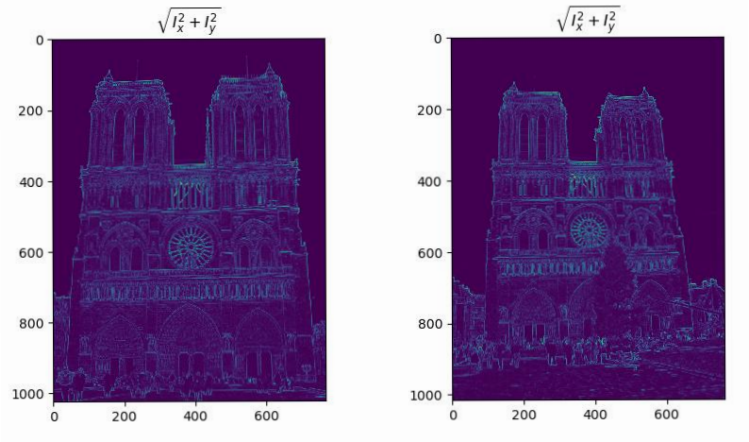
CS 5330 Programming Assignment 2

Amina Tabassum
tabassum.a@northeastern.edu
tabassum.a
002190127

Part 1: Harris corner detector

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

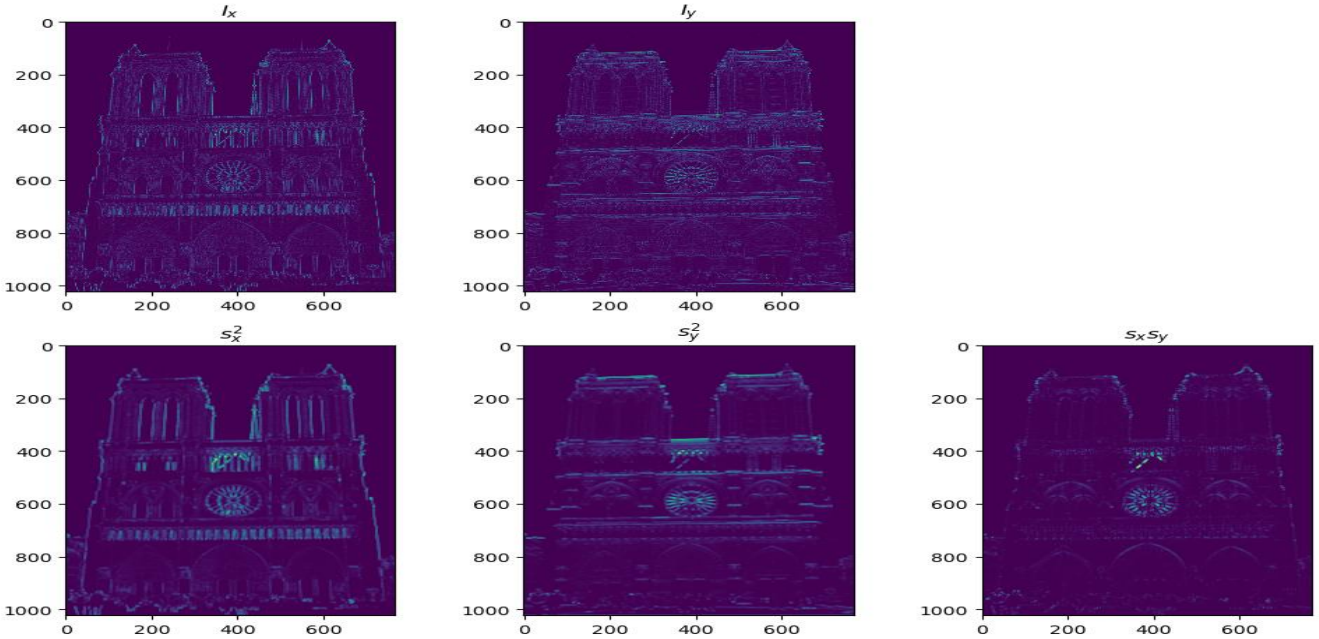


[Which areas have highest magnitude? Why?]

Edges and corners have highest magnitude. These are the points for which rate of change is highest and hence higher magnitude.

Part 1: Harris corner detector

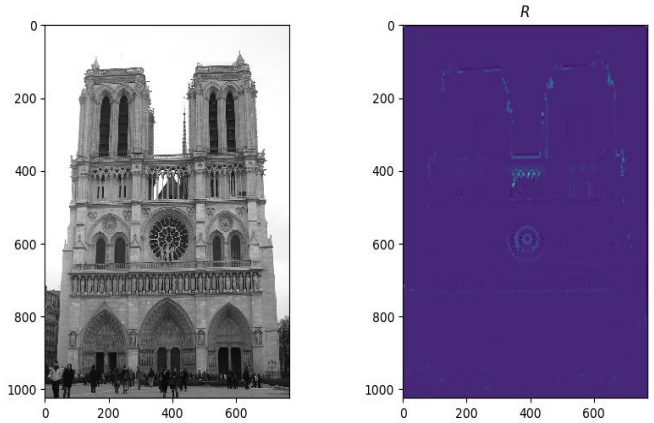
[insert visualization of I_x, I_y, s_x^2, s_y^2, $s_x s_y$ for Notre Dame image pair from pa2.ipynb here]



Part 1: Harris corner detector

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

```
compute_harris_response_map():  "Correct"
Out[505]: <matplotlib.image.AxesImage at 0x7f9715df9190>
```



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

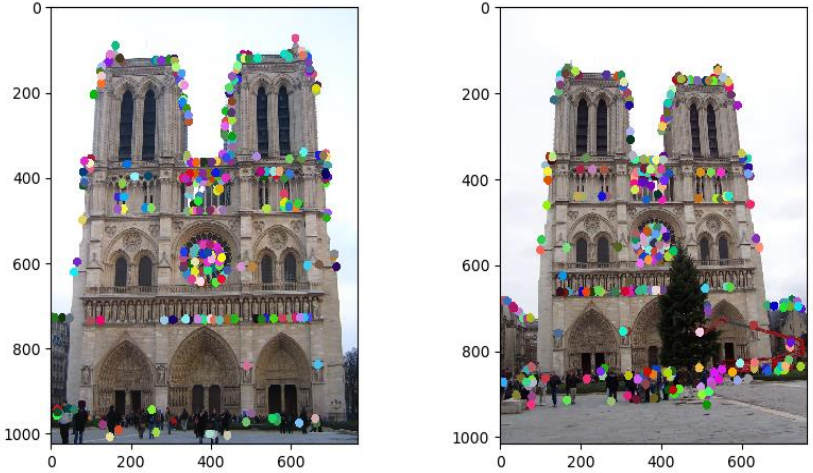
Gradient features are scale invariant. These features give us measure of rate of change which remains same irrespective of scaling. So, these feature are invariant to both additive shift and multiplicative gain.

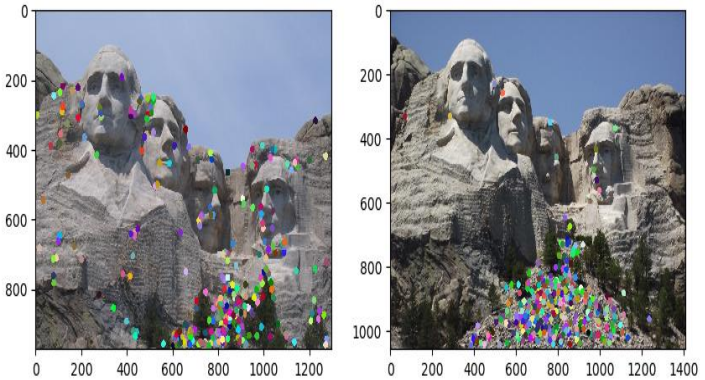
Part 1: Harris corner detector

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

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

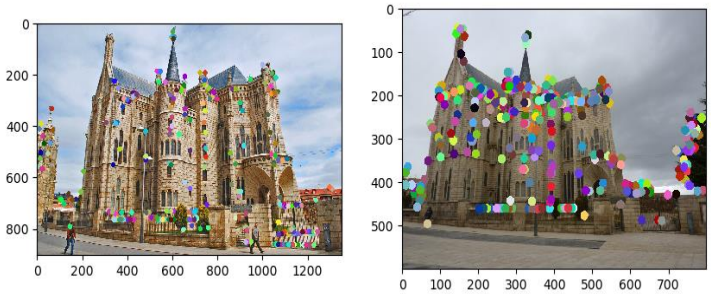
```
get_harris_interest_points() "Correct"
2464 corners in image 1, 2456 corners in image 2
```





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)?]

Maxpooling helps us extract low level features of image.It selects maximum value from certain kernal-window region and stores that value. In this way, it makes non-maximum suppression **computationally efficient** since we only need to compare maxpooling matrix with original matrix and hence select the matched values. We don't need to compute local maximum repetitively since maxpooling already gives us.

However, it **only considers the maximum element from kernel/pooling window and ignores other elements**. So, if certain pooling region has elements of high magnitudes, the discerning features will be ignored resulting selection of only maximum point.

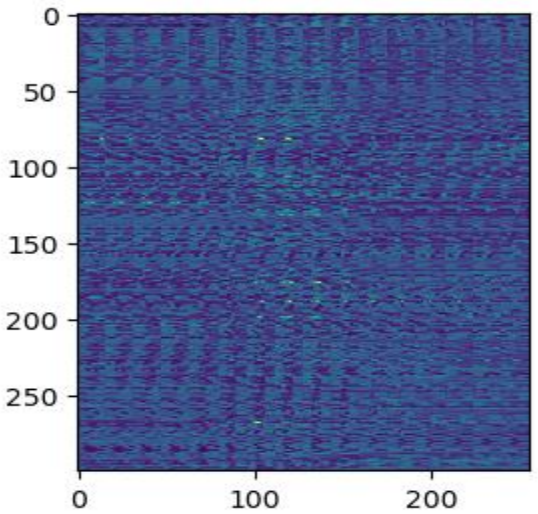
Part 1: Harris corner detector

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

Harris corner detector works by detecting points based on their intensity variation in local region. It selects small window/region around interest points for which intensity change is high as compared to when this window is shifted in other directions. It takes the horizontal and vertical derivatives of image and considers the points where both derivatives are high. Hence, it considers corners as interest points. So, taking corner score response with respect to direction directly makes it effective.

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?]

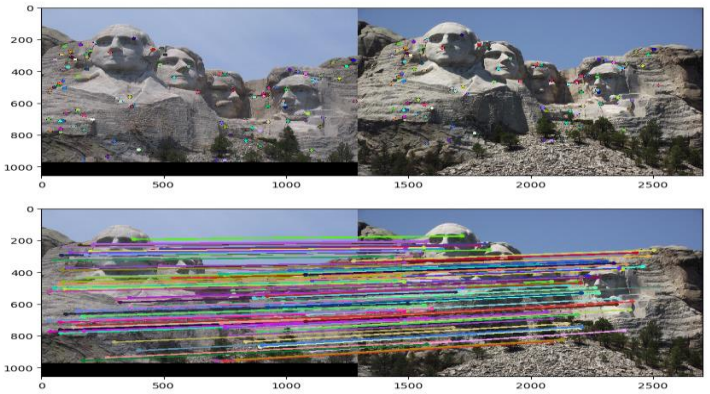
Normalized patches basically form a feature vector by selecting a region around interest point and normalizing that region. These patches are sensitive scaling and rotations and hence, not a good idea.

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): [95]
Accuracy: [0.87]

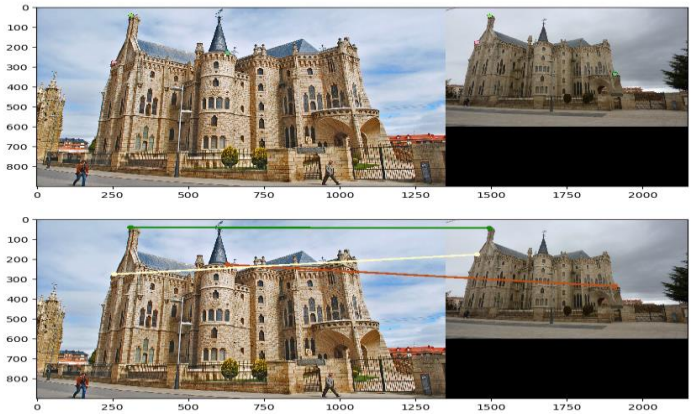
[insert visualization of matches for Mt. Rushmore image pair from pa2.ipynb here]



matches: [91]
Accuracy: [0.83]

Part 3: Feature matching

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

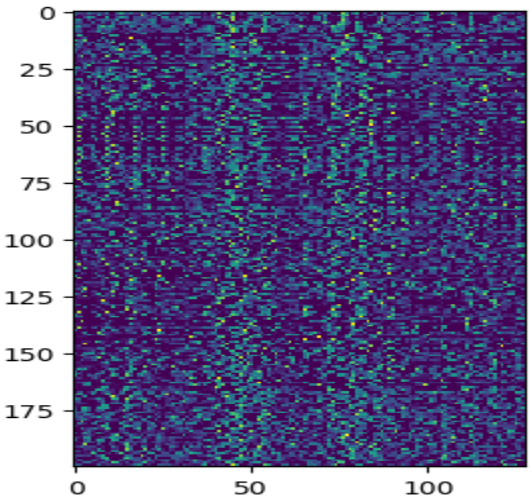


matches: [3]
Accuracy: [0.03]

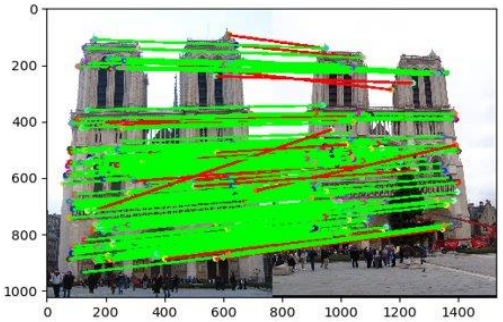
[Describe your implementation of feature matching here]

Part 4: SIFT feature descriptor

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



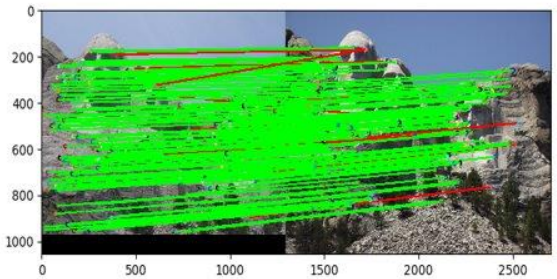
[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from pa2.ipynb here]



matches (out of 100): [95]
Accuracy: [0.87]

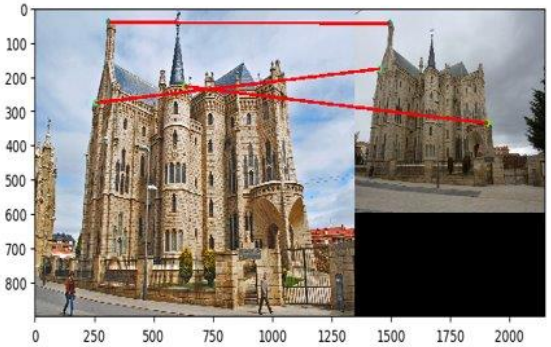
Part 4: SIFT feature descriptor

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



matches: [91]
Accuracy: [0.83]

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



matches: [3]
Accuracy: [0.03]

Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

1. Find magnitude and direction of Ix and Iy and find keypoints.
2. Find gradient histogram vector from patch
    i. Divide 16x16 window into 4x4 girds.
    ii.Compute orientation histogram from each cell
    iii.Return 16cells x 8 orientation =128 x1 feature descriptor
3. Define function that return feature vector for one specific point
4. Use function defined in step3 to find feature vectors for all interest points.
5.Match keypoints.

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

SIFT features are better because these are scale and rotation invariant.

Conclusion

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

Our SIFT version is very basic. We just used normalized feature vector. We did not use adaptive window size. We did not implement rotation according to dominant gradient orientation. We did not implement gaussian smoothing and interpolation.

Incorporating adaptive window size, interpolation, gaussian smoothing and histogram orientation according to dominant gradient can make it rotation invariant.