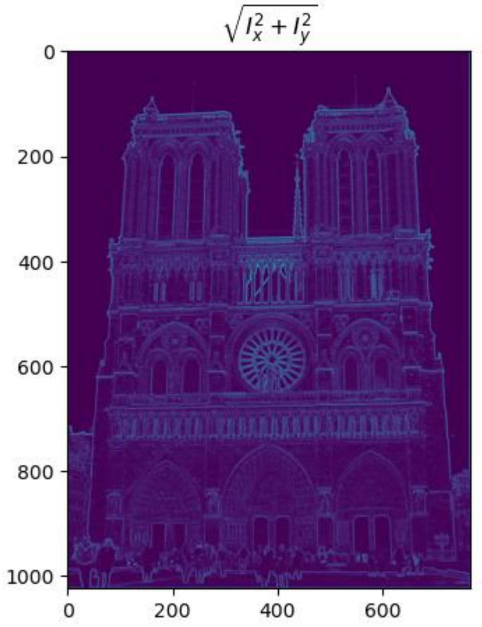
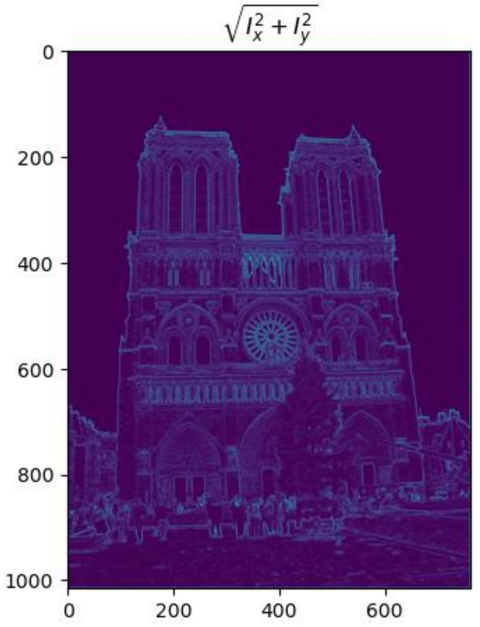
CS 6476 Project 2

Mengying Lin
mlin365@gatech.edu
mlin365
904016037

Part 1: Harris corner detector

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

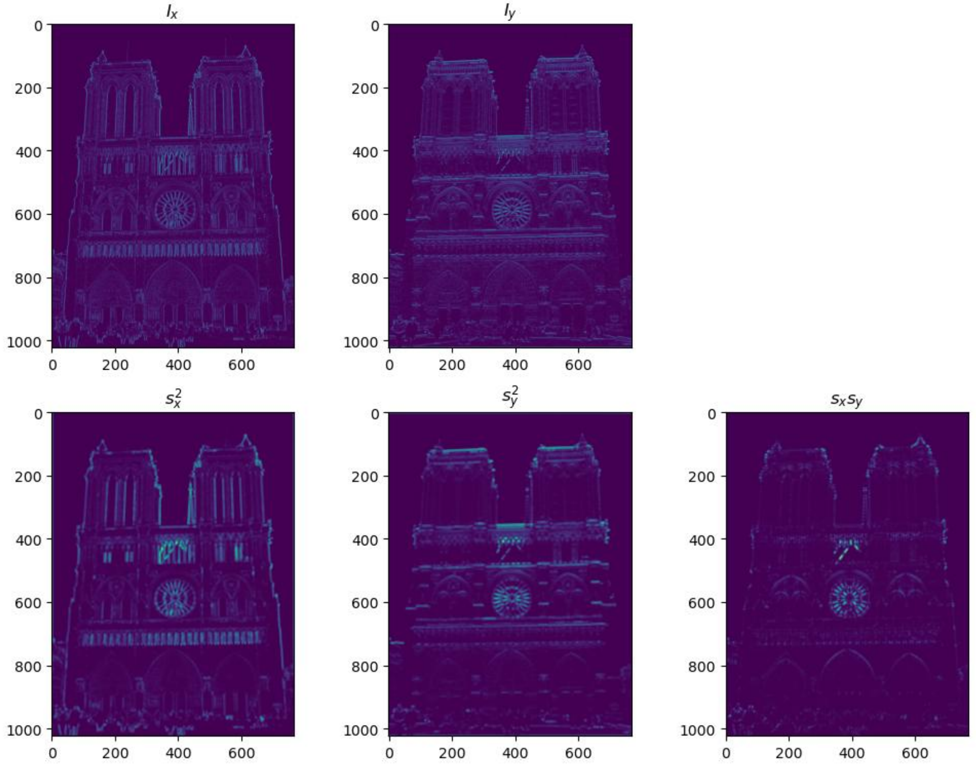




[Which areas have highest magnitude? Why?
The circular ros window, with its bright stone lines against a darker background, creates strong contrasts that result in high gradient magnitudes along the edges of the circular and radial patterns.

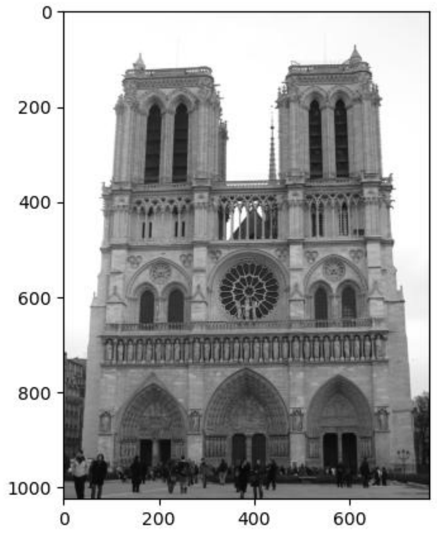
Part 1: Harris corner detector

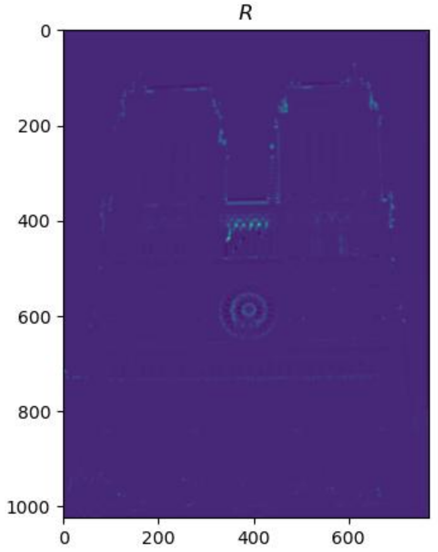
[insert visualization of I_x, I_y, s_x^2, s_y^2, $s_x s_y$ for Notre Dame image pair from proj2.ipynb here]



Part 1: Harris corner detector

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



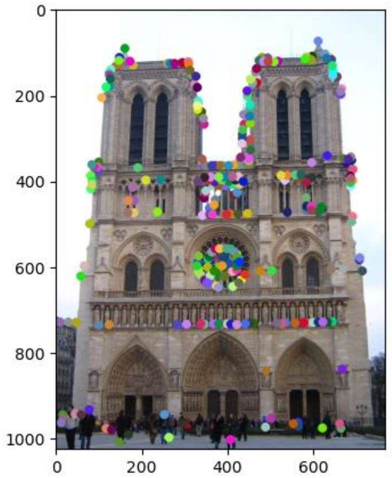


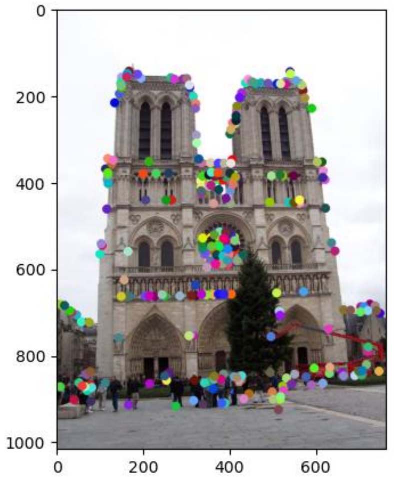
[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

Gradient is invarient to additive shifts but not multiplicative gain. This is because gradients measure the difference in intensity between neighboring pixels, which remains unchanged when a constant value is added to all pixels. However, a multiplicative change in contrast scales the intensity differences between pixels, affecting the magnitude of the gradient.

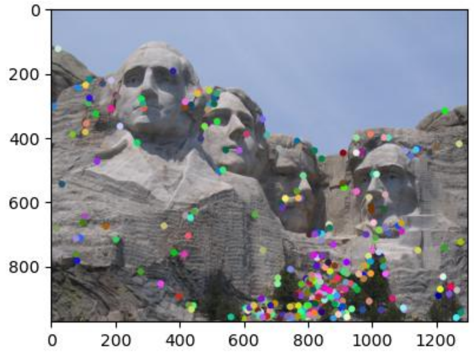
Part 1: Harris corner detector

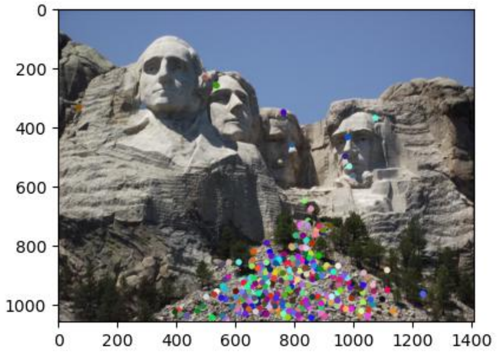
[insert visualization of Notre Dame interest points from proj2.ipynb here]





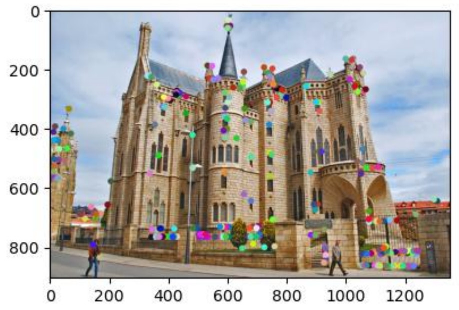
[insert visualization of Mt. Rushmore interest points from proj2.ipynb here]

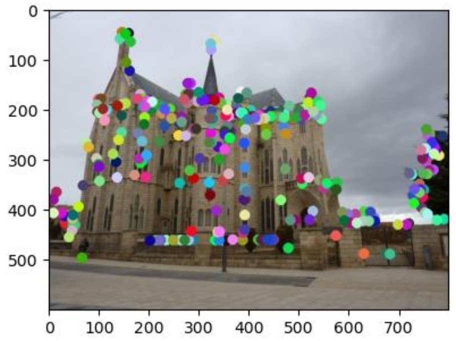




Part 1: Harris corner detector

[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:
Max pooling is a simple and fast operation, and can make the suppression more computationally efficient.
Disadvantages:
Max pooling relies on a fixed window size, which may not adapt well to objects of varying scales

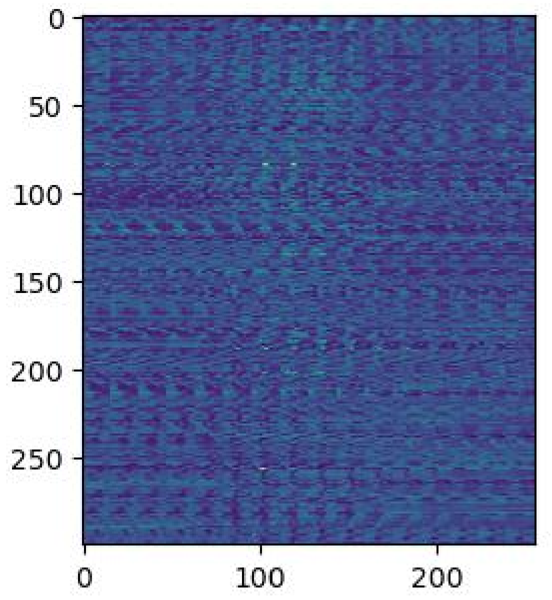
Part 1: Harris corner detector

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

The corners are usually the pixels where intensity changes significantly in both the x and y directions. The Harris corner detector leverages the second moments of each pixel and convolves them with a Gaussian kernel to obtain the response matrix. Eigenvalue analysis of this matrix helps classify regions as corners (both eigenvalues large), edges (one large, one small), or flat areas (both small).

Part 2: Normalized patch feature descriptor

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

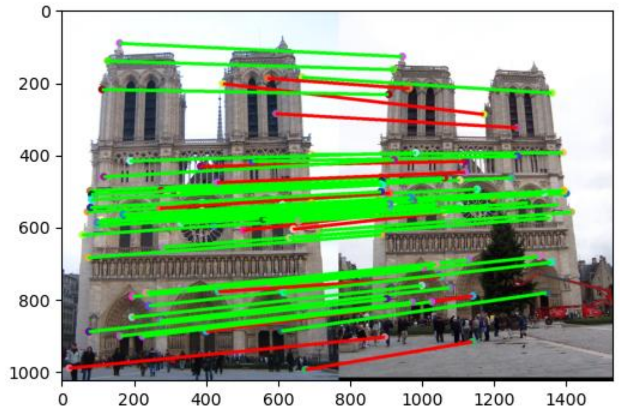


[Why aren't normalized patches a very good descriptor?]

Normalized patches are not ideal descriptors because they are sensitive to illumination changes and lack robustness to rotation and scale variations.

Part 3: Feature matching

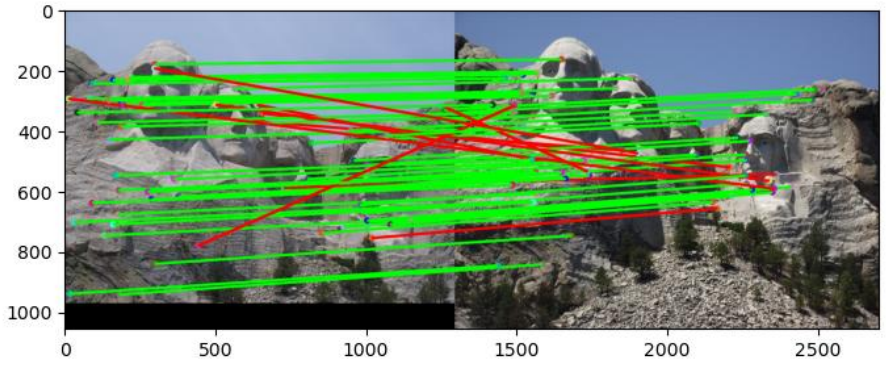
[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from proj2.ipynb here]



# matches (out of 100):64

Accuracy: 0.52

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

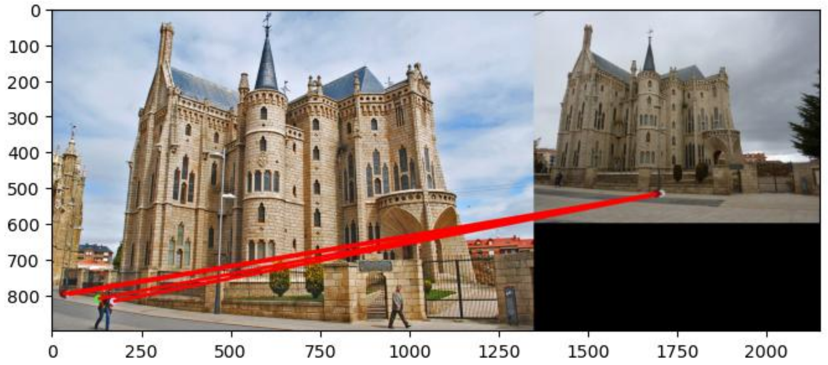


# matches: 61

Accuracy: 0.50

Part 3: Feature matching

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



# matches: 4

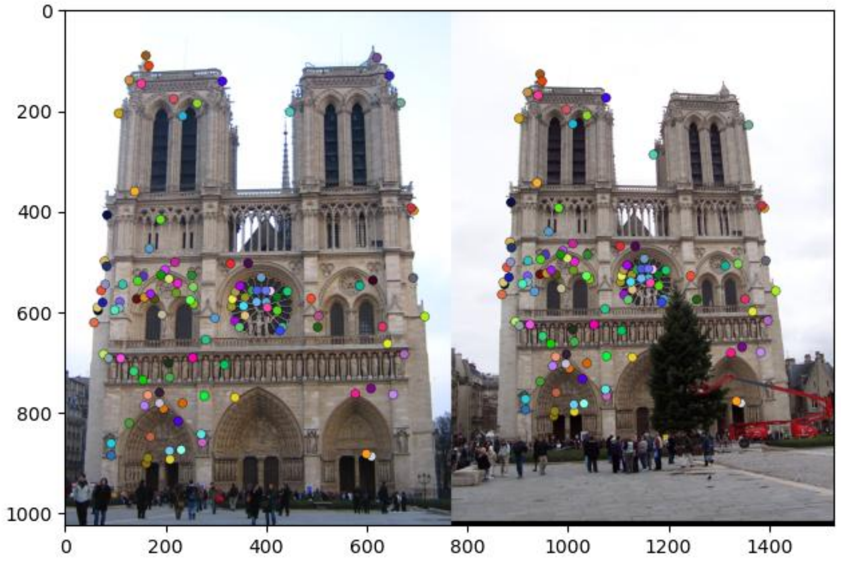
Accuracy: 0.00000

[Describe your implementation of feature matching here]

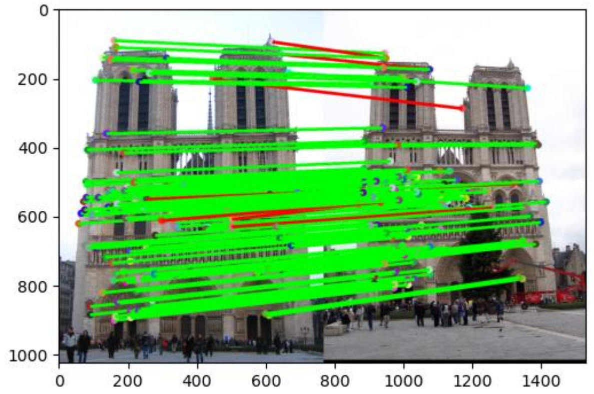
1. Calculate Pairwise Distances: Compute the pairwise distances between the features from the two images.
2. Find Closest Matches: For each feature in image 1, identify the closest and second closest features in image 2.
3. Apply Thresholding: Compute the ratio of the closest distance to the second closest distance. If the ratio is smaller than a predefined threshold, consider it a valid match.

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]

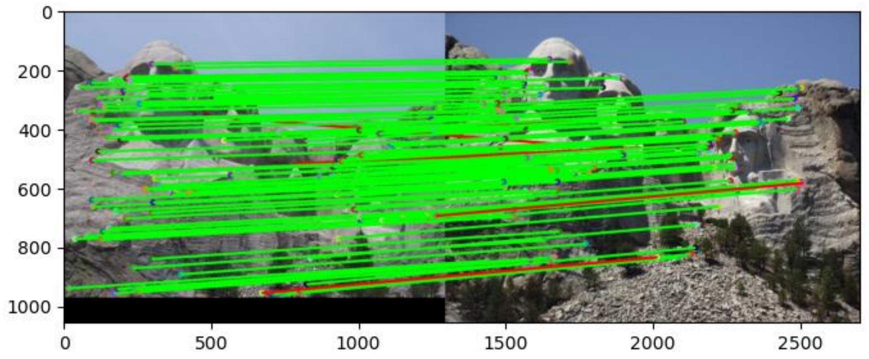


# matches (out of 100): 118

Accuracy: 0.940678

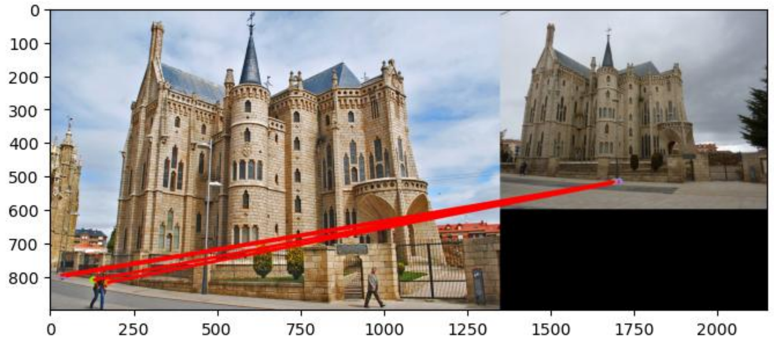
Part 4: SIFT feature descriptor

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



# matches: 108
Accuracy: 0.944444

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



# matches: 4
Accuracy: 0.0000

Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

1. Compute image gradients.
Obtain pixel-wise gradient orientation and magnitude.
2. For each pixel, take its neighboring pixels within a 4x4 grid cells and calculate the histogram of gradient distributions in 8 orientations, which is flattened into a feature vector.
3. Normalize the feature vector using the L2 norm and then take the square root.

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

SIFT features are superior to normalized patches because they offer scale and rotation invariance, making them more robust. Their descriptors contain gradient-based information, which is less sensitive to lighting changes and noise. Additionally, SIFT identifies reliable keypoints in high-contrast regions, while normalized patches often struggle in varying conditions and offer limited discriminative power.

Part 4: SIFT feature descriptor

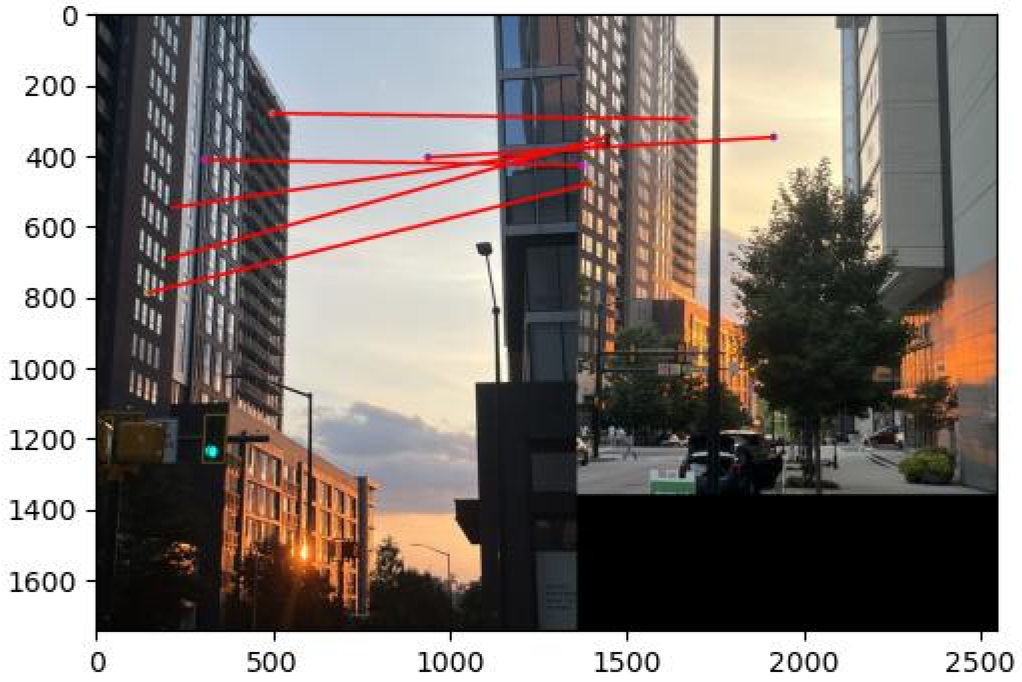
[Why does our SIFT implementation perform worse on the given Gaudi image pair than the Notre Dame image and Mt. Rushmore pairs?]

The Notre Dame and Mt. Rushmore images contain sharp and high-contrast edges which are ideal for SIFT keypoint detection, while the edges in Gaudi image pair are relatively low-contrast.

Also, there are significant scale difference between the two Gaudí images, and the descriptor’s scale and rotation invariance may not fully compensate for such variations.

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]?

The images lack the sharp, high-contrast edges or corners that SIFT relies on for keypoint detection. Additionally, the variations in scale and viewing angles between the images may be too significant for SIFT to handle effectively.

Conclusion

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

To ensure rotation invarience, we might need to compute the dominant orientation for each keypoint and then rotate the keypoint descriptor relative to this dominant orientation.
To ensure scale invarience, we need to detect keypoints at different scales using the Difference of Gaussians (DoG).