Contour Detection

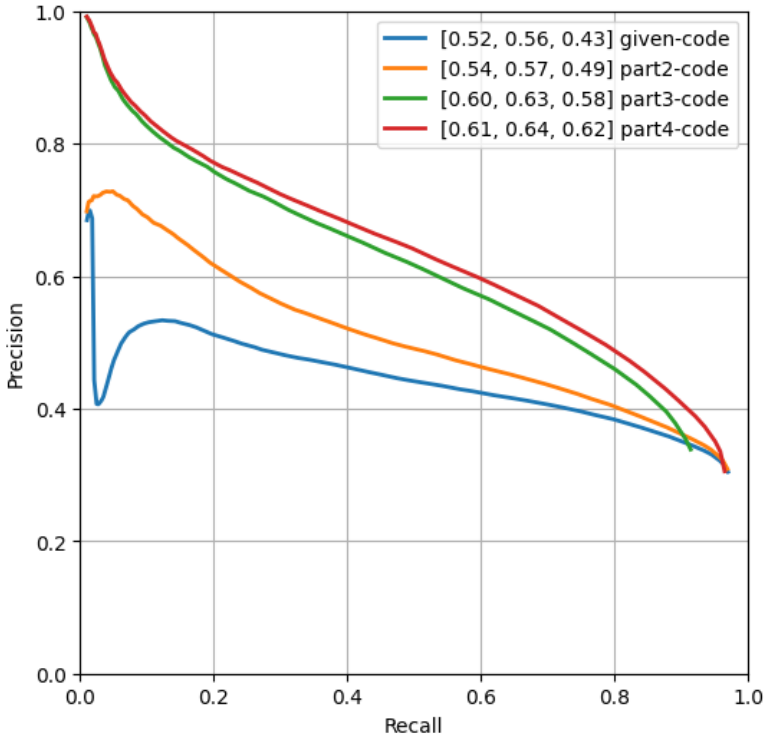
1. **Method Description.**

   For part 2 of this question, I tried different boundary condition for the convolve function. Visually speaking, handling the boundary condition with `symm` gives the best result.

   For part 3 of this question, I tried 1-D gaussian filters with different width and σ value. I also tried to first apply gaussian filter to the image, then apply derivative filter to the image. Base on the output matric, it seems that a wider gaussian filter with a σ about $\frac{1}{6}$ of the filter width gives better result. In particular, a 25-pixel wide gaussian filter with $\sigma = 4$ gives me the highset overall F1 score.

   For part 4 of this question, I tired to perform 2-pixel interpolation first, then perform non-maximum suppression. I did not try bilinear interpolation, which in theory should give better result.

2. **Precision Recall Plot.**



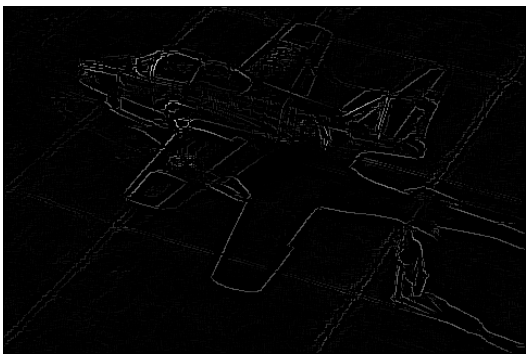
3. **Results Table.**

Method	overall max	average max	AP	Runtime

	F-score	F-score		(seconds)
Initial implementation	0.52	0.56	0.43	0.0117
Warm-up	0.54	0.57	0.49	0.0123
Smoothing	0.60	0.63	0.58	0.0394
Non-max suppression	0.61	0.64	0.62	1.4516
Test set numbers of best model [From gradescope]	0.61	0.64	0.62	1.53

4. **Visualizations.**



From three set of images above, the edge detector works relatively well when the color contrast is high. When the contrast is realtively low, the gradient is smaller and hence the maximum value is lower, and the edge become dimmer in the image.

Corner Detection

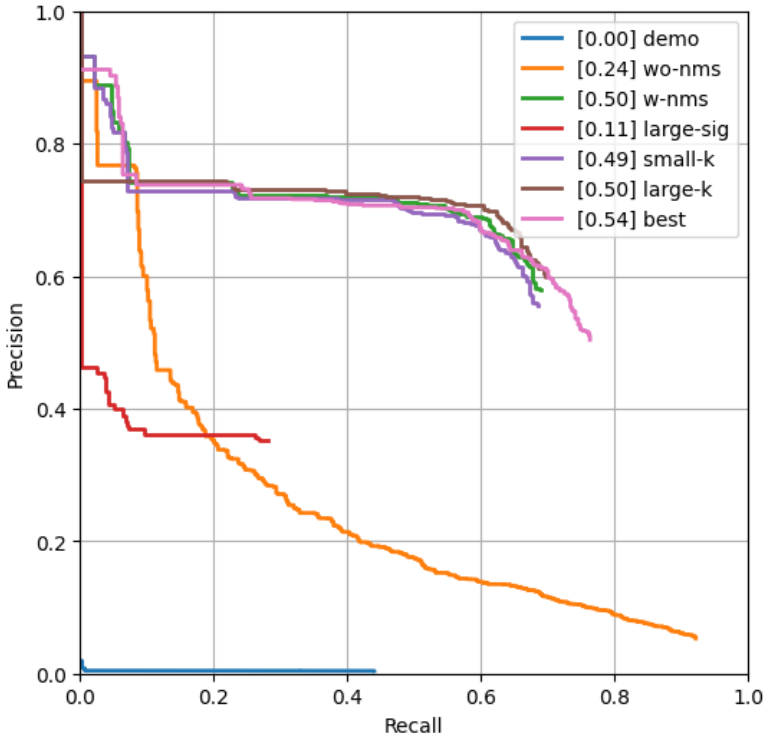
1. **Method Description.**

   For this problem, I first convert the image to grayscale image, then I compute the gradient of the image. From the gradient, I can calculate different term in M matrix. For example, for $\sum w(x,y)I_x^2$, I first compute the $I_x^2 = d_x^2$, where d_x is the gradient along x axis of the image. Then I convolve this image with a gaussian filter, so the result would be the desired $\sum w(x,y)I_x^2$.

   After getting all three terms, I can then calculate the determinent and trance of the matrix M. Then I can compute the response with $r = det(M) - k * (trace^2)$, where k is a constant.

   We know from the equation derivation that any $R > 0$ could represent a corner in the image. So I threshold the response image, then use a non-maximum suppression with a certain window size to determine the corner.

2. **Precision Recall Plot.**

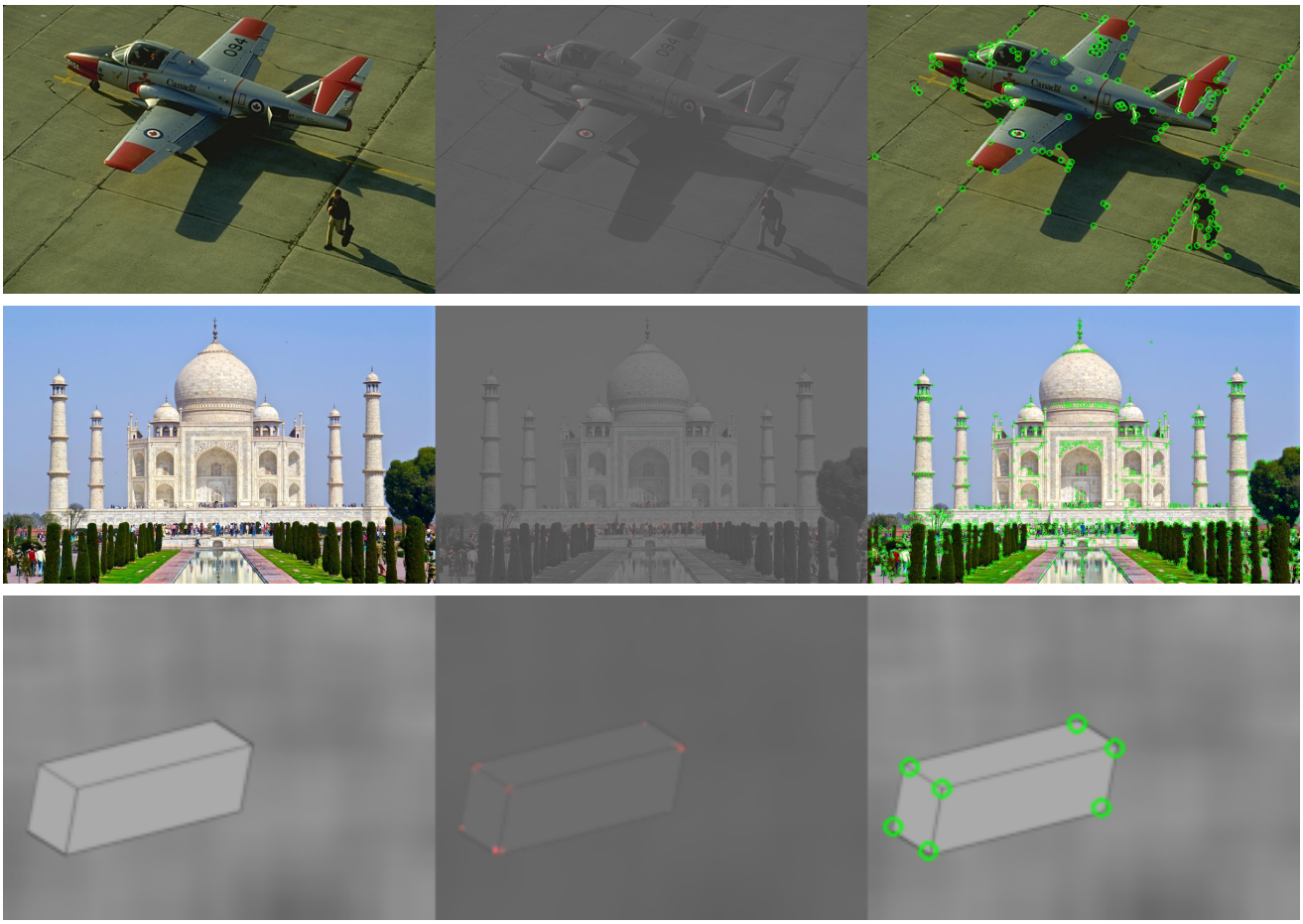


3. **Results Table.**

Method	Average Precision	Runtime

Random	0.001	0.001
Harris w/o NMS	0.242	0.002
Harris w/ NMS	0.503	0.104
Larger σ & NMS window	0.107	0.102
Smaller k value	0.494	0.103
Larger k value	0.505	0.102
Test set numbers of best model [From gradescope]	0.542	0.110

4. **Visualizations.**



This algorithm did relatively weill in these three cases. It successfully marked most major corners in the image. However, in the airplane image, it failed to mark the corner in the shadow of the wing.

5. **Bells and Whistles.**

I tried different method to calculate the gradient. I compared using `numpy.gradient()`, using simple 3-point derivative along two axis, and using a sobel edge filter. It turns out the best result comes with the simple 3-point derivative filter and a $k = 0.5$, $\sigma = 0.85$, and NMS window size of 3.

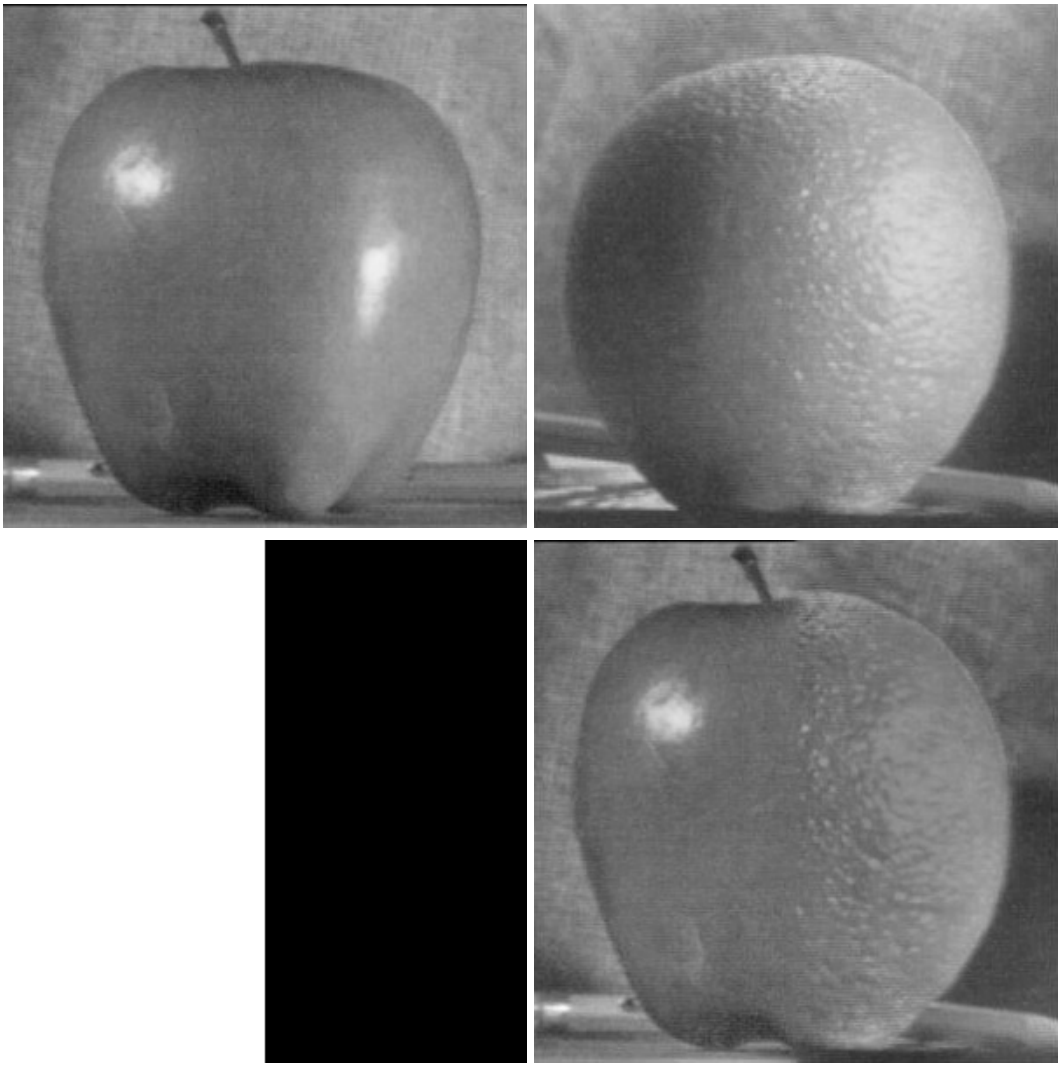
Method	Average Precision	Runtime
Best base Implementation (from above)	0.542	0.110
Using simple 3-point derivative	0.549	0.110
Using sobel filter	0.480	0.112

Multi-resolution Blending

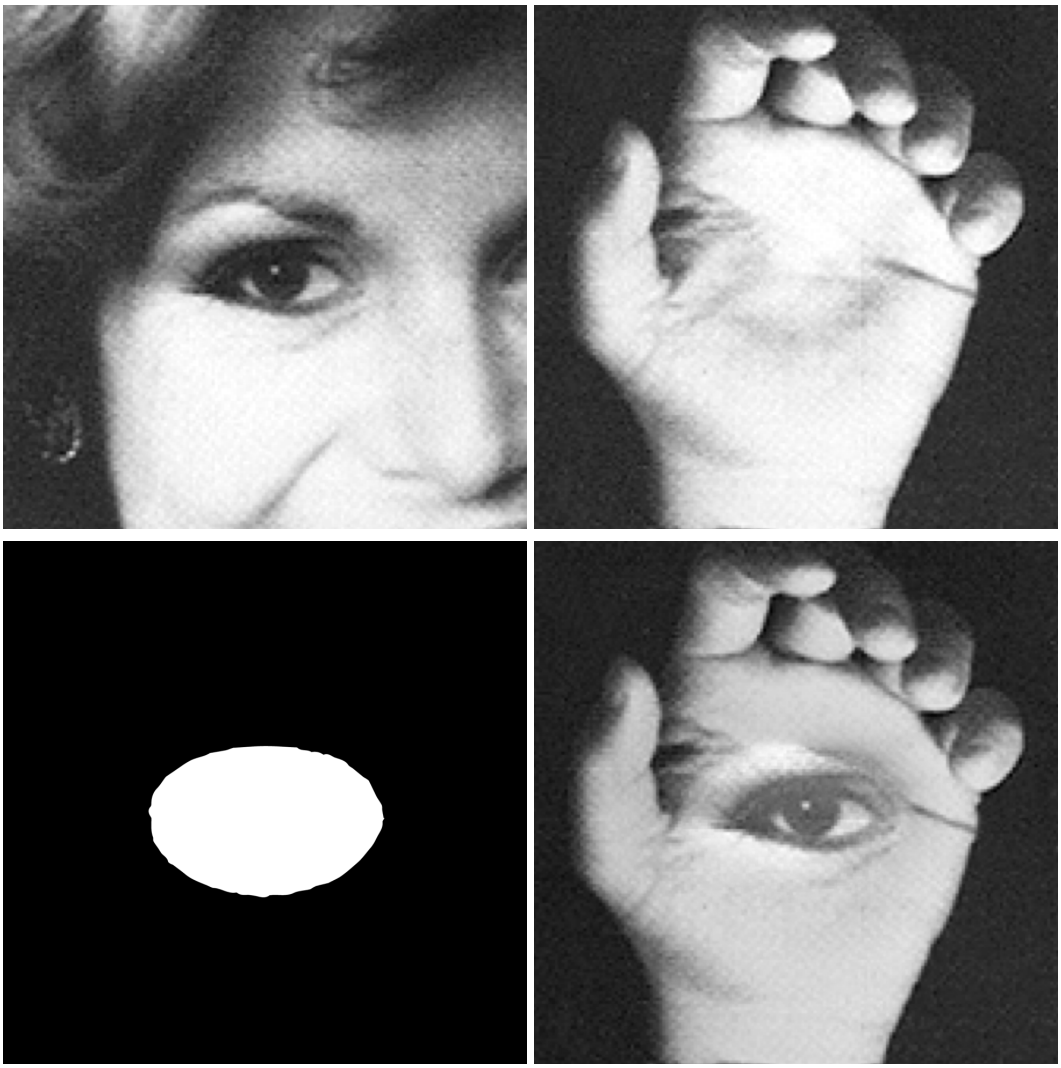
1. **Method Description.**

   For this problem, I first convert all three images to grayscale. Then I compute the laplacian stack of two input image, by computing image with different gaussian filter and find their difference. Then I computed a gaussian stack, by computing the mask with different gaussian filter. Then I combine these three stacks with the equation given in the paper. Lastly, I stack these images together by simply adding them together and normalize the result.

2. **Oraple.**



3. **Blends of your choice.**



I attempted the famous eye on hand image blend. On top of the original oraple blend, I changed the depth of the image stack from 10 to 20 to get a visually pleasing result.