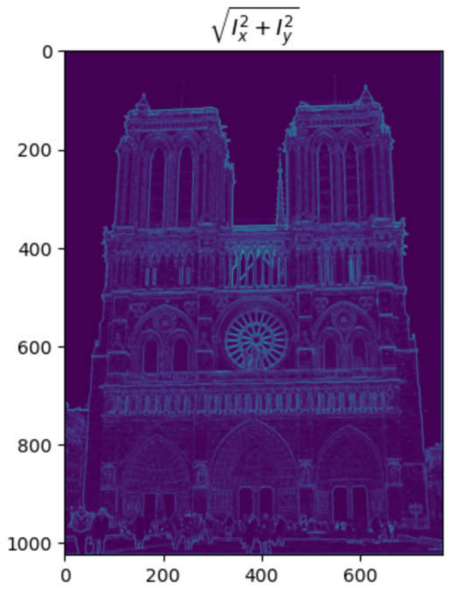
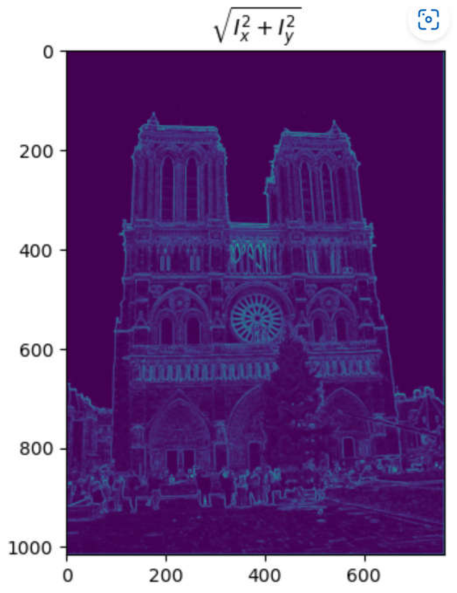
CS 4476/6476 Project 2

[Jiarui Xu]
[jxu605@gatech.edu]
[jxu605]
[903617486]

Part 1: Harris corner detector

[insert visualization of \sqrt(I_x^2 + I_y^2) for Notre Dame image pair from proj2.ipynb here]



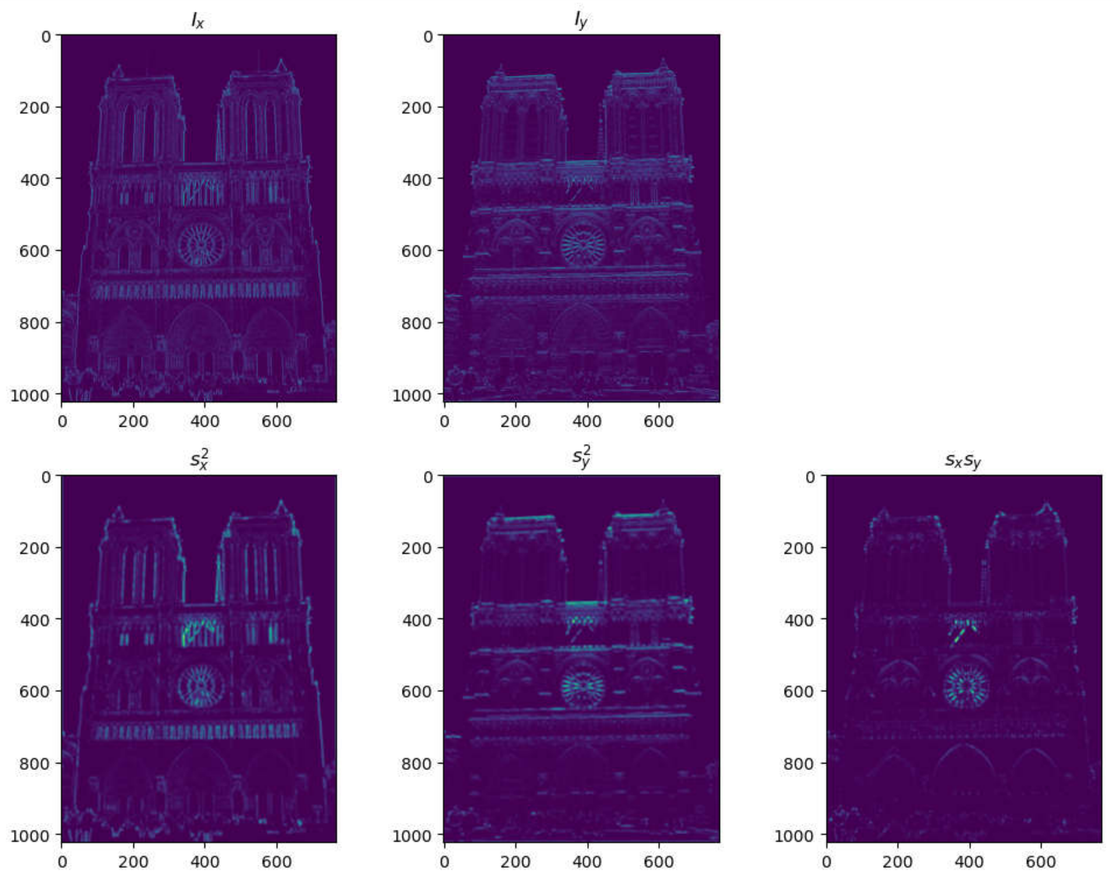


[Which areas have highest magnitude? Why?

The edges of the image.
The value of pixels changes most sharply on the edges, the x gradient or y gradient of the edges should be highest.

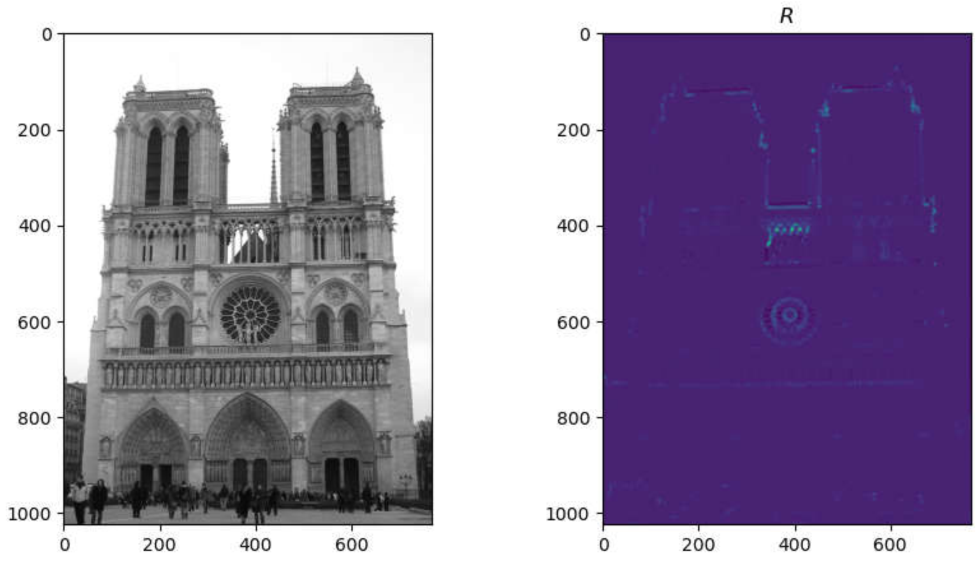
Part 1: Harris corner detector

[insert visualization of I_x, I_y, s_x^2, s_y^2, $s_x s_y$ for Notre Dame image pair from proj2.ipynb here]



Part 1: Harris corner detector

[insert visualization of corner response map of Notre Dame image from proj2.ipynb here]

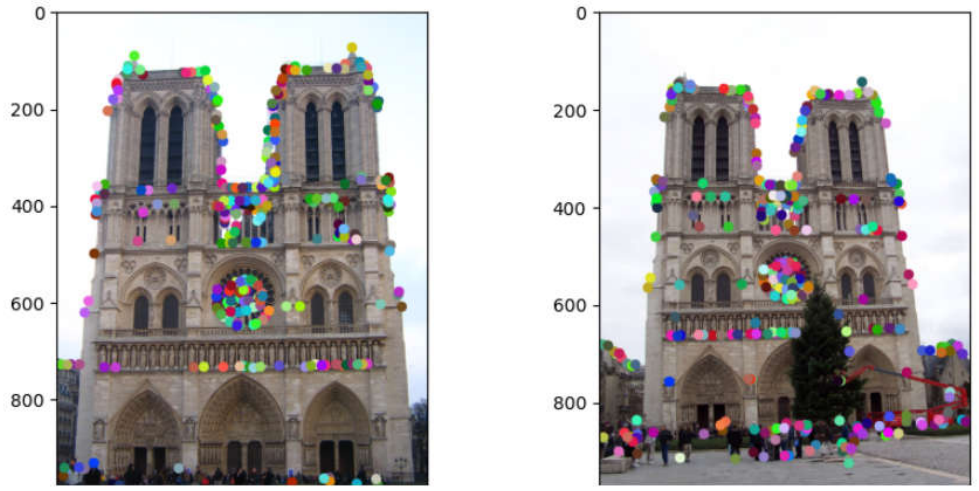


[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

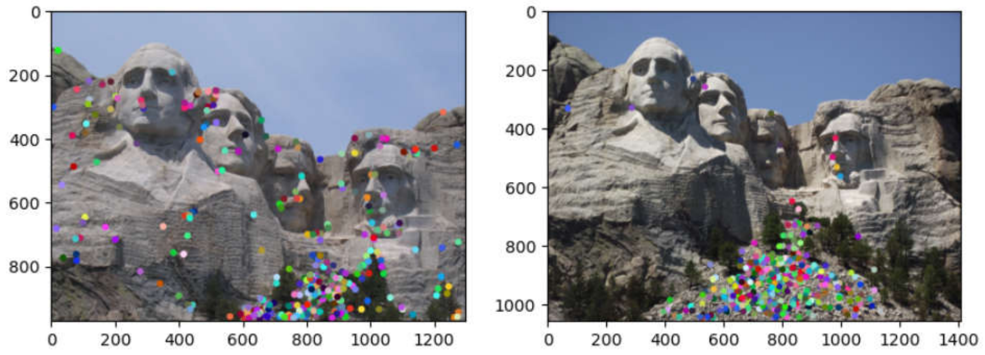
I think gradient features are invariant to brightness, as adding the same value to all the pixels will not change the derivative between pixels.
However, it’s not invariant to contrast, because multiplication will change derivatives.

Part 1: Harris corner detector

[insert visualization of Notre Dame interest points from proj2.ipynb here]

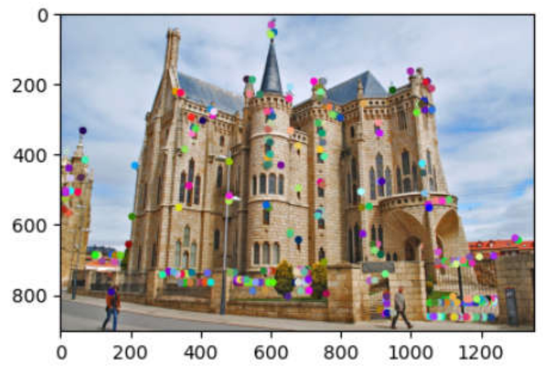


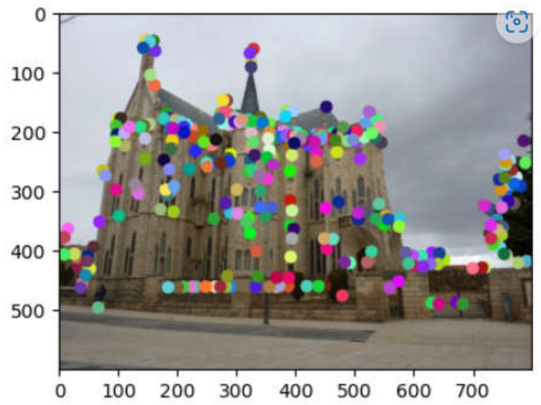
[insert visualization of Mt. Rushmore interest points from proj2.ipynb here]



Part 1: Harris corner detector

[insert visualization of Gaudi interest points from proj2.ipynb here]





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

Advantages:
I think finding the local maxima of the R matrix makes it easier to find top k points in the image. The reason for that is: with much less points to compare, we just have to compare all the local maxima.
Disadvantage:
If there are many corners in a local window, it’s likely to filter out these point with maxpooling, just leaving the point with largest confidence in this window.

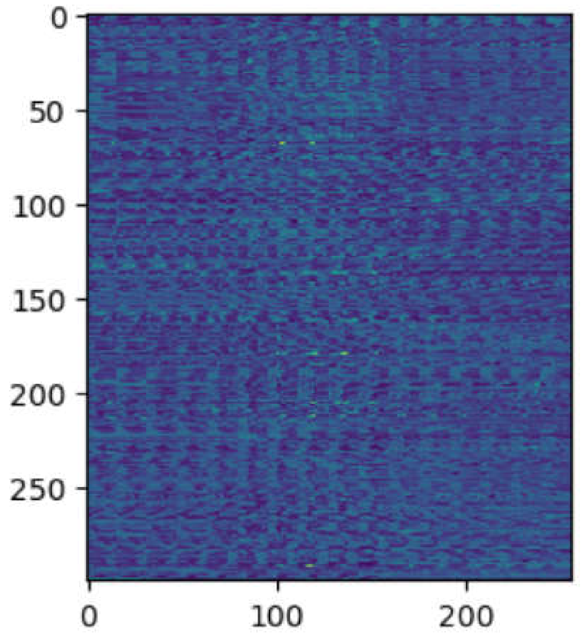
Part 1: Harris corner detector

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

The confidence in the formula of R = det(A) – alpha * trace(A) ** 2, we can have the feeling that Harris corner is based on the gradients in the local window centered at a certain point. If the point is a corner, the values in the window should have a remarkable change with the window moving in every direction.

Part 2: Normalized patch feature descriptor

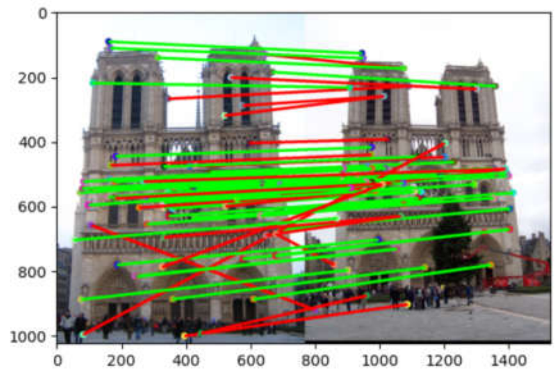
[insert visualization of normalized patch descriptor from proj2.ipynb here]



[Why aren't normalized patches a very good descriptor?]
Because when normalized patches stored, there will be some redundant information, making computing and storing feature descriptor much more complicated.

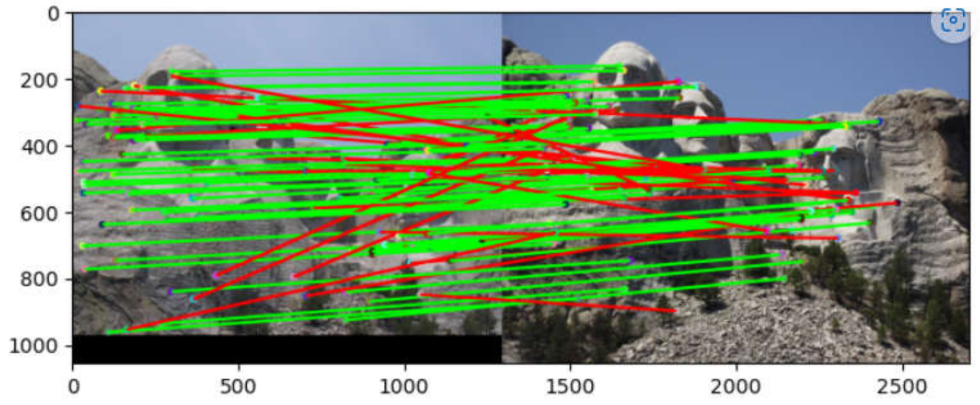
Part 3: Feature matching

[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from proj2.ipynb here]



# matches (out of 100): 74
Accuracy: 0.52

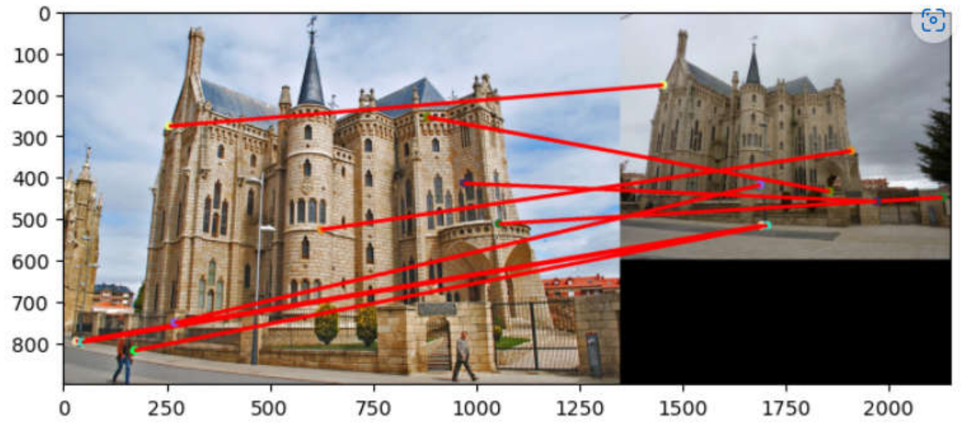
[insert visualization of matches for Mt. Rushmore image pair from proj2.ipynb here]



# matches: 101/100
Accuracy: 0.70297

Part 3: Feature matching

[insert visualization of matches for Gaudi image pair from proj2.ipynb here]



# matches: 9/100

Accuracy: 0.000000

[Describe your implementation of feature matching here]

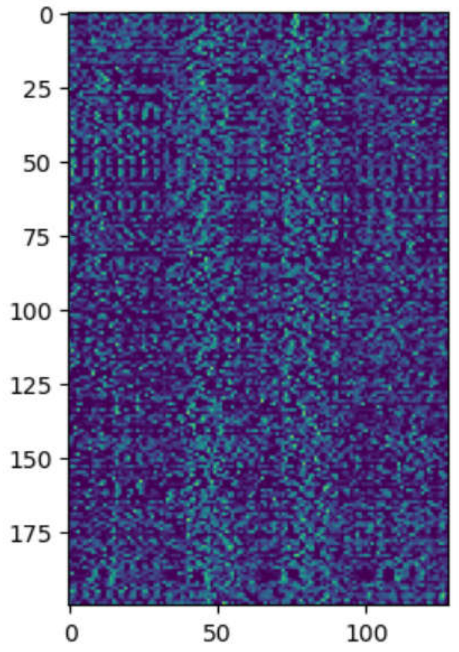
First I calculate the Euclidean distance between features1 and features2.

Then I find the shortest two for each element in features1.

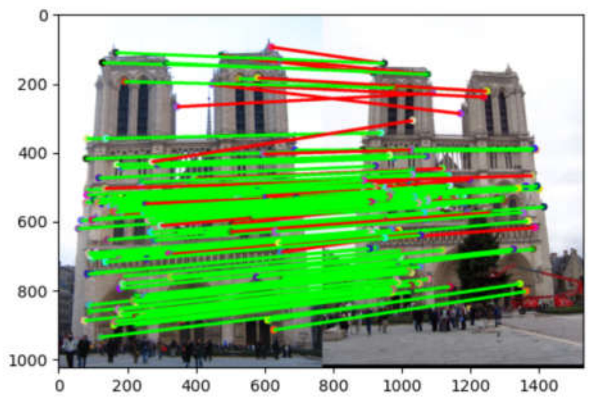
Next, calculate NNDR = d1/d2. If NNDR > 0.8, I consider f1 is matching at least 2 features in f2, meaning it's not appropriately matched. If NNDR <= 0.8, I consider the f1 feature matched successfully.

Part 4: SIFT feature descriptor

[insert visualization of SIFT feature descriptor from proj2.ipynb here]



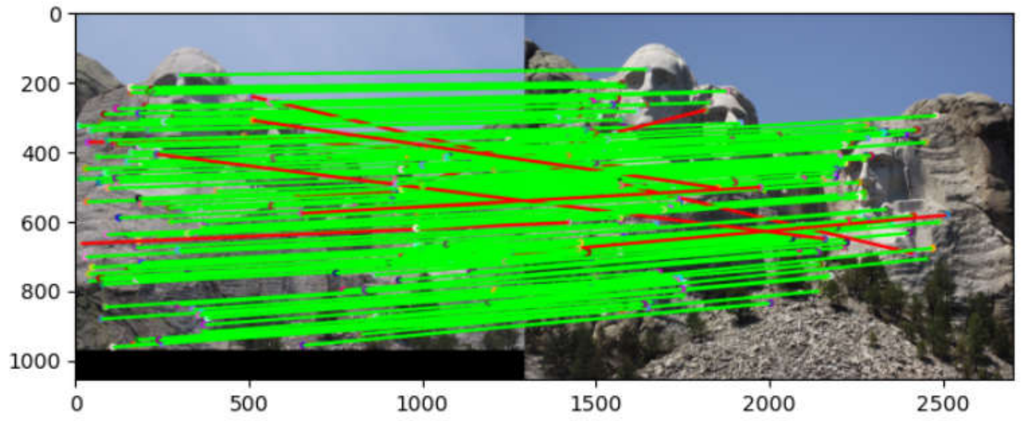
[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from proj2.ipynb here]



# matches (out of 100): 146
Accuracy: 0.835616

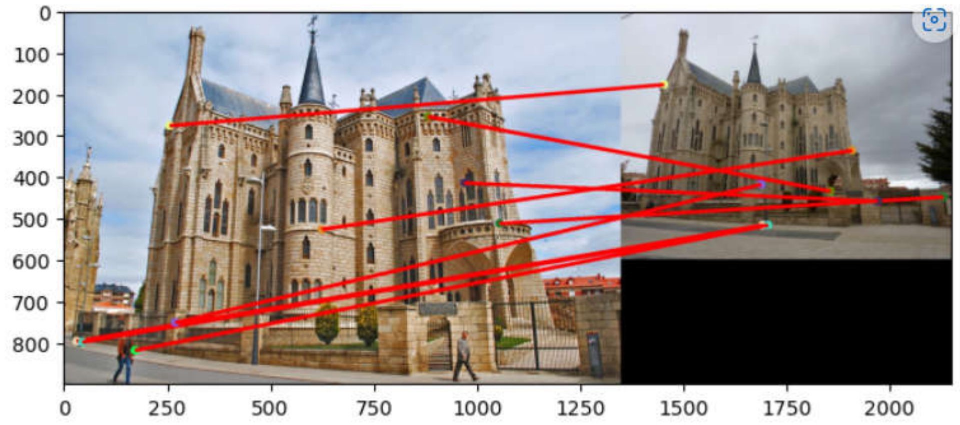
Part 4: SIFT feature descriptor

[insert visualization of matches for Mt. Rushmore image pair from proj2.ipynb here]



# matches: 165/100
Accuracy: 0.93333

[insert visualization of matches for Gaudiimage pair from proj2.ipynb here]



# matches: 9
Accuracy: 0.000000

Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

1. Divide the 16*16 window into 16 4*4 cells.
2. Compute the gradients Ix, Iy in the cell.
3. Compute the orientations(arctan2()) and magnitudes of the vector (Ix, Iy) in every pixel. The orientation is in range [-pi, pi]
4. Divide the range [-pi, pi] into 8 part: -7pi/8, -5pi/5, …, 5pi/8, 7pi/8]
5. Find which part the orientation belongs to, then add the corresponding magnitude to the bin in the histogram.
6. Each cell gives a 1*8 descriptor vector. Concatenating these vectors gives the 1*128 SIFT feature descriptor.

[Why are SIFT features better descriptors than the normalized patches?]
Because it uses 8 values to describe the features in a 4*4 window, filtering out much redundant information. In fact, the orientation and magnitude can be used to describe a window with any shape. With larger window, the SIFT descriptor is more efficient.

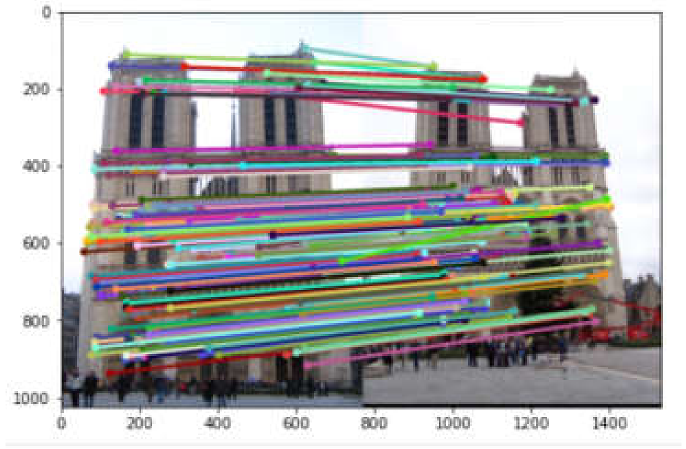
Part 4: SIFT feature descriptor

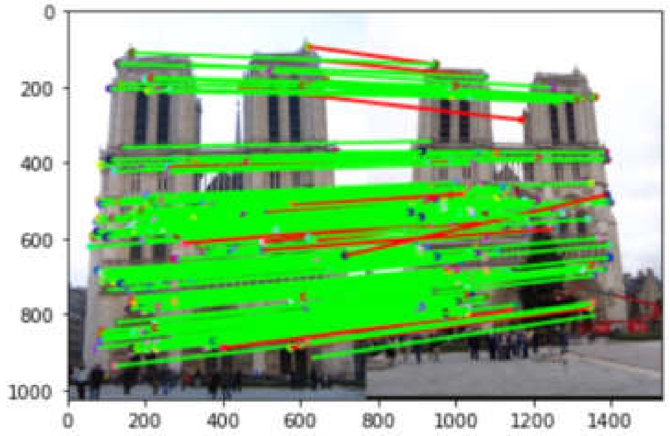
[Why does our SIFT implementation perform worse on the given Gaudi image pair than the Notre Dame image and Mt. Rushmore pairs?]

Because the scale is different. Harris corner is not scale invariant, and if the scale changes, a corner may become an edge. Since our SIFT feature descriptor does not include scale and rotation invariance feature, the SIFT feature descriptor does not perform well comparing with the Notre Dame image

Part 5: SIFT Descriptor Exploration

Describe the effects of changing window size around features. Did different values have better performance?
When you extend the window size. Firstly the result will be better, but later when the size is too large the result will be worse. Yes, appropriate larger window size will have better results. 2 times of current window size would have 93% accuracy.





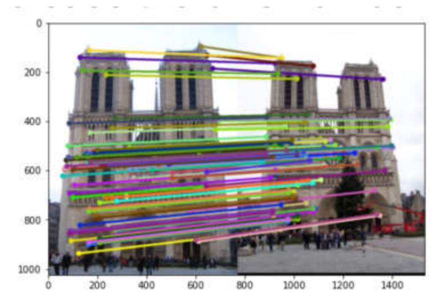
Part 5: SIFT Descriptor Exploration

Describe the effects of changing the number of local cells in a window around a feature? Did different values have better performance?
The result does not improved monotonically with the number of cells. No, from my experiments, 4 cells and 64 cells will have worse result. 16 cells: 83%., 64 cells 81%, 4cells 66%

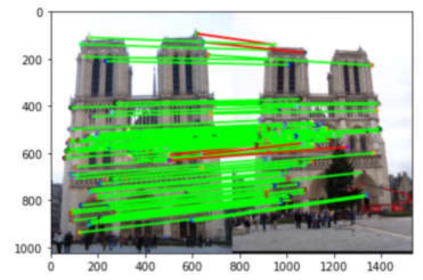
Part 5: SIFT Descriptor Exploration

Describe the effects of changing number of orientations (bins) per histogram. Did different values have better performance?

Increasing the number of orientations (bins) per histogram will improve the result at the beginning because we will have more refined scale. Yes, for 16 bins, the accuracy become 95%

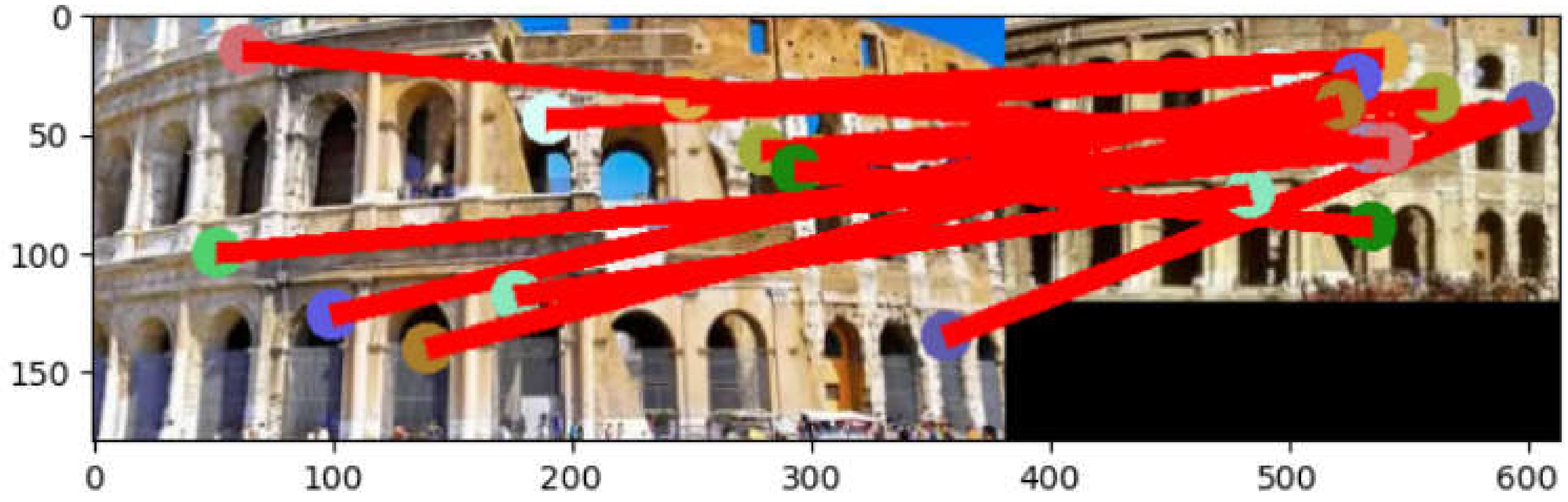


```
You found 101/100 required matches
Accuracy = 0.950495
```



Part 5: SIFT Descriptor Exploration

[insert visualization of matches for your image pair from proj2.ipynb here]



Part 5: SIFT Descriptor Exploration

[Discuss why you think your SIFT pipeline worked well or poorly for the given building. Are there any characteristics that make it difficult to correctly match features]?

Overall, I think the SIFT pipeline works well for architectural images. It works well because the two images have similar proportions and similar orientations. In addition, the brightness and other attributes of the image are almost the same. As a result, the SIFT pipeline can easily identify the matched features and works well.

The scale, light and symmetry are characteristics that make it difficult to correctly match features.
In the picture, the characters and trees in the background and lower part of the tag with different scale and light do not match well. At the same time, I have carefully observed that the two architectural pictures are actually mirror images of each other, so it is difficult for STFT to distinguish the left and right of the pictures with symmetric features, so SIFT recognition and matching features are also a little difficult.

Conclusion

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

Because Harris corner is not scale invariant, and if the scale changes, a corner may become an edge. It's rotation invariant, because rotation doesn't change the shape of the Angle, but the position of the corner isn't invariant, because the position changes with rotation.

To keep its rotation constant, I will choose a dominant direction and calculate the orientation histogram based on it.