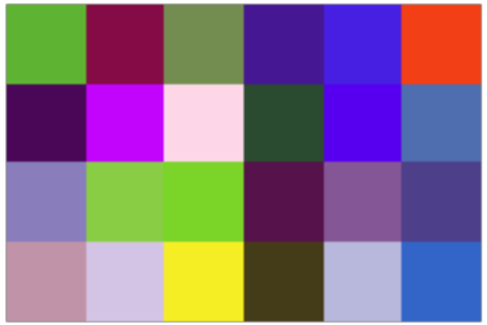




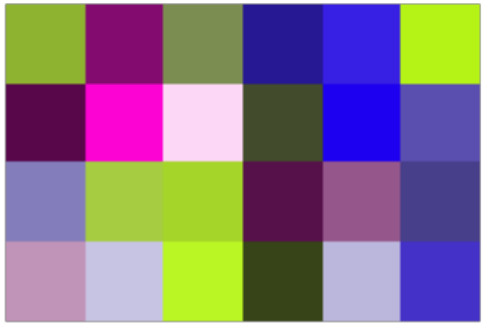
1.2 Color Quantization of HSV Images

< insert visualizations of the quantized HSV image at k = 3, 5, 10 here as noted proj4.ipynb>

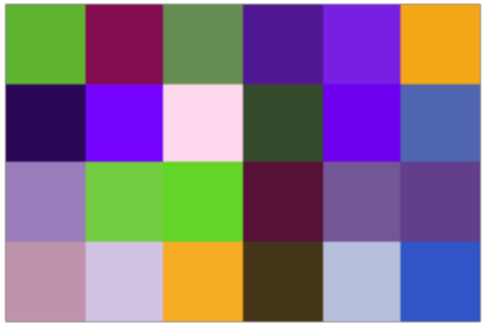
Original



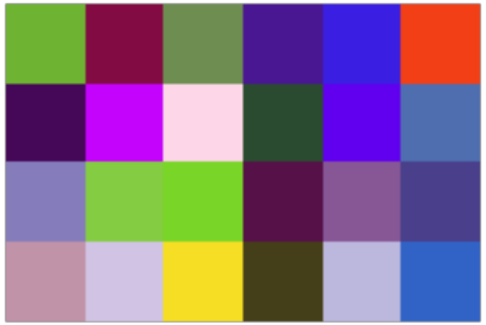
K = 3



K = 5



K = 10



1.3 Logarithmic Quantization Error

< Enter values as $\log_e$(error) >

k	RGB	HSV
3	20.39667281221383	19.19164769405786
5	19.952058124071375	18.074701228036147
10	19.386139229652386	16.981677729401962

1.4 Brief Answers

a) As the number of quantization bins increases, there will be a smaller number of lost/ reduced colors in the image. Therefore, there will be more global color information captured and intact.

b) RGB color quantization uses all 3 channels whereas HSV color quantization only uses its Hue channel. Unlike RGB channels, HSV channels separates color information into Hue, Saturation, and Value (brightness), therefore, the Hue channel is not affected by grayscale level (intensity of light). Thus, using only the Hue channel can present the reduced color information better than RGB. In terms of the qualitative results between RGB and HSV color spaces, the result from HSV quantization will have less information lost, therefore, have a smaller value for SSD error between the original pixel values and the quantized values.

c) 1. Sum of absolute differences (SAD)
It simply calculates the sum of absolute differences. This method will result in a smaller error value compared to SSD. Hence, this less penalizes the difference between the original pixel values and the quantized values.
2. Zero mean normalized cross correlations (ZNCC)
The brightness of two images can have different lighting/ exposure conditions, therefore, the images can be first normalized. Then it uses the local mean value of sub-images & standard deviation to get ZNCC value. ZNCC value that equals to +1 means the image is identical, -1 means one image is the negative of the other one, and finally, 0 means the two images are not so correlated.

2.1 Circle Detection with Hough Transform

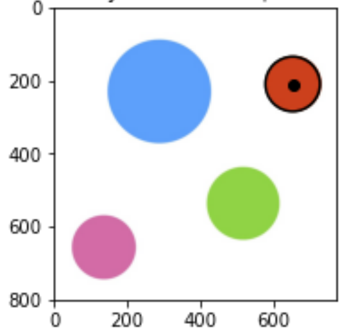
< Briefly explain implementation in concise steps (bullet points / listed steps preferred)>

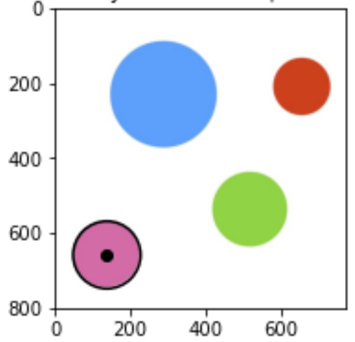
1. Get edges of the image using Canny edge detection algorithm.
2. Initialize an accumulator array to zero.
3. If the algorithm is using the gradient of the image, change the image to grayscale and process filtering by getting the gradient in X & Y directions using Sobel filters. If the algorithm is not using the gradient of the image, only convert the image to grayscale.
4. For each edge point in the image, count circles in the accumulator array.
5. Find values in the accumulator array that have “votes” greater than the thresholded value (= threshold * maximum votes in the accumulator array). This process is called “voting”.
6. After “voting”, the positions found in the accumulator array correspond to the centers of the detected circles in the original image.
7. Using the given radius and detected centers, we can visualize the detected circles!

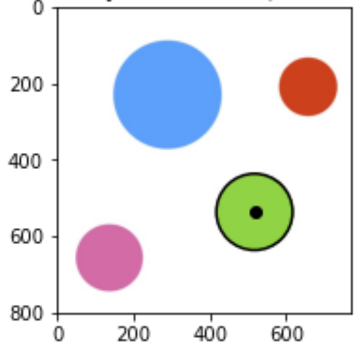
2.2(a) Circle Detection on Synthetic Images

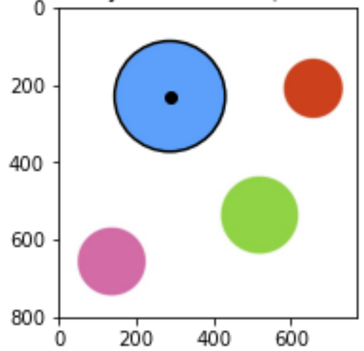
Threshold = 0.95

< Insert useGradient = True images here>

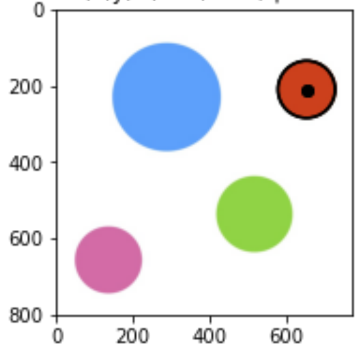

Center (x,y): (654, 210) | Radius: 75

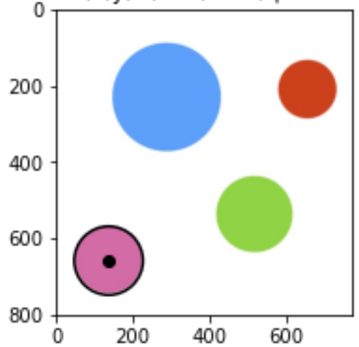

Center (x,y): (137, 659) | Radius: 90

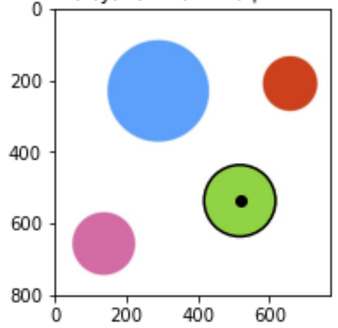

Center (x,y): (517, 537) | Radius: 100

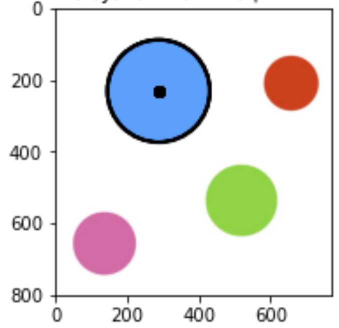

Center (x,y): (287, 230) | Radius: 143

< Insert useGradient = False images here>


Center (x,y): (655, 210) | Radius: 75


Center (x,y): (136, 659) | Radius: 90

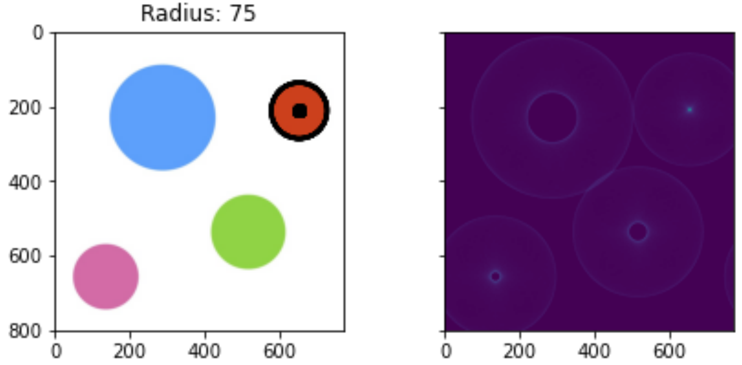

Center (x,y): (517, 537) | Radius: 100

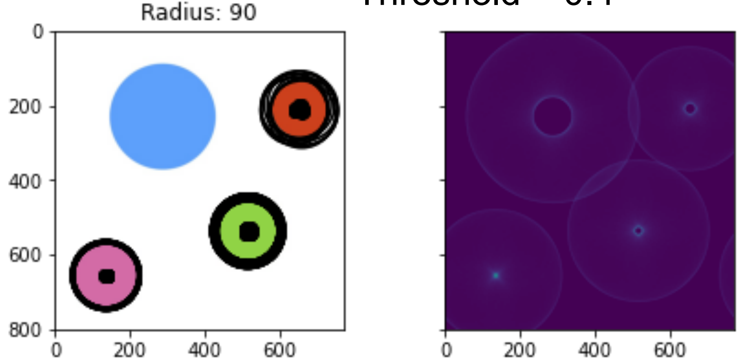

Center (x,y): (289, 233) | Radius: 143

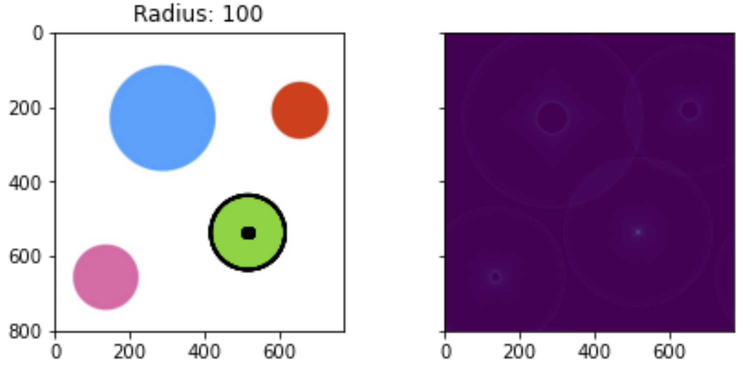
2.2(b) Circle Detection on Synthetic Images

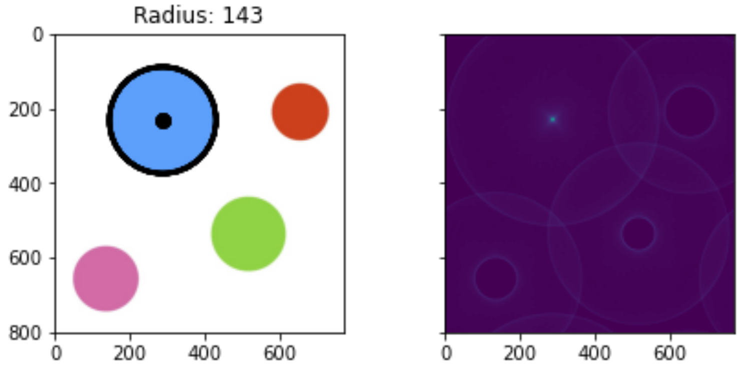
< Insert low threshold images and hough accumulator array here>

UseGradient = False
Threshold = 0.4





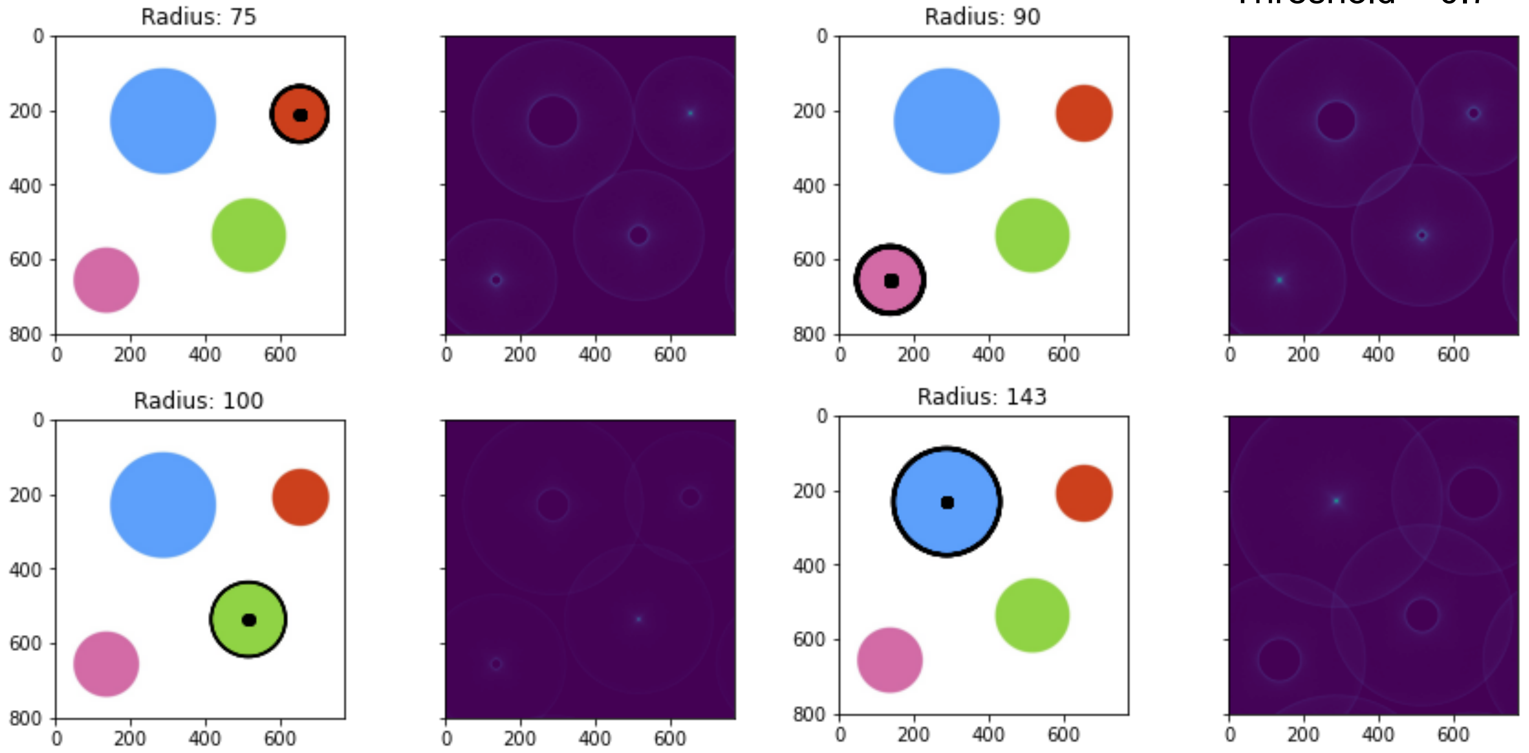




2.2(b) Circle Detection on Synthetic Images

< Insert mid-range threshold images and hough accumulator array here>

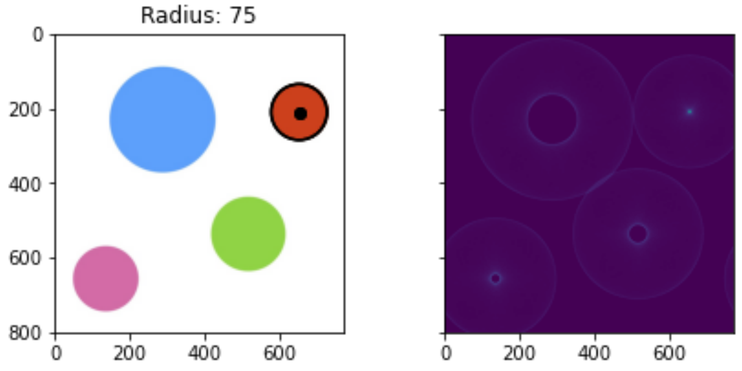
UseGradient = False
Threshold = 0.7

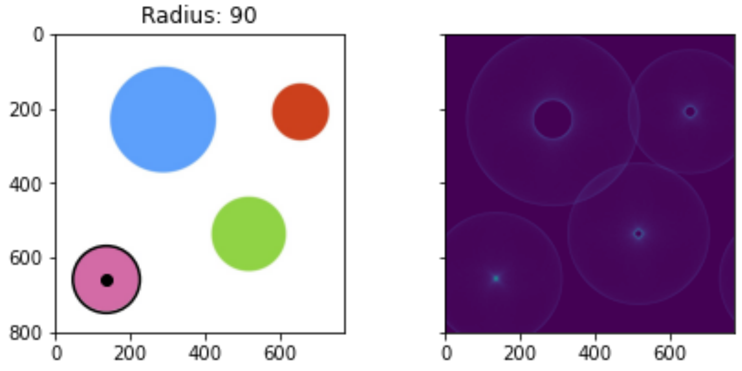


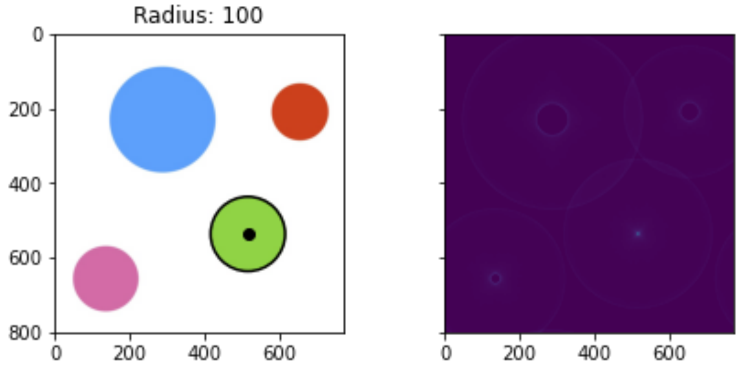
2.2(b) Circle Detection on Synthetic Images

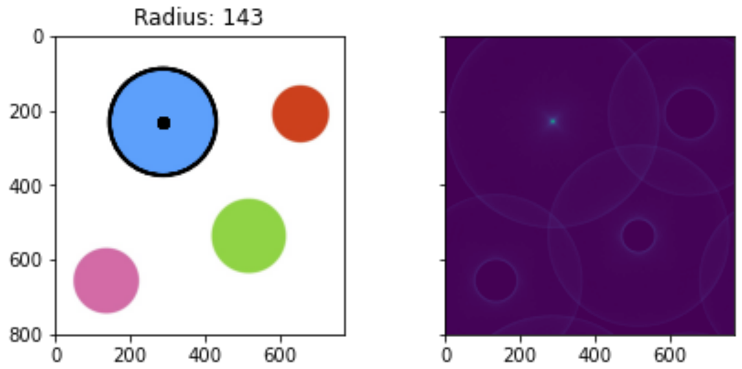
< Insert high-range threshold images and hough accumulator array here>

UseGradient = False
Threshold = 0.95









2.2(b) Circle Detection on Synthetic Images

< Explain how results vary with increasing thresholds>

In Hough circle transform algorithm, the points with ”votes” that are greater than the thresholded value (= threshold * the maximum votes in the neighboring area) are chosen.

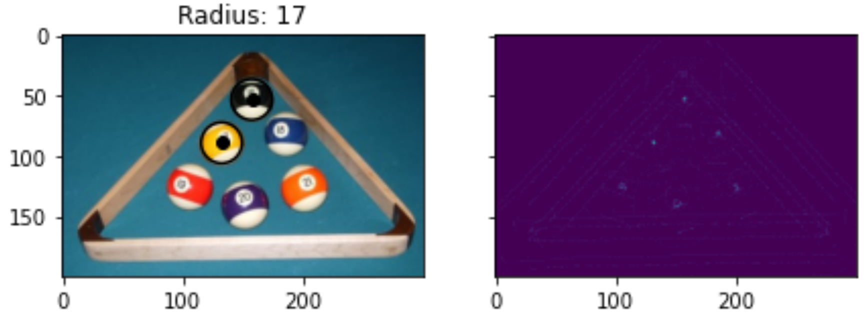
As the threshold value increases, there will be a smaller number of points with ”votes” that are greater than the threshold. This is likely to get rid of more points and leave less points left. Hence, the detected circles will appear more clearly with much less noise.

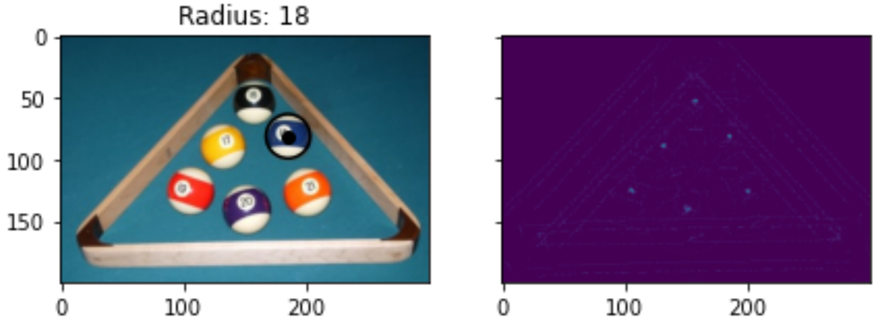
However, if the threshold is too high, it might get rid of too many points, hence, the threshold value should be chosen carefully.

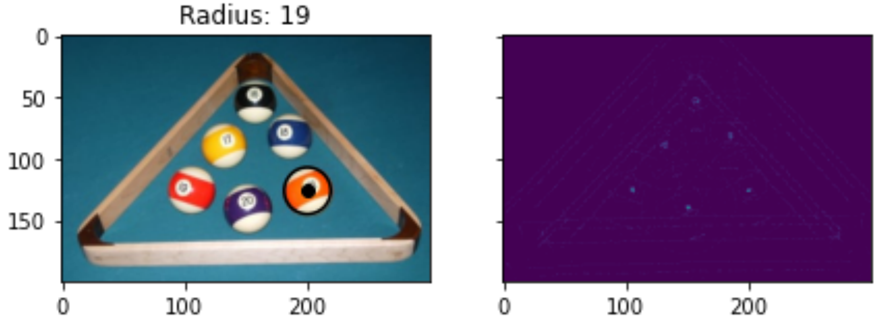
2.3 Circle Detection on Real Images

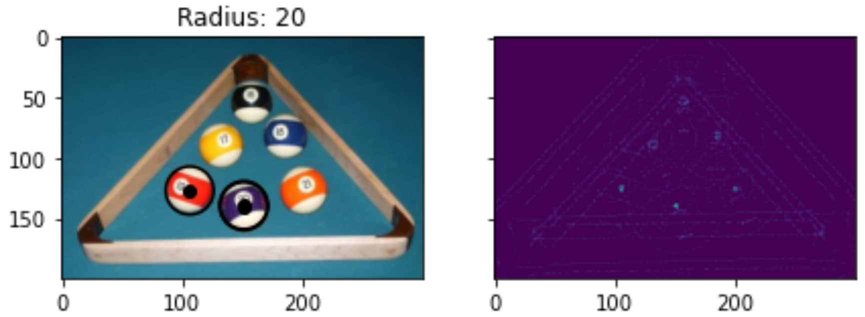
< Include image showing detected circles here >

UseGradient = True
Threshold = 0.85



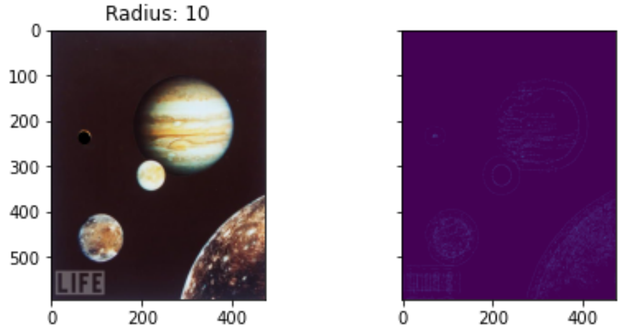


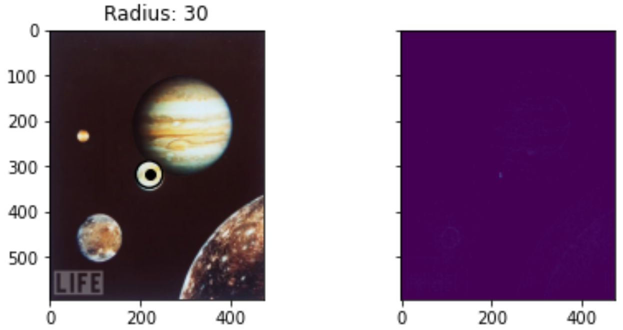


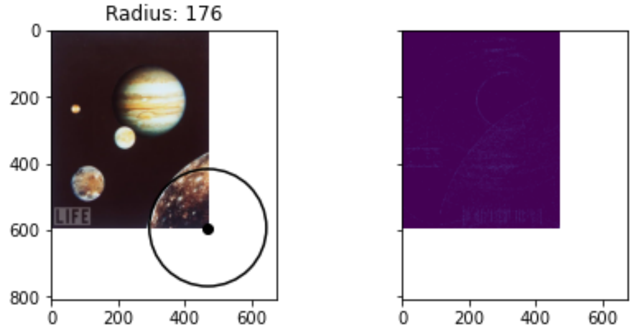


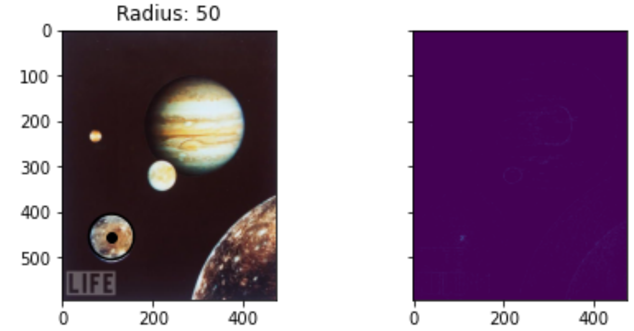
3 Unknown Radii Circle detection

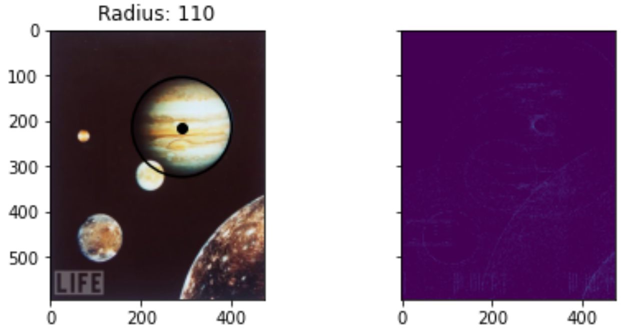
< Include image showing detected circles here >











UseGradient = True
Threshold = 0.95