


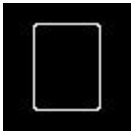


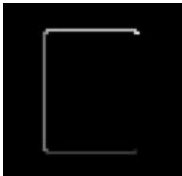
Generalized Hough Transform (GHT)







GHT is an object detection algorithm that takes a template image and finds it in a bigger sample image. For this assignment, the template and sample images are both put through an edge detection filter before the application of the algorithm. From then on, GHT takes these steps:





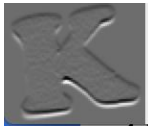

- 1. Interpret the template image as an R-table
- 2. Accumulate points on the sample image where each edge point ‘votes’ for where an object may exist
- 3. Make best guesses on where the object may be, from the accumulator matrix

I used OpenCV to help with low-level graphic functions and displaying the output. To debug, block.tif was used to develop the functions that lead to building the R-table. During development, inspiration was taken from OpenCV library implementation as well as an online tutorial found on the subject.

Parameter Space

Original	Edge	dx	dy	phi
				

Original	Gray	Edge	dx	dy	phi
					

Original	Gray	Edge	dx	dy	phi
					

The images were generated by normalizing the values. The block’s phi values clearly show that the right edge is black. It indicates 0 degrees, as expected.

R-table



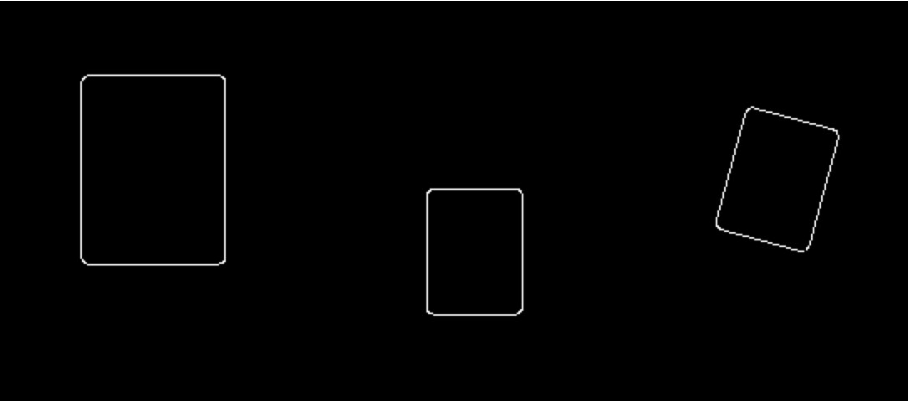
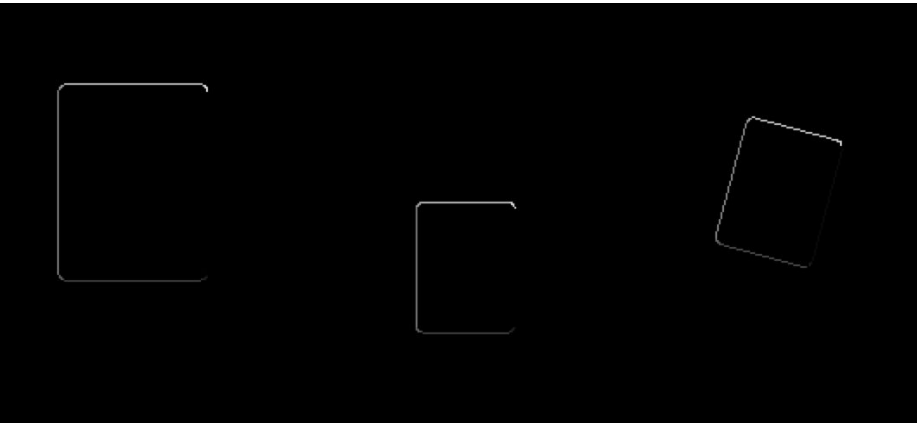
And for each edge point, a pair of relative coordinates were put into phi-bins of the R-table. The following shows the R-table for block.tif for 16 intervals. Angles without entries are omitted.

Φ	Entries
0°	(16 -21), (16 -20), (16 -19), (16 -18), (16 -17), (16 -16), (16 -15), (16 -14), (16 -13), (16 -12), (16 -11), (16 -10), (16 -9), (16 -8), (16 -7), (16 -6), (16 -5), (16 -4), (16 -3), (16 -2), (16 -1), (16 0), (16 1), (16 2), (16 3), (16 4), (16 5), (16 6), (16 7), (16 8), (16 9), (16 10), (16 11), (16 12), (16 13), (16 14), (16 15), (16 16), (16 17), (16 18), (16 19), (16 20)
45°	(15 20)
90°	(-16 21), (-15 21), (-14 21), (-13 21), (-12 21), (-11 21), (-10 21), (-9 21), (-8 21), (-7 21), (-6 21), (-5 21), (-4 21), (-3 21), (-2 21), (-1 21), (0 21), (1 21), (2 21), (3 21), (4 21), (5 21), (6 21), (7 21), (8 21), (9 21), (10 21), (11 21), (12 21), (13 21), (14 21), (15 21),
135°	(-16 20)
180°	(-17 -21), (-17 -20), (-17 -19), (-17 -18), (-17 -17), (-17 -16), (-17 -15), (-17 -14), (-17 -13), (-17 -12), (-17 -11), (-17 -10), (-17 -9), (-17 -8), (-17 -7), (-17 -6), (-17 -5), (-17 -4), (-17 -3), (-17 -2), (-17 -1), (-17 0), (-17 1), (-17 2), (-17 3), (-17 4), (-17 5), (-17 6), (-17 7), (-17 8), (-17 9), (-17 10), (-17 11), (-17 12), (-17 13), (-17 14), (-17 15), (-17 16), (-17 17), (-17 18), (-17 19), (-17 20)
225°	(-16 -21)
247.5°	(-16 -22), (-15 -22), (-14 -22), (-13 -22), (-12 -22), (-11 -22), (-10 -22), (-9 -22), (-8 -22), (-7 -22), (-6 -22), (-5 -22), (-4 -22), (-3 -22), (-2 -22), (-1 -22), (0 -22), (1 -22), (2 -22), (3 -22), (4 -22), (5 -22), (6 -22), (7 -22), (8 -22), (9 -22), (10 -22), (11 -22), (12 -22), (13 -22), (14 -22), (15 -22)

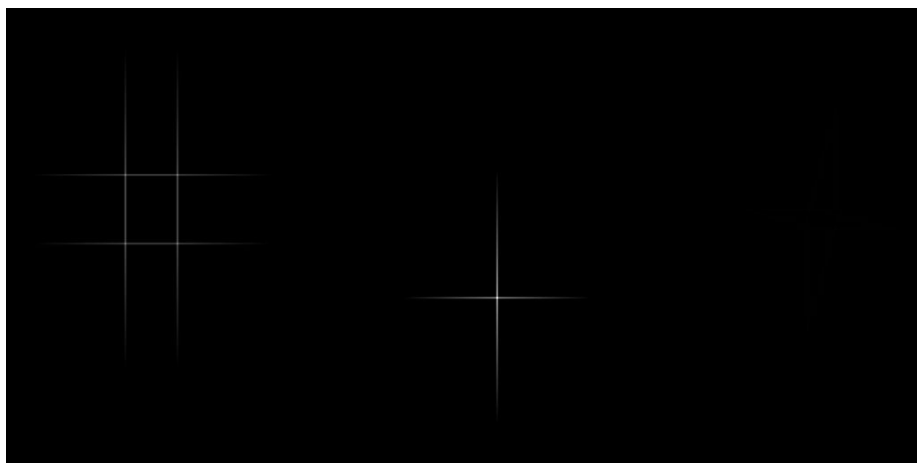
Once the R-table was built for each template image, it was then ready to be compared to the sample image. Based on my observation, I chose a scale of 1.15. Rotation was limited to 20 degrees.

Accumulation

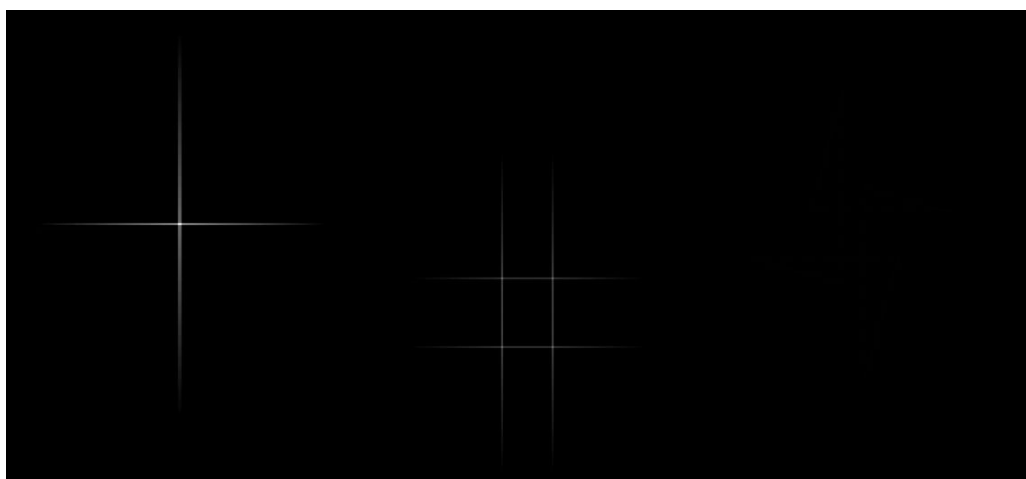
For each edge point on the sample image, its dx/dy was calculated just as the sample image, and its ϕ calculated. If there are any entries on the R-table that matches the ϕ of the sample edge, then votes are casted according to the entries on the R-table.

Template	
Original	
Edge	
Phi	

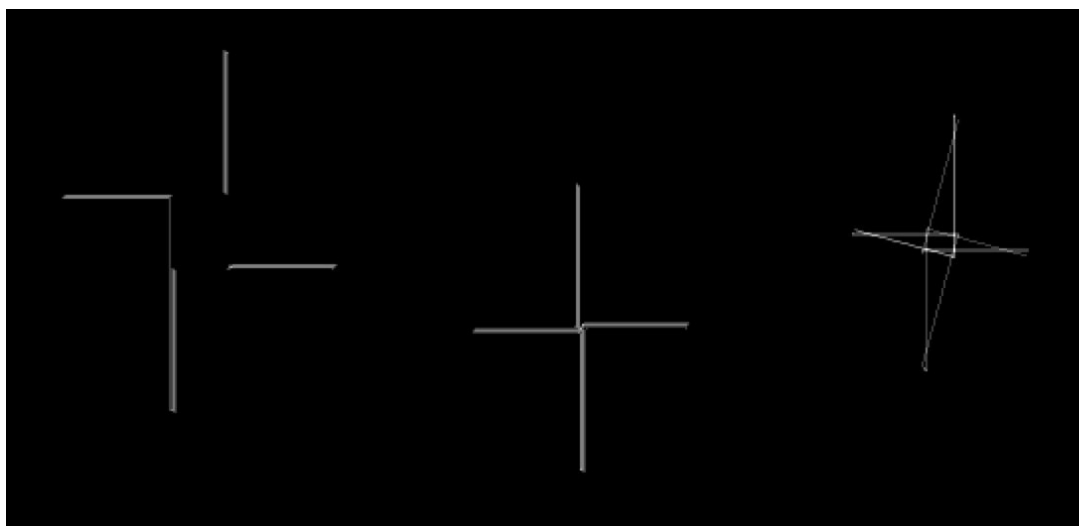
Votes @
Scale 1



Votes @
Scale 1.5

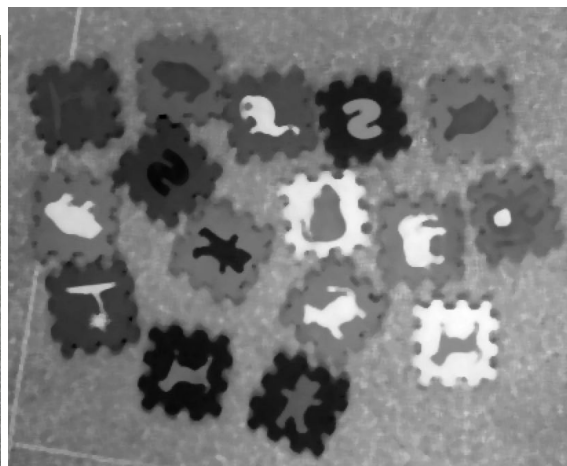


Votes @
scale 1
45° offset



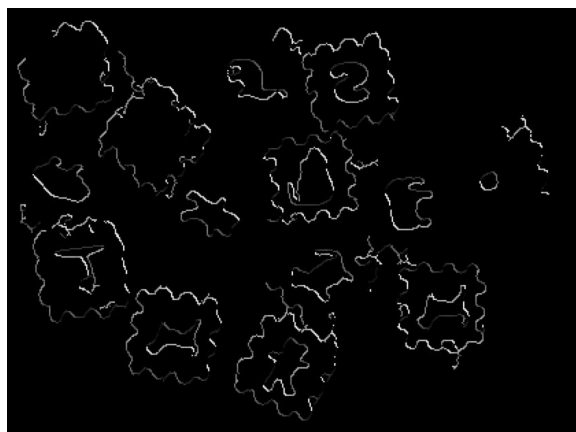
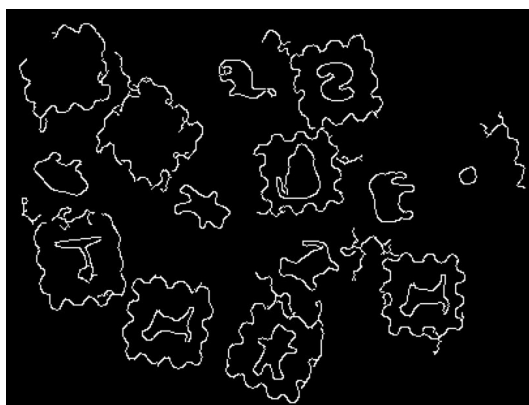
Original

Gray +
median
blur k=5



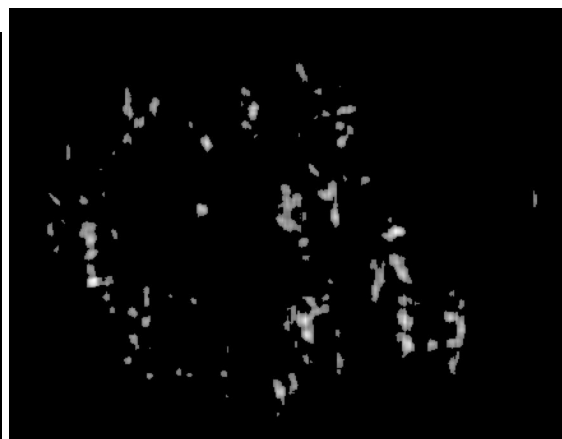
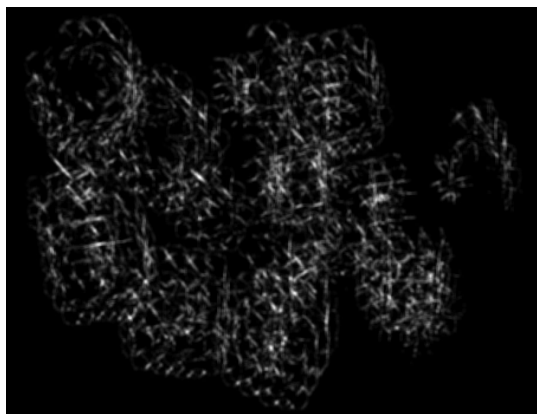
Edge


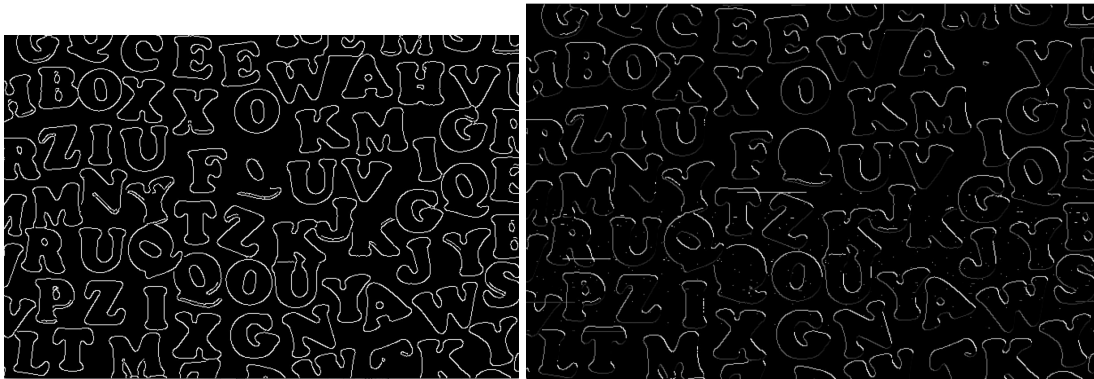
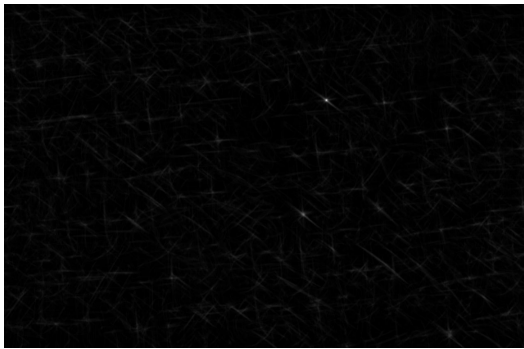
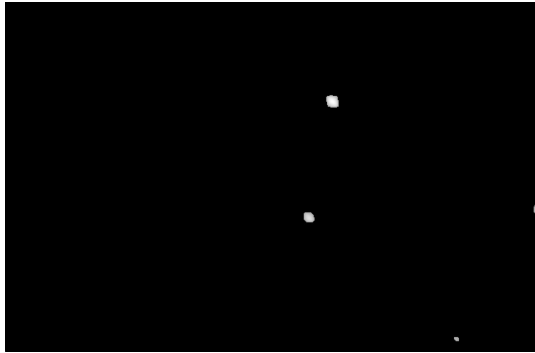
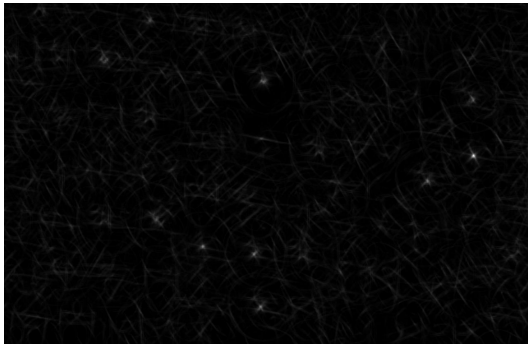
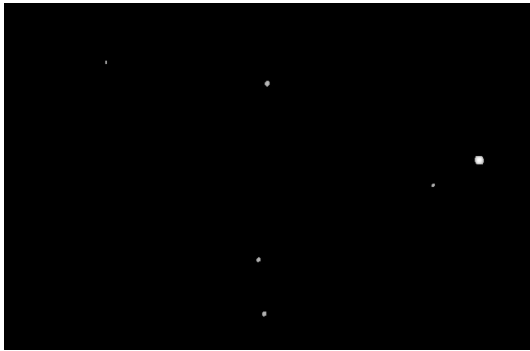
phi



Votes

Votes +
Blur +
High
Pass



Original Gray		
Edge Phi		
K Votes K Votes + Blur + High Pass		
Q Votes Q Votes + Blur + High Pass		

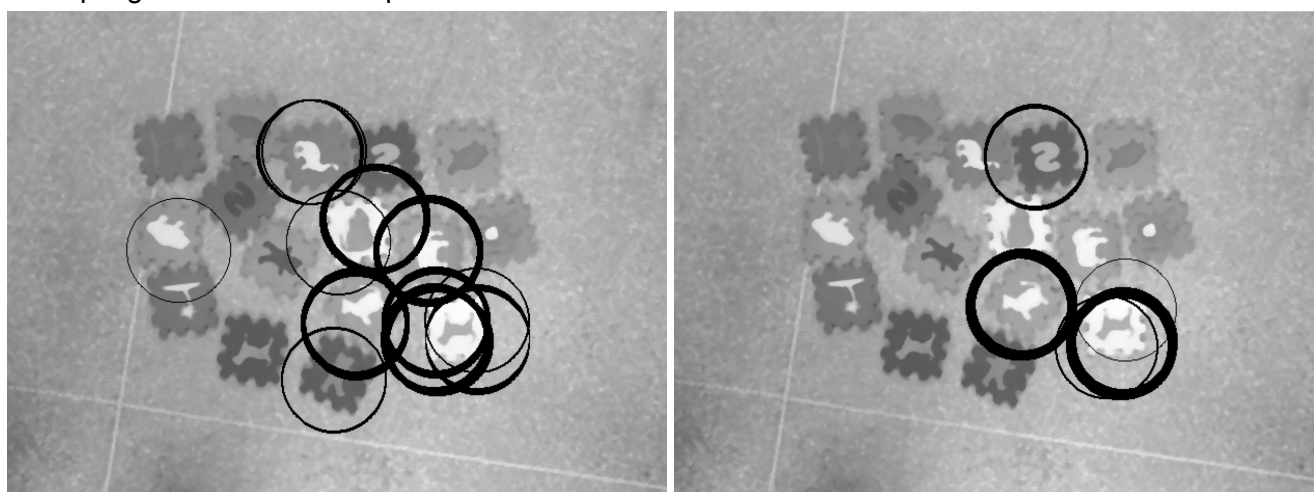
Peak Identification

One challenge for peak identification is to determine the peak among multiple vote maps across different angles and scales. A convenient solution is to add all the vote maps together and find the peak. So after adding all the votes, it was passed through a high pass filter to indicate the regions. After getting the regions, it overlaid its position on top of the original image.

Results



Attempting to find a bear / elephant



Performance was great for K, but it was poor for Q. Because of the similar curvature to O, C, and G, it interpreted all of them as being similar to Q. Even when I manually increased the threshold, it would still interpret Gs as strong candidates to Q.

For bears and elephants, the simple addition/high pass procedure for voting across all angles and scales was much too simple. It got lost between the curvy edges of each tile. Without careful parameters, it had even poorer performance.

Advanced methods for peak identification may include super-votes for each scale/angle combination that accumulate on a super-matrix. But due to time restrictions, I was only able to do simple addition.