Assignment 2 Writeup

Ian Chen

October 9, 2023

1 Short answer problems

1. *Compare the effects of 1) Dilation + Erosion against 2) Erosion + Dilation. Do they have the same effects? Why?*
   They have different effects. Erosion then dilation, also called opening, is used to remove small objects while preserving original shape. This effect removes small regions-like objects like lines or noise, while preserving the larger areas. Dilation then erosion, also called closing, fills holes in regions while preserving region sizes. This effect fills in gaps between objects or within the object itself, while preserving large holes and regions.

2. *List two examples of regular texture and two examples of near-regular texture.*
   Regular- Brick Wall and Checkerboard. Both are exact patterns and identical shapes with no randomness.
   Irregular- Stone Floor Tiles and Wood Planks. Both are identical from far away, but when closely examined, have small details unique to each piece.

3. *What are the cases where optical flow is not well-defined? Please give two concrete examples.*
   One case is the aperture problem, where the object's motion is not aligned with the direction calculated by the optical flow gradient, such as a spinning barber pole, with the pixels seem to move up constantly, while in reality it's moving either clockwise or counter-clockwise. The occlusion of the object's motion causes a false interpretation of its motion.

Another case is the inconsistent brightness, where the object's brightness changes over time, such as a light source being moved around. The optical flow gradient will be inconsistent and not well-defined.
Another case is texture-less regions, where the object has no texture, such as a white wall. The optical flow gradient will not sense motion, since all pixels have the same brightness.
Another case is motion too fast for the frame rate, where the object moves too fast for the camera to capture smooth differences between each frame. This causes the optical flow gradient to be unable to track motion throughout the frames.

4. *What are the advantages of RANSAC when compared with Hough Transform?*
   RANSAC is more robust to outliers than Hough Transform. RANSAC is able to detect and identity more types of shapes than Hough, so it's more applicable to a wider variety of images. RANSAC is also more efficient than Hough Transform, since only a subset of the total points are needed to be sampled to have a confident result of a detected object.

2 Circle Detection

Implement two circle detectors (one based on Hough Transformation and another based on RANSAC) that takes an input image and a fixed (known) radius, and returns the centers of any detected circles of about that size. Include two functions with the following form:

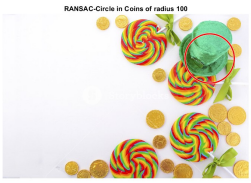
[centers] = detectCirclesHT(im, radius)

[centers] = detectCirclesRANSAC(im, radius)

where 'im' is the input image, 'radius' specifies the size of circle we are looking for. Your detector should not exploit the gradient direction. The output centers is an N x 2 matrix in which each row lists the x,y position of a detected circle's center. Write whatever helper functions are useful. Then experiment with the basic framework, and in your writeup analyze the following:

- *Explain your implementation in concise steps (English, not code).*
  I first found the edges of the image using the Canny edge detector. Then, I created an accumulator array filled with all zeroes. Then, I looped through the edges and incremented the accumular values of the matrix if the center was an edge pixel. Then, I found the maximum value of the accumulator array and determined that it was the center of the circle.

- *Demonstrate the functions applied to the provided images 'coins.jpg' and 'planets.jpg' and one image of your choosing. Display the images with detected circle(s), labeling the figure with the radius. Note: you only need to select one reasonable radius and display all detected circles (i.e., those with highest votes) under that radius. You are not required to consider circles with a center off the image.*

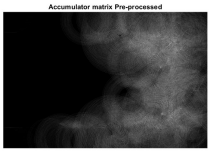


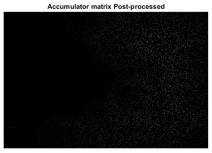
- *For Hough Transform, explain how your implementation post-processes the accumulator array to determine automatically how many circles are present.*
  I first got the local maximums of the accumulator array and zeroed out all values not at a local max. This removes circle overlap of thick edges. Next, I used the maximum of a vote threshold of 80% of the max and a constant threshold of 120, which means one-third of the circle's edge touches this point. This is to account for noise and a circle not being perfectly round. This removes detected circles overlapping due to only the local maximums being considered. Therefore,

all local maxima with at least 120 edge points going through them are considered circles.

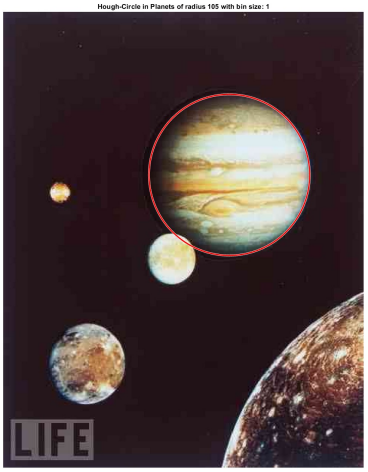
- *For RANSAC, explain how you implement circle fitting.*
  I first randomly selected three points from the edge points. Then, I calculated the center and radius of the circle based on those three points. If the radius was within a threshold of the expected radius, then it was a potential fit. Then, I used a threshold to determine how many edge points were within that threshold from the calculated radius, if enough points were within, based on the statistical equation: $\mathbb{N} = \frac{\log(1-p)}{\log(1-w^n)}$. There had to be enough inliers compared to the total number of edge points.

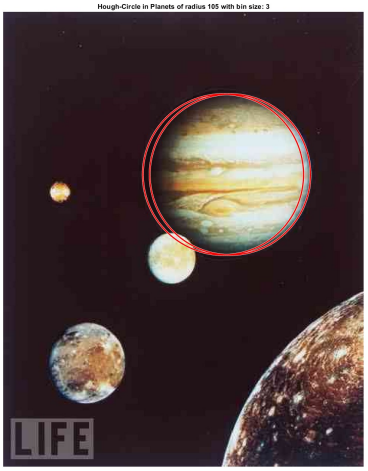
- *For one of the images, display and briefly comment on the Hough space accumulator array.*
  The accumulator array is a 2D array built by looping through the edge points and incrementing the matrix at the circle's circumference assuming the center was that edge. The resulting accumulator array is a 2D matrix with maximum values at circle centers, since all those edges all 1 to the center. Then, I reduce the matrix to include only local maximums, to reduce noise and overlapping circles. The resulting matrix values are either circle centers or random points, which are ignored when post-processing the matrix for circles.





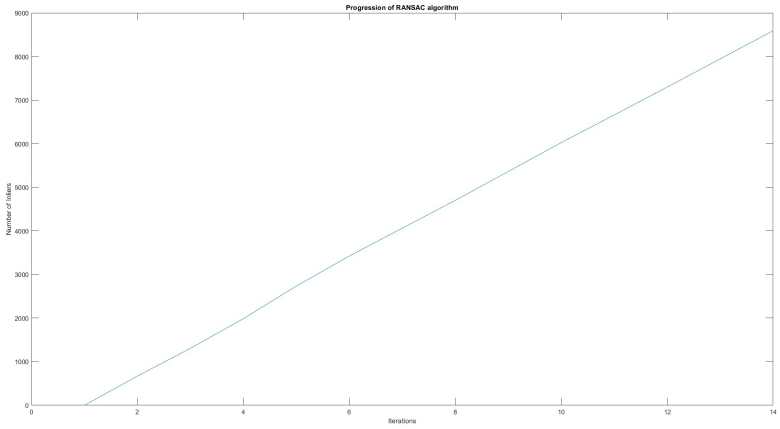
- *For one of the images, demonstrate and explain the impact of the vote space quantization (bin size). In other words, alter the bin size and compare and contrast with a brief explanation why/what happened makes sense.*
  Smaller bin sizes result in a more finely quantized accumulator array. This can lead to higher sensitivity which may produce more noise and false positives. Circles with radii close to the expected radii will be detected, but it may miss some circles with slightly different radii due to the quantization effect. Larger bin sizes result in a coarser quantization of the accumulator array. This can reduce sensitivity but may miss partially occluded circles. It may be more robust against noise but may miss small or slightly distorted circles.
  The choice of the bin size is a fine balance between sensitivity and noise tolerance, depending on the specific application and the expected properties of the circles in the image. Smaller bin sizes are suitable for high-precision applications, while larger bin sizes may be preferred in noisy or less structured images.





The results are interesting, since noise is suppressed by a larger bin size, but in the planets picture, the s̈hadow of Jupiterïs determined a circle, showing the the half circle generated by the shadow was detected .

- *For one of the images, plot the progress of the RANSAC as the number of tries increase. The x axis of the plot should be the number of tries, and the y axis should be the number of inliers that the best model produces.*



3 Image segmentation with k-means

1. *Given an h x w x 3 matrix 'Im', where h and w are the height and width of the image, apply k-means clustering to associate pixels with clusters. Return 'labelIm', an h×w matrix of integers indicating the cluster membership (e.g., from 1 to k) for each pixel. Please use the following form:*

   function [labelIm] = clusterPixels(Im, k)

2. *Detect cluster boundary pixels from 'labelIm'.*

   function [boundaryIm] = boundaryPixels(labelIm)

3. *Please test both functions on the provided images 'gumballs.jpg', 'snake.jpg', and 'twins.jpg' and one other image of your choosing, and then displays the results.*
   I used the RGB values of pixels to determine which cluster it belonged to.

Original image

Edges of original image

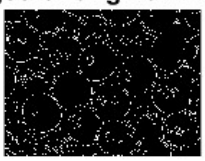
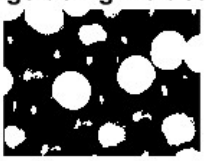
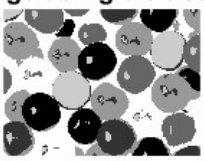


Image using 2 clusters



Edges of the 2 clusters

Image using 6 clusters



Edges of the 6 clusters

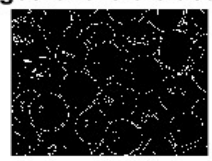
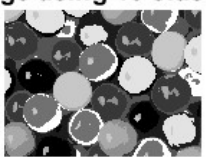
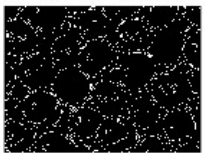
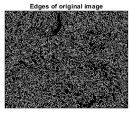


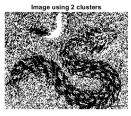
Image using 10 clusters

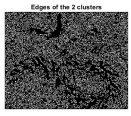


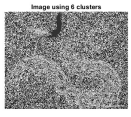
Edges of the 10 clusters

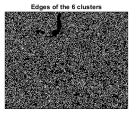


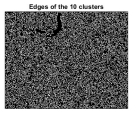


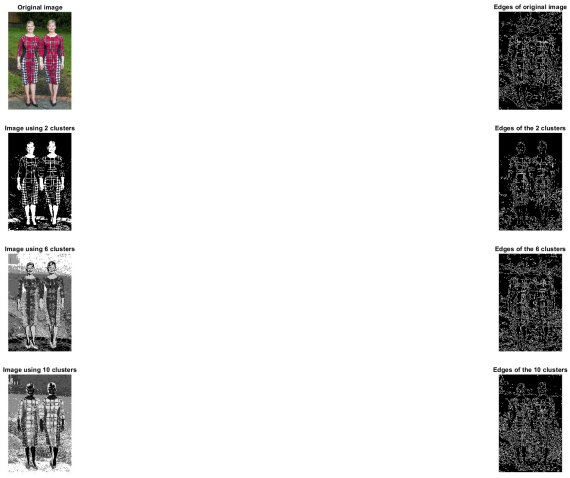











Original image
Edges of original image
Image using 2 clusters
Edges of the 2 clusters
Image using 6 clusters
Edges of the 6 clusters
Image using 10 clusters
Edges of the 10 clusters