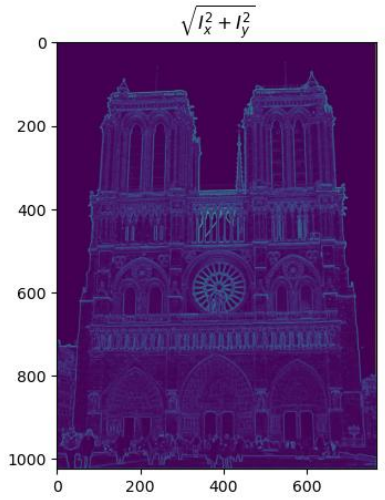
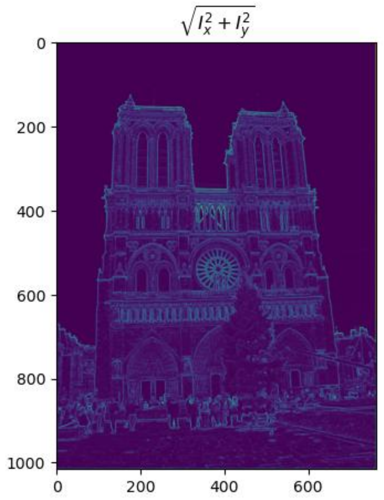
Warning: Do not delete slides.
This includes extra credit slides and any problems you do not complete.
All problems, including extra credit, must be assigned to a slide on Gradescope. Failure to follow this will result in a penalty.

CS 6476 Project 2

Auryn Yamamura
ayamamura6@gatech.edu
ayamamura6
904154249

Part 1: Harris corner detector





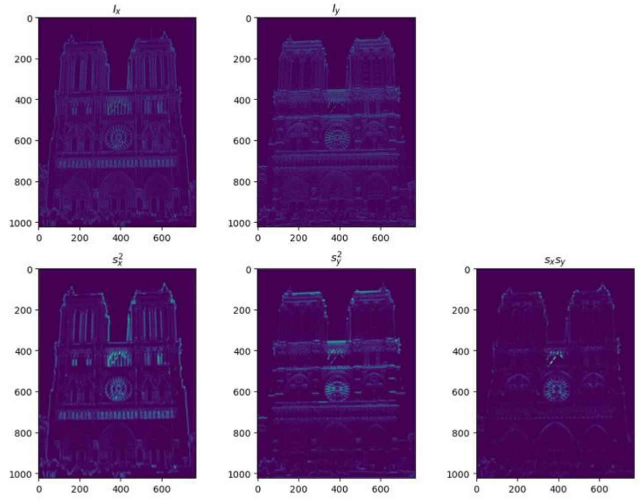
$\sqrt{I_x^2 + I_y^2}$ is the magnitude of the intensity gradient – if this value is large at some point, that tells us that said point is a possible candidate to be a Harris Interest Point (since we want large changes in both x, y-directions).

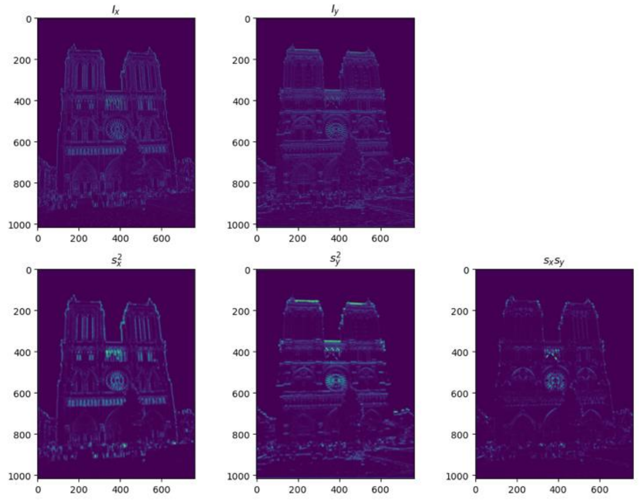
Arguably the edges, the circular window, and especially the arches above said window have the highest gradient magnitudes.

The edges have a high magnitude because of the large intensity change between the Notre Dame and the sky, whereas the window has a high magnitude because of notable changes in both the x, y-directions.

The arches have large intensity changes in both x, y-directions, which gives them the highest gradient magnitudes.

Part 1: Harris corner detector





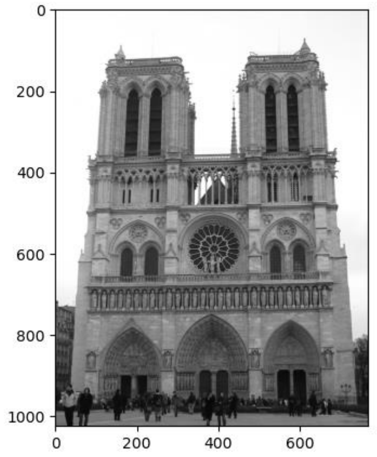
I_x, I_y are the image derivatives/intensity gradients in the x, y-directions. We use them to calculate the second moments S_{xx}, S_{xy}, S_{yy}, which we use for calculating the Harris Cornerness Score for each pixel.

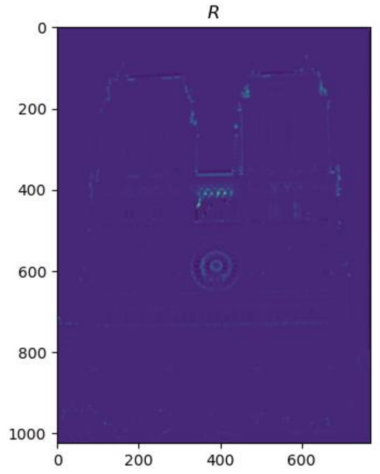
Part 1: Harris corner detector

We blur with a Gaussian filter (could also use a box filter) to get the values of I_x^2, I_y^2, $I_X I_y$ over a small neighborhood (i.e. a window) around each point in our image.

The Gaussian filter also has the added bonus of reducing high-frequency noise in our image derivatives. (Although any blurring filter will work).

Part 1: Harris corner detector





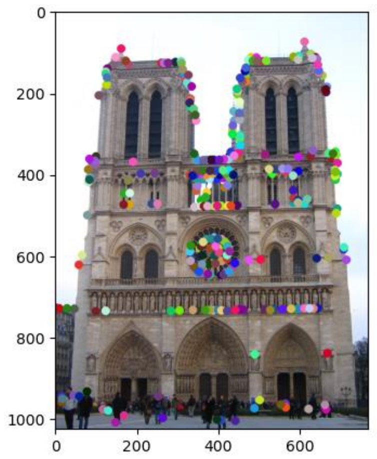
Gradient features are invariant to brightness shifts but not contrast shifts. Suppose we had an affine intensity change $I' = aI + b$ (i.e. both a brightness and contrast shift), then:

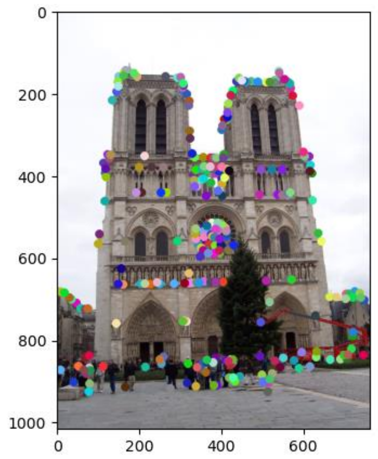
$$I'_x = aI_x, \qquad I'_y = aI_y$$
$$M' = a^2 M$$
$$\lambda'_i = a^2 \lambda_i \implies R' = a^2 R$$

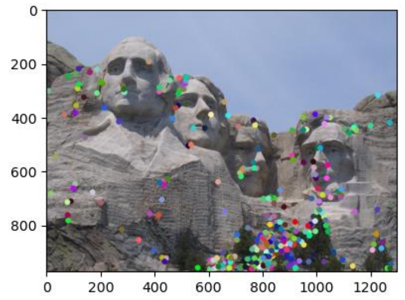
We can see that the brightness shift is ignored when taking the image derivatives while the contrast shift remained, eventually impacting our cornerness score.

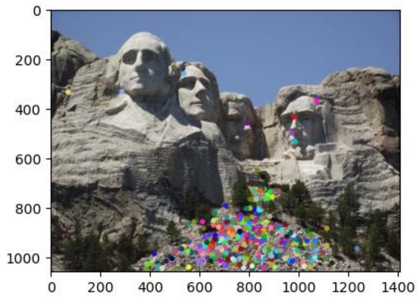
This is significant because it changes our probability of detecting a corner (provided we use thresholding to obtain our interest points).

Part 1: Harris corner detector

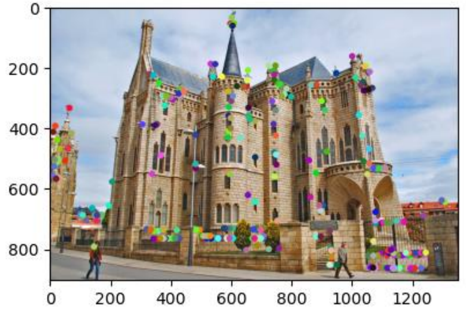


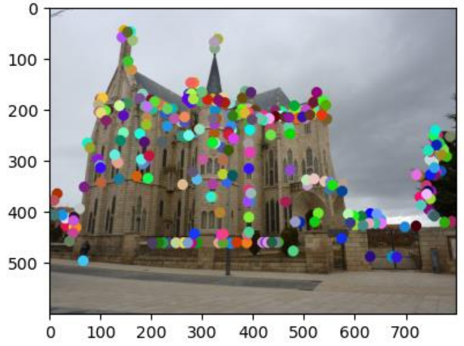






Part 1: Harris corner detector





Maxpooling will always pick the highest value point over a given window, which is needed for finding the local maxima in our image.

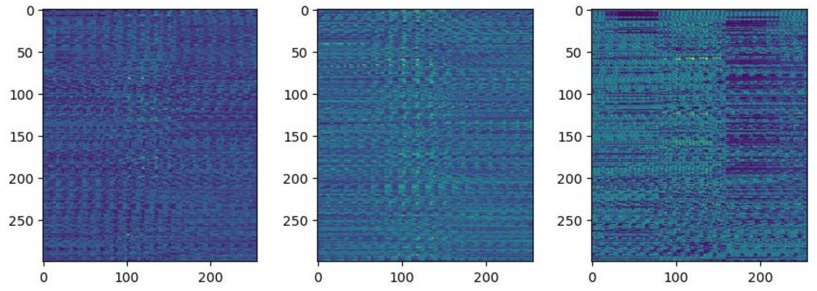
However maxpooling is highly dependent on our window size -- it is possible for maxpooling to miss a local maxima if the window size is too small.

Part 1: Harris corner detector

We want both λ_1, λ_2 to be large at our corners – the Harris cornerness equation allows us to check this without calculating the eigenvalues. $\det(A) = \lambda_1\lambda_2$ will indicate if the gradient magnitude is high, while $trace(A)^2 = (\lambda_1 + \lambda_2)^2$ will penalize cases where one eigenvalue is significantly larger than the other (α determines the weight of the penalty).

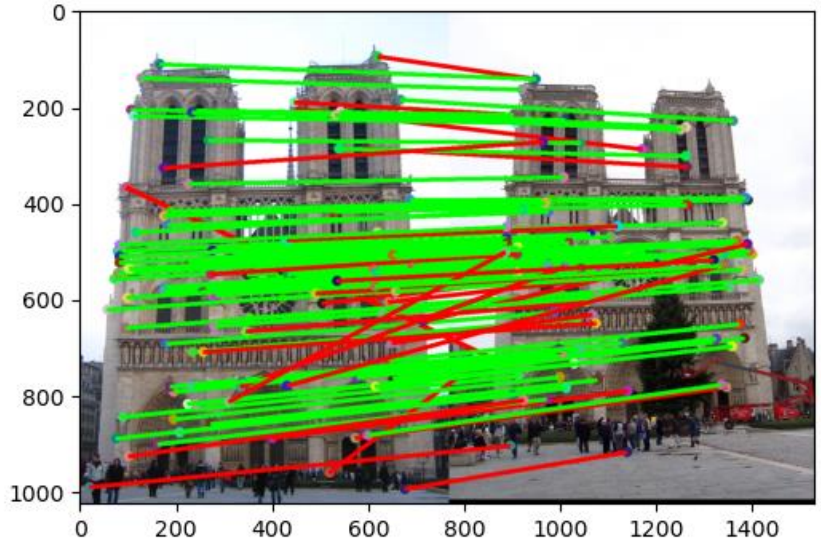
The Harris Corner Detector is effective because it finds points that both distinct and are covariant with respect to translation and/or rotation (albeit not scaling). In other words, it detects points that are comparatively more distinct and also more likely to remain “detect-able” across image pairs.

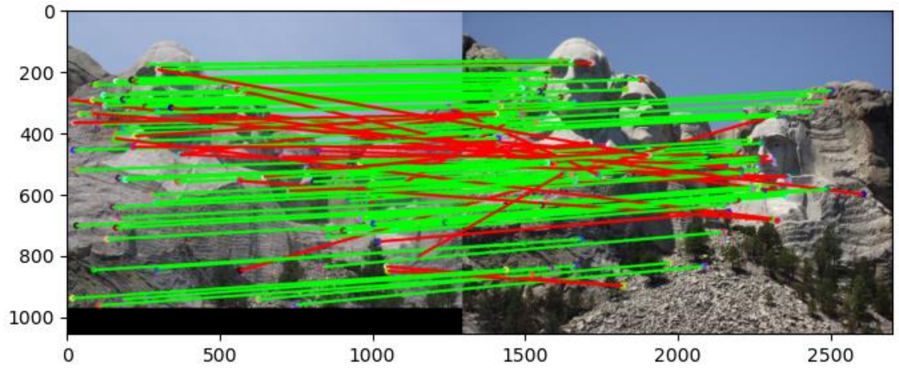
Part 2: Normalized patch feature descriptor

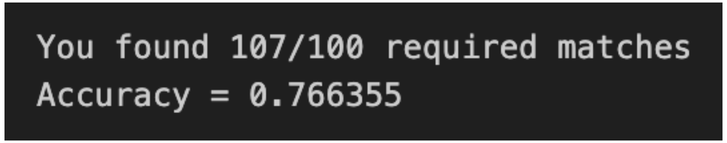


Normalized patch descriptors are unit feature vectors created by normalizing the image intensities for a window around each interest point. They are extremely sensitive to small shifts (i.e. neither translation, rotation, or scaling invariant) due to using image intensities, making them a poor choice for a feature descriptor.

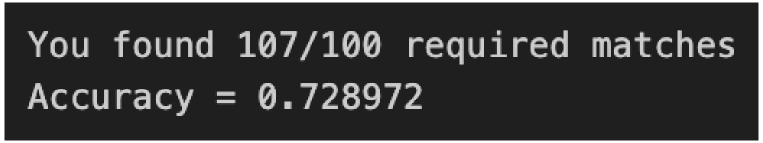
Part 2: Feature matching





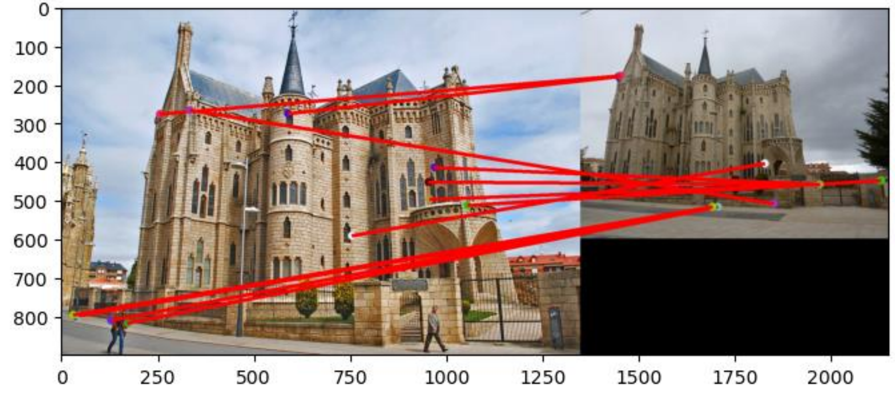


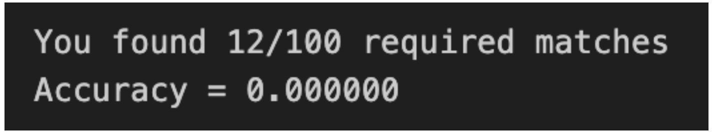
```
You found 107/100 required matches
Accuracy = 0.766355
```



```
You found 107/100 required matches
Accuracy = 0.728972
```

Part 2: Feature matching





```
You found 12/100 required matches
Accuracy = 0.000000
```

match_features_ratio_test works as follows: for each feature (D_A) in image 1, we find its two closest features in image 2 ($D_B,\ D_C$ being the closest and 2nd closest respectively) and form the nearest neighbor distance ratio ($NNDR$):

$$NNDR = \frac{||\, D_A - D_B\, ||}{||\, D_A - D_c\, ||}$$

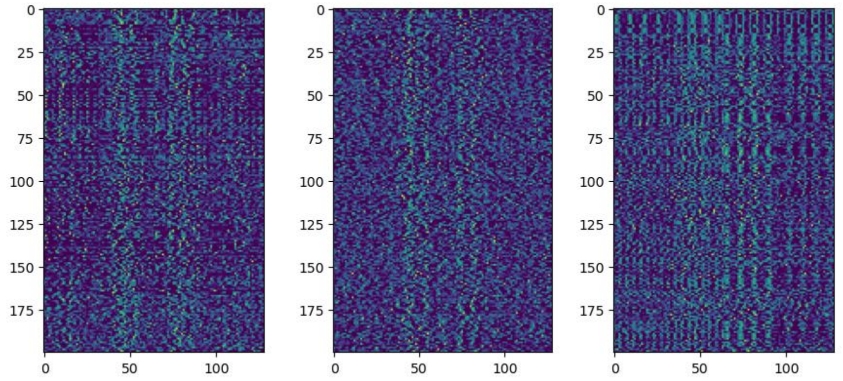
If $NNDR$ was below chosen threshold (0.8 in this case), then we would accept D_A, D_B as a matching feature pair.

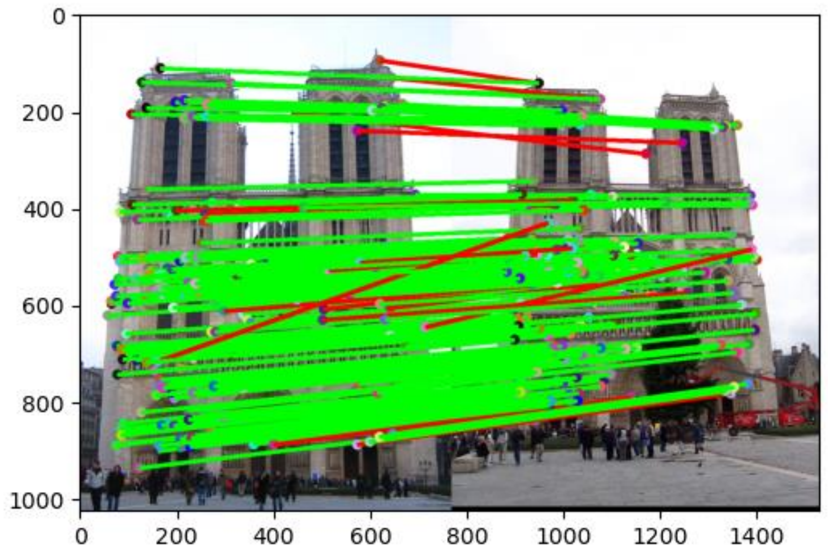
The confidence score is calculated as $1 - NNDR$ and represents how confident we are that our feature pair is actually matching.

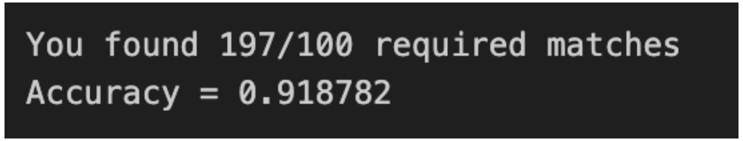
Part 2: Feature matching

Features matches that easily pass the NNDR test tend to be more isolated from other points, and thus more distinct. This makes it less likely that another nearby feature point will be chosen instead of the correct one during feature matching.

Part 3: SIFT feature descriptor

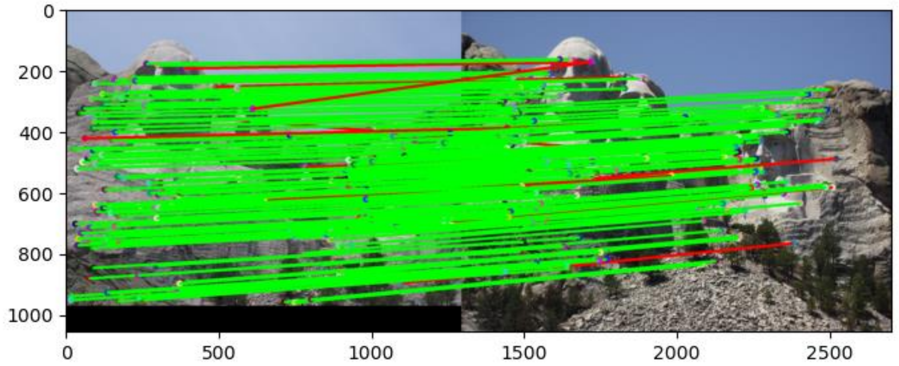


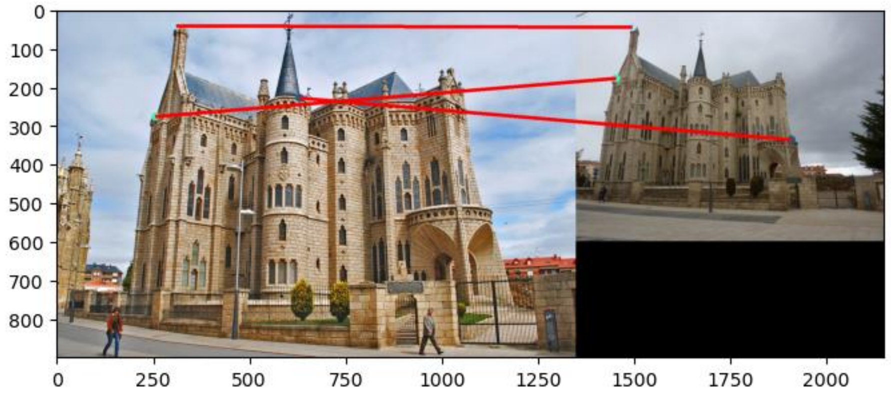




```
You found 197/100 required matches
Accuracy = 0.918782
```

Part 3: SIFT feature descriptor





```
You found 178/100 required matches
Accuracy = 0.932584
```

```
You found 3/100 required matches
Accuracy = 0.000000
```

Part 3: SIFT feature descriptor

We take a 16x16 patch around each point and then subdivide that into 16 4x4 grids of cells. Over each grid, we construct a histogram by “taking a vote” of each pixel’s gradient orientation weighted by their gradient magnitude. Since each histograms is 8-dimensional, then our SIFT descriptors are 128-dimensional.

These SIFT descriptors are later used during feature matching, where we compare our SIFT feature vectors in the NNDR ratio test instead of the features themselves.

SIFT descriptors are better than normalized patch descriptors because they are invariant to brightness shifts and translations and are also potentially invariant to rotation/scaling changes, depending on implementation. In short, they preserve more information about our interest points. They are also less susceptible to noise.

[What is the advantage of Square-Root SIFT?] (1.5)
From the 2012 CVPR paper linked in the project pdf, Square-Root SIFT (or RootSIFT) improves our performance without increasing or storage requirements.

Part 3: 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?]

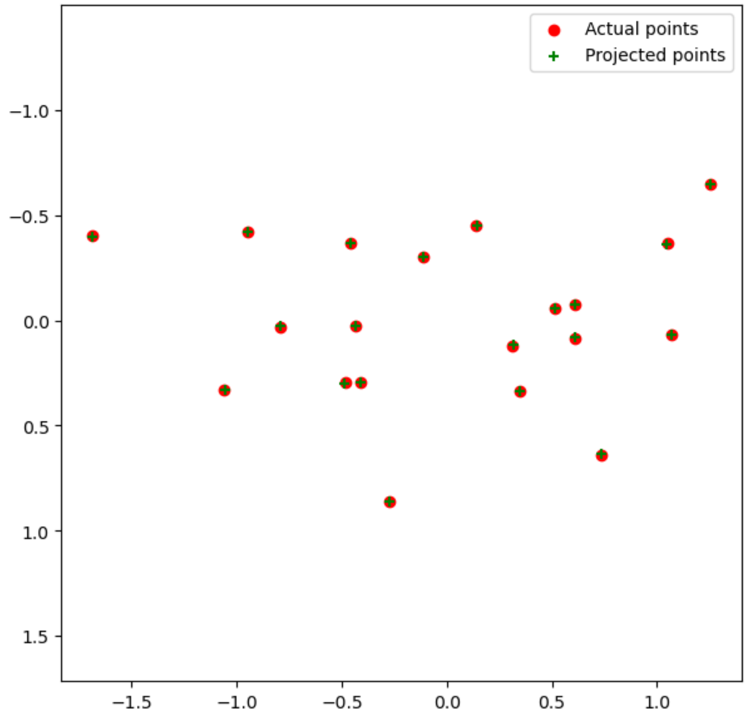
We use a fixed window size in our SIFT implementation, our SIFT descriptors are not scale-invariant. Since the scale of the Gaudi building is *noticeably* different across our image pair, our SIFT descriptors perform extremely poorly (even worse than our Nearest-Neighbor matches).

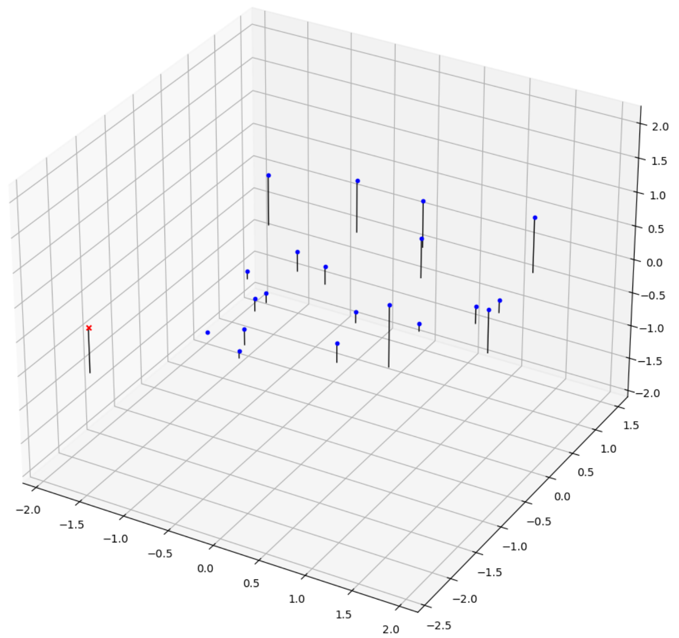
Part 3: SIFT feature descriptor

Our SIFT features do not normalize the orientations of our patches making them rotation-invariant. To amend this, we could compute the orientation histogram and select the most dominant orientation and normalize our patch with respect to said orientation.

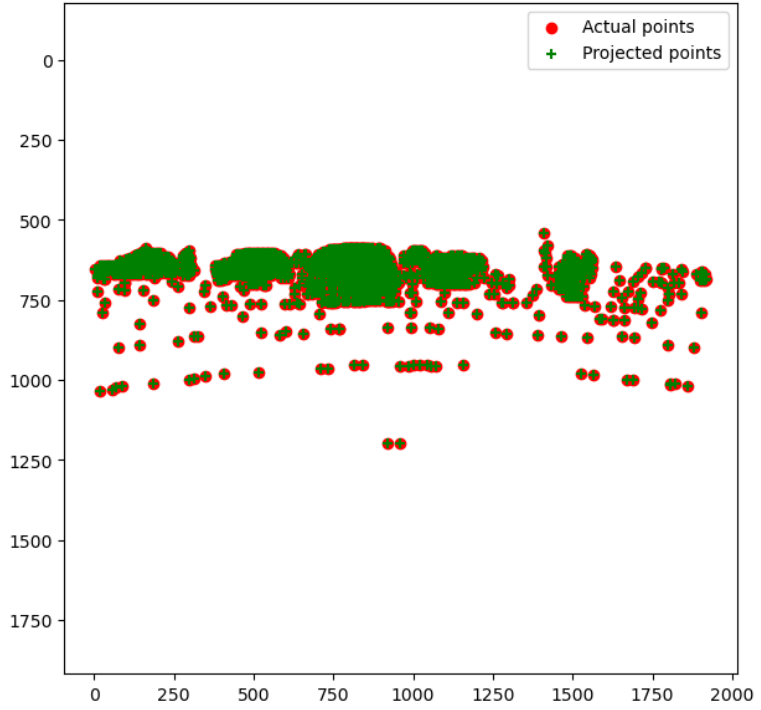
As SIFT features are also not scale-invariant because they use a fixed window size. To solve this, for each interest we could try different window sizes and use the window size that maximizes the Harris cornerness score for said point (meaning that each point could potentially have varying window sizes).

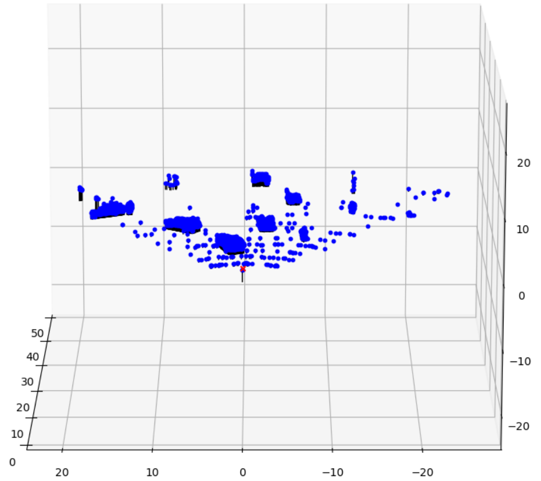
Part 4 (Extra Credit): : Projection matrix





Part 4 (Extra Credit): : Projection matrix





Part 4 (Extra Credit): : Projection matrix

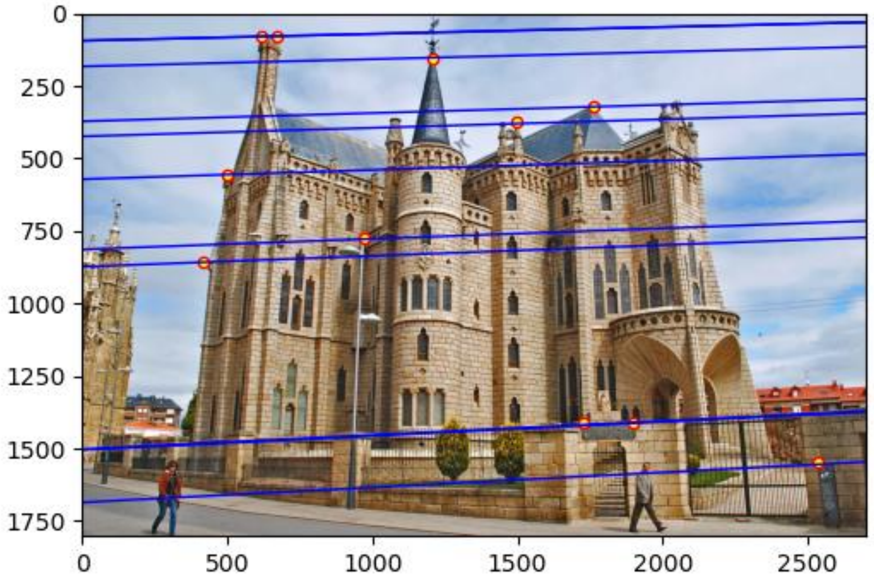
The camera matrix translates between 3d world coordinates and our 2d image coordinates by projecting our 3d (homogeneous) coordinates onto our image plane.

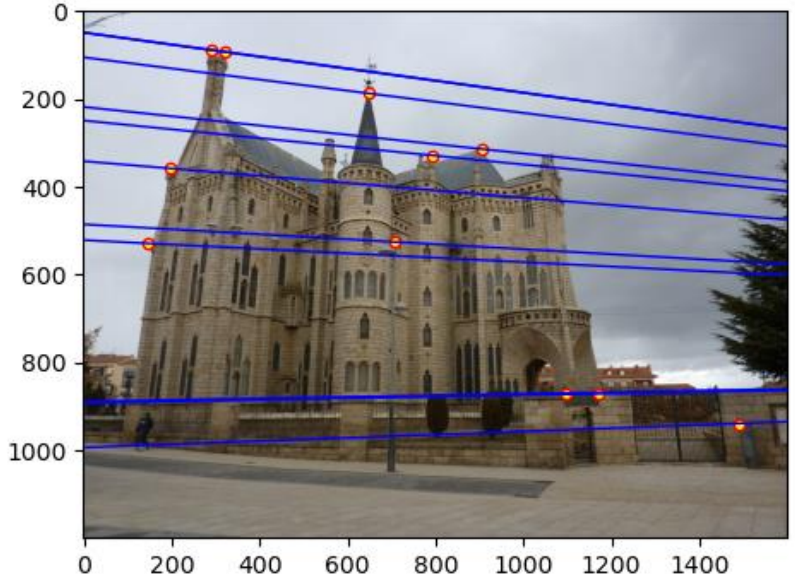
Our camera matrix can be decomposed into the product of an intrinsic matrix K describing the parameter of our camera, and an extrinsic matrix $[R \mid T]$ which translates our 3d world coordinates in terms to our 3d camera coordinates using rotation and translation.

[List any 3 factors that affect the camera projection matrix.]

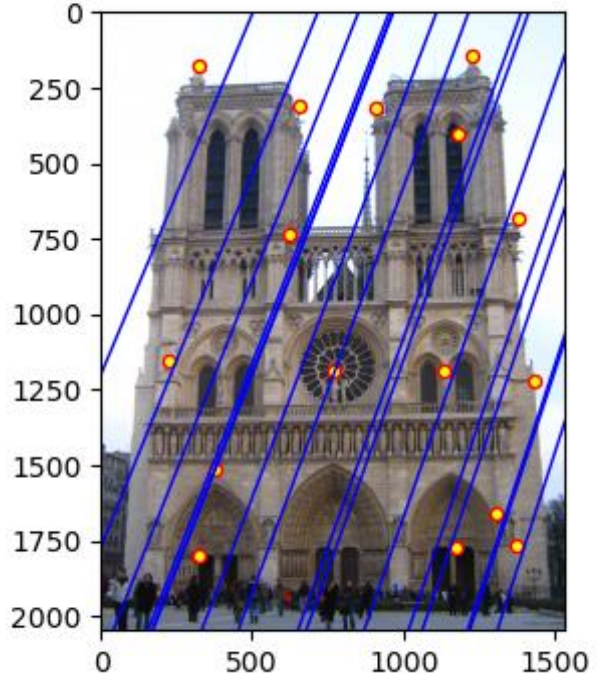
- Position of our camera in the world ($C = -R^{-1}T$)
- Skew between the sensor axes
- Focal length (in either x/y-directions)

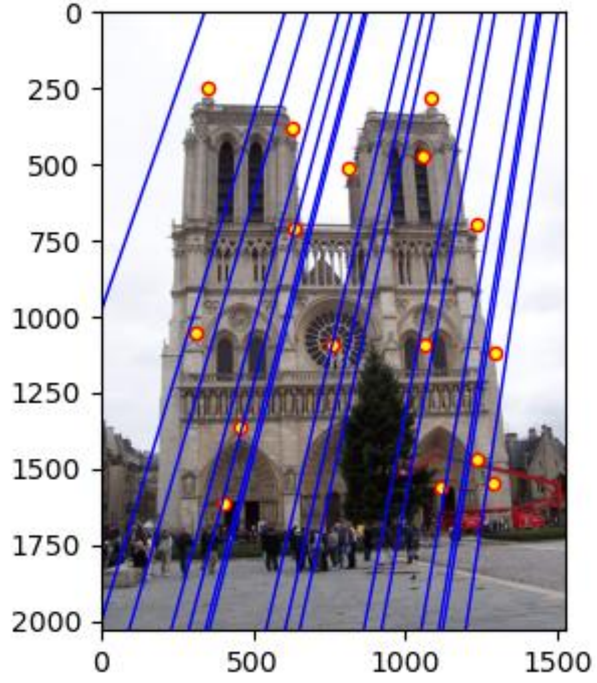
Part 5 (Extra Credit): Fundamental matrix





Part 5 (Extra Credit): Fundamental matrix





Part 5 (Extra Credit): Fundamental matrix

Since 2d images coordinates are merely projections of our 3d world coordinates, then that means that each point corresponds to ray from the camera center to the 3d world. When this ray is projected onto another image plane, it would appear as an epipolar line.

If Camera A is able to view the Camera B’s center, then Camera B’s epipole will be projected into the image captured by Camera A. Additionally, all epipolar lines projected onto Image A will intersect at the epipole corresponding to Camera B.

Part 5 (Extra Credit): Fundamental matrix

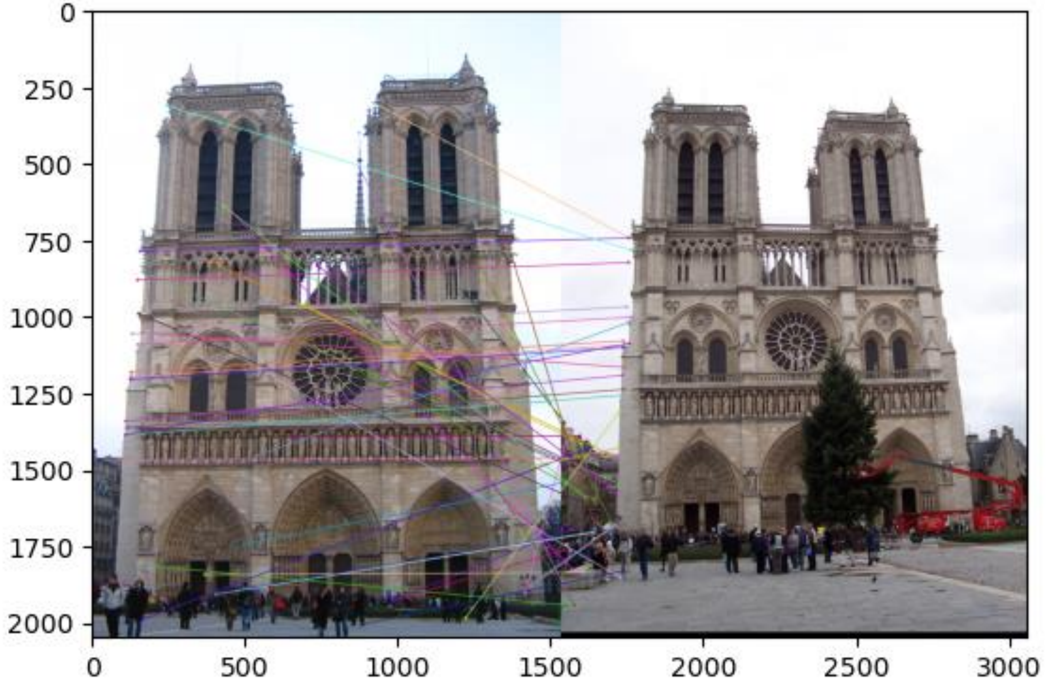
If the epipolar lines are all horizontal across the two images, it means both cameras are facing (roughly) the same direction at (about) the same height and distance away from the object being captured. In other words, the camera centers will be horizontal translations of each other.

We calculate the fundamental matrix F using $p'^T Fp = 0$ where p, p' are points in our left/right images respectively. Since we can get more solutions by multiplying F by any constant a ($p'^T aFp = 0$), we say that F is defined up to a scale.

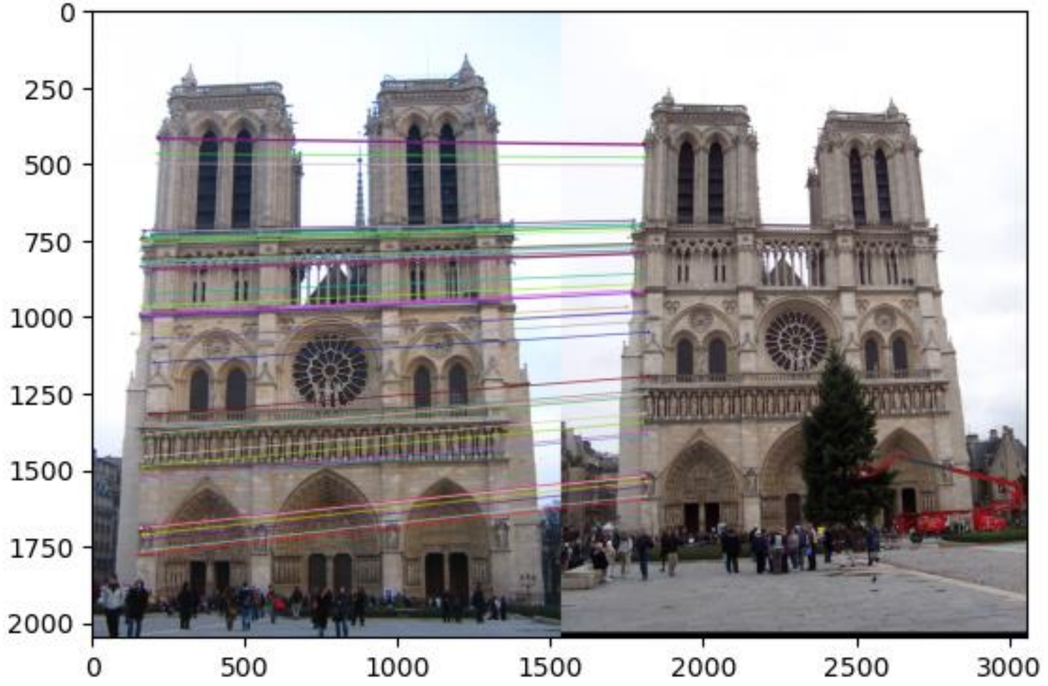
We know that the fundamental matrix cannot be full rank as it must map to the nullspace as dictated by the equation we use to solve for it.

Since rays corresponding to points in one image are projected as epipolar lines in the other image plane, then the fundamental matrix must be at least rank 2.

Part 6: RANSAC



Part 6: RANSAC



Part 6: RANSAC

[How many RANSAC iterations would we need to find the homography with 99.9% certainty for your Notre Dame SIFT results? Assume a hypothetical inlier ratio (correspondence accuracy) of 90% and that you are using the minimal sample size of 4 points]

6 iterations

[One might imagine that if we have many correct matches, it would be better to use more than the minimum number of points to compute the homography in each sample. Investigate this by finding the number of RANSAC iterations you would need if you used a sample size of 8 points instead of 4 (keeping the same 99.9% certainty and 90% inlier ratio).

12 iterations

[If our dataset had a lower point correspondence accuracy (i.e., a lower inlier ratio), say 70%, what is the minimum number of RANSAC iterations needed to find the homography with 99.9% certainty? Remember to use the minimal sample size of 4 points in your calculation.]

25 iterations

Part 6: Performance comparison

Image A warped onto B using Naive Homography (NO RANSAC)

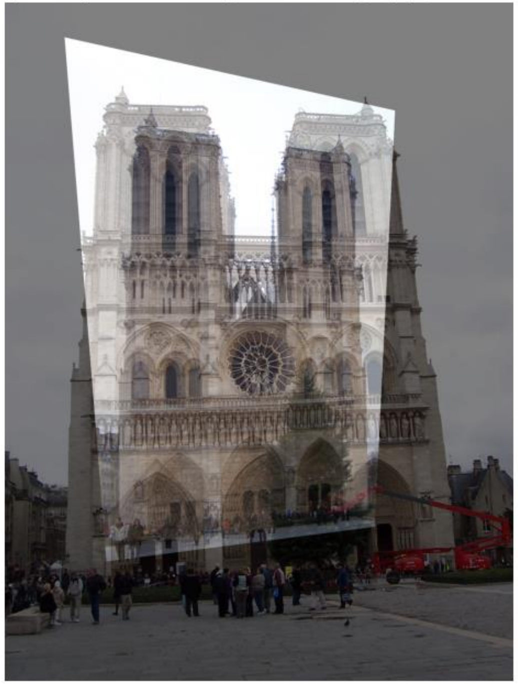
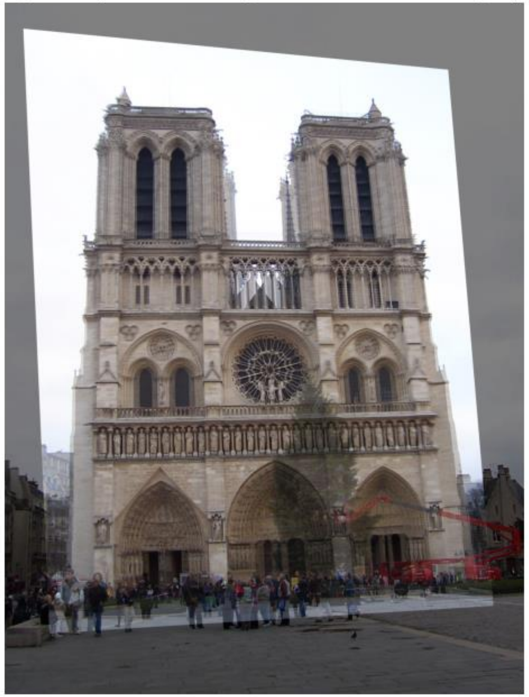


Image A warped onto B using Custom RANSAC Homography



Part 6: Performance comparison

The Naïve Homography performed significantly worse than our RANSAC Homography: the warping produced by the RANSAC could seen as a single image with some noisiness while same can not be said for the Naïve warping.

The Naïve Homography is produced using Least Squares, which is very sensitive to outliers. Our RANSAC implementation is more robust since:

1) We’re less likely to use incorrect matches since we only use four feature pairs to calculate our homography.

2) We’re not trying to find the homography that perfectly fits to all points (which would include outliers) but rather one that maximizes the number of inliers.

Our RANSAC implementation would be more robust in real applications, since in practice it is likely that some incorrect matches will pass through the feature matching pipeline.

Conclusion

RANSAC-based homography was more significantly effective because:

1) It’s less likely to use incorrect feature matches since in each iteration only four feature pairs are randomly to calculate our homography, reducing the risk of selecting an incorrect pair.

2) Rather than find the homography that perfectly fits to all points (including outliers) like in the naïve estimation, RANSAC select the homography that maximizes the number of inliers across numerous iterations.