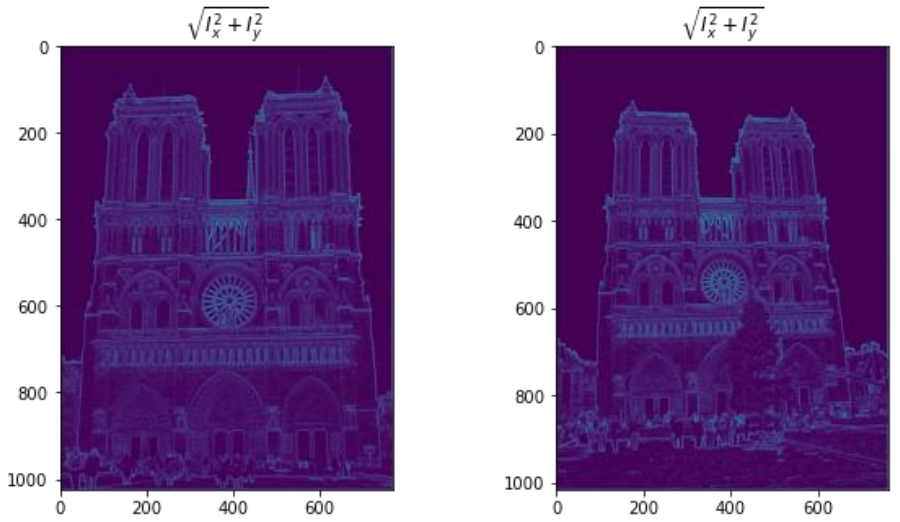
CS 4476/6476 Project 2

[Manan Patel]
[mpatel608@gatech.edu]
[mpatel608]
[903748003]

Part 1: Harris corner detector

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



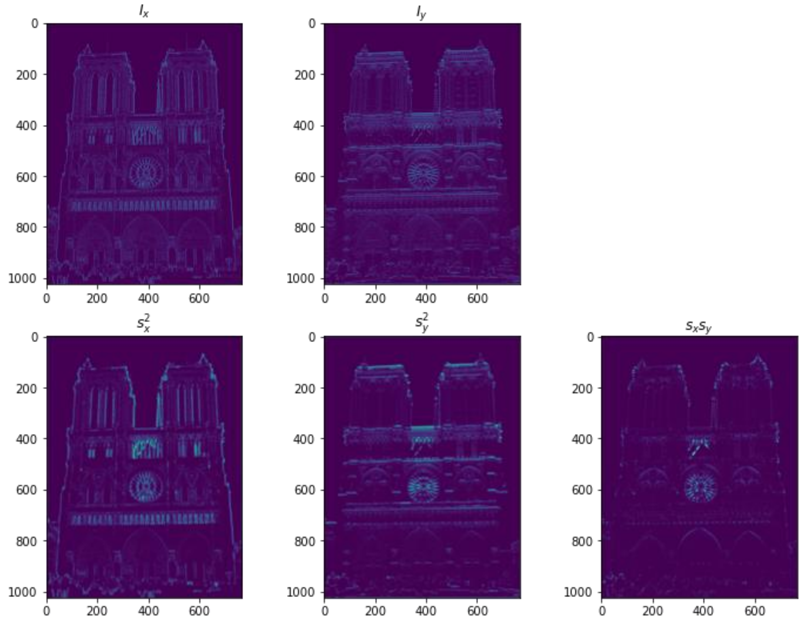
[Which areas have highest magnitude? Why?

The edges which are slim and edges which are not particularly straight have the highest magnitude.

- The slim edges because there is a high change in pixel values
- The non-straight edges because the total of change in X and Y would be higher than edges which have gradient only in one direction

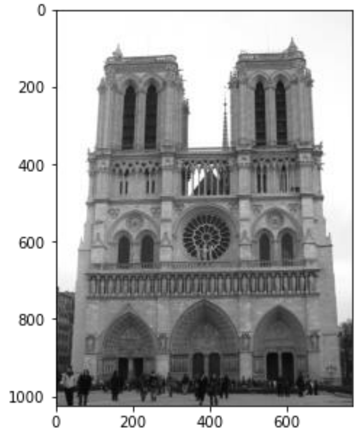
Part 1: Harris corner detector

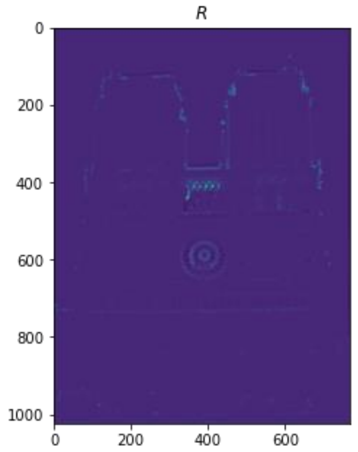
[insert visualization of I_x, I_y, s_x^2, s_y^2, $s_x s_y$ for Notre Dame image pair from proj2.ipynb here]



Part 1: Harris corner detector

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





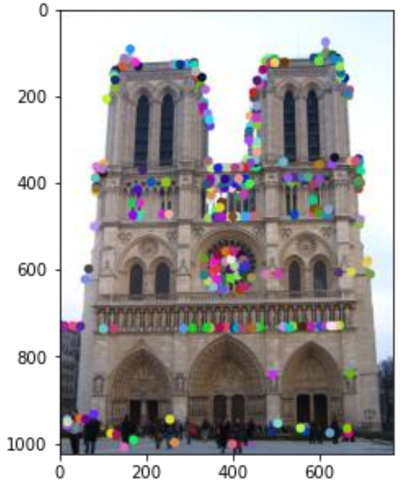
[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

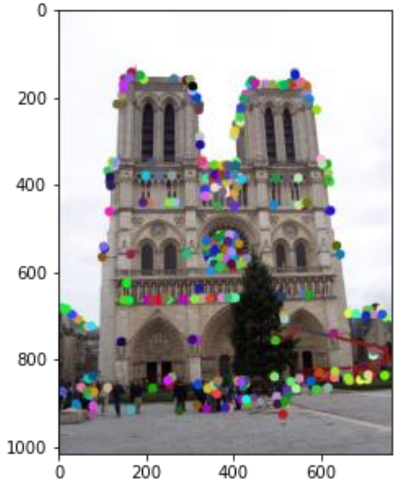
- The gradient features are invariant to additive shifts since we are taking derivatives.
- However, they are variant to gain since the derivative is doubled if the gain is 2.

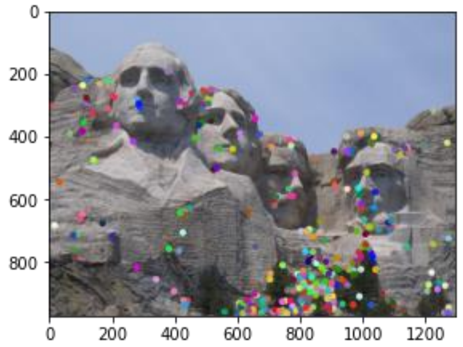
Part 1: Harris corner detector

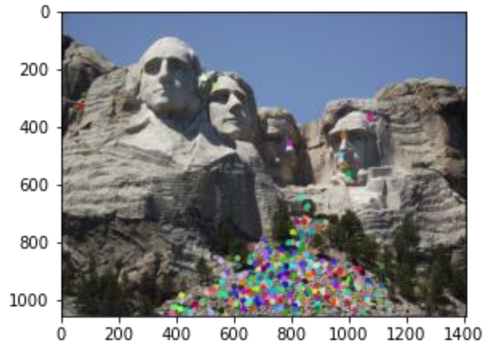
[insert visualization of Notre Dame interest points from proj2.ipynb here]

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



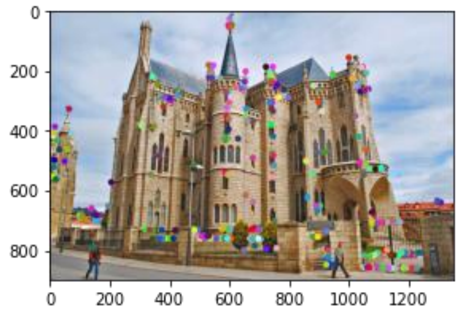


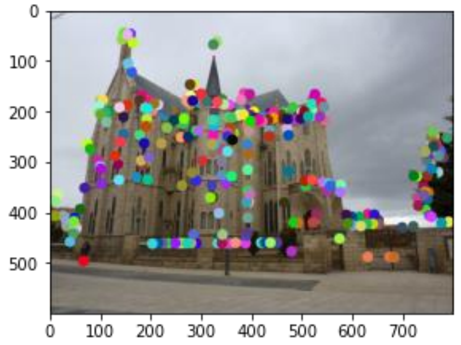




Part 1: Harris corner detector

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





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

- The advantage is that it keeps the best interest points and ignores the ones with low scores exhaustively
- However, this might result in overfitting to a major feature in an image and filter out the minute details.

Part 1: Harris corner detector

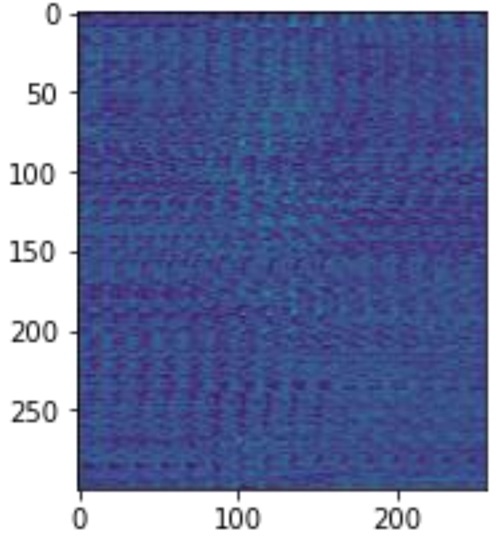
[What is your intuition behind what makes the Harris corner detector effective?]

- The harris corner detector is effective because we not only consider the derivatives in two directions, but for a given patch, we look at the eigen values which represent the maximum change in a given patch.

- Using this feature, we are also able to filter out the edges which are not oriented in the direction in which the derivatives were taken, keeping the patches which are most “cornery”.

Part 2: Normalized patch feature descriptor

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

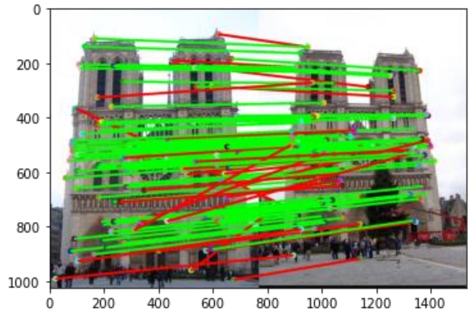


[Why aren't normalized patches a very good descriptor?]

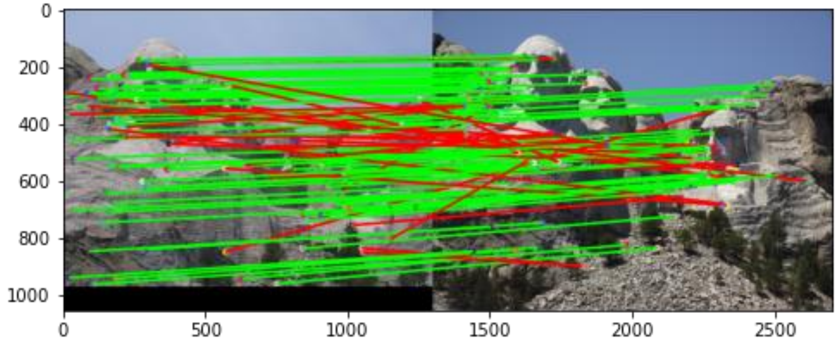
- The purpose of a descriptor is to distinguish between the interest points. However, if we use normalized patches, they are not invariant to affine and intensity transforms.
- Thus, even if two patches are “close”, it is not a good heuristic to use to match interest points.

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]



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

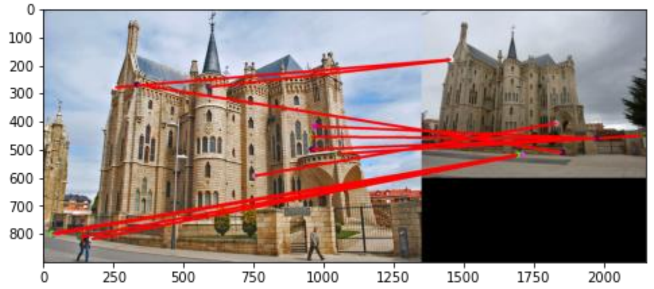


# matches (out of 100): [107]
Accuracy: [77]

# matches: [107]
Accuracy: [73]

Part 3: Feature matching

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



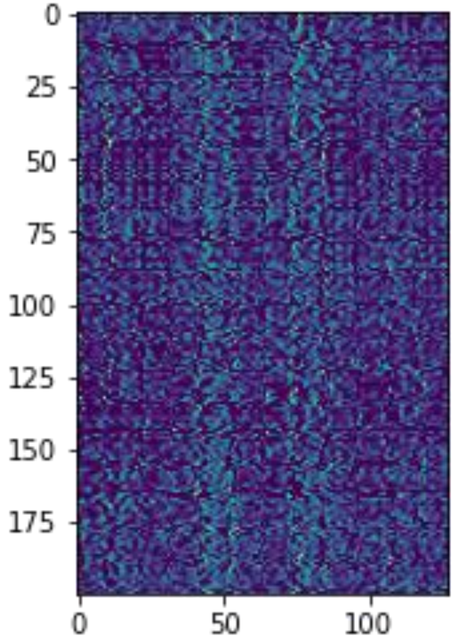
# matches: [12]
Accuracy: [0]

[Describe your implementation of feature matching here]

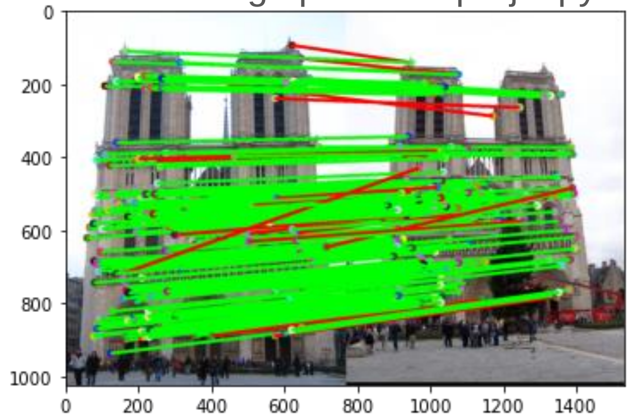
- Get the image gradients in X and Y directions using SOBEL filter for each pixel
- Calculate the second moments for each pixel
- Calculate the k scores
- Perform non-maximum suppression over a batch of 7x7 pixels throughout the image
- Select the first k interest points based on the k scores
- For the descriptor, take a 16x16 patch of pixel to be the dimensions used for comparison
- Calculate Euclidean distance between interest points in images and select value based on NNDR ratio test

Part 4: SIFT feature descriptor

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



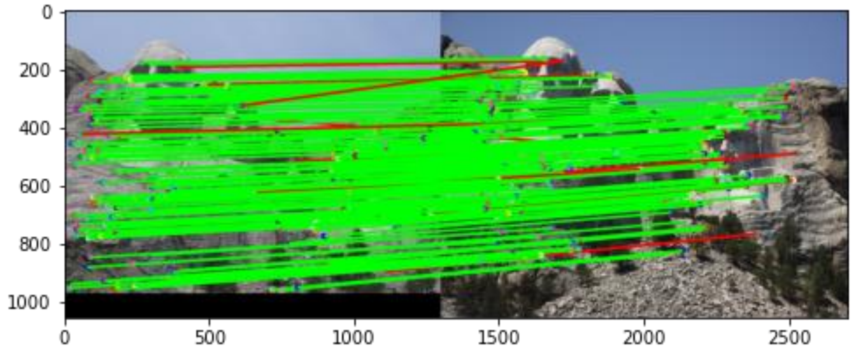
[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from proj2.ipynb here]



# matches (out of 100): [197]
Accuracy: [91.88]

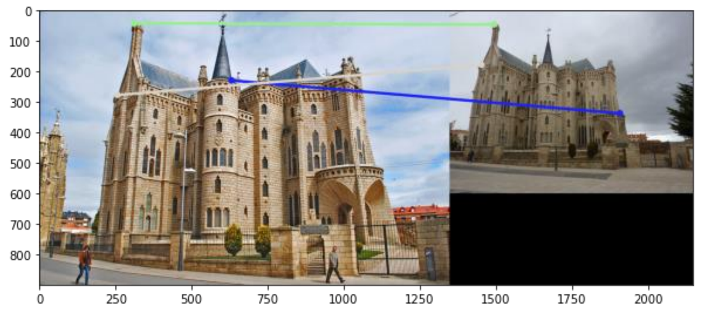
Part 4: SIFT feature descriptor

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



# matches: [178]
Accuracy: [93.26]

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



# matches: [3]
Accuracy: [0]

Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

- First a feature width (16x16), bin size(4x4) and histogram size (8) is chosen
- For every interest point, calculate the orientation of derivative from Ix and Iy generated using SOBEL filter
- For every pixel in every bin, assign it to a cell in histogram based on the orientation and magnitude of Ix and Iy as weights
- Generate a 128 dimension feature vector describing the descriptor

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

- Invariance to orientation: We take into account the direction of derivative while describing our feature vector.
- Invariance to displacement: Although not done in this project, we can weight each derivatives contribution to histograms in different bins as well to have a more robust implementation

Part 4: SIFT feature descriptor

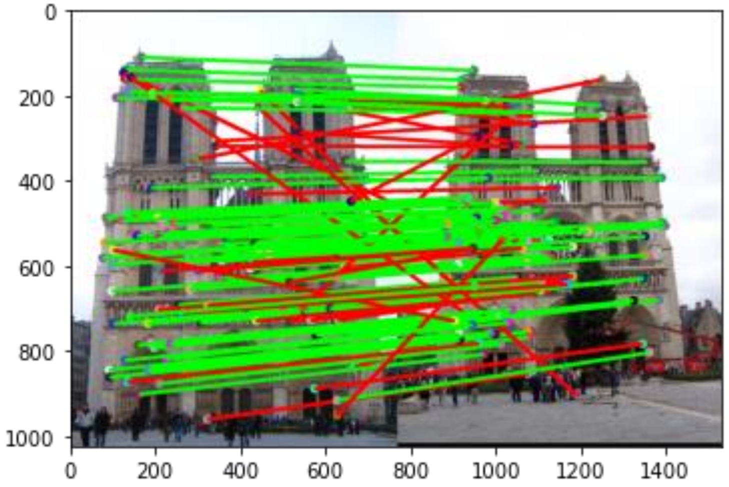
[Why does our SIFT implementation perform worse on the given Mt. Rushmore and Gaudi image pairs than the Notre Dame image pair.]

- In my implementation, SIFT actually performed slightly better for Mt. Rushmore (93% accuracy) than Notre Dame (92% accuracy). This might have happened because the a and b images for Notre Dame are significantly cluttered and have varied number of external elements than in Mt. Rushmore, albeit Mt. Rushmore being more scaled up. One would expect Mt. Rushmore to perform worse since it is scaled up and our implementation did not consider weighting the contributions to all the bins. However, due to picture being more consistent than Notre Dame, it performed better.
- Gaudi image performed worse because the second image is scaled as well as displaced majorly. Also, there is a significant difference in lighting conditions between the two images which gives rise to low accuracy.

Part 5: SIFT Descriptor Exploration

Describe the effects of changing window size around features. Did different values have better performance?

Window size = 8, accuracy = 67.5%



- As can be seen, decreasing the feature_width by half, reduces the accuracy significantly.
- Thus, however, increasing the window size to 20 did not result in significant accuracy improvement.
- Thus, we can say that it levels off.

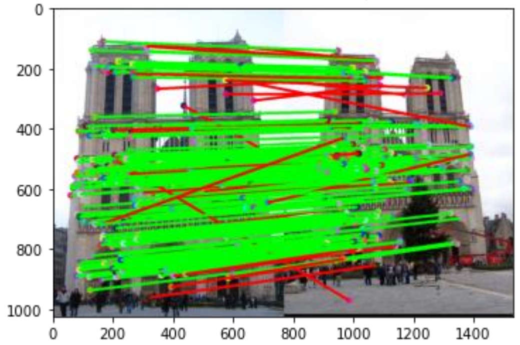
Part 5: SIFT Descriptor Exploration

Describe the effects of changing the number of local cells in a window around a feature? Did different values have better performance?

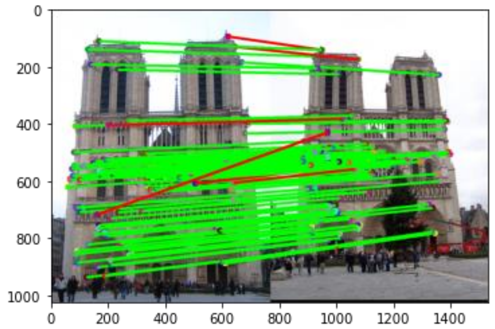
Part 5: SIFT Descriptor Exploration

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

Bins = 4, accuracy = 85%



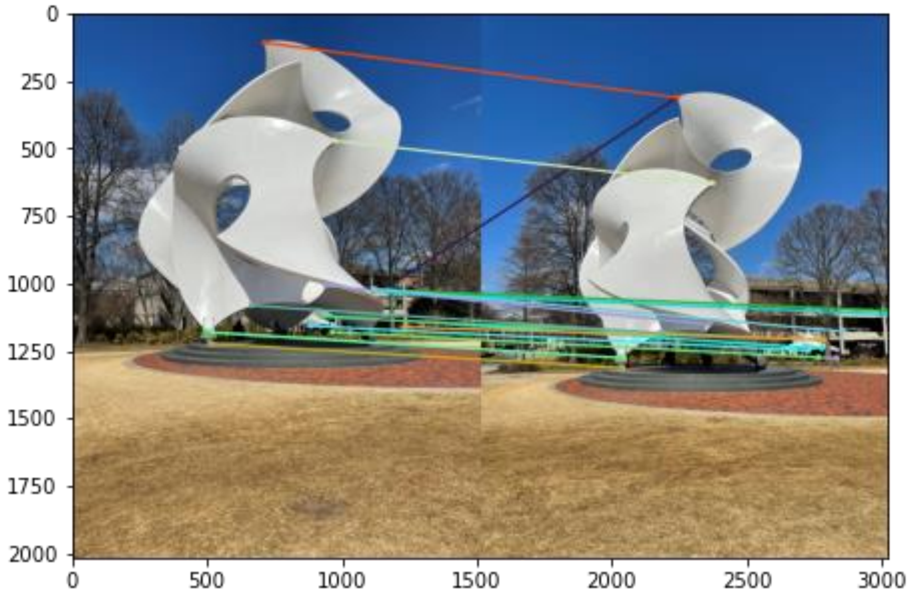
Bins = 16, accuracy = 94%



- Thus it is evident that increasing the bin size significantly increases the accuracy

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]?

- In my case, it seems SIFT performs well enough given the limited number of distinct corners in the image with only a couple of mis-matches.
- However, if our metric is to maximize the number of matches, then it certainly does poorly in that respect due to lack of distinct corners in the image.
- The change in perspective in the second image also covers the building in the background which leads to major alteration in the image composition, as a result, reducing the number of overall matches.

Conclusion

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

- To be co-variant to rotation, we would have to keep track of an up-vector which is the direction in which we find the maximum magnitude. Now, depending on the up-vector, create the descriptor with respect to this direction in both the images.

- For scale-invariance, we can have the magnitudes be weighted across the bins as well. The farther away from a bin, the less its influence to other bins and so on.