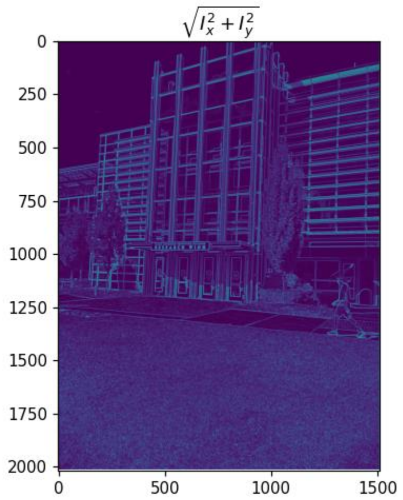
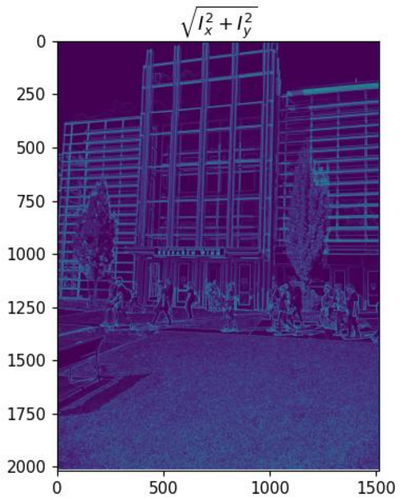
CS x476 Project 4

Zhaodong Yang
halyang@gatech.edu
zyang645
903748903

Part 1: Harris corner detector

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



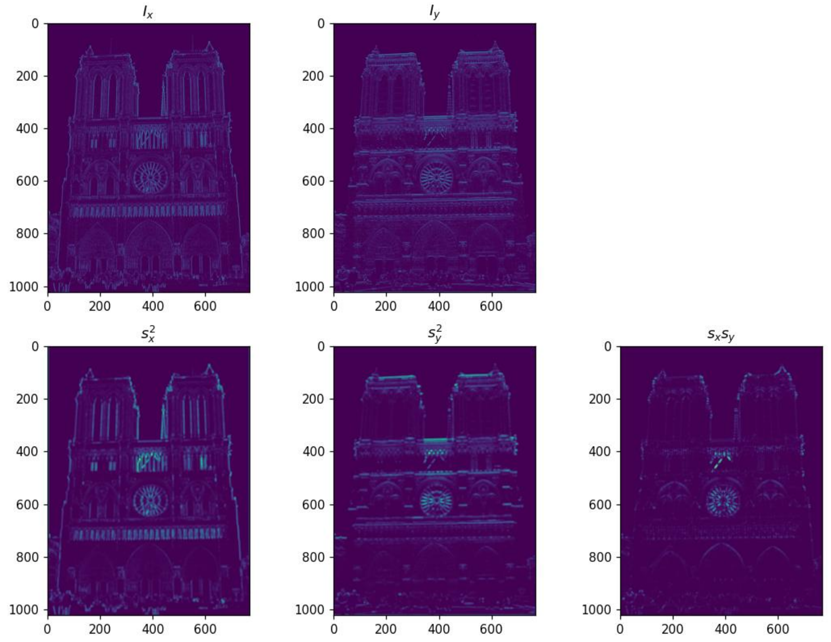


[Which areas have highest magnitude? Why?]

The steel frame of the building and the outline of the building have the highest magnitude. Because these are the contours in the image, where the pixel values vary greatest. Which means the pixel value derivatives of these areas are the 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]

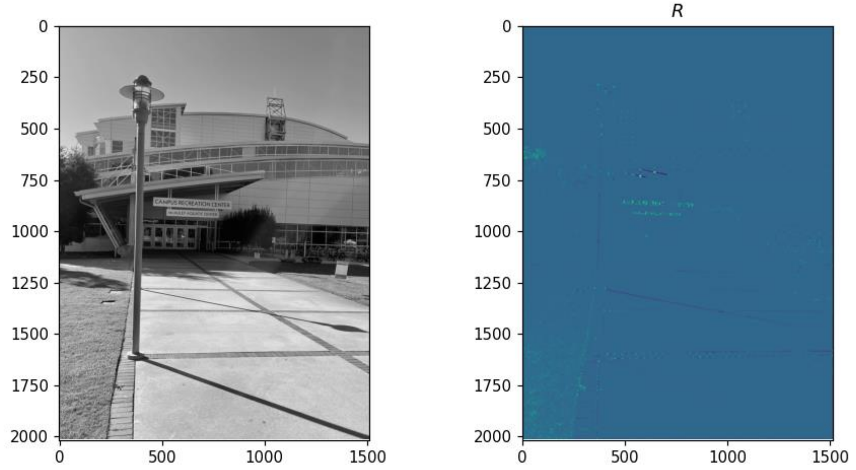


Part 1: Harris corner detector

[Describe your implementation of generating the Harris cornerness score response map]
First, I used compute_image_gradients() to apply sobel kernel on image to get the image gradients. Then I used second_moments() to apply gaussian convolution to get the second moments of the image. In the end, I used vectorized equation R = sxx * syy - sxsy * sxsy - alpha * (sxx + syy) * (sxx + syy) to get the Harris response map.

Part 1: Harris corner detector

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

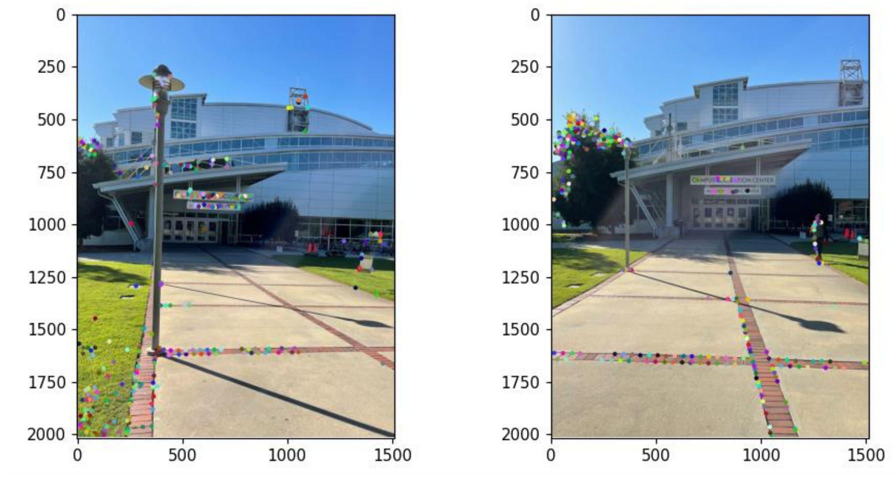


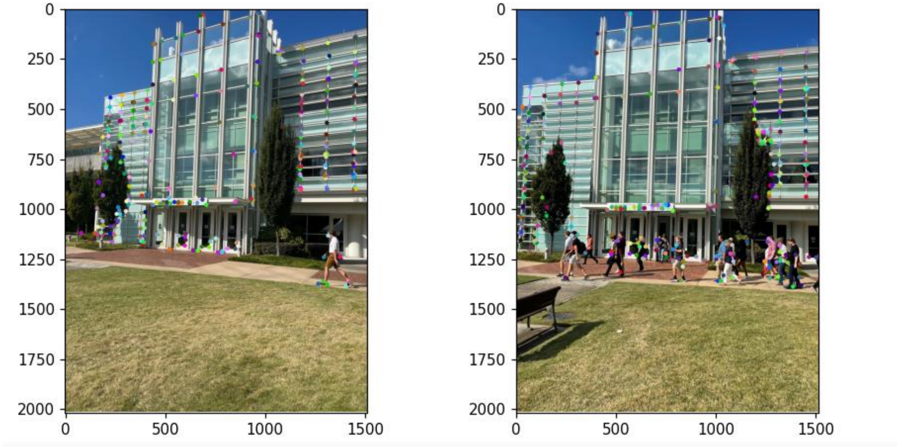
[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not?]
Gradient features is not invariant to both additive shifts and multiplicative gain. Because the when brightness of an image becomes bigger, the gradients becomes bigger. When contrast of an image becomes bigger, the gradients becomes bigger as well.

Part 1: Harris corner detector

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

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





Part 1: Harris corner detector

[What are the advantages and disadvantages of using maxpooling for non-maximum suppression (NMS)?]
Using maxpooling for non-maximum suppression can select corners from all areas evenly. And maxpooling can avoid repeatedly selecting corners from the same neighbor area. But When several important corners are too close, maxpooling will ignore most of them and just pick one with highest confidence from them, which may lead to the ignorance of some important information.

Part 1: Harris corner detector

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

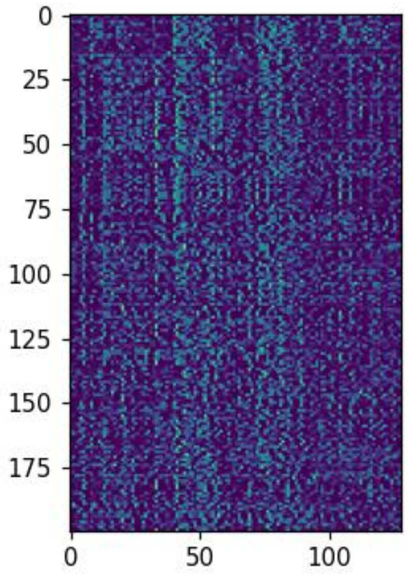
The threshold we choose when threshold globally on the score response map.

The kernel size of maxpooling.

The sigma and kernel size of gaussian kernel when calculating second moments.

Part 2: SIFT feature descriptor

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



[Describe your implementation of SIFT feature descriptors here]
At first, I calculated the magnitude and orientation of gradients. Then I wrote get_gradient_histogram_vec_from_patch(), which use window magnitudes and window orientations, dividing each of them to 4*4 grids. Then I compute histogram of orientation from each of the cell from 4*4 grid, with the magnitudes as the weight of histogram. Then I applied this procedure on every selected corners in the image.

Part 2: SIFT feature descriptor

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

Because we didn’t determine a main direction when calculate orientation histogram. And we didn’t use LoG pyramid to determine the scale to use.

We need to determine a main direction when calculate orientation histogram. And we need to use LoG pyramid to determine the scale to use.

Part 2: SIFT feature descriptor

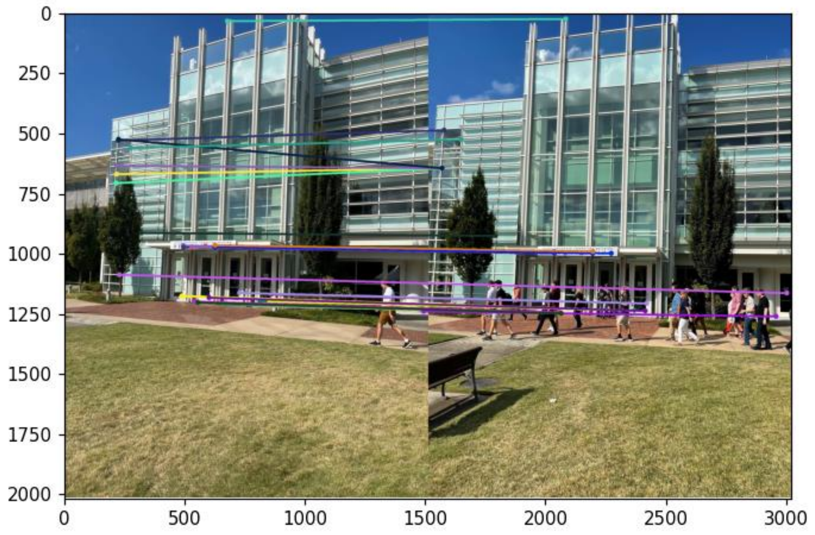
[Why are SIFT features better descriptors than simplying normalizing image intensities in a local window?]

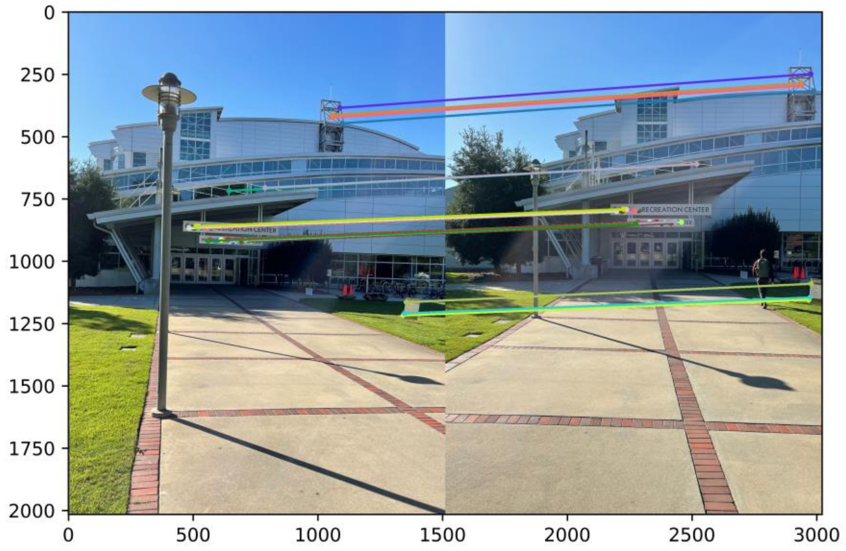
Because SIFT is additive-shifts-invariant, scale-invariant, and rotate-invariant.

Part 3: Feature matching

[insert visualization of matches for Klaus image pair here with num_pts_to_visualize = 30]

[insert visualization of matches for CRC image pair here with num_pts_to_visualize = 30]





Part 3: Feature matching

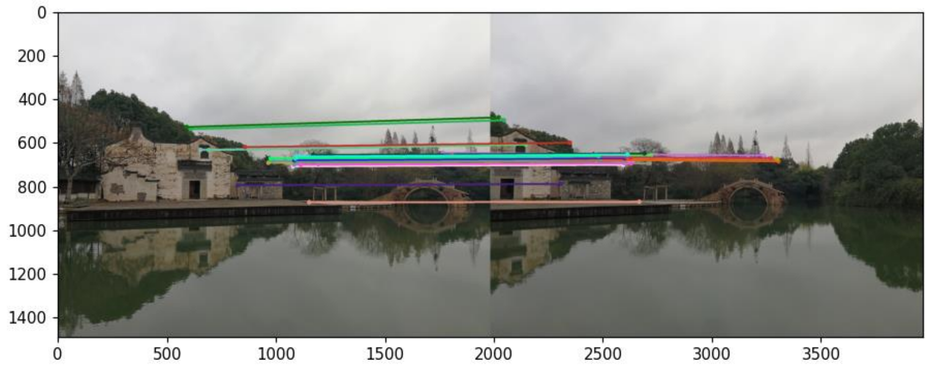
[Describe your implementation of feature matching here]
At first, I wrote compute_feature_distances() to calculate the distance between the features of the two images. Then I used torch.topk to found out the 2 smallest distance for every feature1. I calculated the nndr to determine if I should keep this match using a nndr threshold.

[Look at some of the mismatched features in your picture. Why might this have occurred?]

Some features are just very similar. For instance, the boundary of sky and building are all very similar.

Part 4: SIFT feature descriptor (Extra Credit)

[insert visualization of matches for your own image pair here]



What makes our feature matching pipeline work well or poorly for your image pair?

Because we are using sift descriptor, which is invariant to scale. And the NNDR matching also makes it perform well.