Short Answer 1

The representation based on the filter bank is invariant to orientation because it has filters that have been rotated across both scales.

Short Answer 2

A likely clustering assignment would split the circles down the center since there aren't any outliers to throw off the grouping.

Short Answer 3

Out of the three, mean-shift would be the most appropriate for a continuous voting space because it will converge to the center of the most votes. By contrast, k-means would only find the center of a cluster while graph-cut will cut along lines not in the center.

Short Answer 4

For each blob begin with a point inside. Then for each point on the boundary of the blob, measure the vector between the inside point and the boundary point and store this vector in a HashMap indexed by gradient orientation. For the other blobs, use the HashMap vectors on each point on the edges to vote for a center. The blobs with the most votes belong in the same group.

2.1-1

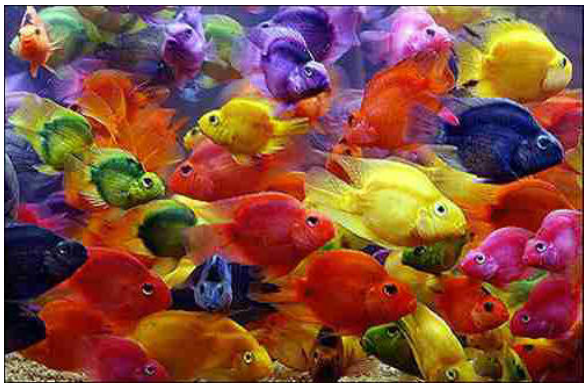


Figure 1: Original RGB Image

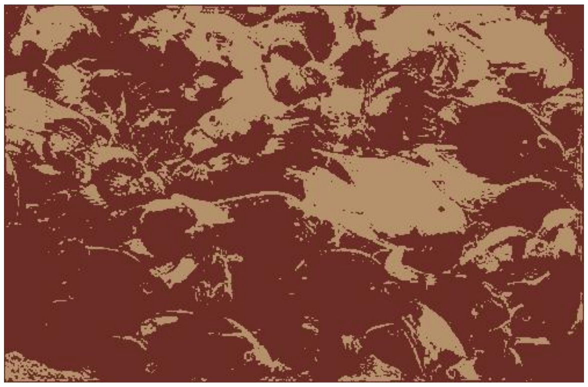
(a) 

Figure 2: Quantized RGB k = 2

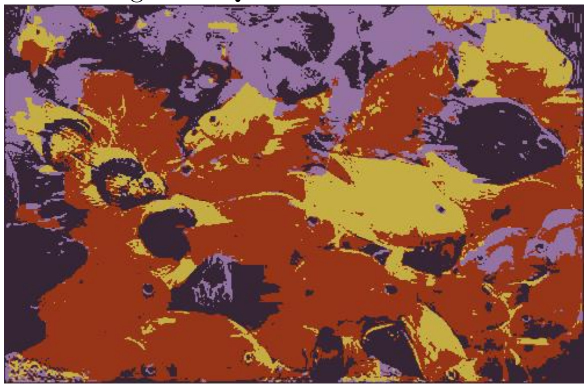


Figure 3: Quantized RGB k = 4

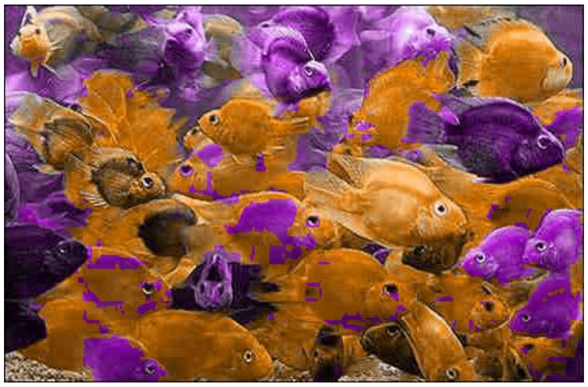
(b) 

Figure 4: Quantized HSV k = 2

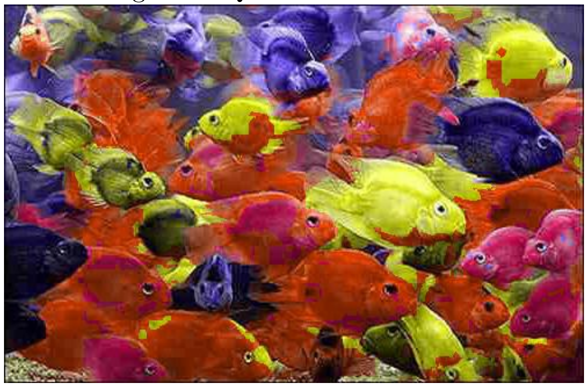
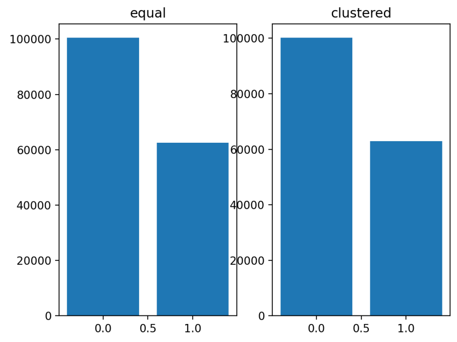


Figure 5: Quantized HSV k = 4

(c) SSD Error RGB (k = 2) = 1018982460.0
SSD Error RGB (k = 4) = 563902000.0

(d) SSD Error HSV (k = 2) = 400070430.0
SSD Error HSV (k = 4) = 90368480.0



(e)

Figure 6: Hue Histograms for k = 2

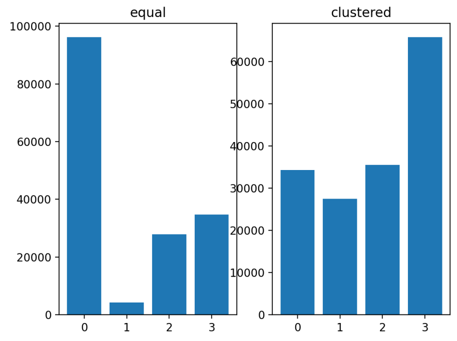


Figure 7: Hue Histograms for k = 4

2.1-2

Between the different k values, the histograms differ both in the size of the bins and the number of bins. The equal histograms divided the hue space into k equal size bins while the clustered histograms divided the hue space into k bins based on k-means clustering. The results for the RGB space differ from the results of the HSV space because in the case of RGB, the image is quantized into k RGB values while in the case of HSV, the hue channel is quantized into k values. This allows the HSV to preserve the brightness and saturation of the original, giving it a smaller error than the RGB. k determines the number of hues or RGB values. As k increases, the error falls since there are more RGB values to choose from, allowing for a closer approximation of the original image. Since HSV allows for more RGB values, this also contributes to HSV's lower error than RGB.

2.2-1

For the Hough transform implementation, first I converted the input image to grayscale. Then the Canny edge detection method was used (jupiter.jpg used a sigma of 4 while egg.jpg used a sigma of 2) to get the edges of the image. Then the gradient direction matrix was derived using the x and y components of np.gradient and arctan. For the case of the gradient not being used, for all coordinates that contain an edge, a and b are calculated using $a = x + cos(theta)$ and $b = y + sin(theta)$ where $0 \leq theta \leq 2\pi$ and vote for (b, a) in the accumulator array. In the case where the gradient is being used, $theta$ is derived directly from the gradient direction at (x,y) and two points are voted for. The points with the maximum votes in the accumulator array are determined to be the centers of the circles in the image that match the radius.

2.2-2

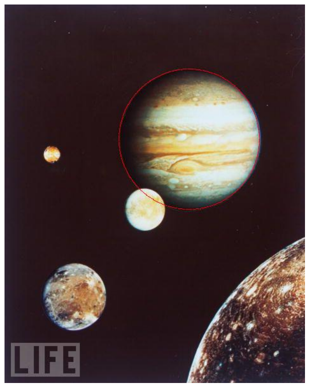


Figure 8: Jupiter image with no gradient and radius = 110

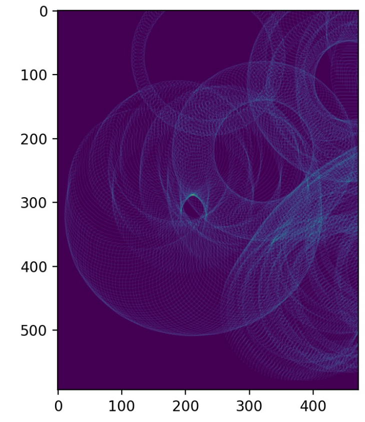


Figure 9: Jupiter image accumulator with no gradient and radius = 110

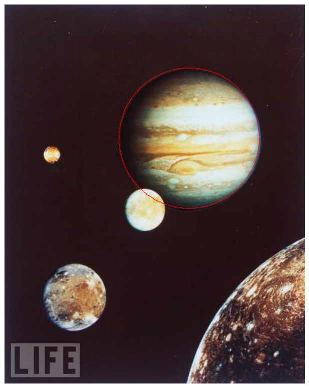


Figure 10: Jupiter image with gradient and radius = 110

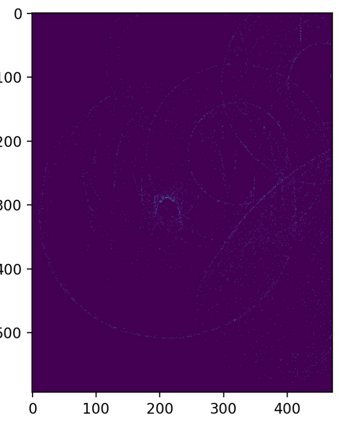


Figure 11: Jupiter image accumulator with gradient and radius = 110

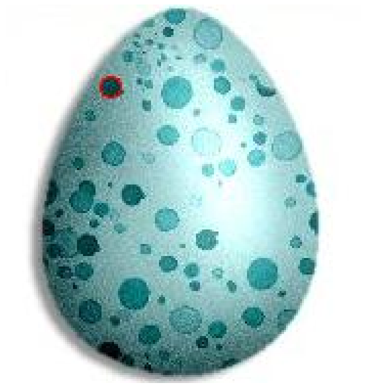


Figure 12: Egg image with no gradient and radius = 5

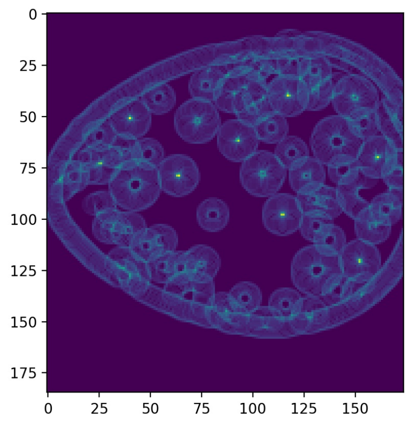


Figure 13: Egg image accumulator with no gradient and radius = 5

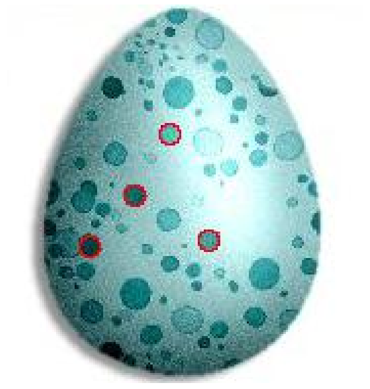


Figure 14: Egg image with gradient and radius = 5

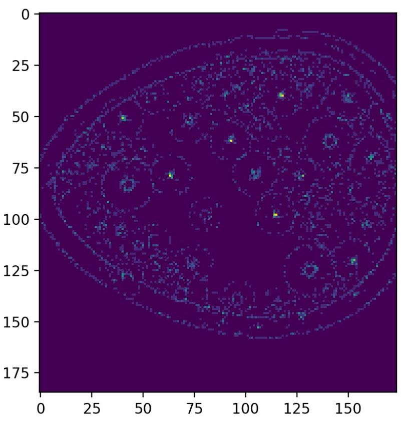


Figure 15: Egg image accumulator with gradient and radius = 5

2.2-3

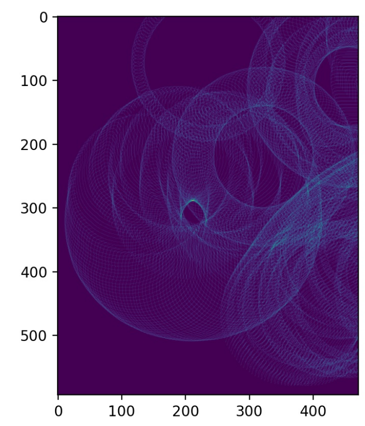


Figure 16: Jupiter image accumulator with no gradient and radius = 110

In the accumulator array, one can see the circles made by each point and in the case of Jupiter, there are circles overlapping around Jupiter's center, and those overlaps increases the score around there, leading to the center being determined.

2.2-4

More circles can be displayed by decreasing the threshold of what centers can be accepted by the detect circles function (using np.argwhere($hough > x * maximum$) where $x < 1$ instead of np.argwhere($hough == maximum$)

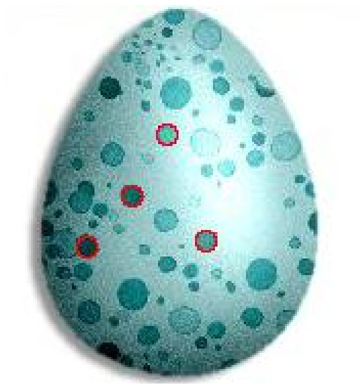


Figure 17: Egg gradient where $x = 0.9$

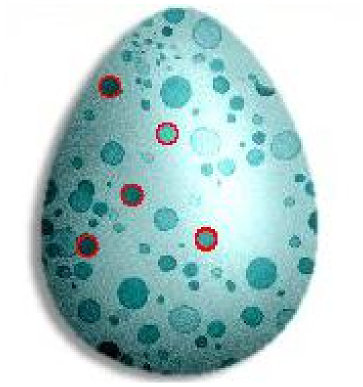


Figure 18: Egg gradient where $x = 0.8$

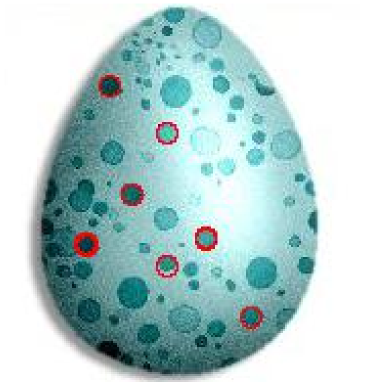


Figure 19: Egg gradient where $x = 0.7$

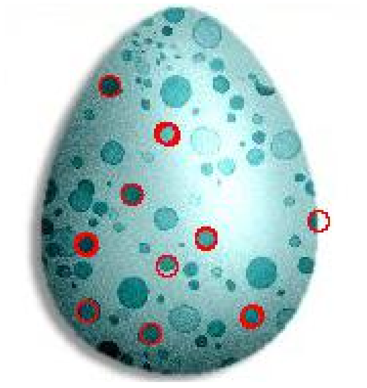


Figure 20: Egg gradient where $x = 0.6$

2.2-5

The following accumulator array with an increased bin size has higher vote numbers since adjacent pixels are in the same bin.

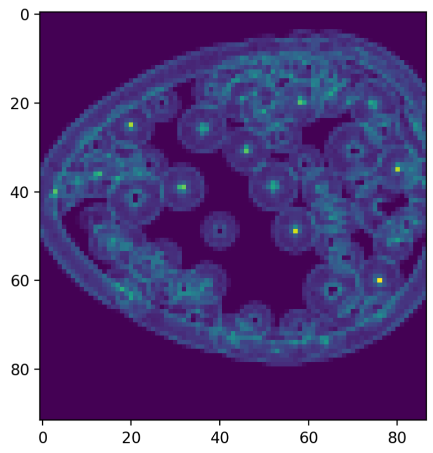


Figure 21: Egg gradient where the bin size has been increased