Jonathan Pearce

260672004

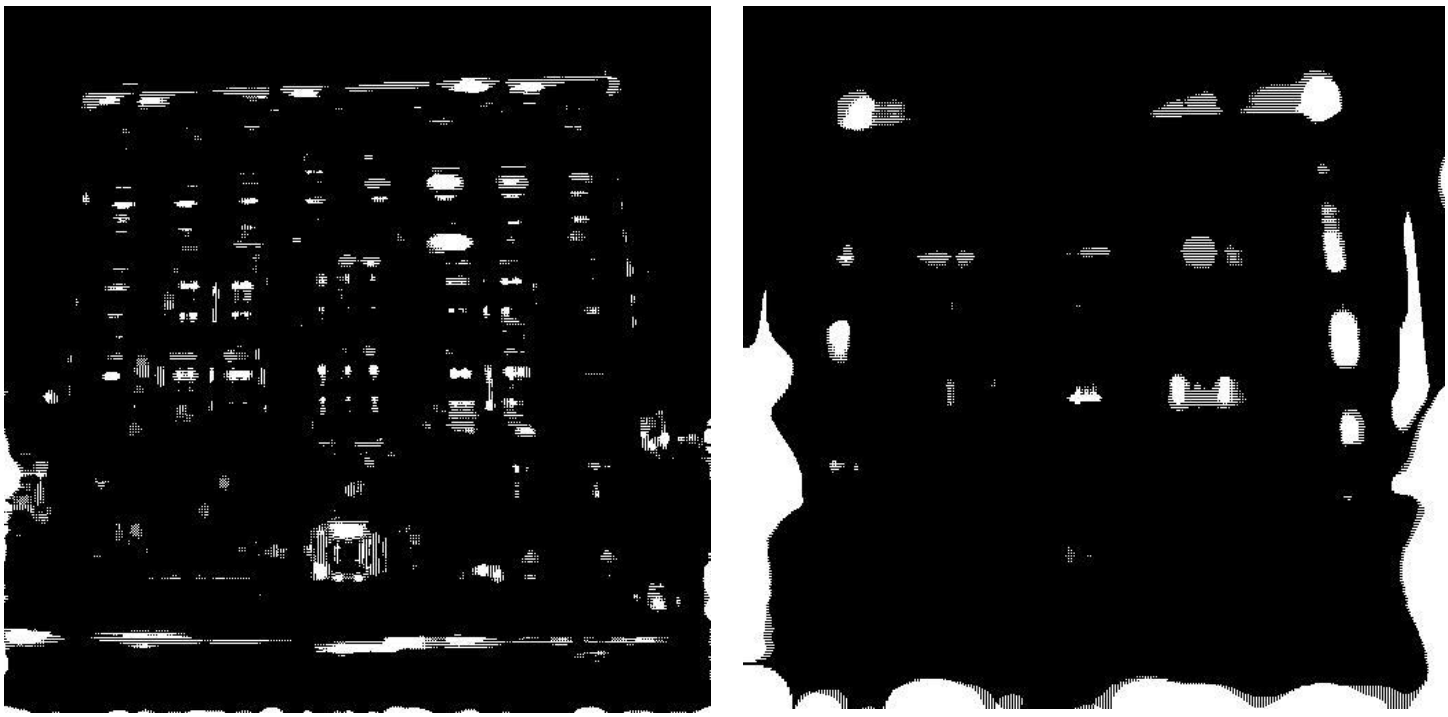
Comp 558

October 26, 2018

Assignment 2

1. Gaussian blurred images (left column), heat equation images (right column). Smoothing Scales of $\sigma = 4, 8, 12,$ and 16 for Gaussian blurring going from top to bottom of column respectively. Corresponding number of iterations for discretized heat equation with $\Delta t = 0.25$ and $t = 0.5\sigma^2$ going from top to bottom respectively.

Binary image of intensity differences for Gaussian blurring with σ = 4 and corresponding heat equation image (left). Gaussian blurring with σ = 16 and corresponding heat equation image (right). White pixels are locations where the absolute difference in intensity values were greater than 0.01.



In general the discretized heat equation does a good job of performing the same image smoothing as convolution with a Gaussian (when $t=0.5\sigma^2$). However there are two areas where the heat equation seems to struggle to mirror the Gaussian convolution, along the borders of the images and along edges within the image.

The error at the borders of the image occur from the two smoothing methods handling the computations at image borders differently. From the images above we see that early on the error at the borders is minimal (left image) but as the heat equation continues these errors propagate further into the image (right image).

The error at the edges within the image come from the fact that the heat equation needed to be discretized in order for this method to be implemented. As shown in the paper mentioned in the question description,

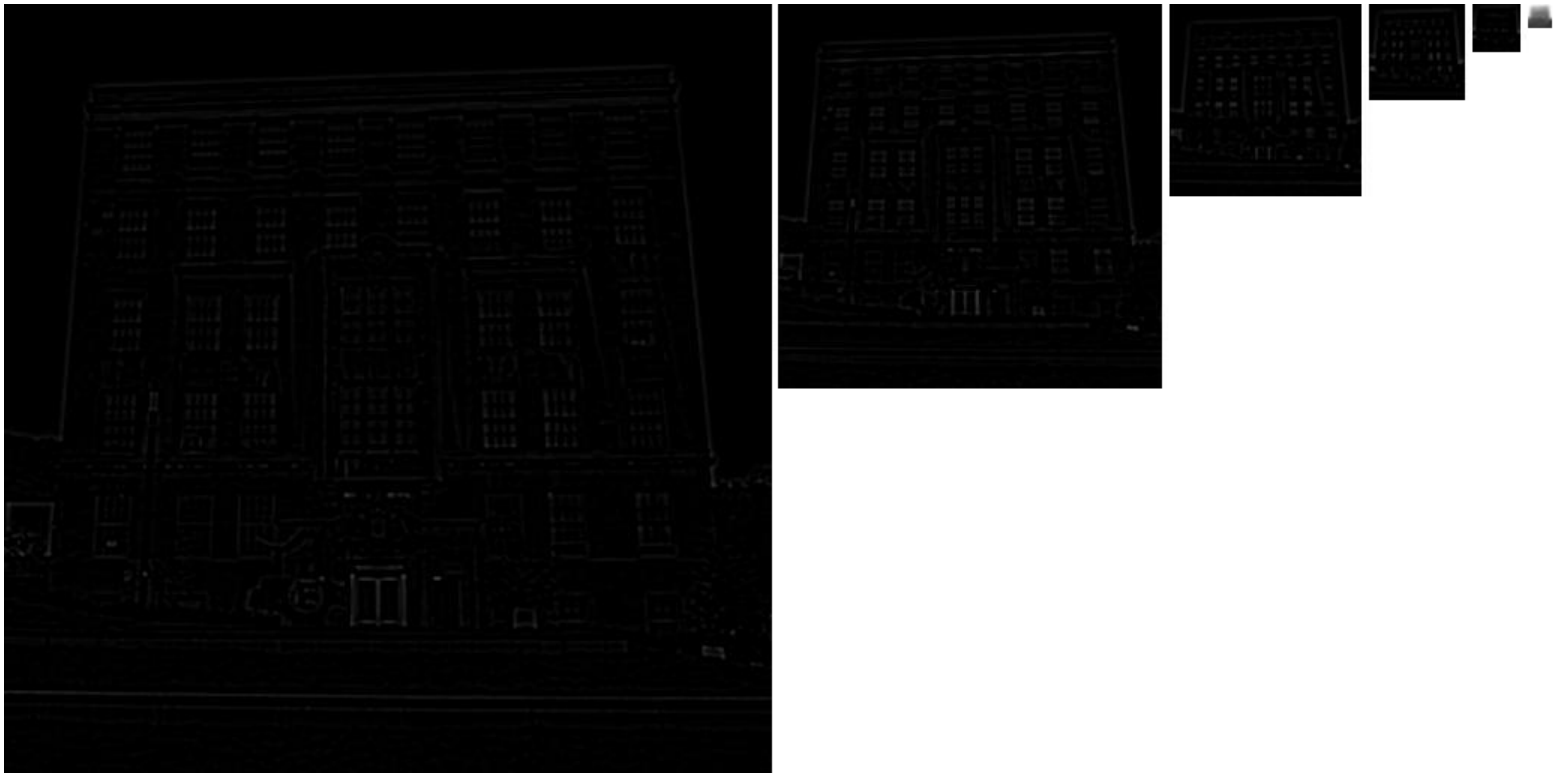
$$(\Delta K)(x,t) = \frac{\partial K}{\partial t}(x,t) = \lim_{\tau \to 0}(K(x,t+\tau) - K(x,t))/\tau.$$

Therefore the difference in image intensities between the convolution images and the heat equation images stems from my selection of $\Delta t = 0.25$. In order for the two methods to produce the same image (excluding image boundaries) Δt would need to approach 0. For this assignment and in practice, this is not feasible since the runtime of the heat equation would increase significantly as Δt goes to 0.

2.

Gaussian Pyramid

Laplacian Pyramid



Laplacian pyramid with intensity increased by a factor of 10 (note 6th image becomes all white and is lost in diagram)

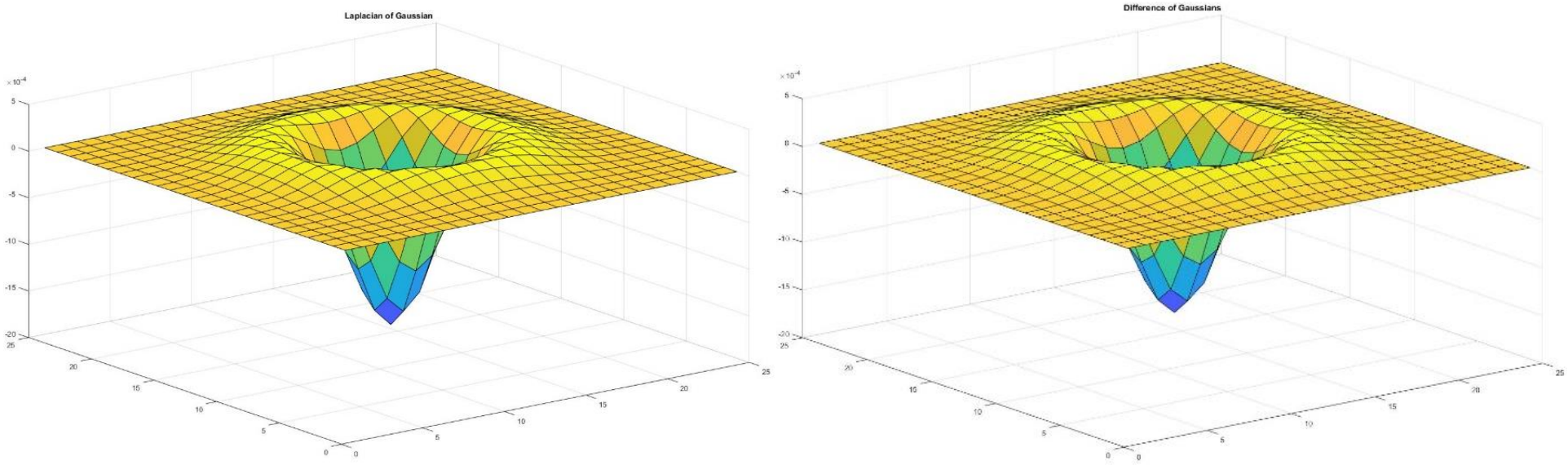
The scaling of intensities shows that the Laplacian pyramid identifies edges from the original image

Part 3: Lowe first uses the heat equation parameterized with respect to σ (instead of t=σ^2) to show that the derivative of a Gaussian with respect to σ is equal to the Laplacian of a Gaussian multiplied by a constant factor σ. Next he discretizes the derivative of the Gaussian in a similar manner to what we did in Question 1 of this assignment, using two different Gaussians of similar scales kσ and σ (i.e. k ~ 1), this gives him an approximation of the Laplacian of a Gaussian multiplied by a constant factor σ. Finally multiplying the denominator of the approximation (kσ – σ) out, Lowe shows that if kσ and σ are nearby scales then the difference of Gaussians with scales kσ and σ respectively gives an accurate approximation of the Laplacian of a Gaussian multiplied by a constant factor (k-1) σ^2.

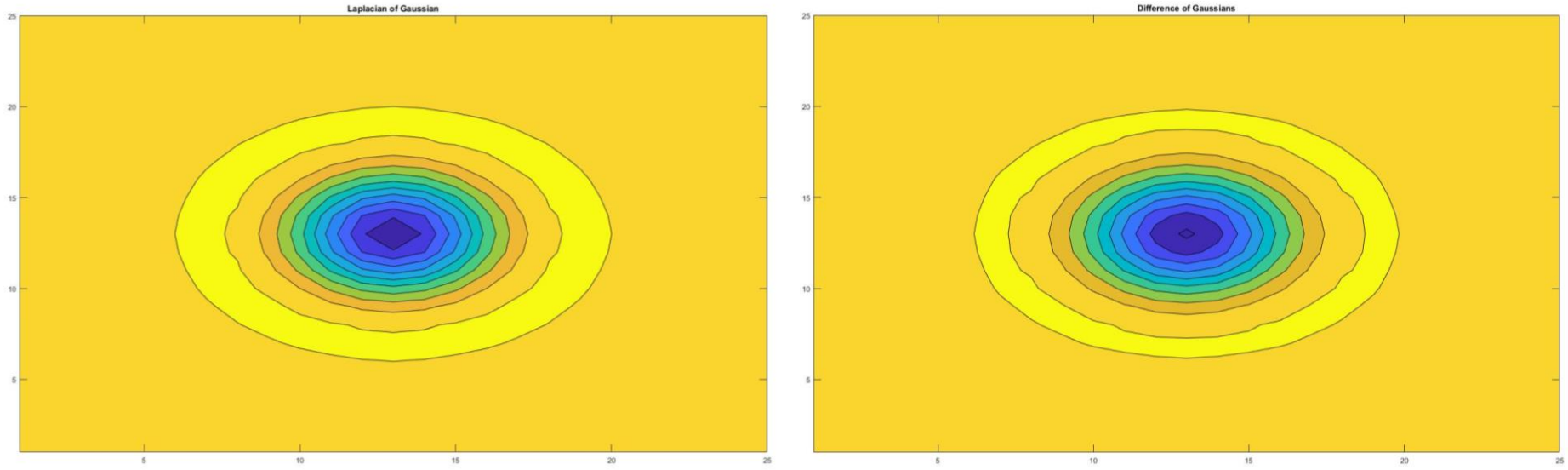
On the next page, with k = 1.05 and σ = 3 I demonstrate Lowe's derivation visually, specifically that

$$G(x, y, k\sigma) - \mathrm{G}(\mathrm{x}, \mathrm{y}, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

Surf diagram for Laplacian of Gaussian (left) and Difference of Gaussians (right). These 3D diagrams looks very similar, difficult to spot the exact differences using this qualitative method.



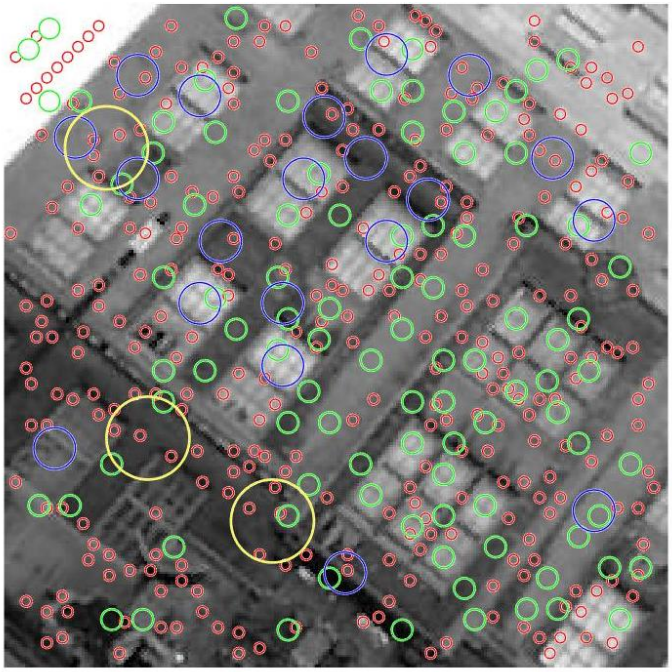
Contours for Laplacian of Gaussian (left) and Difference of Gaussians (right). The shape and value pattern of the contours are the same. However, the width of the two bands before the functions zero out are different for each function, also the DOG appears to have a different center value (region).



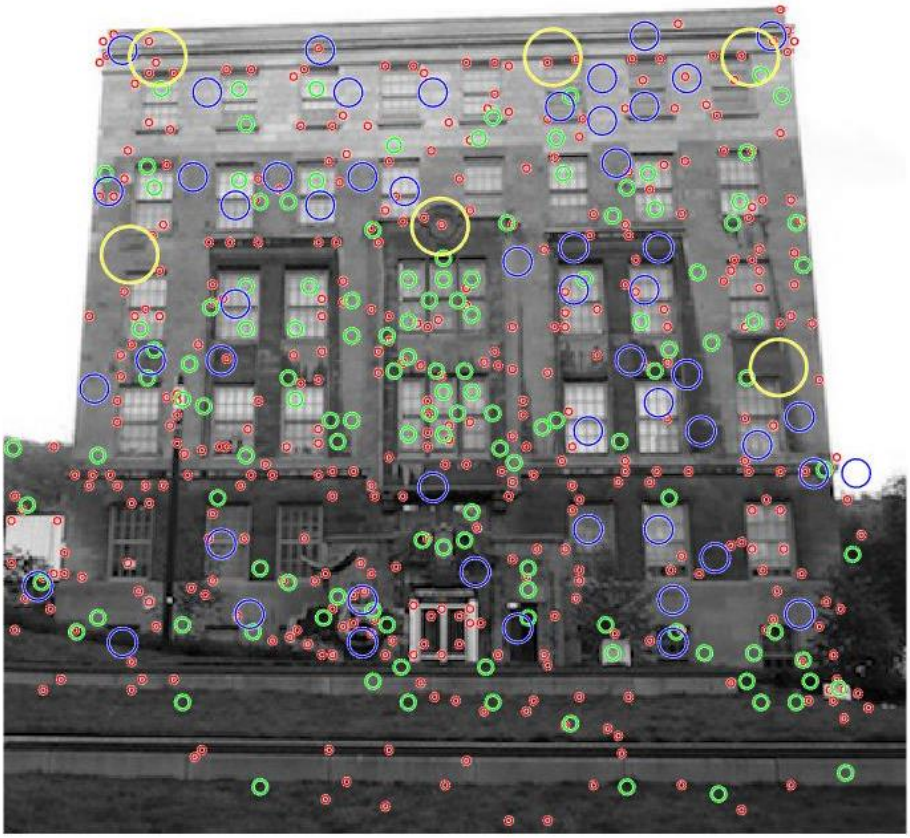
Part 4.

SIFT key points (no threshold for extrema)

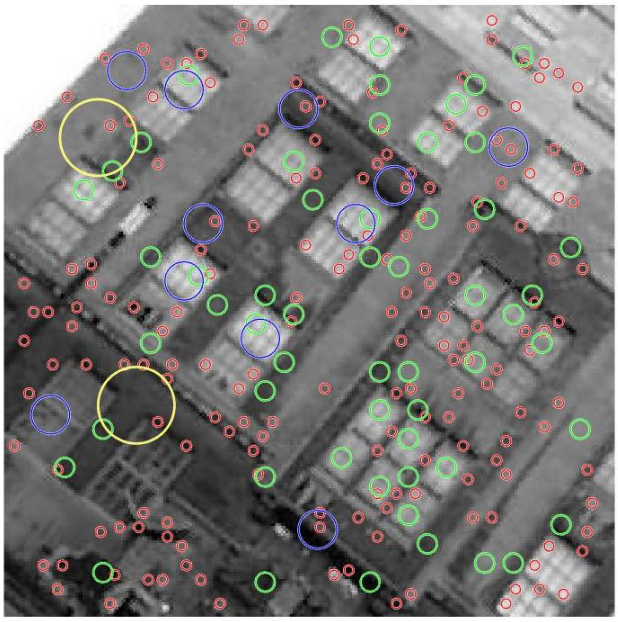
SIFT key points on rotated image (no threshold for extrema)



SIFT key points (extrema threshold = 0.002)



SIFT key points on rotated image (extrema threshold = 0.002)

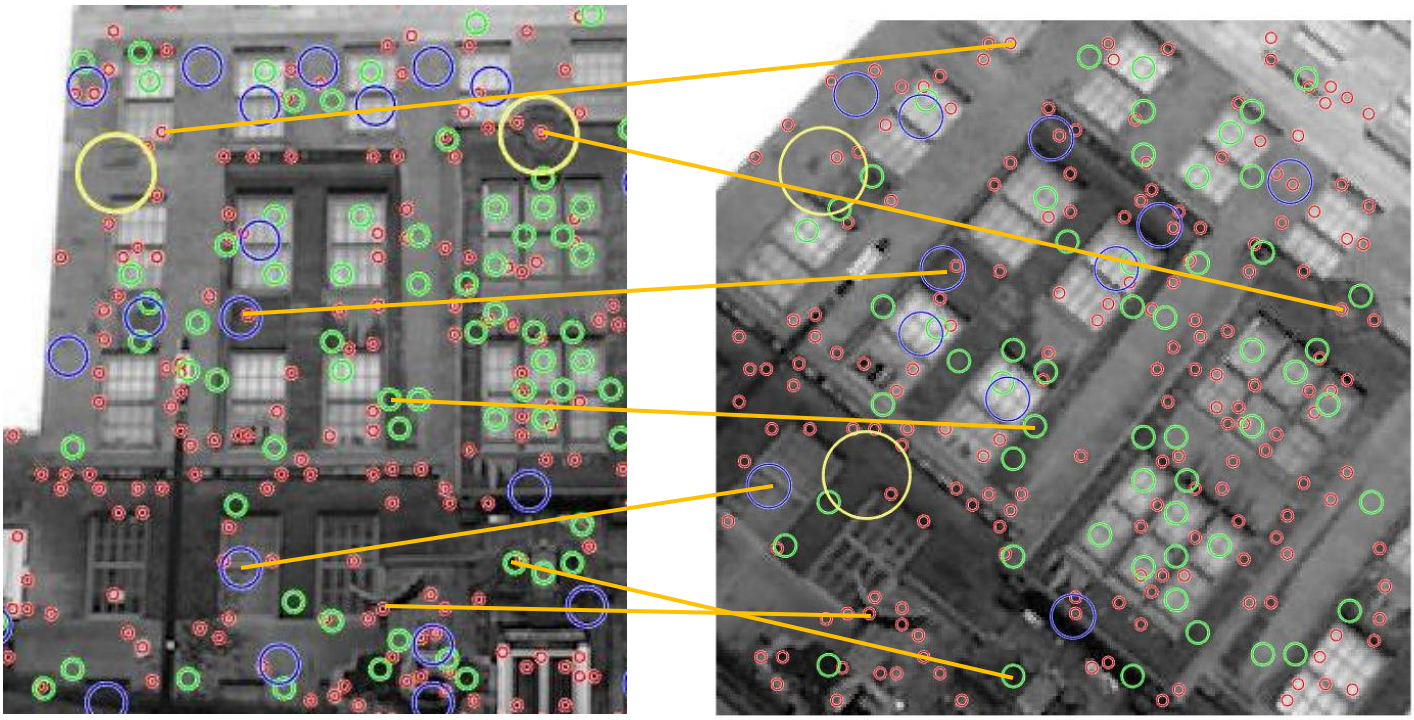


NOTE: extrema threshold was a method I added to my SIFT key point detector to filter poor quality key points out of my set. What originally brought about this idea were some of the key points you see in the first image, specifically where there are key points being found in the uniform sky of the image. I know that SIFT has more advanced techniques to remove low quality key points, but since we were just doing detection this quick fix worked. I would find a key point (max or min) and get the value at that pixel location and scale, then in the same neighbourhood I would find the second largest pixel intensity in the case of a maximum (second smallest for minimum). Next I would take the absolute difference of those two values and it would have to be above the pre-set threshold to remain a key point. In other words the extrema had to be of some significance.

In comparing the key points of the regular images with the key points of the rotated images it appears that many key points are robust to scale and orientation. One has to allow for a certain amount of error in checking key points in 'corresponding' locations since my method of mapping key points onto the original image has an error of 2^(n-1) where n is the level of the Laplacian pyramid that key point was found.

My key point detector is not perfect however, there are cases where a key point appears in the standard image and not in the rotated image or vice versa. I suspect that these are most likely key points that would be filtered out by the more advanced techniques featured in the full SIFT algorithm.

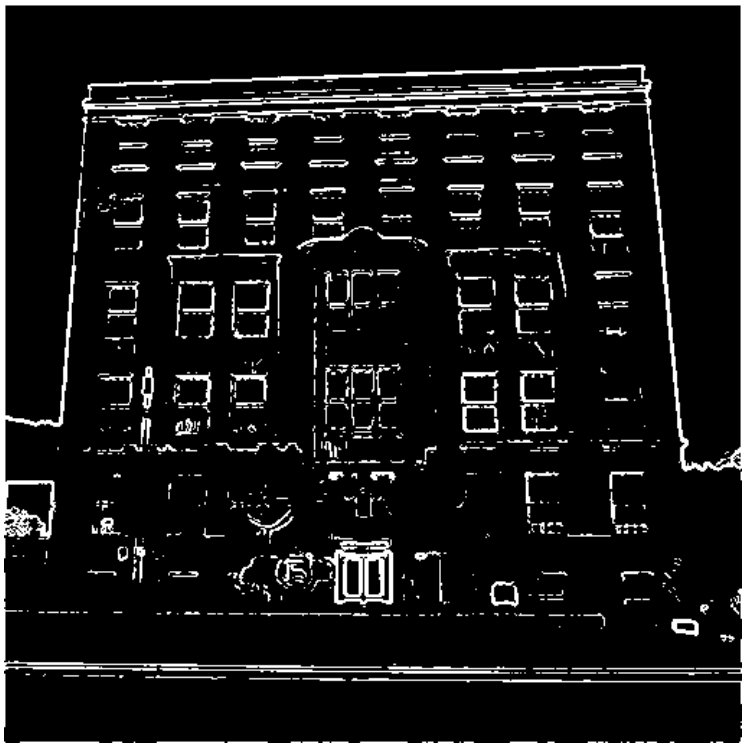
Using the two photos from the previous page (threshold = 0.02), with a crop on the standard image to focus on the area that was rotated. I visually demonstrate a few example of key point matches.



3.

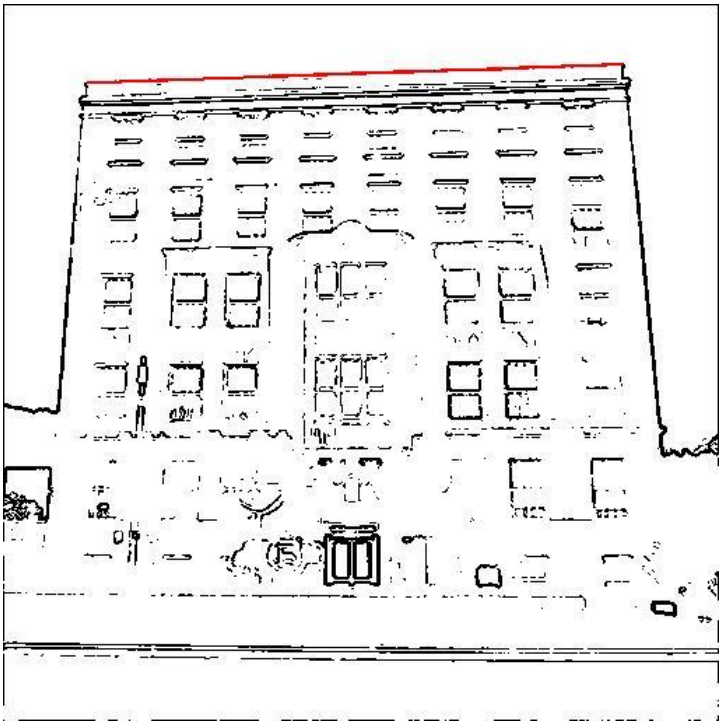
Binary Edge Image used for question 3

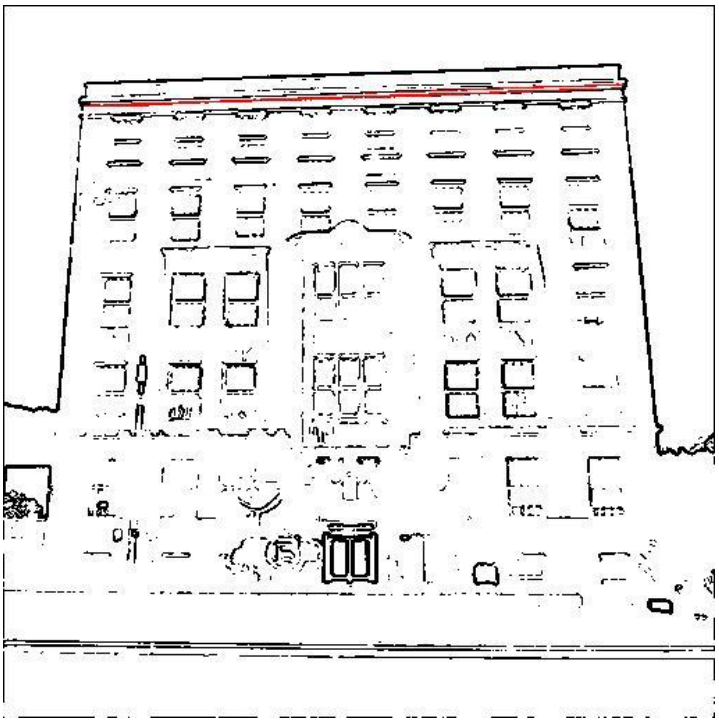
$\tau = 0.09$ (Image gradient magnitude threshold)



Edge Images (T=25)

RANSAC would commonly find these 4 edges (or edges in their neighbourhood)





Edge Images (T=250)

When T was raised to 250, RANSAC would commonly find these 2 edges (or edges in their neighbourhood)

My RANSAC was successfully able to find dominant edges in the image of the James Admin Building. However there were a few issues.

When T was low (e.g. 25) an issue that was encountered was that we might sample a point on a dominant edge however the gradient was not exactly perpendicular to the edge it was 'representing', this discrepancy in gradient orientation could be from image noise or minor variations in lighting. When this occurred our line model did not fall directly along the edge. Thus when checking for inliers, our distance measure would permit pixels not on the physical edge to be

counted if these pixels were on an edge that intersected (or nearly intersected) our skewed line model (assuming gradient orientation criteria was met as well). An example of this can be seen in the bottom left image of the T=25 images, where my RANSAC algorithm tags pixels on multiple lines of the steps outside the building as inliers.

When T is raised to 250 the previous issue tends to go away because we are more likely to sample a pixel on a dominant edge that is almost exactly perpendicular to the physical edge and thus our line model is correct.

Another issue that persists regardless of the value of T is multiple lines in the same neighbourhood in the image. Examples of this can be seen in the top right image of the T=25 set of pictures and the right image of the T=250 set, both are of the lines just below the top of the James Admin Building. The problem here is multiple horizontal edges grouped very close together. It is possible to isolate the correct line by lowering the distance comparison to nearly 1 pixel however, you begin to lose points that lie directly on the line in this case. In order for the RANSAC algorithm to correctly identify only points on the true edge as inliers the lines must be isolated from each other to an extent, lines that are 2 or 3 pixels apart will usually become grouped unless you can accept a greater error on inlier identification.