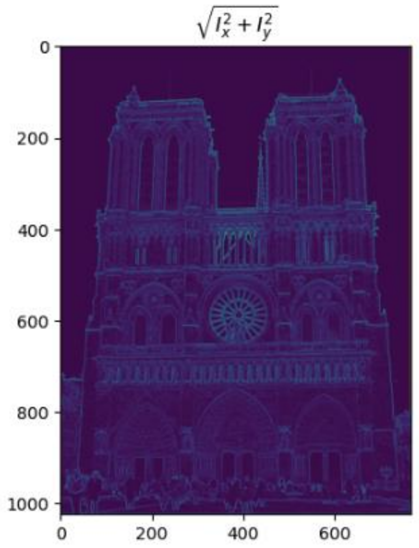
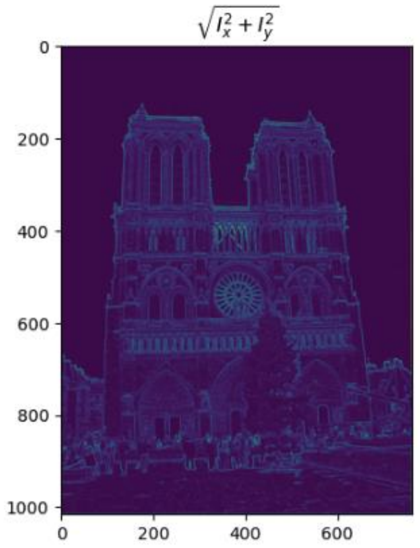
Warning: Do not delete slides.
This includes extra credit slides and any problems you do not complete.
All problems, including extra credit, must be assigned to a slide on Gradescope. Failure to follow this will result in a penalty.

CS 6476 Project 2

Anirudh Arunkumar
aarunkumar8@gatech.edu
aarunkumar8
903572206

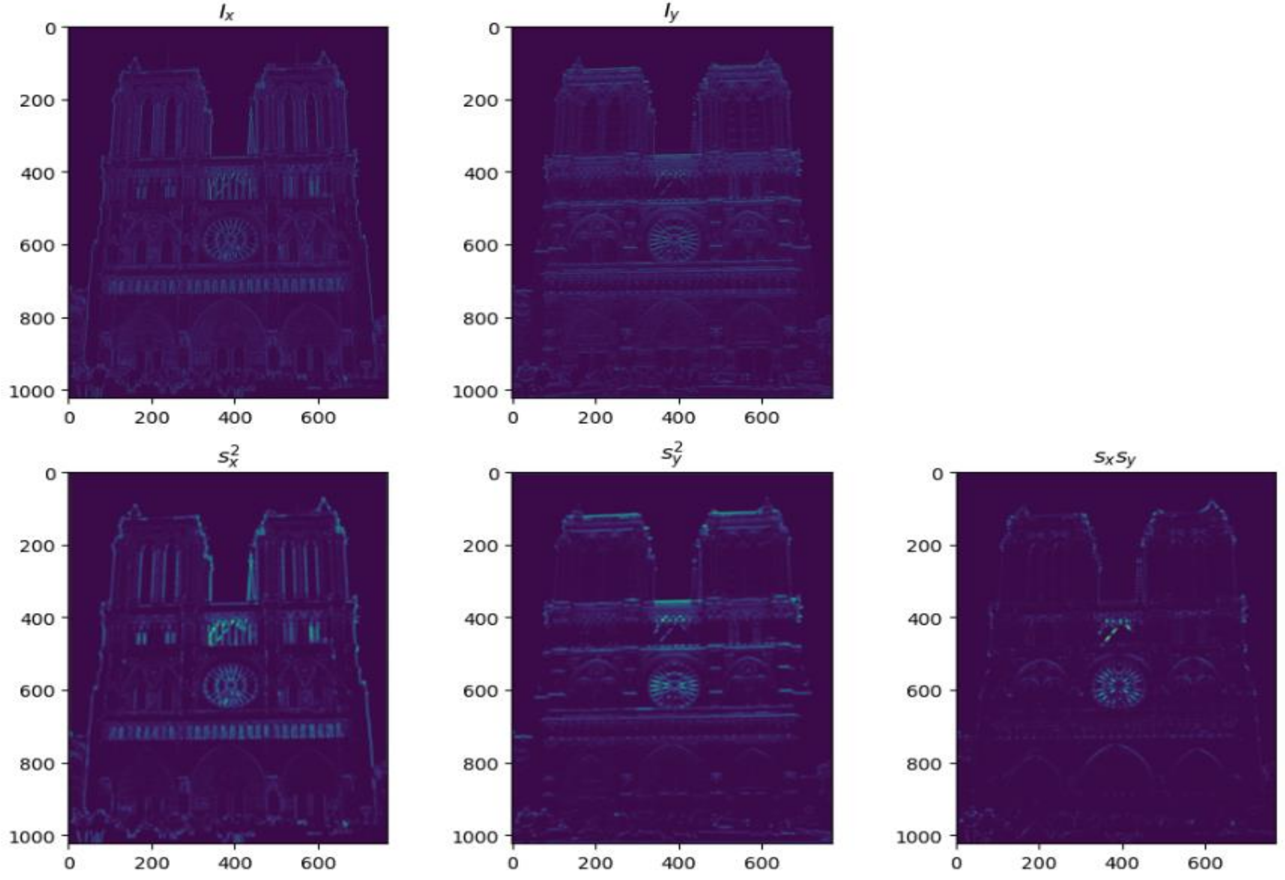
Part 1: Harris corner detector



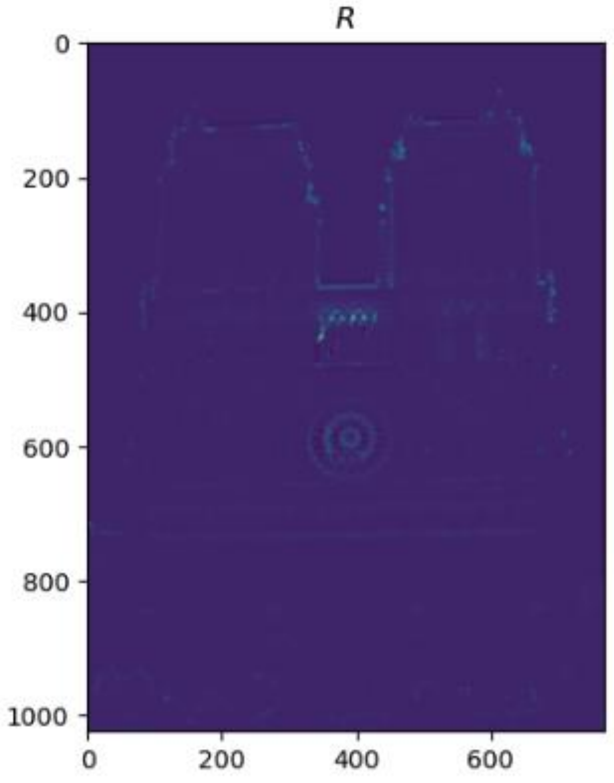


The highest magnitude happens at the corners of an image. This is because there is higher image intensity and two large eigenvalues which in turn causes high magnitudes. It is also why we can detect corners in an image.

Part 1: Harris corner detector



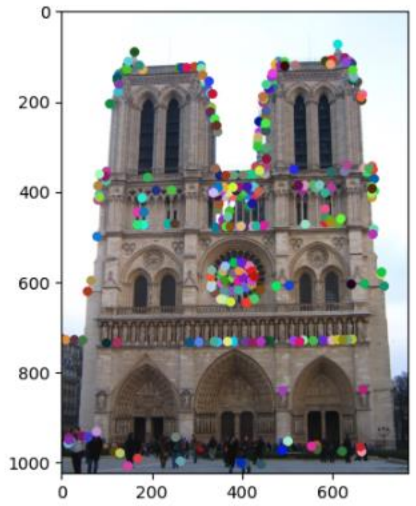
Part 1: Harris corner detector

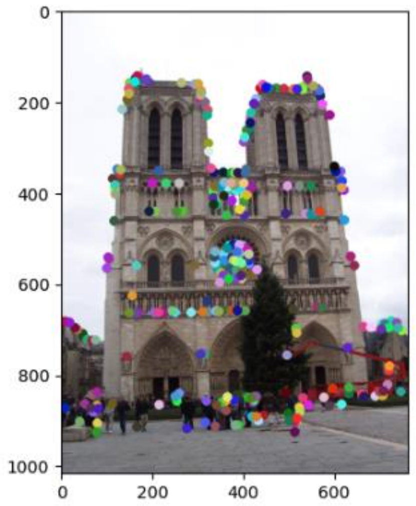


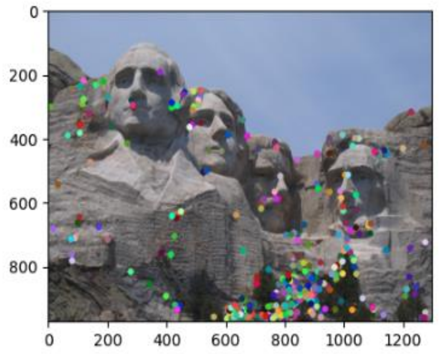
[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

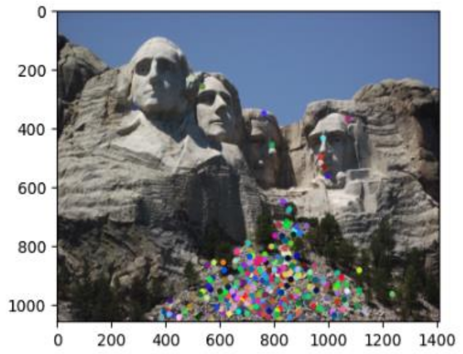
Gradient features are invariant to additive shifts in brightness. This is because the derivatives are unchanged when a constant value of certain intensity is added. They are not invariant to multiplicative gain though. When scaled by a factor then the gradients are also scaled. This has to do with how scalars are handled vs constants when taking the derivative

Part 1: Harris corner detector

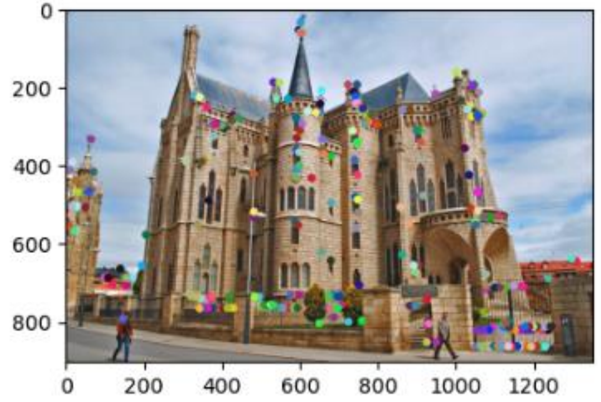


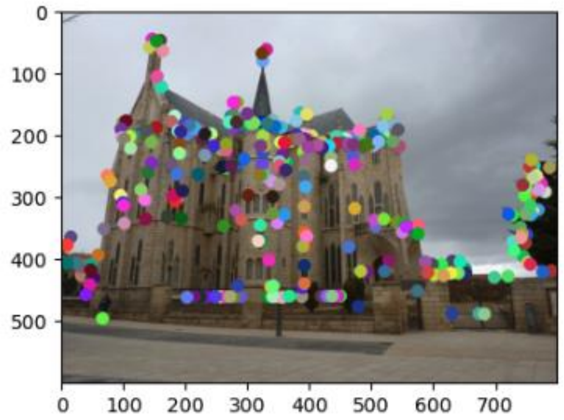






Part 1: Harris corner detector





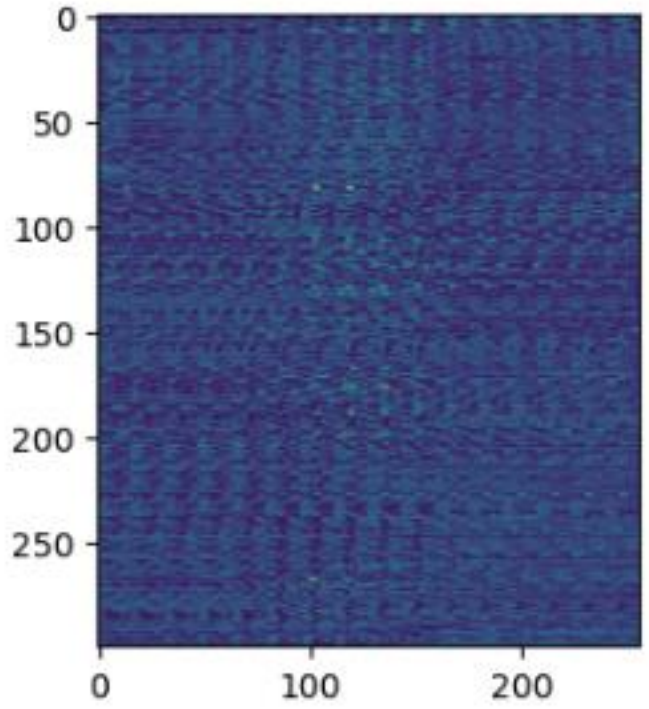
[What are the advantages and disadvantages of using maxpooling for non-maximum suppression (NMS)?]

Using Maxpooling for NMS has advantages such as removing weak responses, differentiable, and only dependent on fixed kernel sizes. Some disadvantages are the fact it works on a fixed grid and also ignores the entire image (only kernel size). It might struggle with overlapping objects or when multiple things are close together in the image.

Part 1: Harris corner detector

This corner detection method is effective because it takes points where the gradient intensities change in multiple directions (corners). Corners of an image are highly distinctive which makes the object detection or recognition task much simpler. They are defined so that when eigenvalues indicate the intensity changes so the corners can be identified easily. The Harris detector is able to pinpoint these corners because it is also invariant to brightness and transformations.

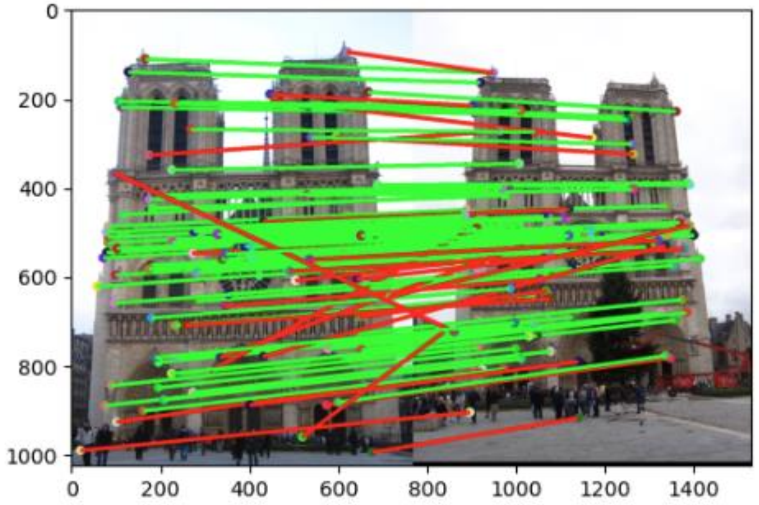
Part 2: Normalized patch feature descriptor

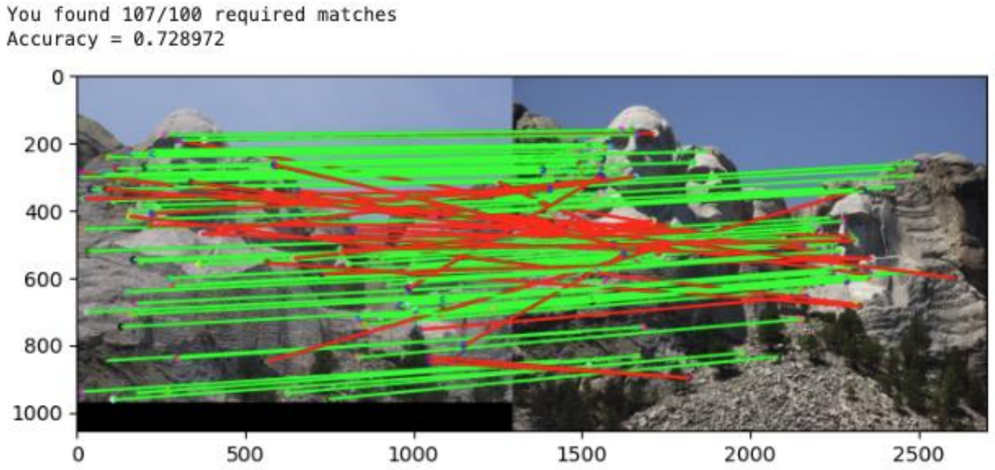


[Why aren't normalized patches a very good descriptor?]

They are not got because it is a fixed size 16x16 grid which changes if there are rotations or scaling transformations. This can be an issue when viewing an image from different views. Another reason are a high number of false positives due if there are repetitive textures. There is variance to affine transformations further make a bad descriptor.

Part 2: Feature matching



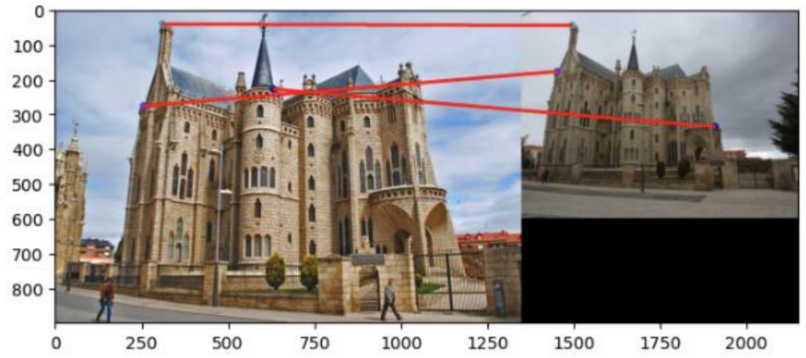

```
You found 107/100 required matches
Accuracy = 0.728972
```

# matches (out of 100): 107
Accuracy: 77%

# matches: 107
Accuracy: 73%

Part 2: Feature matching

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

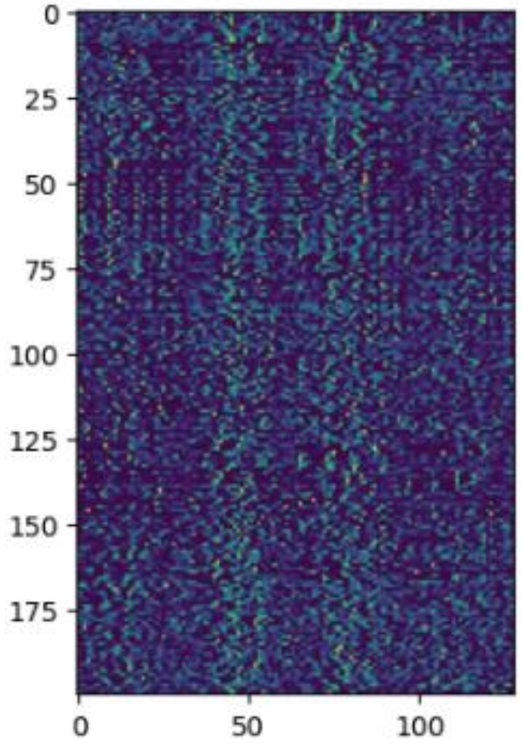


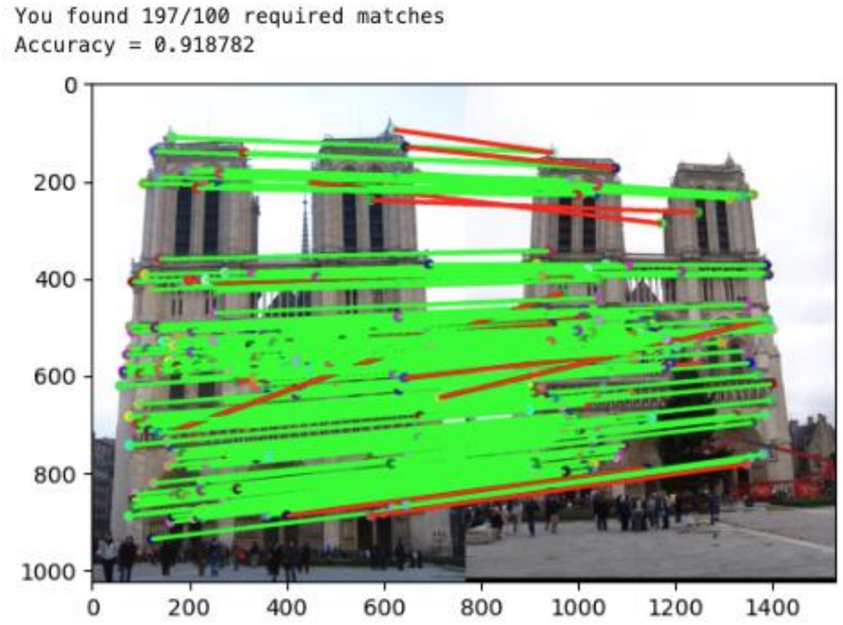
# matches: 3

Accuracy: 0%

We use the Lowe’s ratio test from his SIFT paper. We create a distance matrix. Then for each feature we compute the distance using Euclidean formula for features 1 and features 2. Then in the match_features_ratio_test() we find the correspondences using Lowe’s ratio test. After getting the distances from the previous function we get the nearest neighbors and sort them. The ratio test uses 0.8 and then we compute the confidence score.

Part 3: SIFT feature descriptor



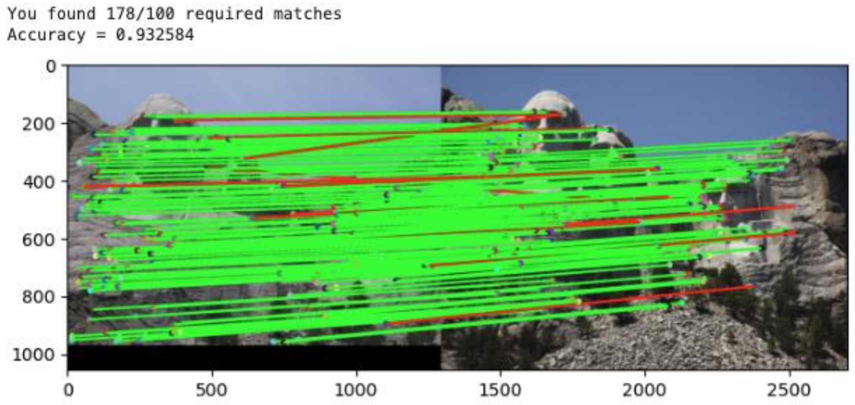


```
You found 197/100 required matches
Accuracy = 0.918782
```

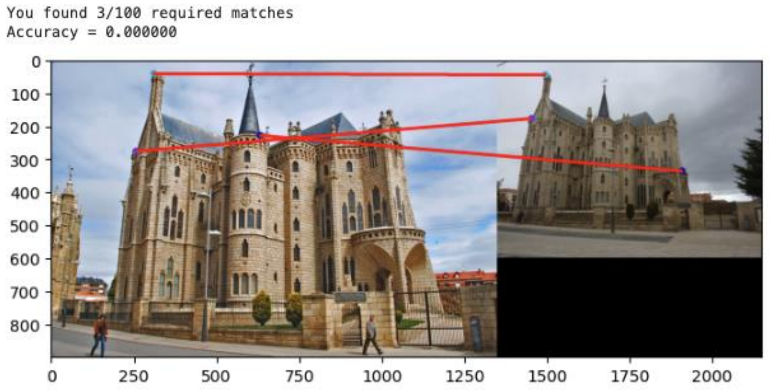
/red
) for
ere]

# matches (out of 100): 197
Accuracy: 92%

Part 3: SIFT feature descriptor



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



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

# matches: 178
Accuracy: 93%

# matches: 3
Accuracy: 0%

Part 3: SIFT feature descriptor

We have a a get_magnitude_and_orientations functions that converts Ix and Iy to a float and computes the magnitude using their formulas to get the dominant gradients. The get_gradient_histogram_vec_from_patch() divides a 16x16 into 4x4 cells forming histograms of 8 bins and concatenates them based on magnitudes (128 vector). Get_feat_vec takes the SIFT descriptor for each keypoint by taking a 16x16 patch and applying square root SIFT formula after normalizing the feature vector. Finally get_SIFT_descriptors() computes gradients, gets magnitudes and orientations and iterates over all key points and stores the 128 descriptors in a matrix.

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

This is because they are invariant to scale and rotation. It is also robust when there are changes in brightness due to the computation of magnitudes and orientations. Using the histograms and 128 dimensions makes it better at matching and distinctiveness in images.

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?]

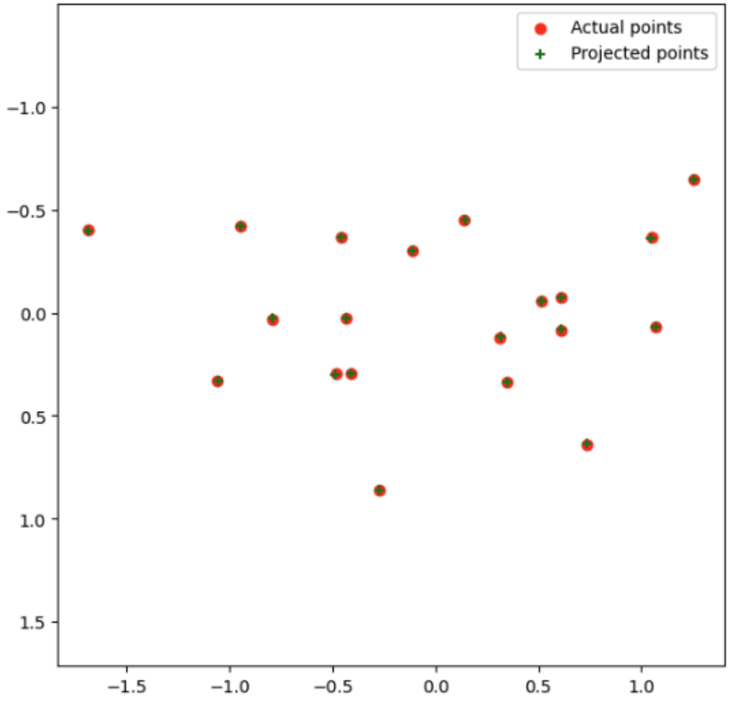
It performs worse because of extreme scale differences. SIFT is designed to be invariant to scale but large variances can make it struggle and give only a few matches. The scaling of the other two image pairs are much more consistent. Gaudi has a lot of complex patterns and repetitive patterns so it might be hard to find the difference between them. Another thing I noted was the difference in color between the Gaudi pairs further increasing inaccuracy.

Part 3: SIFT feature descriptor

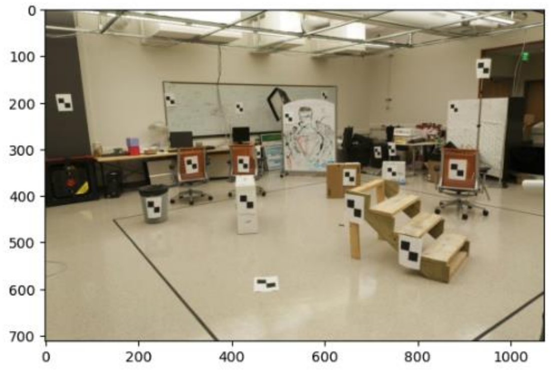
[Why aren't our version of SIFT features rotation- or scale-invariant? What would you have to do to make them so?]

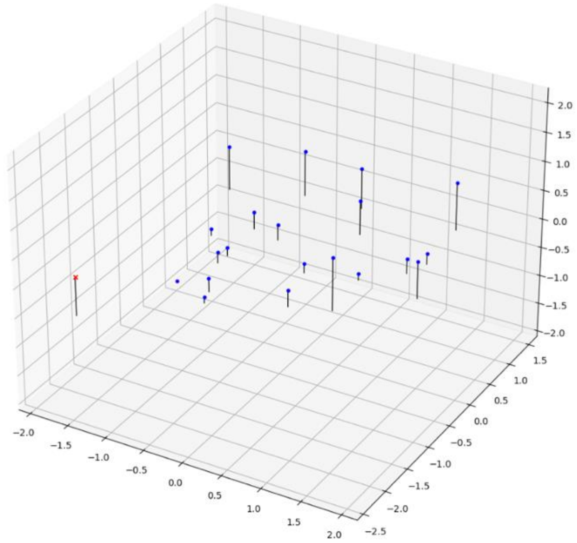
Our version is not rotation or scale invariant because we did not implement a key point orientation and scale space to account for this. In the methods we do not have orientation alighment causing the descriptor to change with rotations. Since we do not have a Gaussian scale space size variations can also change the result. For rotation invariance we need to compute the gradient historgram for each key point and align descriptor to the most dominant orientations. For scale invariance we need a pyramid that detects key points on octaves using DOG.

Part 4: Projection matrix

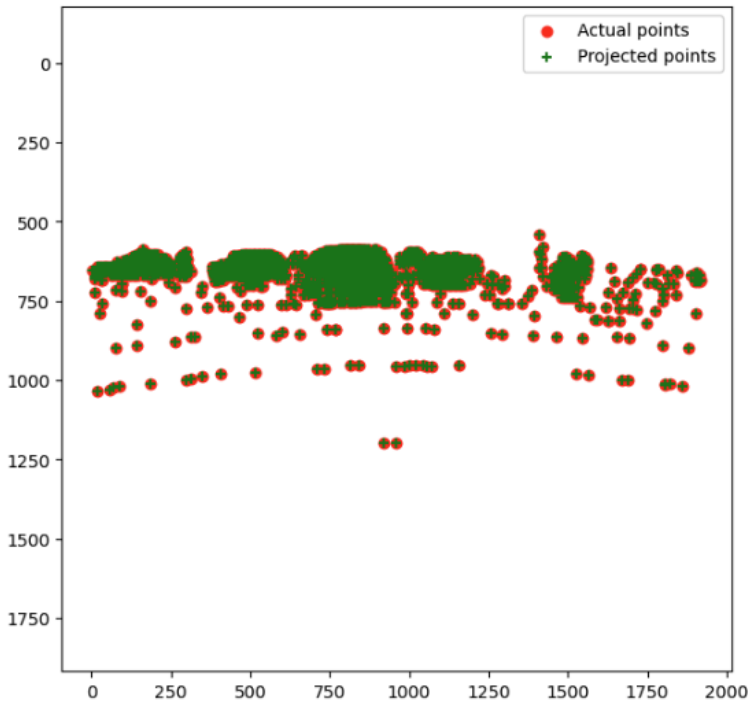


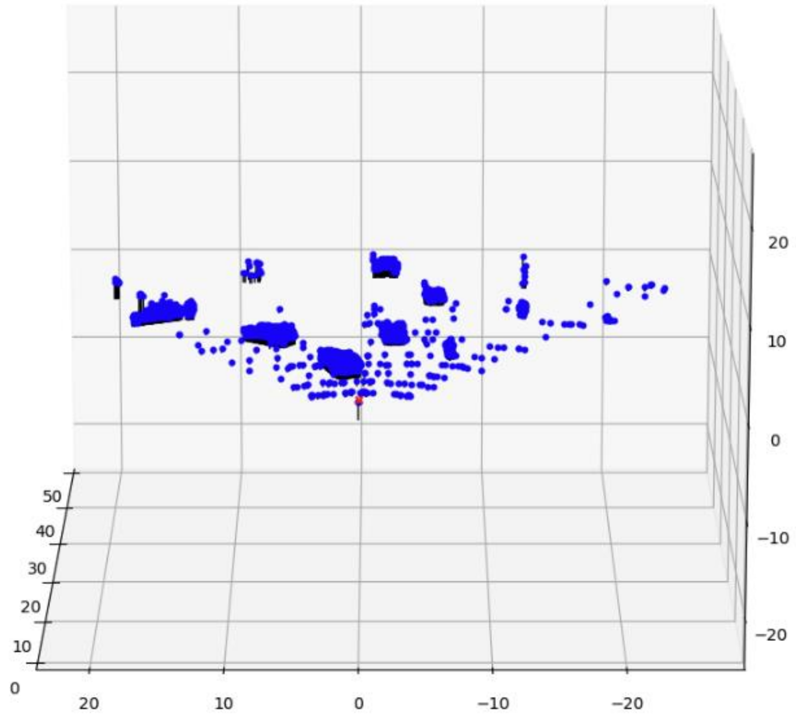
```
The estimated location of the camera is <-1.5127, -2.3517, 0.2826>
```





Part 4: Projection matrix





Part 4: Projection matrix

[What two quantities does the camera matrix relate?]

It relates 3d world coordinates to 2d image coordinates

[What quantities can the camera matrix be decomposed into?]

There is a K instrinsic matrix with focal lengths, optical center, and skews. There is a R rotational matrix showing camera orientation in the world. There is a t a translation vector showing camera position

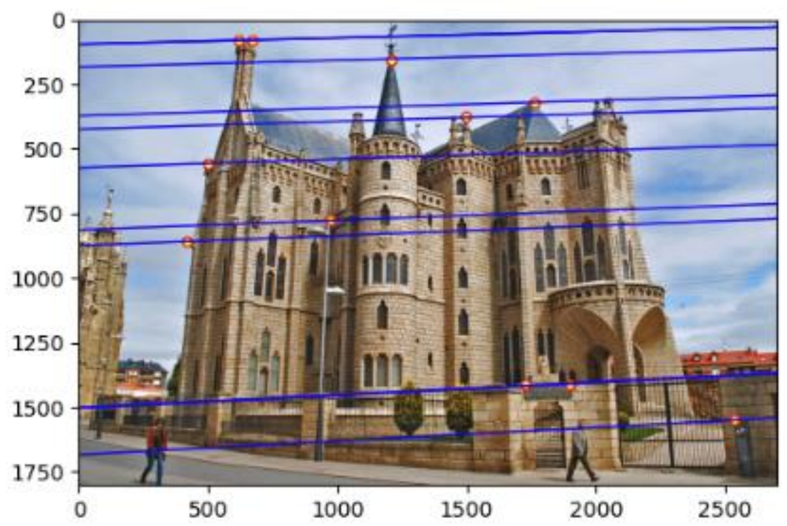
[List any 3 factors that affect the camera projection matrix.]

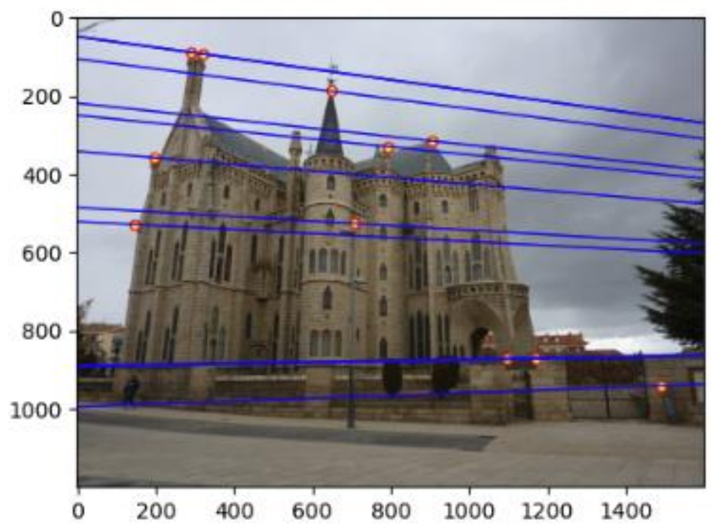
The focal length f_x and f_y can impact the zoom and field of view.

The Extrinsic matrices R and t determine orientation and translations.

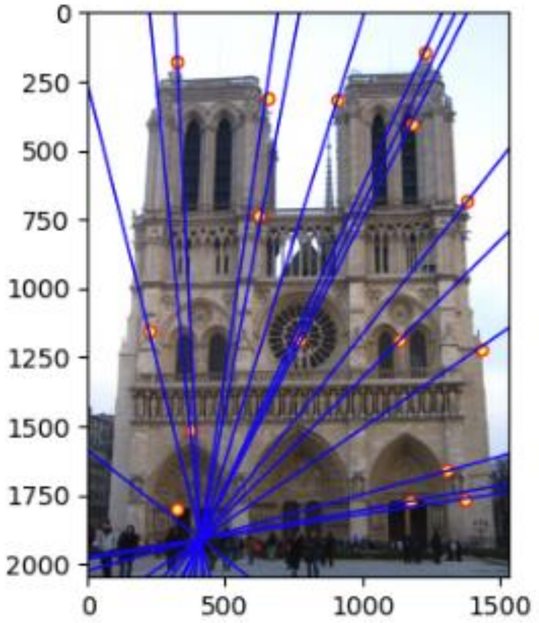
There could be distortion due to camera lens differences.

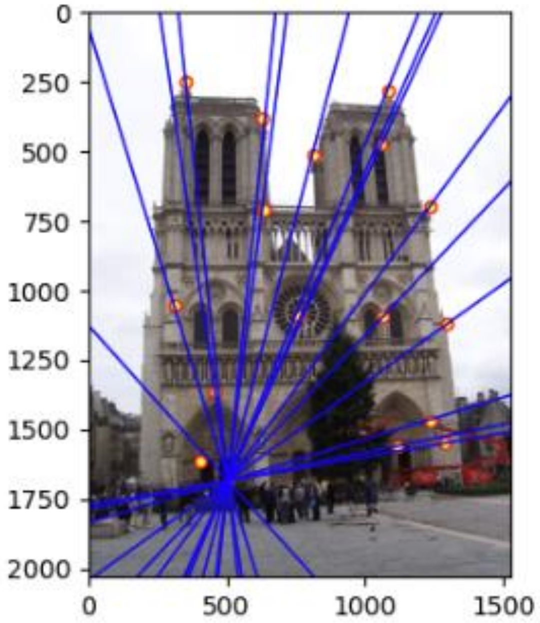
Part 5: Fundamental matrix





Part 5: Fundamental matrix





Part 5: Fundamental matrix

[Why is it that points in one image are projected by the fundamental matrix onto epipolar lines in the other image?]

The fundamental matrix has an epipolar constraint that makes sure that a point in an images maps to the epipolar lines. This is because the 3d point project onto two images in the same line of sight.

[What happens to the epipoles and epipolar lines when you take two images where the camera centers are within the images? Why?]

The camera centers appearing in the images cause epipoles to shift into the images. This makes lines that converge toward them. The reason for this is because epipoles are vanishing points.

Part 5: Fundamental matrix

[What does it mean when your epipolar lines are all horizontal across the two images?]

When they are horizontal that means the axes is parallel and the image planes are aligned. The fundamental matrix has zeros.

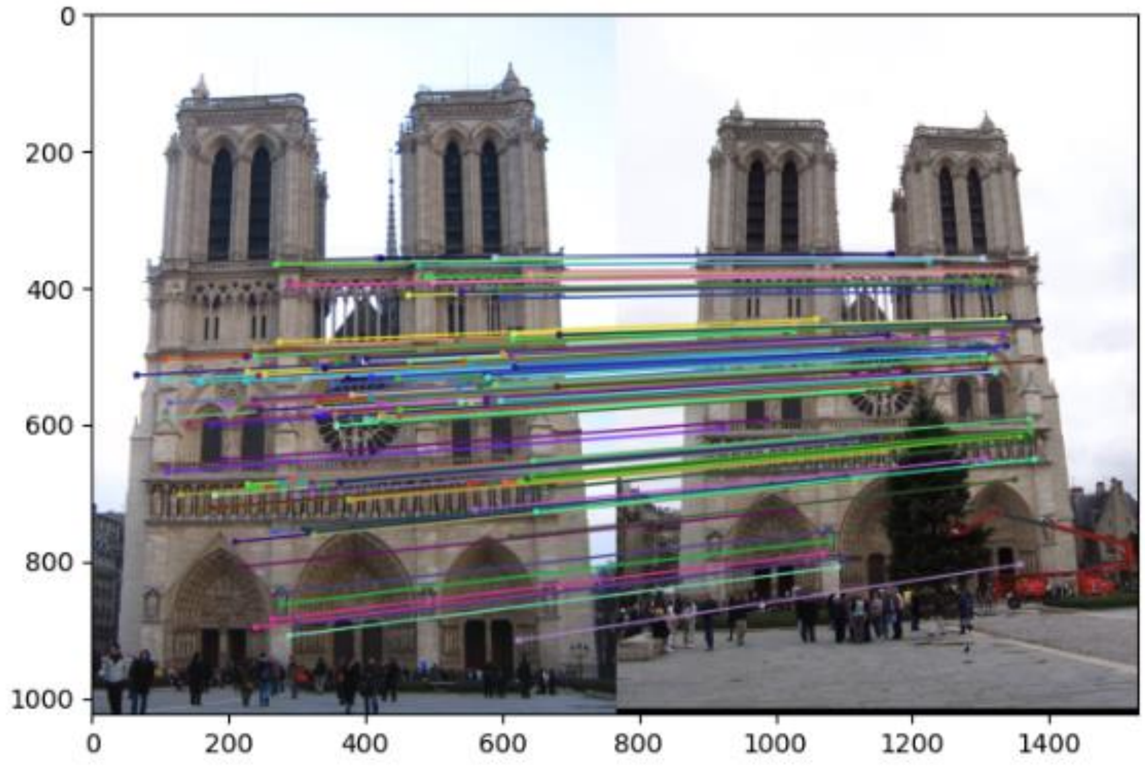
[Why is the fundamental matrix defined up to a scale?

The F is taken from homogeneous linear systems so it determines relatives not absolute values. If a linear constraint is satisfied them the scaled constraint is also satisfied.

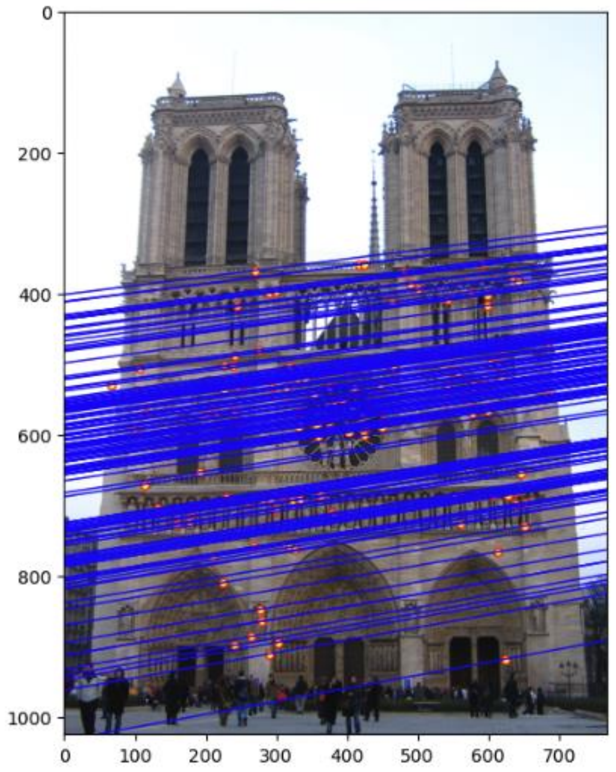
[Why is the fundamental matrix rank 2?

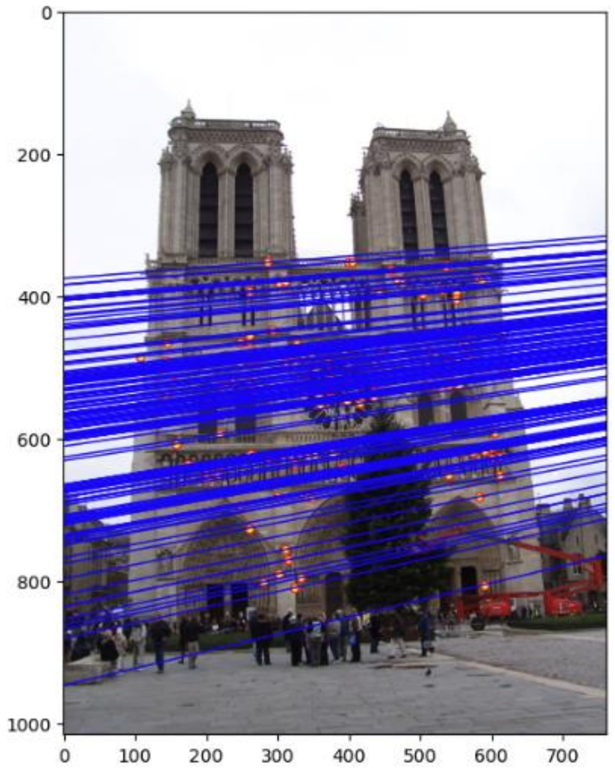
It has a rank 2 because it is derived from the essential matrix. One column is a linear combindation of the other two columns. Epipoles lie on epipolar lines. It described a two dimensiona relatins between corresponding points on two images.

Part 6: RANSAC



Part 6: RANSAC





Part 6: RANSAC

[How many RANSAC iterations would we need to find the fundamental matrix with 99.9% certainty from your Mt. Rushmore and Notre Dame SIFT results assuming that they had a 90% point correspondence accuracy if there are 9 points?]

Using 1177 iterations was good enough for 8 points on Mount Rushmore and Notre Dame. Notre dame I used 2356 iterations for 9 points. Mt Rushmore required 2356

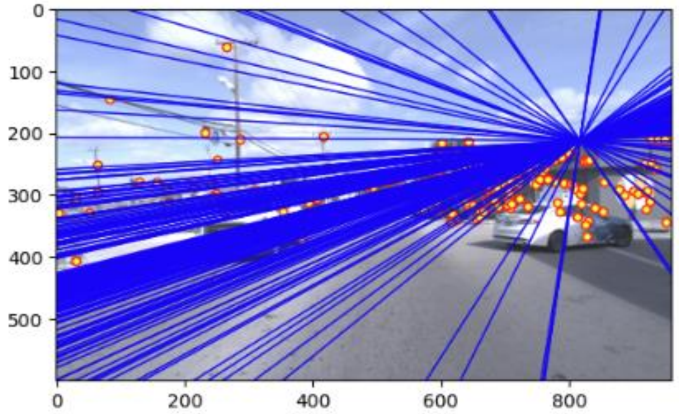
[One might imagine that if we had more than 9 point correspondences, it would be better to use more of them to solve for the fundamental matrix. Investigate this by finding the # of RANSAC iterations you would need to run with 18 points.]

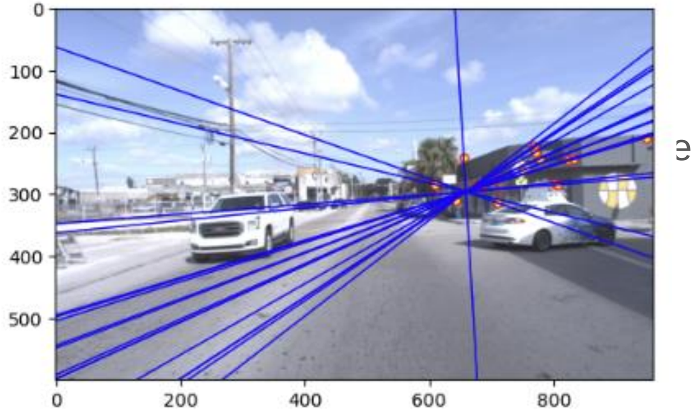
2414431 iterations ?

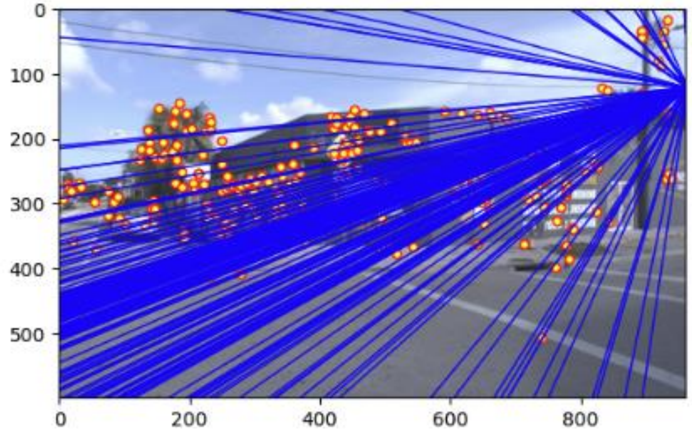
[If our dataset had a lower point correspondence accuracy, say 70%, what is the minimum # of iterations needed to find the fundamental matrix with 99.9% certainty?]

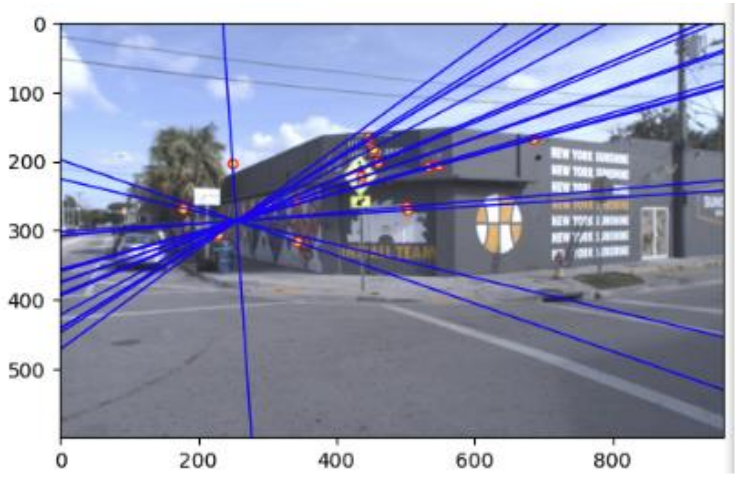
168 iterations

Part 6: Performance comparison









Part 6: Performance comparison

[Describe the different performance of the two methods.
Least square is efficient and works well on noise free data but it is sensitive to outliers which can distort the fundamental matrix. Ransac mitigates outliers because it takes random samples and forms the matrix that maximizes inliers. Using RANSAC on large datasets can be compute intensive.

[Why do these differences appear?

These appear because of the way each of the two methods work.

[Which one should be more robust in real applications? Why?]

RANSAC is more robust for real world because it handles outlier by getting a lot of inliers and taking out false matches. It also handles distortions which least squares does not handle. There is a better fundamental matrix in RANSAC because of these things and how it handles points.

Conclusion

[What made our epipolar-based matching more effective than naive matching from part 2? What image pairs was it most successful on?

The epipolar matching was more successful because there are more geometric constraints. Naïve mathching entirely relies on feature similarity making it less effective. We eliminate more false matches by restricting the correspondences to 1d epipolar lines that do not fit the fundamental matrix. It handle different viewpoints well which is why it performed better on Notre Dame and Mt. Rushmore. When there are high textured and distorted images it performs bad like Gaudi. The epipolar constraints improve feature matching due to these things.