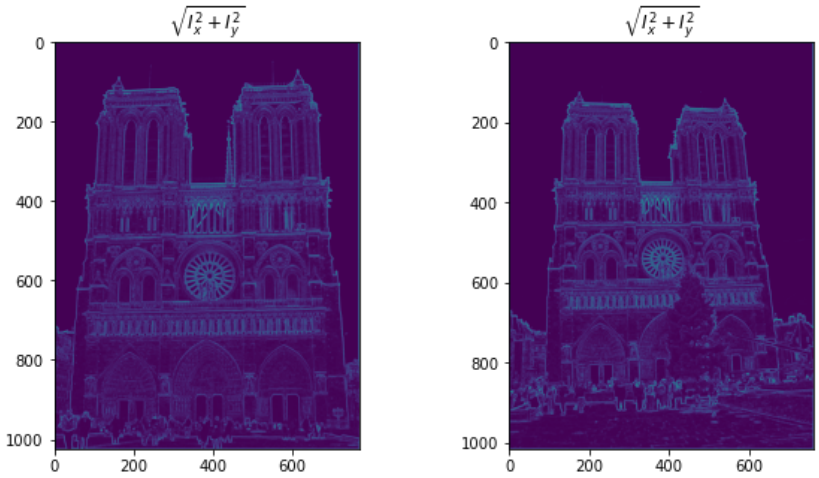
# CS 5330 Programming Assignment 2

[Sai Nikhil Thirandas]
[thirandas.s@northeastern.edu]
[thirandas.s]
[001564864]

# Part 1: Harris corner detector

[insert visualization of \sqrt($I_x^2$ + $I_y^2$) for Notre Dame image pair from pa2.ipynb here]

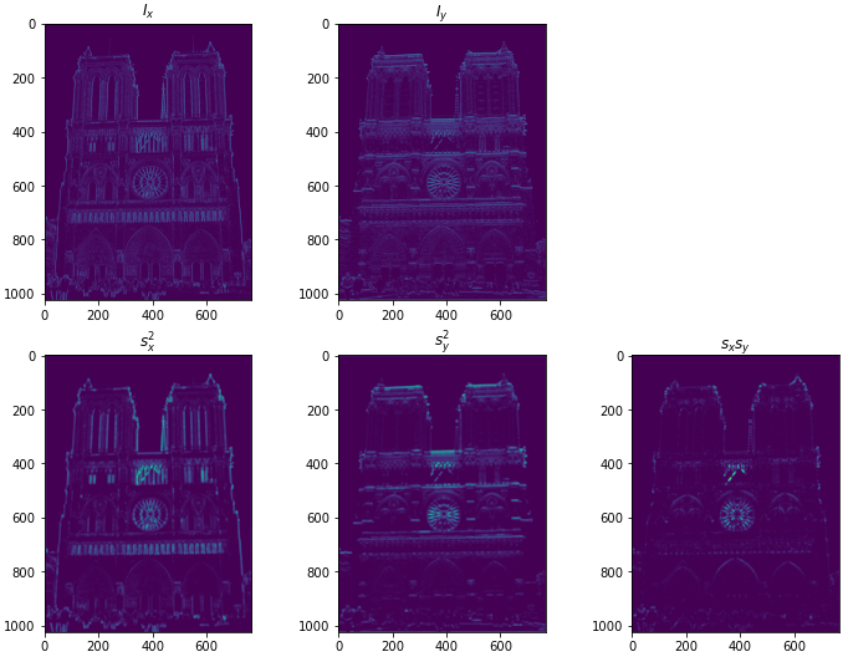



[Which areas have highest magnitude? Why?]

As shown in the the image edges (horizontal/vertical) and corners are areas with highest magnitude. This is because the change is intensity of the pixel value is high when there is an edge or a corner. Gradient measures the change in pixel intensity. Higher the gradient more likely there is an edge or a corner.

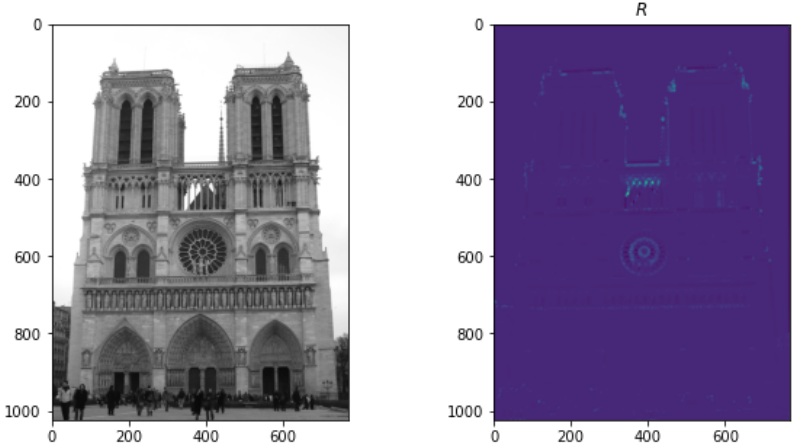
# Part 1: Harris corner detector

[insert visualization of $I_x$, $I_y$, $s_x^2$, $s_y^2$, $s_x s_y$ for Notre Dame image pair from pa2.ipynb here]



# Part 1: Harris corner detector

[insert visualization of corner response map of Notre Dame image from pa2.ipynb here]



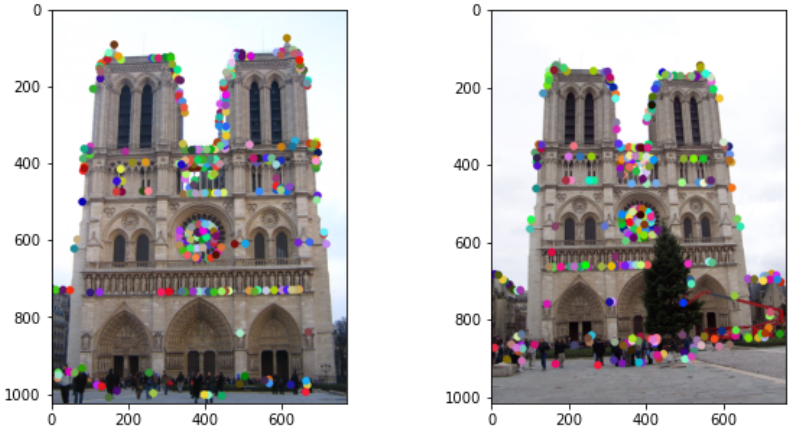

[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

Increasing brightness (additive offset) is addition of constant to intensity of every pixel. Since gradient measures change in intensity, it doesn't change when you add brightness to image.

Increasing contrast (multiplicative gain) is multiplication of constant to intensity of every pixel. This means the gradient is also multiplied by the same constant. However, if we use normalization, we can remove the effect of constant.

# Part 1: Harris corner detector

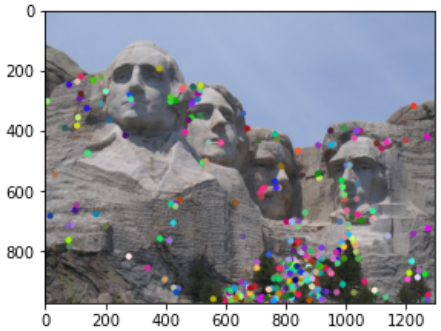
[insert visualization of Notre Dame interest points from pa2.ipynb here]

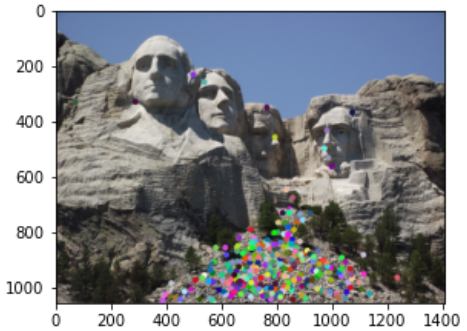






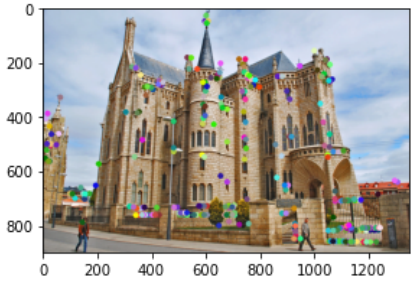
[insert visualization of Mt. Rushmore interest points from pa2.ipynb here]

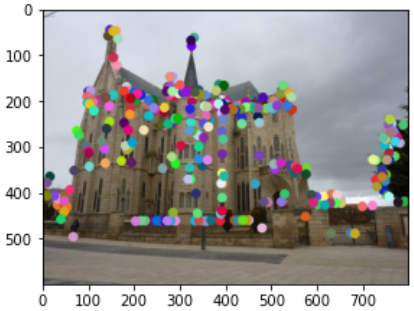


# Part 1: Harris corner detector

[insert visualization of Gaudi interest points from pa2.ipynb here]




[What are the advantages and disadvantages of using maxpooling for non-maximum suppression (NMS)?]

The advantage of Local Non Max Suppression (using maxpooling) is gain in time complexity (O(w*h), where w, h are height and width of image). It is also easy to implement. The main disadvantage is it is a greedy technique. The points can sometimes be still uneven. Hence, in order to have a better distribution of points, we need to use Adaptive Non Maximal Suppression. Most of the times ANMS outperforms LNMS in terms of accuracy, but not always. ANMS is quadratic in the number of key points detected. O(n^2), where “n” is the number of key points detected. In worst case, n ~ O(w*h) that implies TC of ANMS is O(w^2 * h^2).

# Part 1: Harris corner detector

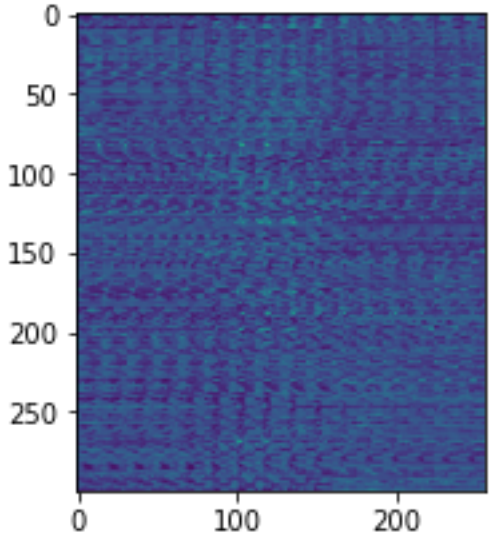
[What is your intuition behind what makes the Harris corner detector effective?]

Let $\lambda_1$ and $\lambda_2$ are the eigenvalues of Autocorrelation matrix (**A**). So, the values of these eigenvalues decide whether a region is a corner, edge or flat. Let the Harris corner score be $\boldsymbol{R} = \det(\boldsymbol{A}) - \alpha\, trace(\boldsymbol{A})^2 = (\lambda_1\lambda_2) - \alpha(\lambda_1+\lambda_2)^2$.

- When |**R**| is small, which happens when $\lambda_1$ and $\lambda_2$ are small, the region is flat.
- When |**R**| < 0, which happens when $\lambda_1 >> \lambda_2$ or vice versa, the region is an edge.
- When |**R**| is large, which happens when $\lambda_1$ and $\lambda_2$ are large and $\lambda_1 \sim \lambda_2$, the region is a corner.

# Part 2: Normalized patch feature descriptor

[insert visualization of normalized patch descriptor from pa2.ipynb here]

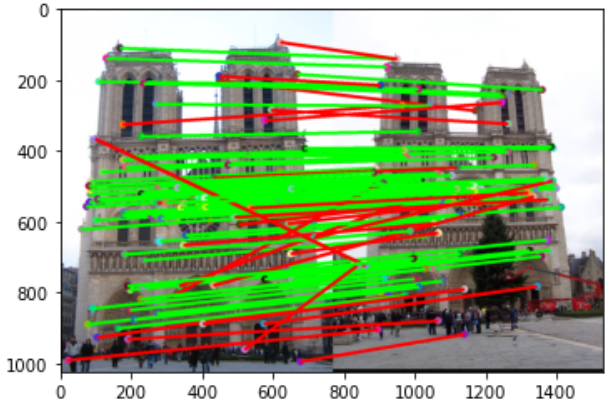



[Why aren't normalized patches a very good descriptor?]

In normalized patch descriptor, we take corner points and construct a 16x16 window and flatten it to get a 256-size vector and compute norm between another 256-size vector in a different image. Clearly if the second image is rotated or scaled, we may still have a larger distance between visually same key points. In short it is not scale/rotation invariant.

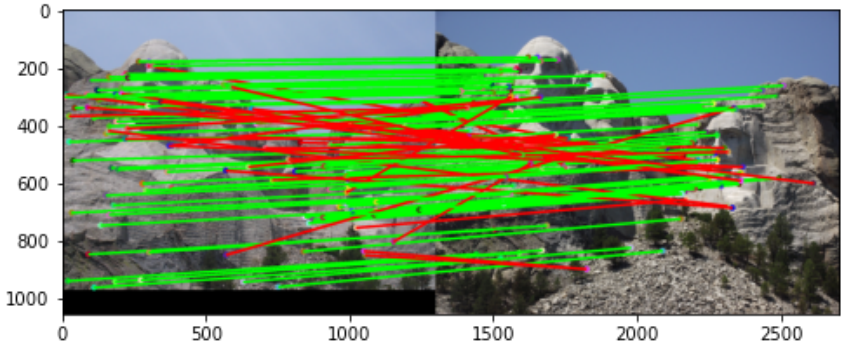
# Part 3: Feature matching

[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from pa2.ipynb here]




# matches (out of 100): [105]
Accuracy: [75.24 %]

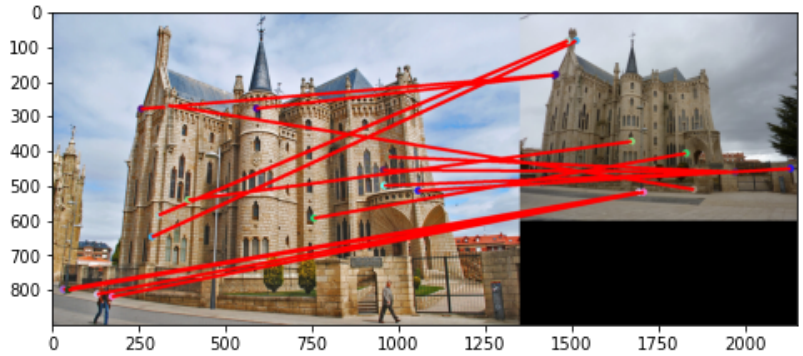
[insert visualization of matches for Mt. Rushmore image pair from pa2.ipynb here]




# matches: [111]
Accuracy: [73.874 %]

# Part 3: Feature matching

[insert visualization of matches for Gaudi image pair from pa2.ipynb here]



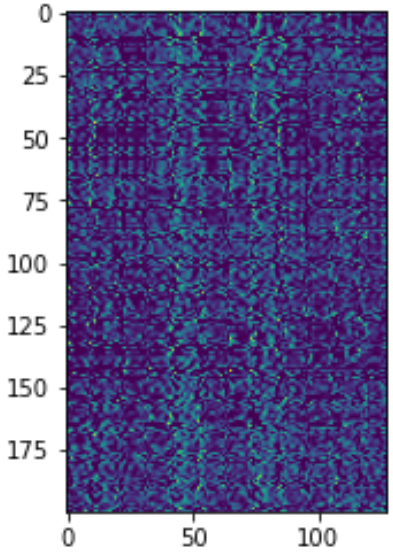

# matches: [15]
Accuracy: [0.00 %]

[Describe your implementation of feature matching here]

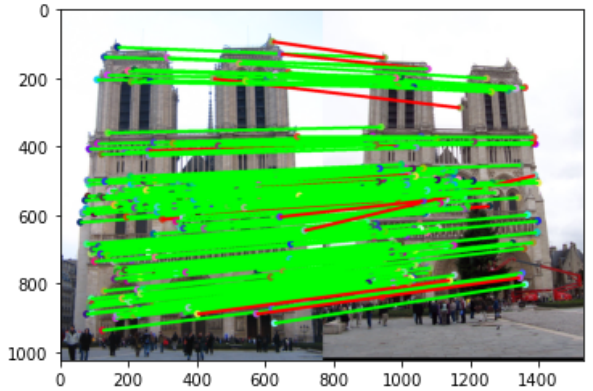
I first compute all pair distances between key points in image1 and image2. Then I compare the top 2 nearest neighbors for every point of image1 in image 2. I compute the ratio of smaller distance and second smaller distance. If this ratio is smaller than a thresholder value (for example 0.8), then I ignore these points. Otherwise, I say it is a matching pair. I return the list of matches sorted by confidence score and their corresponding confidence values.

# Part 4: SIFT feature descriptor

[insert visualization of SIFT feature descriptor from pa2.ipynb here]




[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from pa2.ipynb here]

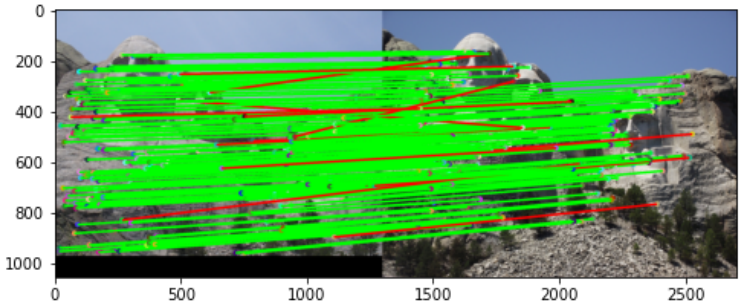



# matches (out of 100): [196]

Accuracy: [93.37 %]

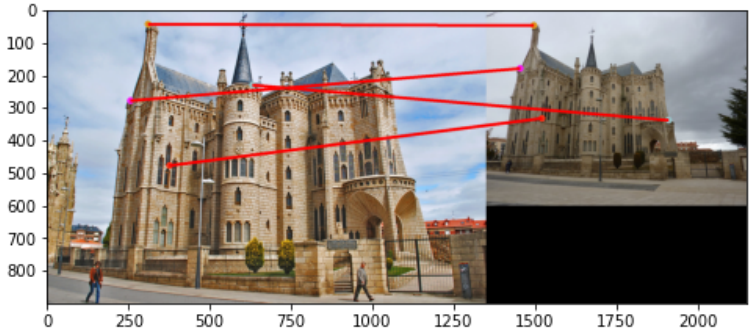
# Part 4: SIFT feature descriptor

[insert visualization of matches for Mt. Rushmore image pair from pa2.ipynb here]




# matches: [181]
Accuracy: [92.82 %]

[insert visualization of matches for Gaudiimage pair from pa2.ipynb here]




# matches: [4]
Accuracy: [0.00 %]

# Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

I retrieve gradient magnitudes and orientations of the image. For every 16x16 patch around a corner point, I construct 16 histograms (one for each 4x4 window) of orientations with 8 bins of equal width equal to pi/4. The heights of histogram are equal to the sum of magnitude of gradients falling in respective bins. I reshape this 4x4x 8 3d array in to (128,1) size 2d array.

[Why are SIFT features better descriptors than the normalized patches?]

As SIFT is relying on gradients instead of image intensity values, and since we already know that gradients are invariant to brightness and their effects due to change in contrast can be nullified by normalization, SIFT features are better descriptors than normalized patches.

# Conclusion

[Why aren't our version of SIFT features rotation- or scale-invariant? What would you have to do to make them so?]

Because the direction of gradients change upon both rotation and scale, and SIFT features are relying on them, they are not rotation or scale invariant. In order to account for scale variance, we can construct a scale space (a collection of images having different scales generated from a single image) and enhance them using a Difference of Gaussian as described in the class. In order to make it rotation invariant, we can choose a dominant direction after constructing histogram as described for SIFT descriptor. This is done by finding the significant peaks in the histogram distribution as described in Lowe's paper.