MSML640 Assignment 3

Ayush Chaudhary (120429125)

March 24, 2024

1 Short Answer Problems

1. **Texture description using textons:** The resulting representation using textons from a filter bank of anisotropic derivatives of Gaussian filters at two scales and six orientations is sensitive to orientation.

   Explaination: The filter bank consists of multiple filters, each designed to respond maximally to specific patterns or textures in the image. These filters are oriented differently (e.g., horizontal, vertical, diagonal), capturing various texture features. Since the filters are sensitive to different orientations, the textons will vary based on the orientation of the texture pattern. Hence, When the orientation of a texture changes (e.g., rotation), the filter responses and resulting textons will also change. Therefore, the representation is sensitive to orientation.

2. **Most likely clustering assignment:** The points will likely form two clusters corresponding to the two concentric circles. The outer circle's points will belong to one cluster, and the inner circle's points will belong to another cluster.

   Explaination: The K-means algorithm aims to partition the data into K clusters by minimizing the sum of squared distances between data points and their respective cluster centers. In this case, the data points are distributed in two concentric circles, with the outer circle forming one cluster and the inner circle forming another cluster. The points in each circle are closer to each other than to points in the other circle, making them likely to be assigned to the same cluster. Therefore, the most likely clustering assignment will be two clusters corresponding to the two concentric circles.

3. The Hough Transform works by transforming the spatial domain representation of lines into a parameter space, where the parameters of interest typically represent the slope and intercept of the lines. Here's how the algorithm works

   (a) **Edge Detection:** Before applying the Hough Transform, edges in the image are typically detected using techniques like the Canny edge detector. This step highlights potential lines in the image.

   (b) **Parameter Space Representation:** Each edge pixel in the image contributes to possible lines in the parameter space. For example, in the case of detecting lines using the equation $y = mx + c$, where m represents the slope and c represents the intercept, each edge pixel corresponds to a sinusoidal curve in the parameter space.

   (c) **Voting:** Each edge pixel votes for the possible lines it could be part of in the parameter space. The intersection of these sinusoidal curves in the parameter space indicates the presence of a line in the image.

   (d) **Accumulator Thresholding:** After the voting process, the accumulator space is examined to find peaks. These peaks indicate the parameters of the lines detected in the image.

   (e) **Line Detection:** Once the parameters of the lines are identified, they can be converted back to spatial coordinates to draw the detected lines on the image.

   Limitations and Challenges:

(a) **Computational Complexity:** The Hough Transform can be computationally expensive, especially for images with a large number of edge pixels. This can make it impractical for real-time applications or processing high-resolution images.

(b) **Parameter Sensitivity:** The performance of the Hough Transform is sensitive to parameters such as the resolution of the parameter space and the threshold for peak detection. Choosing these parameters optimally can be challenging and may require tuning.

(c) **Accurate Edge Detection:** The accuracy of line detection heavily relies on the accuracy of edge detection. If edges are not detected properly, it can lead to incorrect line detection.

(d) **Limited to Staright Lines:** The traditional Hough Transform is designed to detect straight lines. It may struggle with detecting curves or other non-linear shapes.

Potential Solutions and Enhancements:

(a) **Fast Hough Transform:** Several variants of the Hough Transform, such as the Probabilistic Hough Transform or the Randomized Hough Transform, aim to reduce computational complexity by sampling edge pixels or parameter space.

(b) **Adaptive Parameter Selection:** Implementing techniques to automatically adapt the parameters of the Hough Transform based on image characteristics can improve robustness and performance.

(c) **Integration with Deep Learning:** Deep learning techniques can be combined with the Hough Transform to improve edge detection accuracy and handle complex shapes beyond straight lines.

Application Scenario: An application scenario where line detection using the Hough Transform is crucial is in autonomous driving systems. In this scenario, accurate detection of lane markings is essential for lane keeping and autonomous navigation. The Hough Transform can be used to robustly detect lane markings on roads, even in challenging lighting and weather conditions. Accurate lane detection contributes to the safety and efficiency of autonomous vehicles, highlighting the significance and potential impact of the Hough Transform in real-world computer vision tasks.

4. To group blobs based on the similarity of their outer boundary shape into a specified number of groups, you can follow these steps:

(a) **Extract outer boundary:** For each blob in the binary image, extract its outer boundary. This can be achieved using contour tracing algorithms like the Moore-Neighbor tracing algorithm or the Freeman chain code.

(b) **Shape Descriptor Calcualtion:** Calculate a shape descriptor for each blob's outer boundary. A common shape descriptor is the Fourier descriptor, which captures the shape characteristics of the boundary in frequency domain.

(c) **Clustering:** Apply a clustering algorithm to group blobs based on their shape descriptors. One option is to use a clustering algorithm like K-means or hierarchical clustering. Specify the desired number of clusters based on the number of groups you want.

(d) **Assign labels:** Assign a label to each blob based on the cluster it belongs to.

(e) **Visualization:** Optionally, visualize the grouped blobs by overlaying the original image with different colors or labels corresponding to each group.

Key concepts and variables involved:

(a) **Blob:** A connected component in the binary image, representing a region of interest.

(b) **Outer Boundary:** The contour or edge of each blob, which is used to characterize its shape.

(c) **Shape Descriptor:** A numerical representation of the shape characteristics of the outer boundary.

(d) **Clustering:** The process of grouping similar blobs together based on their shape descriptors.

(e) **Number of Groups:** The desired number of clusters or groups into which the blobs will be grouped.

(f) **Clustering algorithm:** A method used to partition the blobs into clusters based on their shape descriptors.

By following these steps, you can effectively group blobs based on the similarity of their outer boundary shape into a specified number of groups, which can be useful in various image analysis and computer vision applications such as object recognition and classification.

2 Programming

1. **Color Quantization with K-Means:**

   (a) **K = 3:**

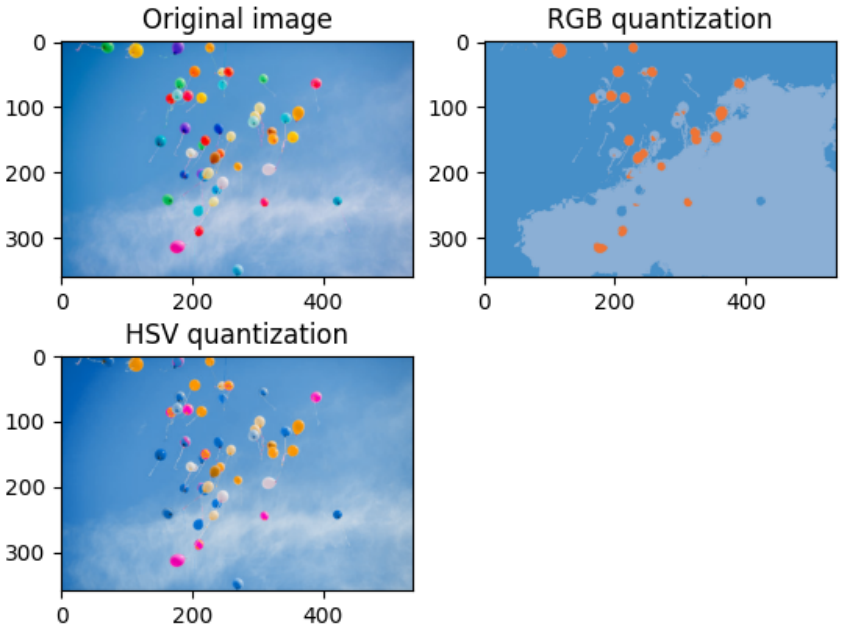
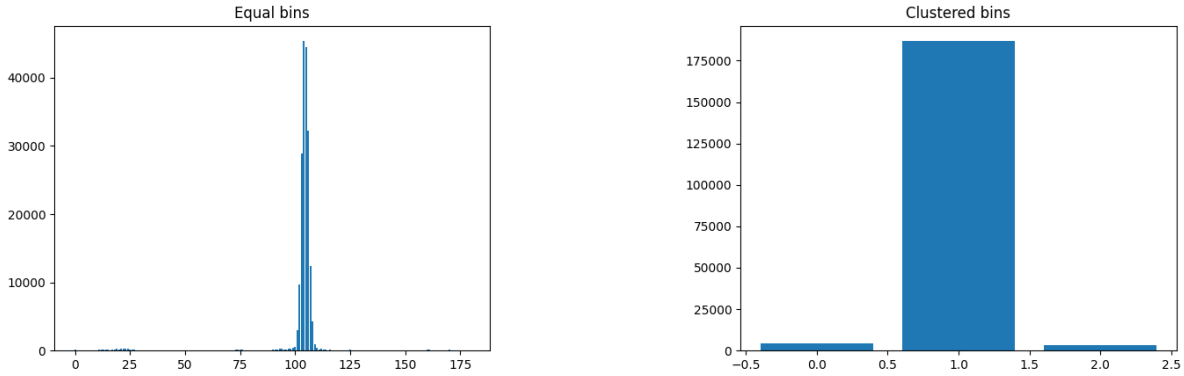


Figure 1: RGB and HSV quantization with k = 3

SSD error for RGB quantization with k = 3: 41266980
SSD error for HSV quantization with k = 3: 5334217



(a) histogram with equally spaced bins

(b) histogram with 3 clusters

Figure 2: Histograms of color quantization with k = 3

Figure 1 shows the histograms of color quantization with k = 3 for both RGB and HSV color spaces. Figure 2 compares the histograms of color quantization with k = 3 using equally spaced bins and 3 clusters.

(b) **K = 5:**

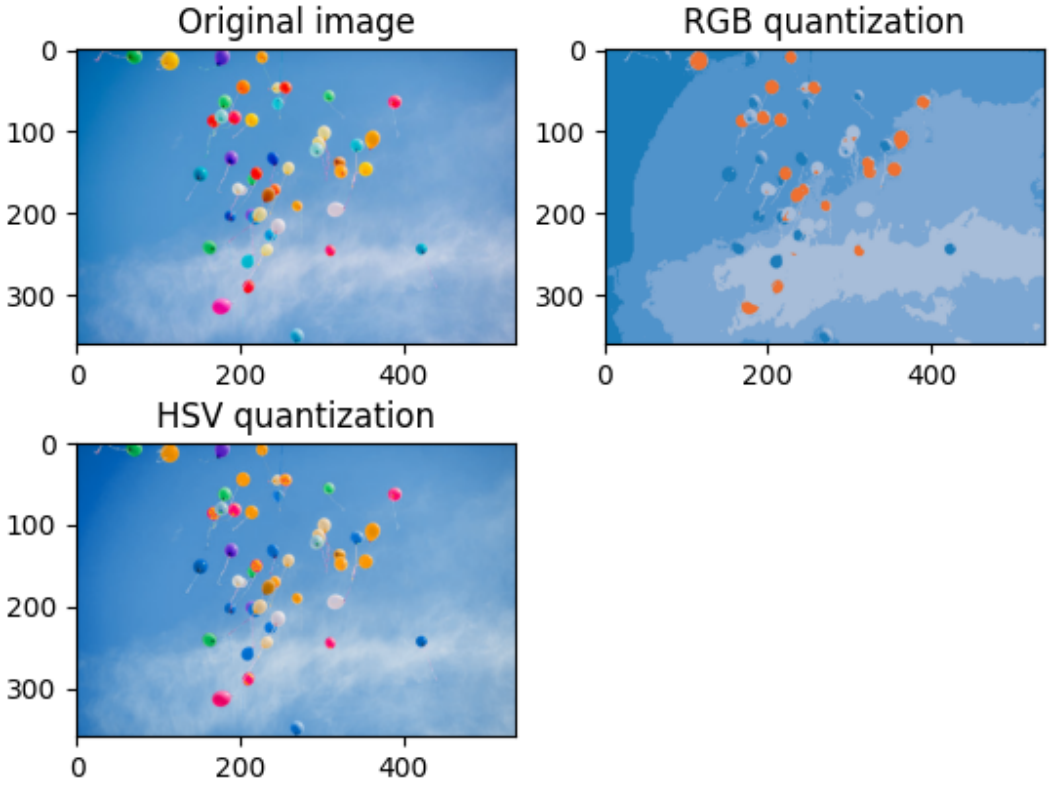
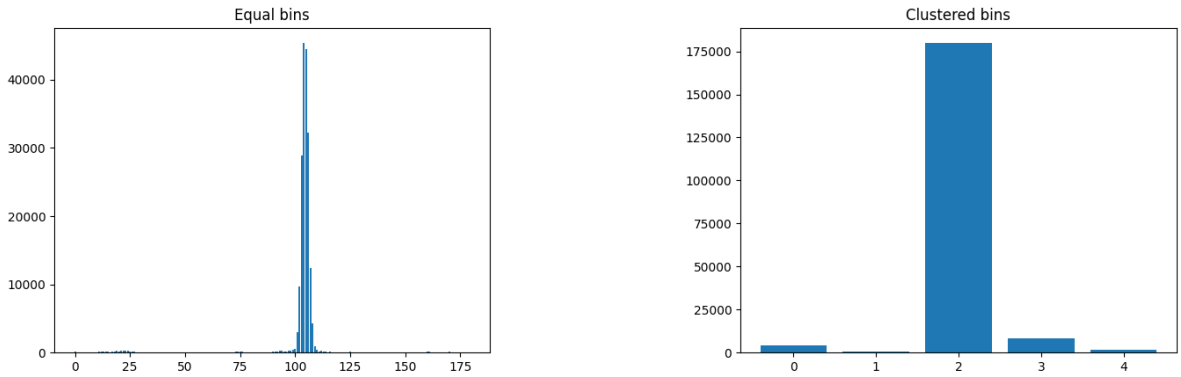


Figure 3: RGB and HSV quantization with k = 5

SSD error for RGB quantization with k = 5: 29841047
SSD error for HSV quantization with k = 5: 5018501



(a) histogram with equally spaced bins

(b) histogram with 5 clusters

Figure 4: Histograms of color quantization with k = 5

Figure 3 shows the histograms of color quantization with k = 5 for both RGB and HSV color spaces. Figure 4 compares the histograms of color quantization with k = 5 using equally spaced bins and 5 clusters.

(c) **K = 7:**

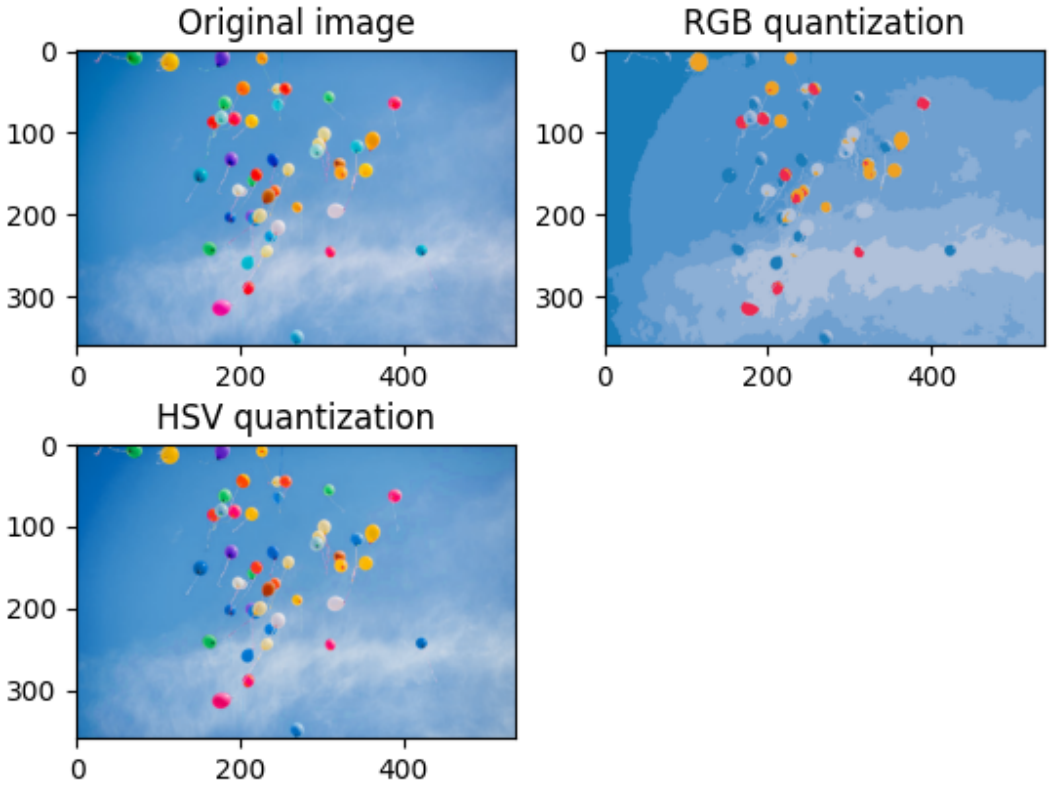
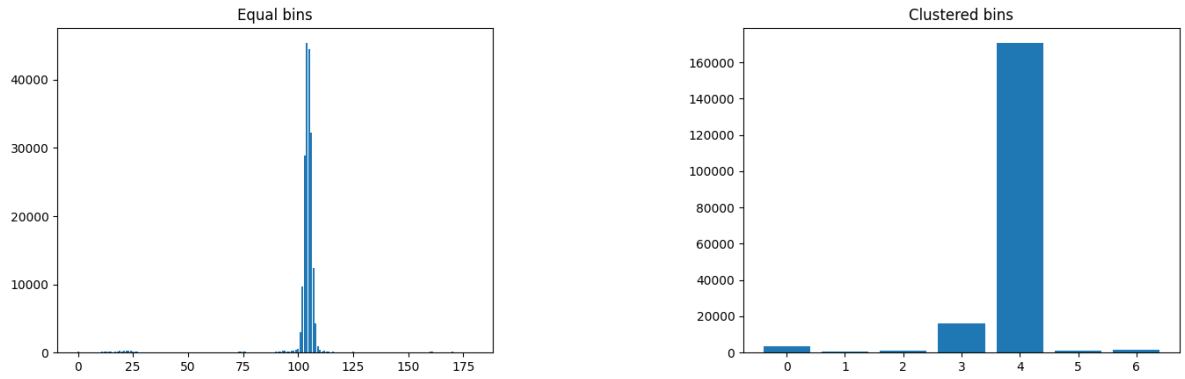


Figure 5: RGB and HSV quantization with k = 7

SSD error for RGB quantization with k = 7: 25768552
SSD error for HSV quantization with k = 7: 2977123



(a) histogram with equally spaced bins

(b) histogram with 7 clusters

Figure 6: Histograms of color quantization with k = 7

Figure 5 shows the histograms of color quantization with k = 7 for both RGB and HSV color

spaces. Figure 6 compares the histograms of color quantization with k = 7 using equally spaced bins and 7 clusters.

(d) **K = 10:**

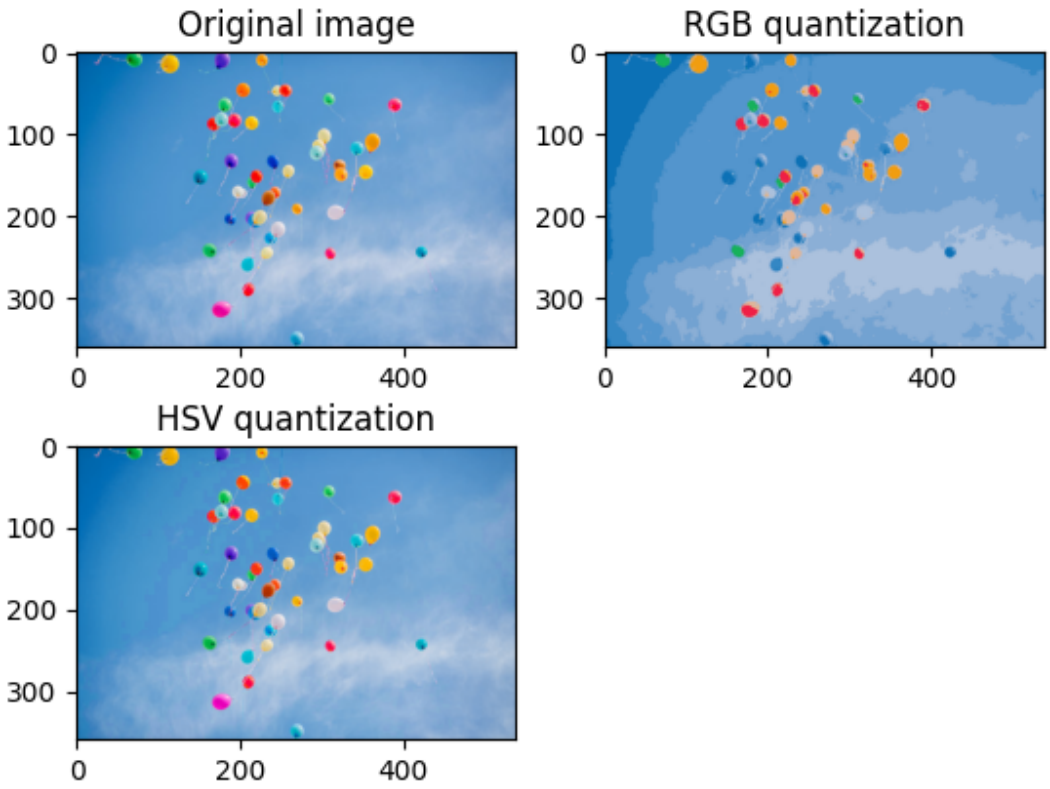
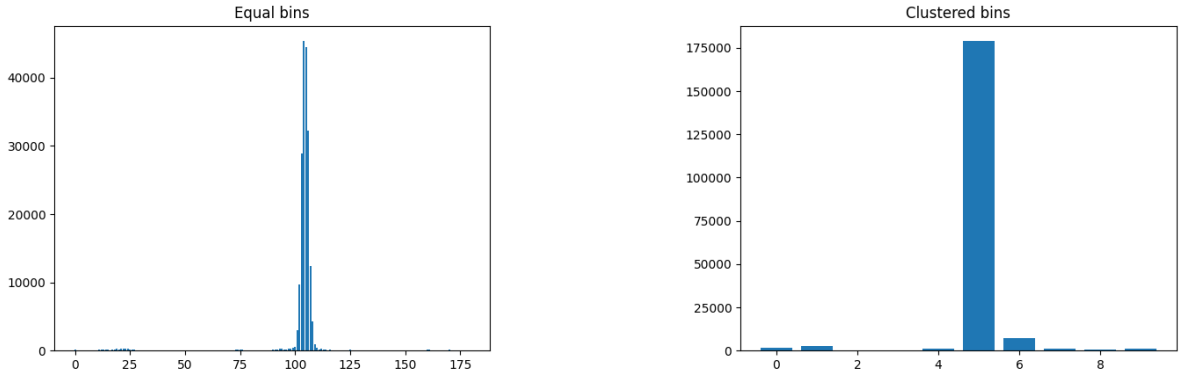


Figure 7: RGB and HSV quantization with k = 10

SSD error for RGB quantization with k = 10: 22065037
SSD error for HSV quantization with k = 10: 2551576



(a) histogram with equally spaced bins

(b) histogram with 10 clusters

Figure 8: Histograms of color quantization with k = 10

Figure 7 shows the histograms of color quantization with k = 10 for both RGB and HSV color spaces. Figure 8 compares the histograms of color quantization with k = 10 using equally spaced bins and 10 clusters.

(e) **Observations:**

i. **Difference in the Histograms:** The histograms of color quantization with k clusters differ from the histograms with equally spaced bins in terms of the distribution of colors. The histograms with k clusters show the distribution of colors based on the cluster centers obtained from K-means clustering, while the histograms with equally spaced bins show a uniform distribution of colors based on the bin boundaries.

ii. **Effect of choosing color space:** The results of color quantization can vary depending on the color space used (RGB or HSV). In the RGB color space, the quantization is based on the individual color channels (Red, Green, Blue), while in the HSV color space, the quantization is based solely on the hue component while keeping the saturation and value components constant. By keeping the saturation and value components constant, the SSD error for HSV quantization is generally lower than RGB quantization, and this leads to better color quantization results in the HSV color space.

iii. **Effect of varying k:** The value of k affects the quality of color quantization. As k increases, the number of clusters used for quantization increases, leading to a more detailed representation of colors in the image. However, increasing k can also lead to overfitting and loss of generalization. Therefore, choosing an optimal value of k is crucial for achieving a balance between color representation and computational complexity.

2. **Circle Detection with Hough Transform:**

(a) **Implementation Explaination:**

i. The detectCircles function takes an input image, the radius of the circles to detect, and optional parameters like the bin threshold, useGradient flag, and post-processing flag.

ii. The input image is converted to grayscale and then smoothed using Gaussian blur to reduce noise.

iii. Canny edge detection is applied to the smoothed image to detect edges.

iv. Parameters for the Hough transform, such as theta range, cosine, and sine values, are computed.

v. Circle candidates are generated based on the given radius and theta values.

vi. A defaultdict named accumulator is created to accumulate votes for circle centers.

vii. If useGradient is True, the gradient direction is calculated using the Sobel operator. Otherwise, the original circle candidates are used.

viii. For each edge pixel in the edge image, the circle candidates are adjusted based on gradient direction (if useGradient is True) and then voted for in the accumulator.

ix. After voting, circles with votes exceeding the bin threshold are shortlisted.

x. Post-processing is optionally applied to remove duplicate circles that are too close to each other.

xi. Finally, the shortlisted circles are drawn on the output image, and the detected circles along with output image(optionally) are returned.

(b) **Demonstration:**

i. **Deer Eyes Image:** Figure 9 shows the results of circle detection on the deer eyes image with a radius of 3 units. The detected circles are highlighted in the image.

A. **Accumulator Array:** Figure 10 shows the accumulator array generated during the circle detection process on the deer eyes image. The peaks in the accumulator array correspond to the detected circles in the image. As it can be observed from the images, the peaks are more defined when the useGradient flag is enabled, leading to more accurate circle detection.

B. **Note:** For the given results, bin threshold is set to 0.7, post-processing is enabled and num thetas is set to 100.

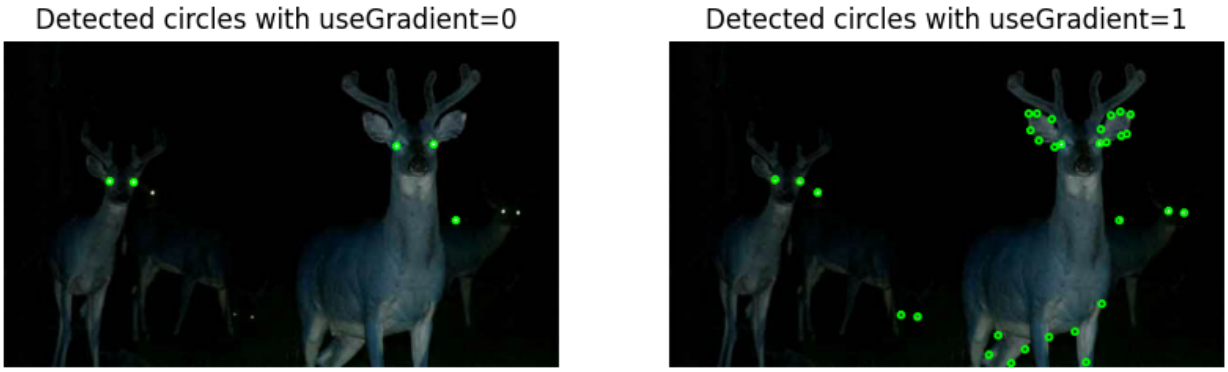


Figure 9: Circle detection on deer eyes image with radius 3 units

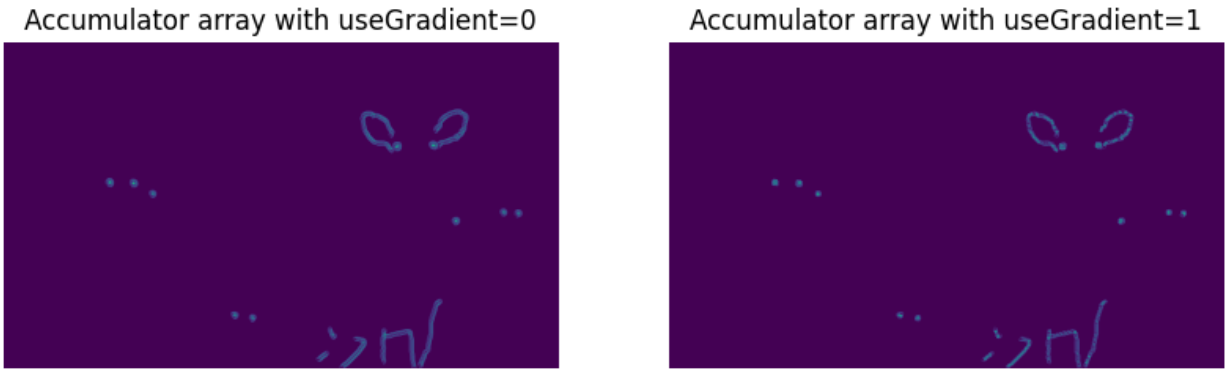


Figure 10: Accumulator array on deer eyes image

ii. **Balls Image:**

A. **Radius 25:** Figure 11 shows the results of circle detection on the balls image with a radius of 25 units. The detected circles are highlighted in the image.
Accumulator Array: Figure 12 shows the accumulator array generated during the circle detection process on the balls image with a radius of 25 units. The peaks in the accumulator array correspond to the detected circles in the image.
Note: For the given results, bin threshold is set to 0.4, post-processing is disabled and num thetas is set to 100. The bin threshold had to be reduced to detect the balls which are not perfect circles.

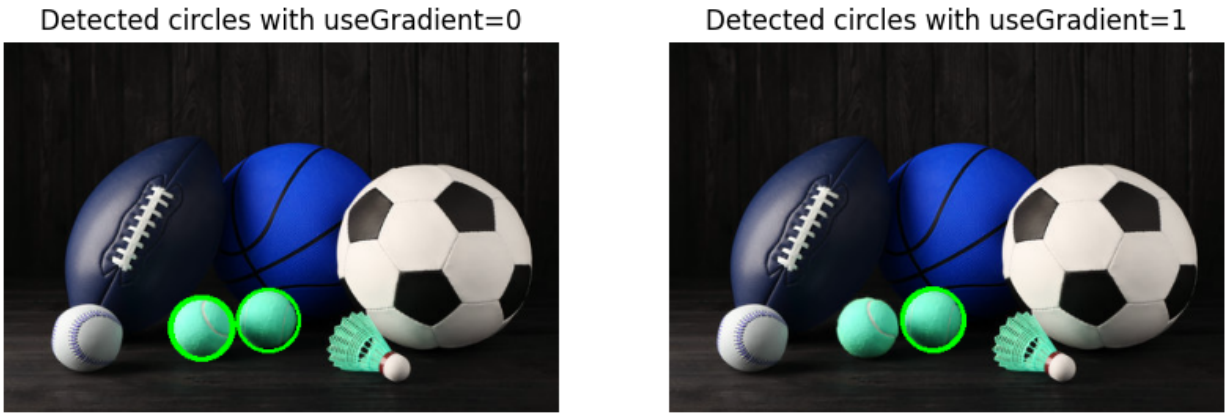


Figure 11: Circle detection on balls image with radius 25

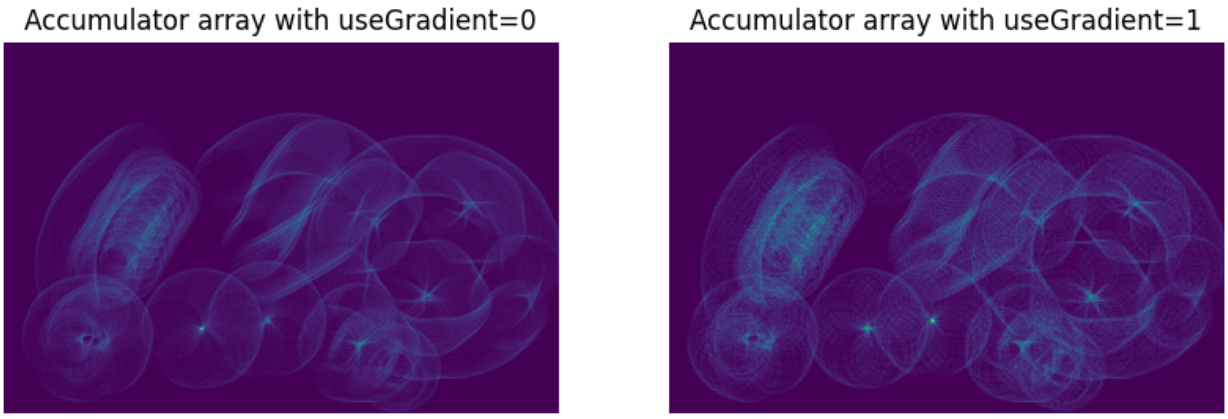


Figure 12: Accumulator array on balls image with radius 25

B. **Radius 78:** Figure 13 shows the results of circle detection on the balls image with a radius of 78 units. The detected circles are highlighted in the image.
Accumulator Array: Figure 14 shows the accumulator array generated during the circle detection process on the balls image with a radius of 78 units. The peaks in the accumulator array correspond to the detected circles in the image.
Note: For the given results, bin threshold is set to 0.25, post-processing is disabled and num thetas is set to 100. The bin threshold is reduced even more as the bigger balls farther from being perfect circles.

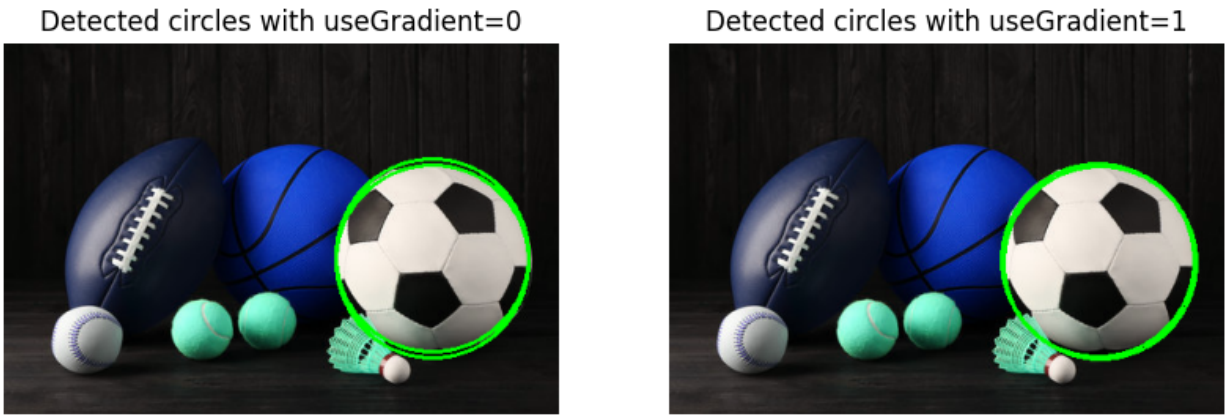


Figure 13: Circle detection on balls image with radius 78

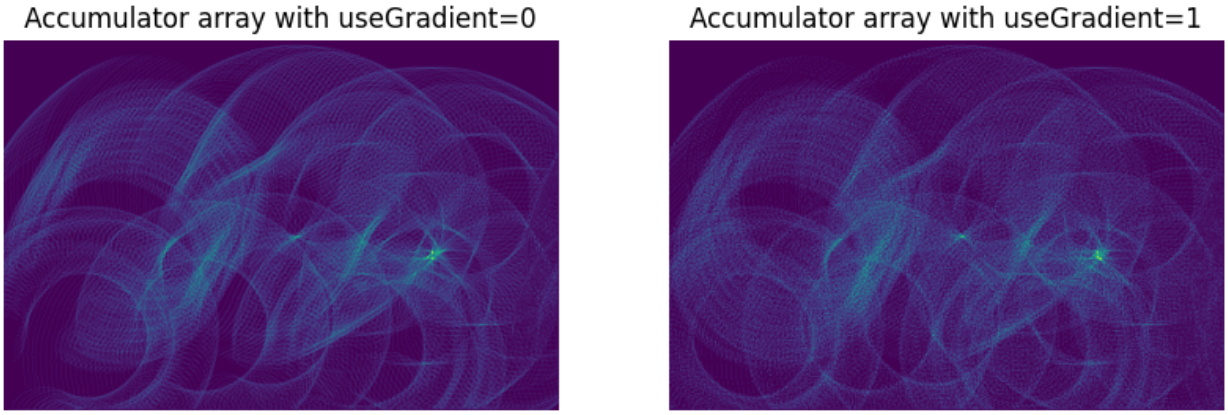


Figure 14: Accumulator array on balls image with radius 78

(c) **Accumulator array:** The accumulator is used to accumulate votes for potential circle centers in the Hough space. Each key-value pair in the accumulator dictionary represents a potential circle center (x_center, y_center) as the key and the number of votes it receives as the value. During the detection process, when an edge pixel is encountered, votes are cast for possible circle centers that could pass through that pixel. The more votes a particular center accumulates, the more likely it is to be the center of a detected circle. This accumulator array helps identify prominent peaks in the Hough space, indicating the presence of circles in the input image.

Figures 10, 12, and 14 show the accumulator arrays generated during the circle detection process on the deer eyes and balls images. The peaks in the accumulator arrays correspond to the detected circles in the images, providing insights into the circle detection process and the distribution of votes for potential circle centers.

(d) **Effect of Post Processing:** Post-processing is an optional step that can be applied after circle detection to remove duplicate circles or circles that are too close to each other. This step helps refine the detected circles and improve the accuracy of the results. Post Processing is applied on the circles detected in the accumulator array to remove duplicate circles or circles that are too close to each other.

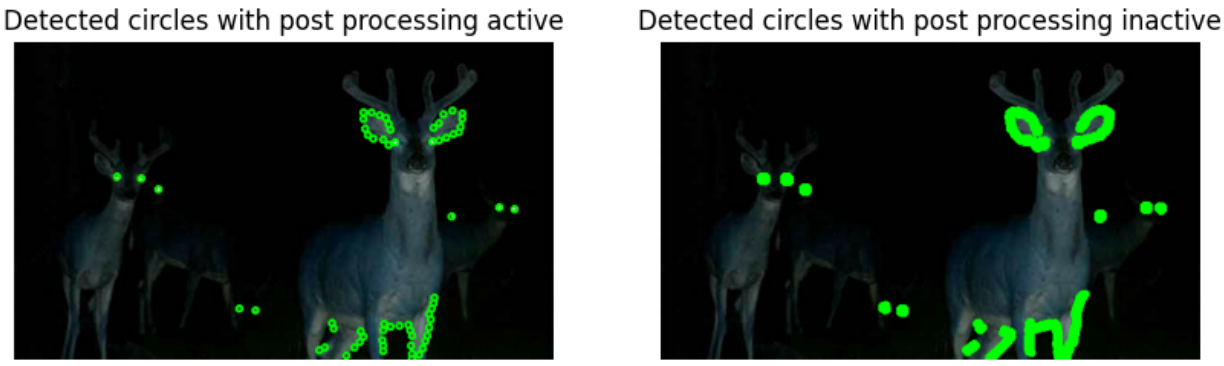


Figure 15: Comparing circles detected with and without post-processing on deer eyes image with radius 3

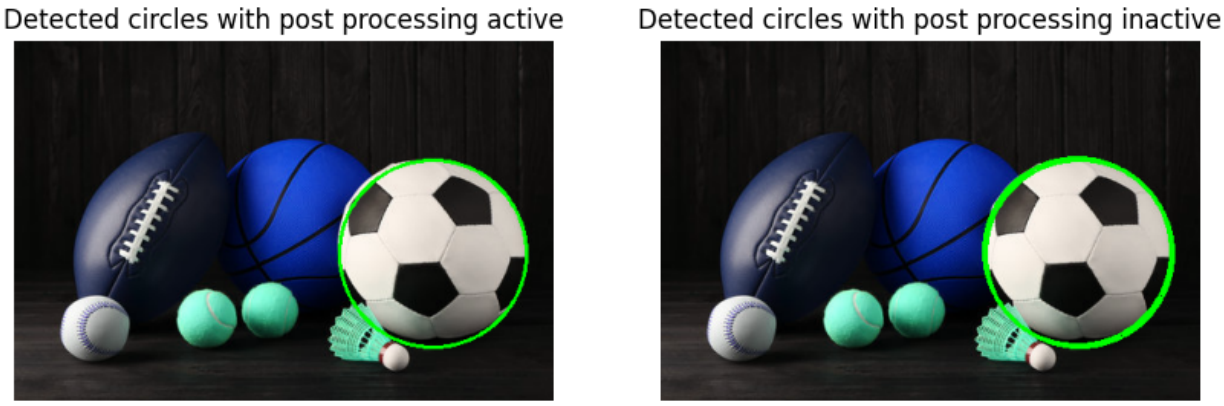


Figure 16: Comparing circles detected with and without post-processing on balls image with radius 78

Figures 15 and 16 show the circles detected with and without post-processing on the deer eyes and balls images, respectively.

(e) **Effect of Binsize:** When the bin size is increased, the granularity of the accumulator decreases, causing multiple circle centers to be grouped together, which can result in the detection of redundant circles with similar parameters.

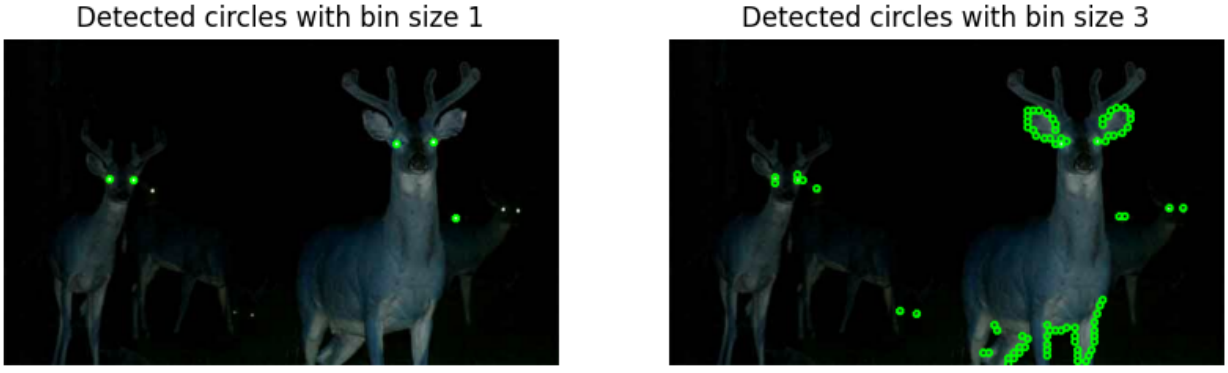


Figure 17: Effect of bin size on circle detection on deer eyes image with radius 3

Figure 17 shows the effect of bin size on circle detection on the deer eyes image with a radius of

3 units. As the bin size increases, the circles detected become less defined, and multiple circle centers are grouped together, leading to less accurate results.