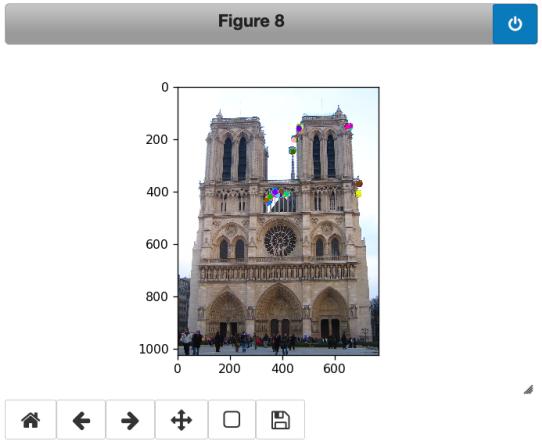
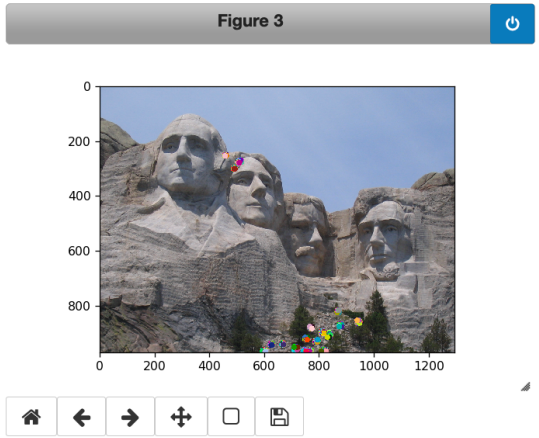
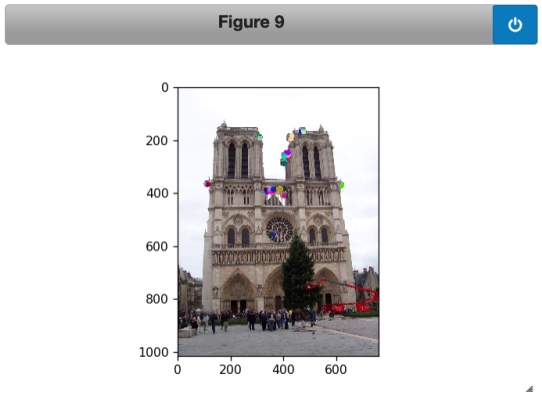
CS 4476 Project 3

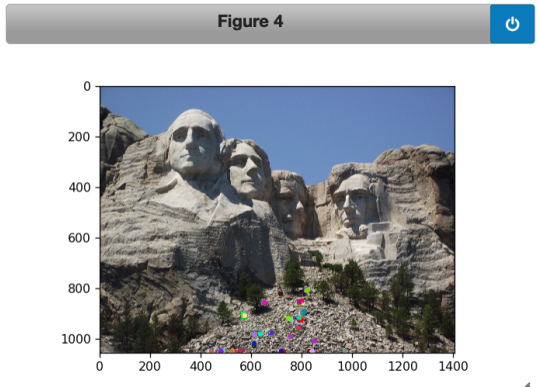
Sreyans Sipani
ssipani6@gatech.edu
ssipani6
903310164

Part 1: Harris Corner Detector

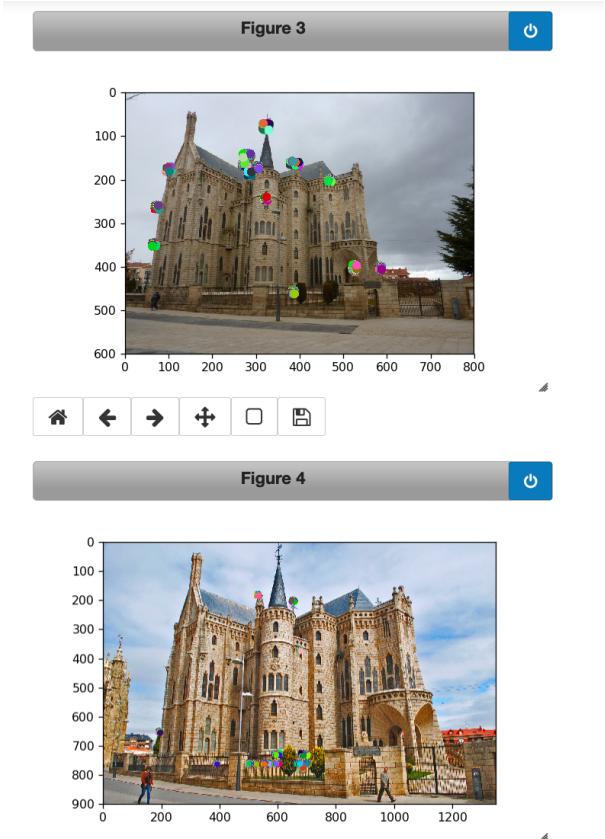








Part 1: Harris Corner Detector



Part 1: Harris Corner Detector

We do a step by step comparison of our implementation and the Harris implementation –

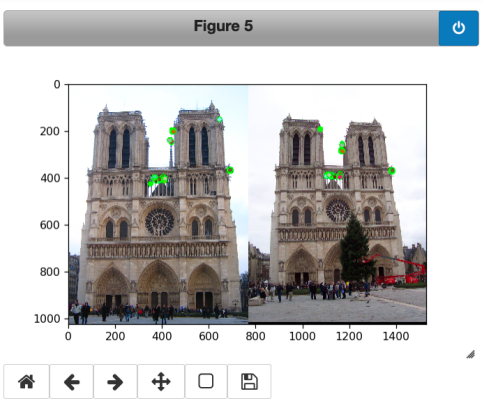
1. Convert image to grayscale (Same as Harris)
2. Compute vertical and horizontal gradients of image by convolving with the Sobel filter. [get_gradients]
3. Then we compute the second moments by multiplying and convolving with a gaussian kernel. [second_moments]
4. Then we compute the R by using the formula – det(A) – alpha * (trace(A)^2). [corner_response]
   We can use the formula – det(A) / trace(A)
5. Then we perform non-max suppression. We remove all values below median. Then we run a maximum filter on our matrix. [non_max_suppression
6. We set our threshold to zero. Then we keep the points with confidence above the threshold. [get_interest_points]
7. Then we remove the points near the border. [remove_border_vals]

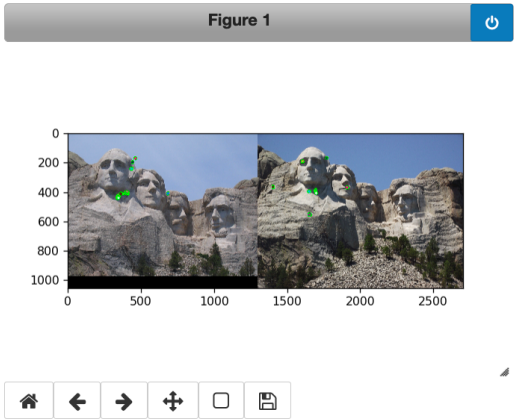
Part 2: Sift

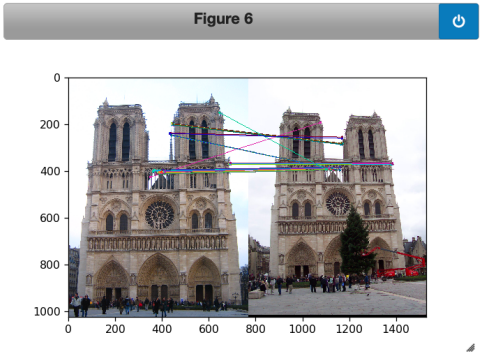
Implementation and SIFT's implementation –

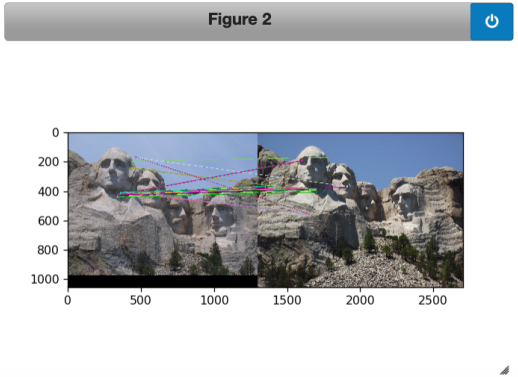
1. Get magnitude (= (dx^2 + dy^2)^(1/2)) and orientation (= arctan2(dy/dx)) of the gradients
2. Get feature vectors (128) by weighting the histogram of orientations with the magnitude into 8 bins [get_feat_vec]. We started by taking the 16x16 window around the interest point (the original implementation convolves it with a gaussian filter). In the actual SIFT implementation, each of the original 256 weighted gradient magnitudes is softly added to 2 x 2 x 2 histogram bins using trilinear interpolation and apply Hough transforms or local histogram equalization. On the other hand, we apply a very naïve histogram approach.
3. We also normalize to unit length and raise it to a power of 0.9
4. Use above function to generate every feature. [get_features]

Part 3: Feature Matching

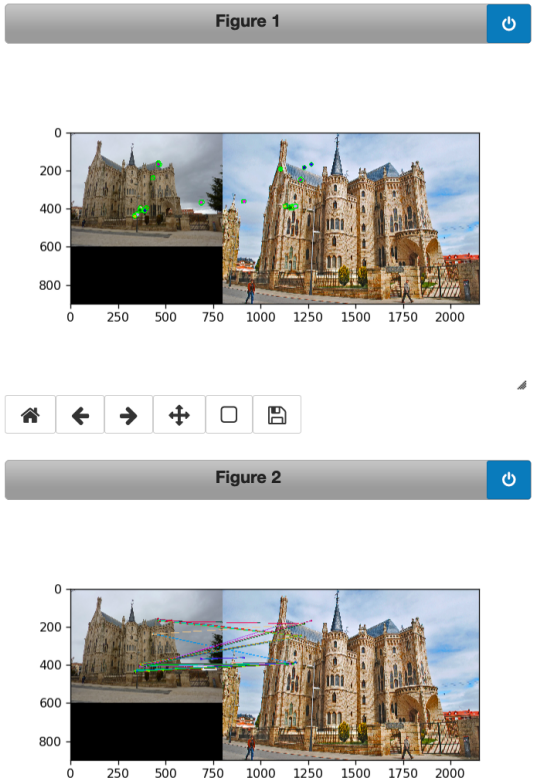






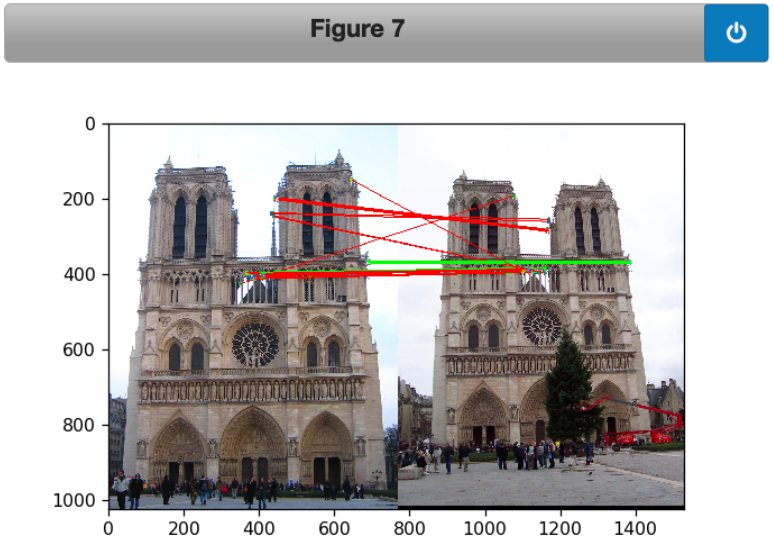


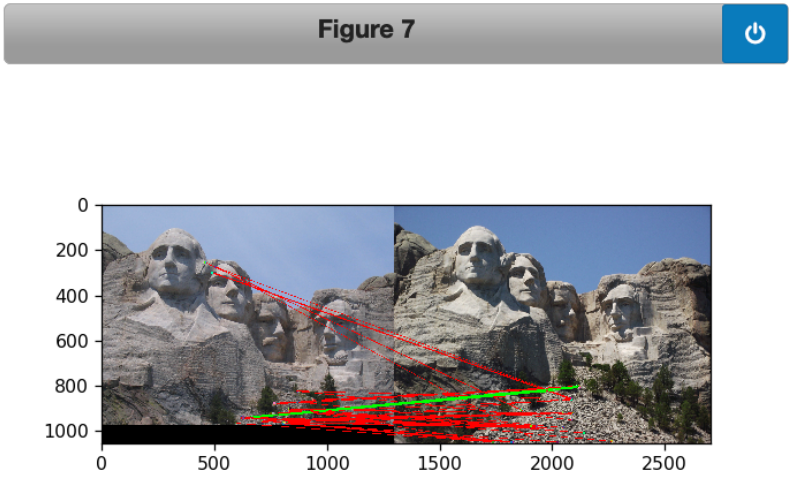
Part 3: Feature Matching



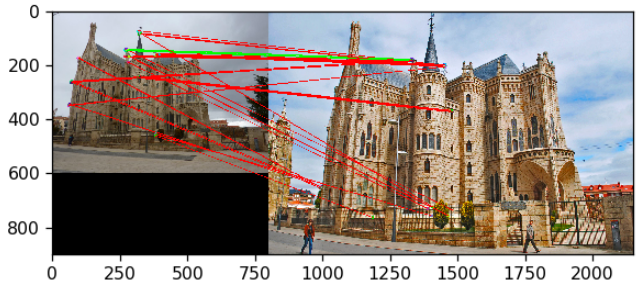
1. Calculate pairwise distances between all features of both images
2. Then we take the closest 2 features to every feature and perform the ratio test (= closest distance / second closest distance)
3. The confidence in the inverse of the ratio.
4. Then we output the matches based on decreasing order with their confidences.

Results: Ground Truth Comparison





Results: Ground Truth Comparison



Notre Dame – 35%
Rushmore – 40%
Gaudi – 21%

Changing the subgrid would affect the dimension of our feature vector i.e. larger the subgrid, lesser the dimensions. Therefore there would be lesser information in the features and harder to make accurate matches.

Extra Credit: Hyperparameter Tuning part 1

<Insert images of the ground truth correspondence and their corresponding accuracies for varying sigma in the second moments [3, 6, 10, 30] >

When changing the values for large sigma (>20), why are the accuracies generally the same?

Extra Credit: Hyperparameter Tuning part 2

<Insert images of the ground truth correspondence and their corresponding accuracies for varying feature width in the SIFT [8, 16, 24, 32] >

What is the significance of changing the feature width in SIFT?

Extra Credit: Accelerated Matching

<Insert Runtime/Accuracy of your faster matching implementation. What did you try and why is it faster?>