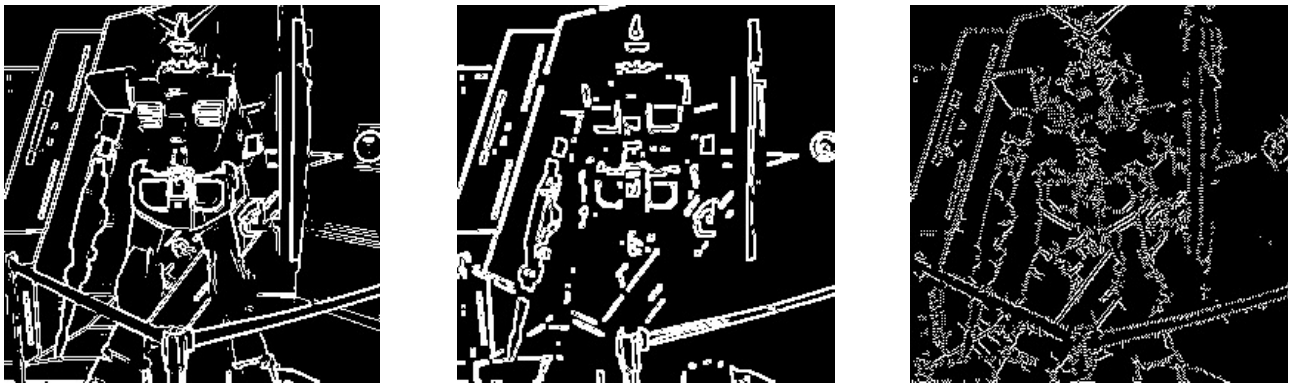
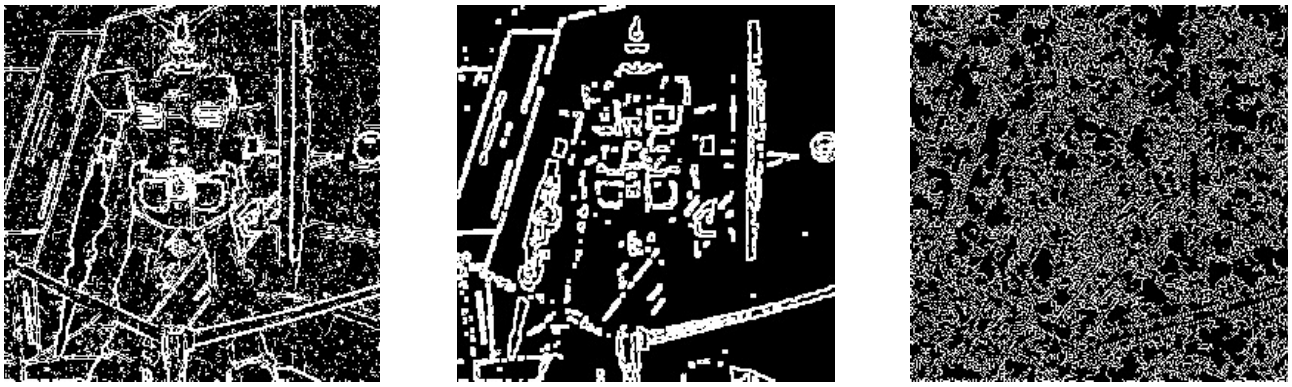
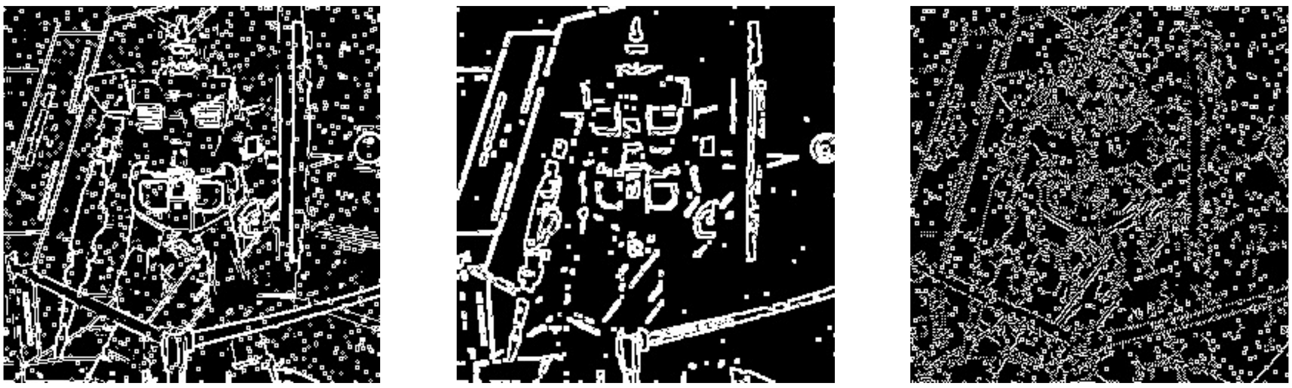
Name: Steven Lin. 林于智
Student ID: R04922170

Problem 1

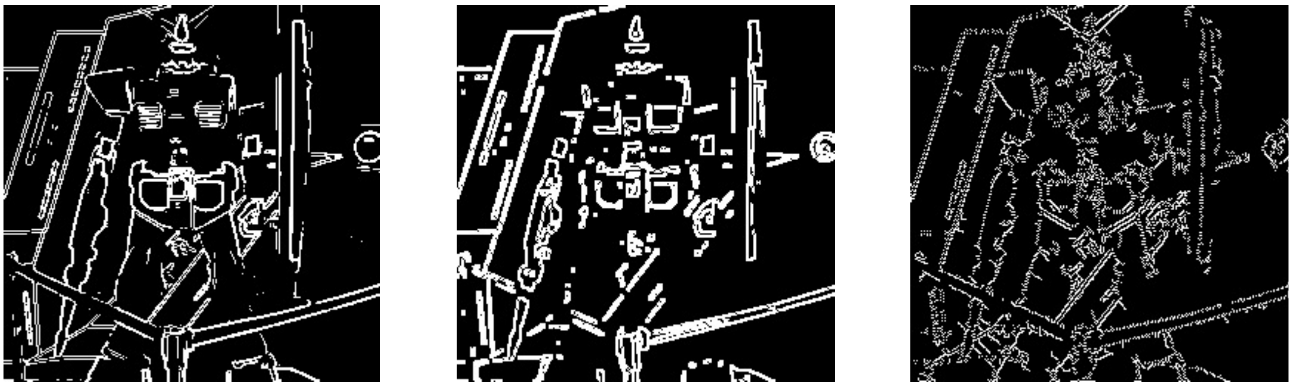
The 9 images below are 3 original images with 3 different edge operators, Sobel(150), 2nd Order(400), and Canny(250, 50).


sample1
(left: Sobel, mid: 2nd Order, right: Canny)

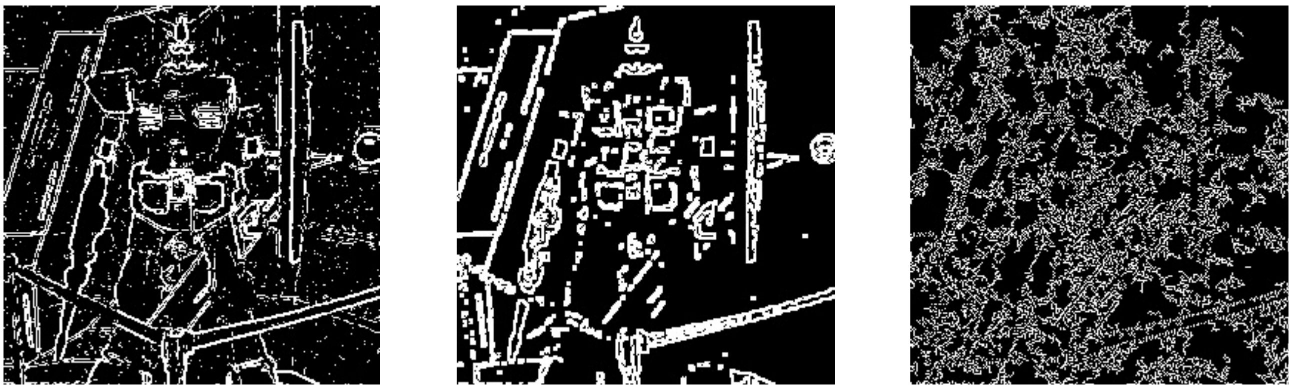

sample2
(left: Sobel, mid: 2nd Order, right: Canny)


sample3
(left: Sobel, mid: 2nd Order, right: Canny)

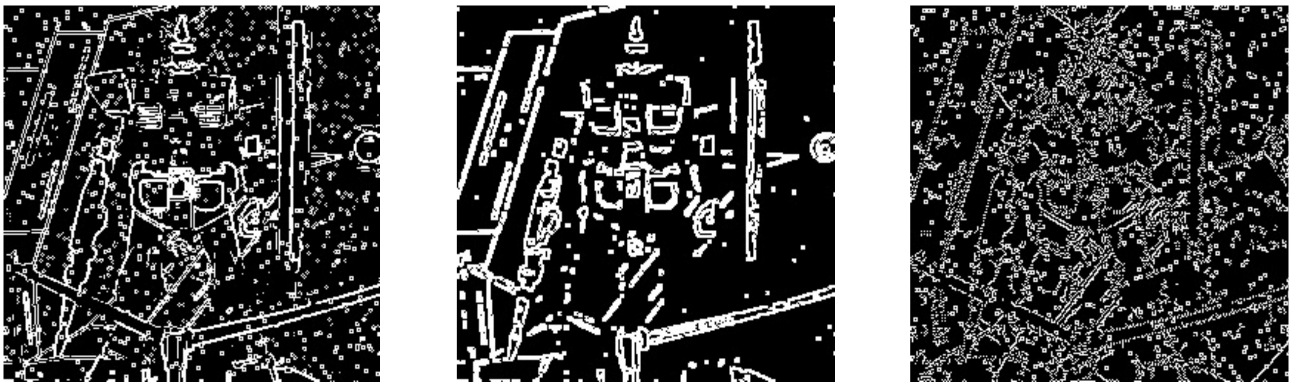
These following images are of different threshold values with the 3 operator mentioned above. (Sobel(200), 2nd Order(500), and Canny(300, 30)



sample1
(left: Sobel, mid: 2nd Order, right: Canny)



sample2
(left: Sobel, mid: 2nd Order, right: Canny)



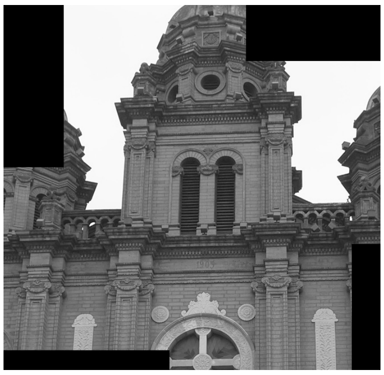
sample3
(left: Sobel, mid: 2nd Order, right: Canny)

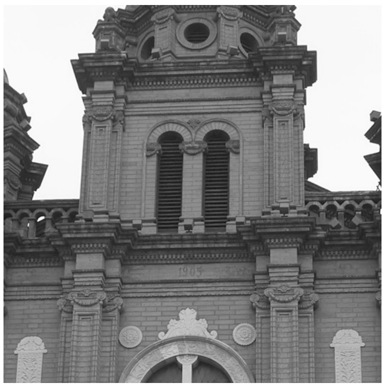
Observation

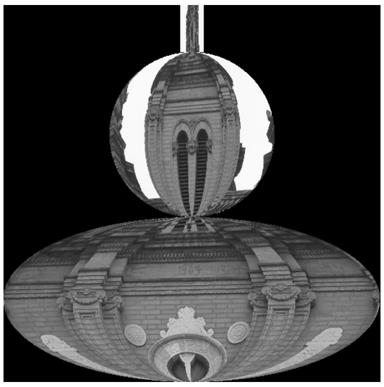
We can infer from the pictures above that first order derivative methods don't really handle noise well, no matter it's Gaussian noise or Salt-and-Pepper noise. On the other hand, second order methods did well on Gaussian noised images.

Problem 2

The resulted image are shown as follows.


R: stitched result


S: the cropped result.


G: warped image.

Problem 3

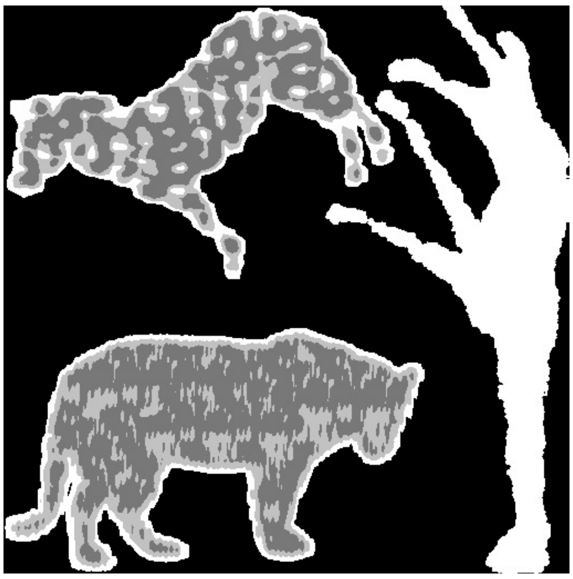
My steps:

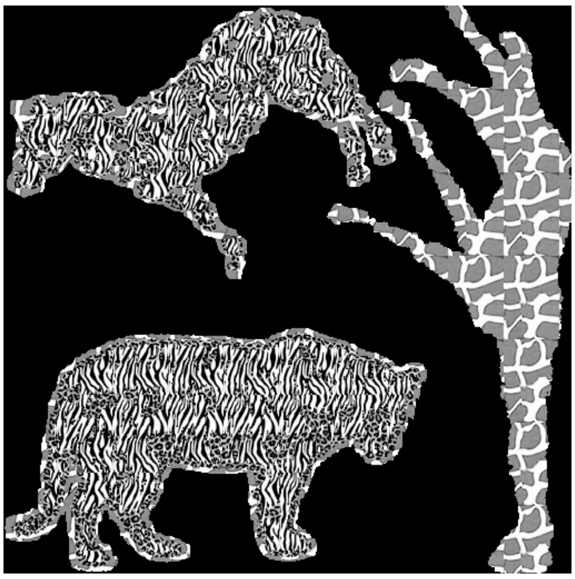
For (a), the feature vectors are calculated after the convolution with 9 masks and the standard deviation measurement. In my implementation, M[i][height][width] stores the results of convolution where i = 0 ~ 8, indicating which mask is involved. And we use the 16.2-2 formula on textbook to compute the T[i][height][width], the window size for this step is set to 15.

For (b), the k-means method starts with four randomly generated centroids. During each iteration, all pixels are classified into one cluster (cluster0 ~ cluster3) each, and the new centroid is calculated as well. Basically we can do as many iteration as we want, but it'll gradually converge to a stable condition and therefore the while-loop can stop. **The resulted 4 clusters are drawn in 4 different intensity: (0*255/3), (1*255/3), (2*255/3), and (3*255/3).**

For (c), my program needs response from user since the program does not know the intensity value of the zebra region, etc. My program will ask three times in order for the positions of the three animals. The answers are among {black, dark-gray, light-gray, white} as described in the last paragraph.

The resulted images of my program are shown below. And it's actually not very good because the texture on zebra and the one on jaguar cannot be distinguished from each other!


L: after texture analysis.


C: after texture attaching.