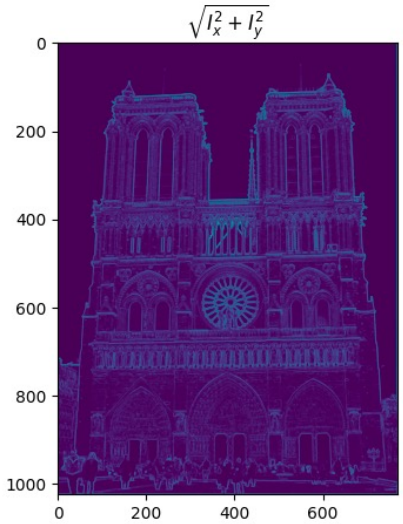
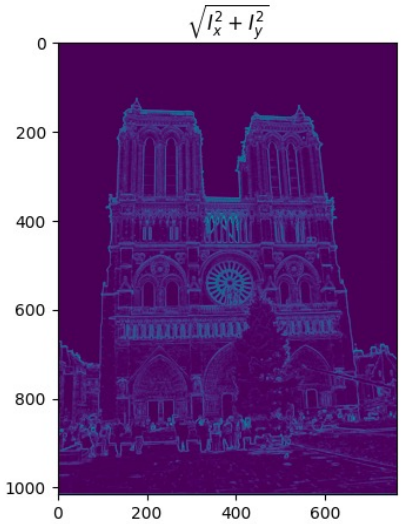
CS 4476 Project 2

Vidit Pokharna
vidit@gatech.edu
vpokharna3
903772087

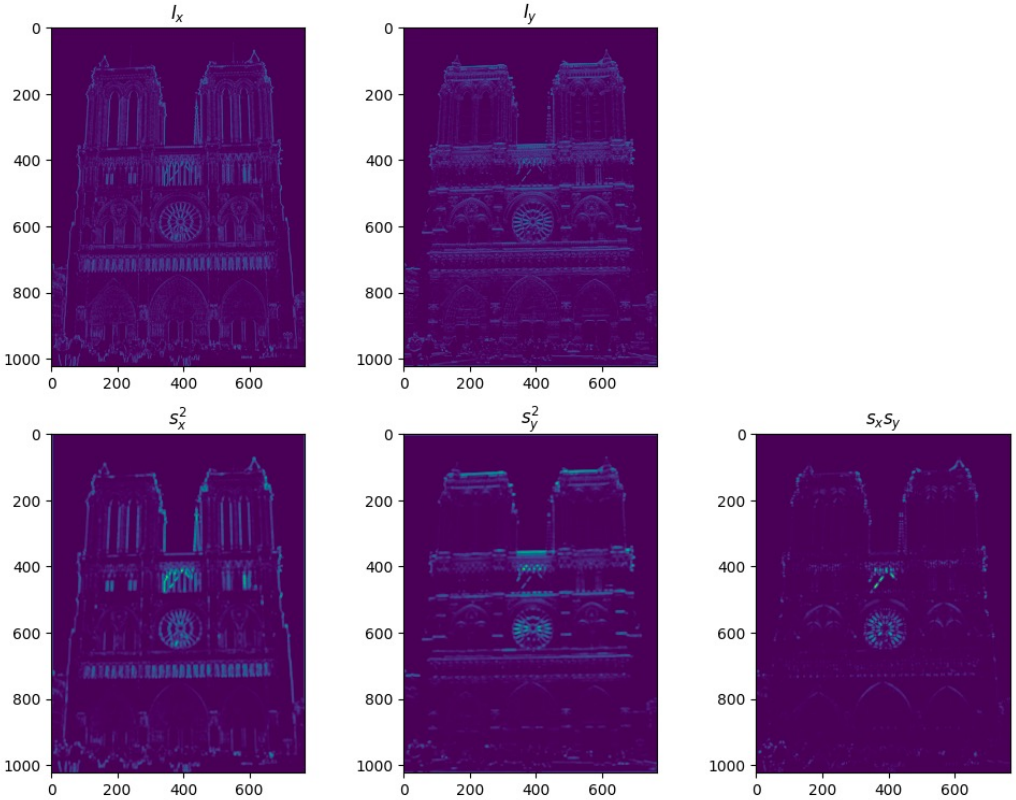
Part 1: Harris corner detector



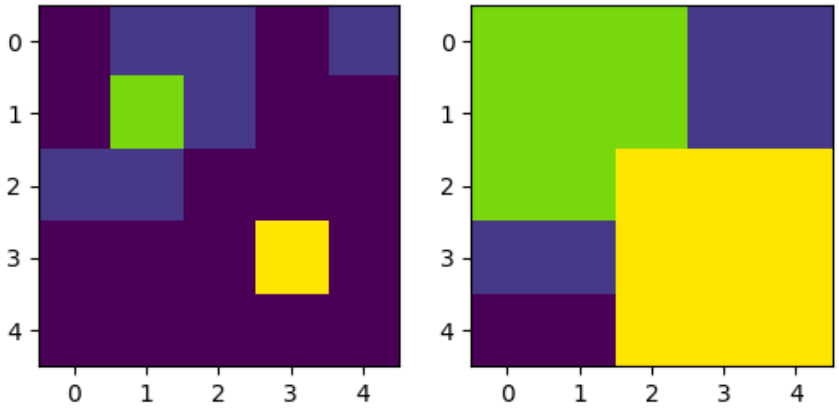


It’s apparent that the highest magnitudes of the gradient are around the contours of the Notre-Dame Cathedral, as indicated by the brighter regions on the gradient magnitude representation. This is because the edges of the building have the most significant changes in intensity compared to the relatively uniform sky or the less textured surfaces of the building's walls.

Part 1: Harris corner detector



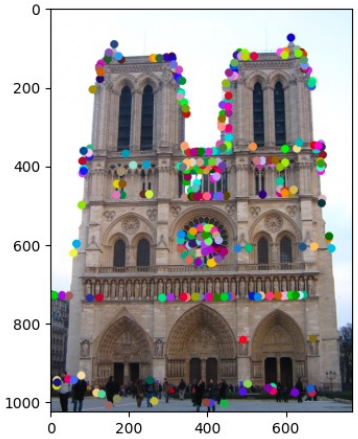
Part 1: Harris corner detector

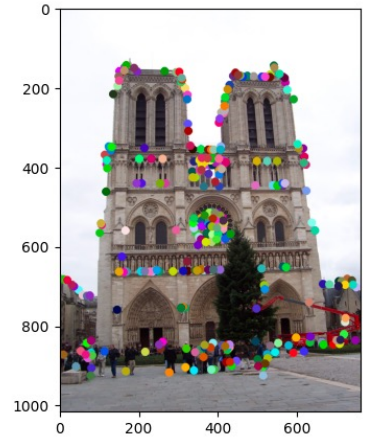


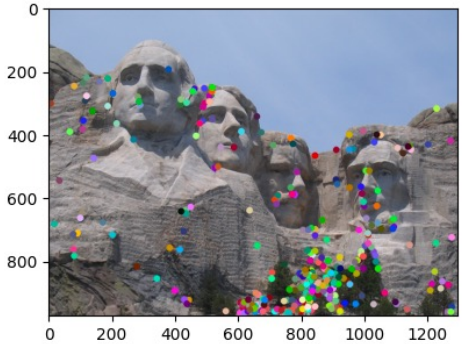
Gradient features are typically invariant to additive shifts in brightness. This is because when you add a constant value to all pixels in an image (additive brightness change), the difference between the pixel intensities (which gradients measure) does not change. Hence, the gradient magnitude remains the same.

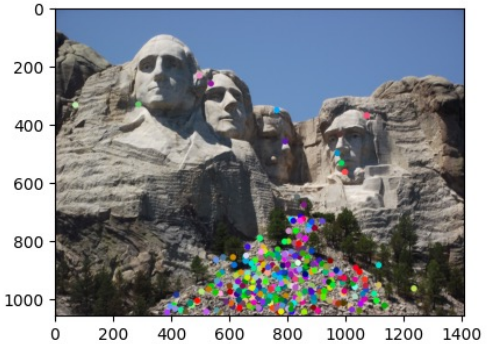
On the other hand, gradient features are not generally invariant to multiplicative changes in contrast. If you multiply all pixel values by a constant (multiplicative contrast change), the differences between pixel intensities are scaled by that constant. This scaling will change the magnitude of the gradient but not the direction. For normalized gradients (where only the direction of the gradient is considered, not the magnitude), the invariance to multiplicative changes might hold.

Part 1: Harris corner detector

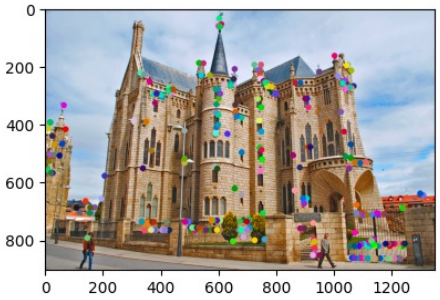


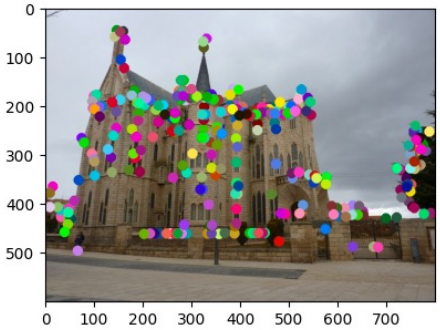






Part 1: Harris corner detector



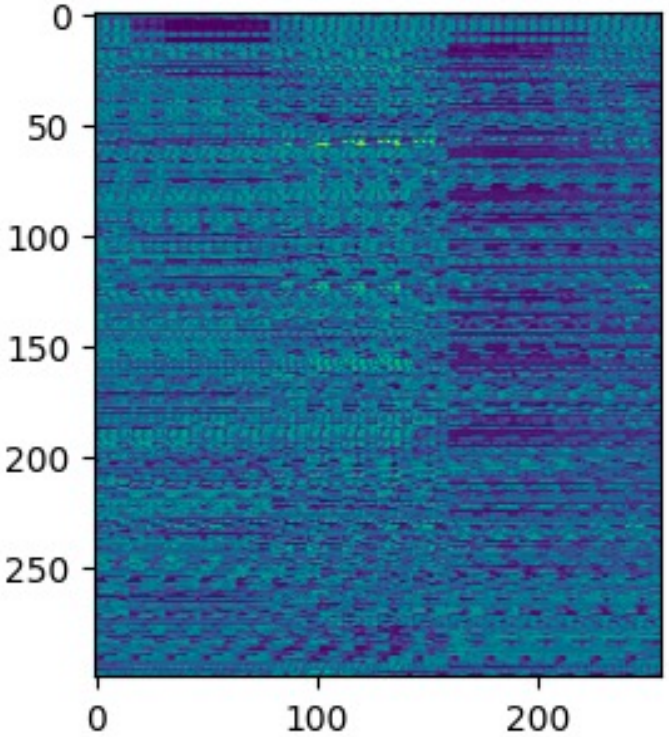


Using maxpooling for NMS in CV tasks like object detection is advantageous due to its efficiency, particularly on GPUs, and its simplicity in implementation, making it suitable for real-time applications. It's also robust to variations and noise, considering only the strongest features in a local neighborhood. However, this approach has its downsides, such as the loss of detailed information since only the maximum value is retained, and a lack of adaptability to varying object scales due to fixed pooling regions, which can lead to missed small objects or oversimplified large ones. Additionally, maxpooling can result in positional inaccuracies, as it may distort the exact location of detected features, reducing the precision of object localization.

Part 1: Harris corner detector

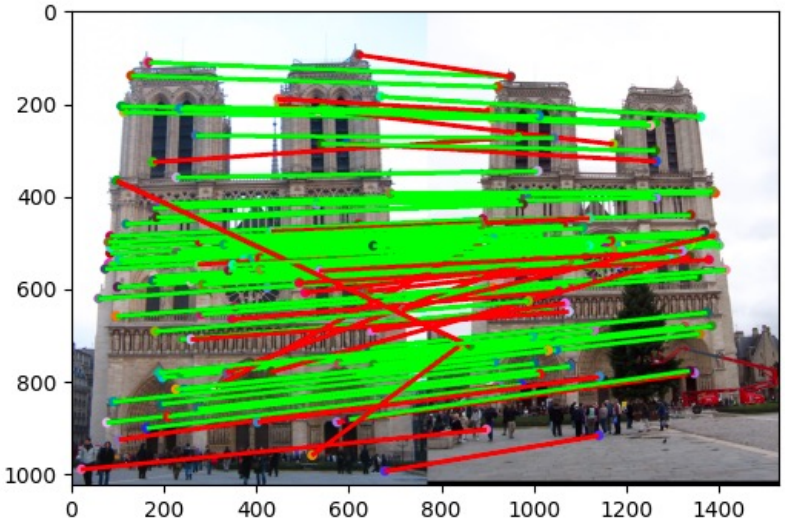
The Harris corner detector is effective because it specifically looks for regions in an image where there are significant changes in intensity in all directions. The intuition behind its effectiveness lies in its ability to identify points that have a unique signature of intensity variation, which corresponds to corners. Corners are interesting features in images as they typically correspond to areas of high information content and are invariant to rotation, scale, and illumination changes to some extent.

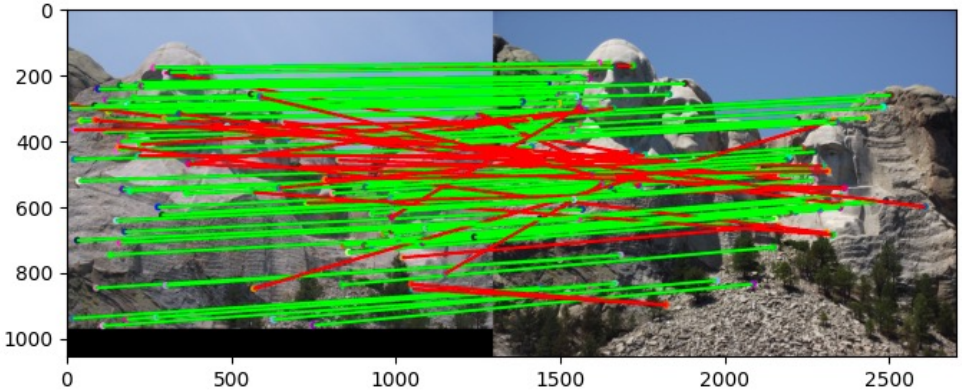
Part 2: Normalized patch feature descriptor



Normalized patches aren't considered very good descriptors due to their sensitivity to lighting and viewpoint changes, their high dimensionality leading to increased computational and storage demands, and their vulnerability to noise amplification. Additionally, the process of normalization might strip away distinctive features necessary for accurate matching, and variations in appearance due to scale or rotation changes can render them unreliable.

Part 3: Feature matching

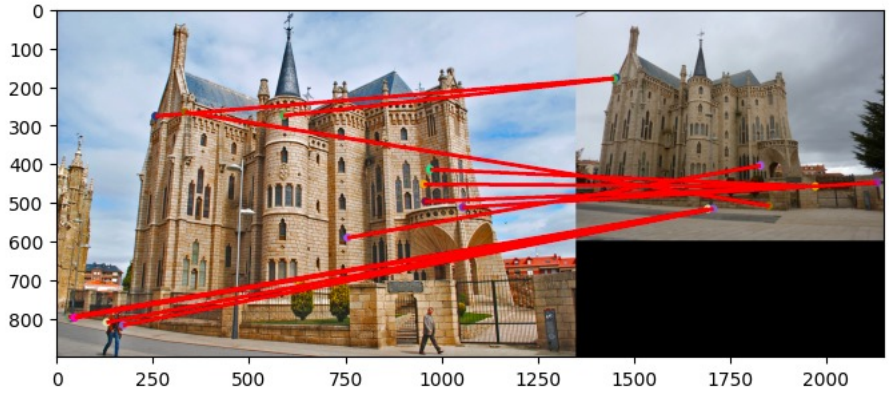





```
You found 107/100 required matches
Accuracy = 0.766355
```

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

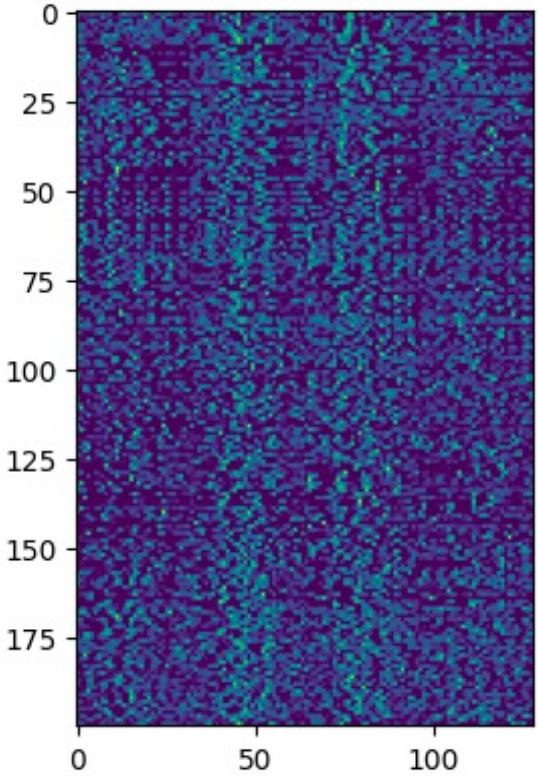
Part 3: Feature matching

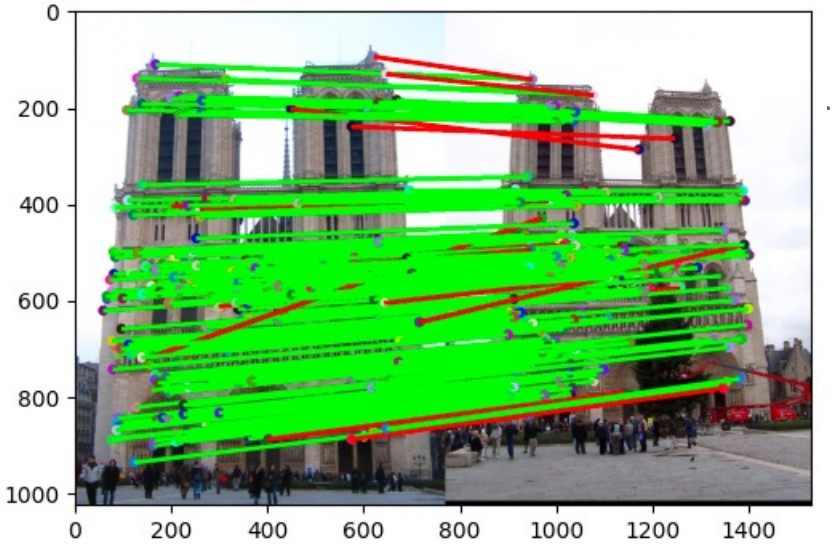



```
You found 12/100 required matches
Accuracy = 0.000000
```

The code implements feature matching by employing the nearest-neighbor distance ratio test, a method that identifies correct matches by examining the relative distances between the closest and second-closest feature matches. The compute_feature_distances function calculates pairwise Euclidean distances between two feature sets using vectorized operations, which ensures efficiency and avoids excessive memory usage. Within the match_features_ratio_test function, it first sorts the distances for each feature in features1 against all features in features2. It then computes the ratio of the smallest to the second smallest distance for each feature, leveraging the ratio test to filter out non-distinctive matches. If the ratio is below a certain threshold (set to 0.8), it indicates a good match. The matches passing this threshold are stored with their corresponding confidence values, which inversely relate to the ratio—lower ratios yield higher confidence. Finally, the matches are sorted by their confidence in descending order to prioritize the most reliable matches. This implementation is highly effective for sifting through repetitive features to find the most distinctive matches, and is computationally efficient, making it well-suited for real-time applications.

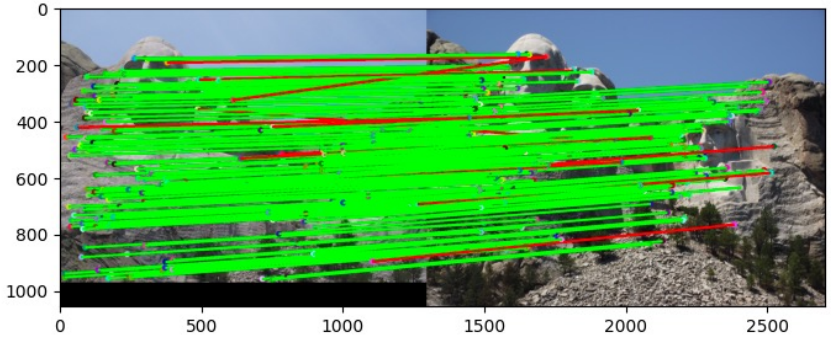
Part 4: SIFT feature descriptor

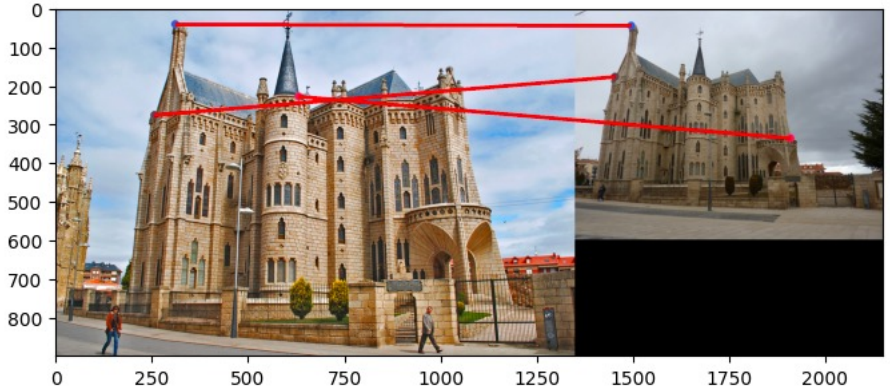





```
You found 197/100 required matches
Accuracy = 0.918782
```

Part 4: SIFT feature descriptor






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

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

Part 4: SIFT feature descriptor

Initially, the function computes the gradients of the grayscale image, obtaining both magnitude and orientation for each pixel. These gradients are essential for capturing the local texture and shape information. The feature vectors are constructed around the given points of interest (X, Y coordinates) by considering the magnitudes and orientations within a window defined by feature_width. Each local region is divided into sub-regions, and for each sub-region, a histogram of orientations is computed. These histograms are concatenated to form a feature vector, which is then normalized to enhance invariance to changes in illumination and contrast. This process results in a 128-dimensional descriptor for each point of interest, capturing the essential local gradient information in a compact, yet descriptive form.

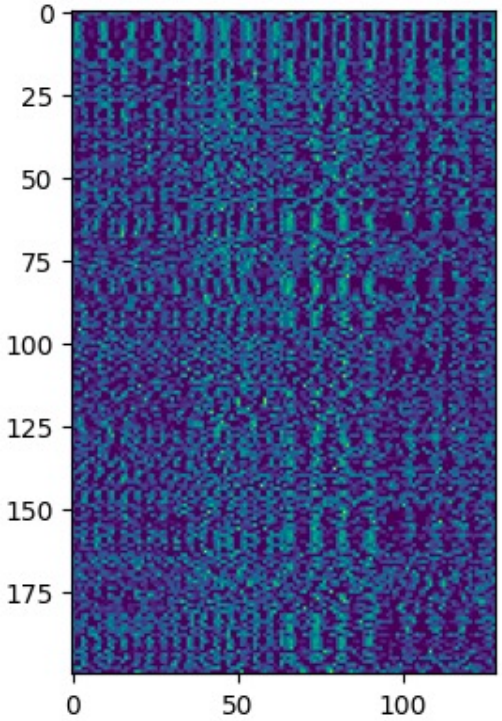
SIFT descriptors are invariant to scale and rotation, meaning they remain effective even when the object is viewed from different angles or distances. They achieve this by constructing orientation histograms over multiple scales and normalizing them. The SIFT descriptor captures more detailed information by encoding the direction and magnitude of gradients, providing a richer representation of the local image structure. SIFT features are less sensitive to changes in illumination and contrast due to their normalization process. Normalized patches, while somewhat robust to linear intensity changes, can lose important shape information and fail to capture the more complex variations in an image. Lastly, SIFT descriptors are more distinctive, which makes them better suited for matching across different images with high levels of accuracy, particularly in scenes with repetitive textures or similar features.

Part 4: SIFT feature descriptor

Our SIFT implementation may perform worse on the given Gaudi image pair compared to the Notre Dame and Mt. Rushmore pairs due to several factors, one of which is the difference in brightness between the images in the pair. Extreme differences in brightness can affect the detection and description of key points. The Gaudi images might have such variations in lighting that the key points are not consistently detected across the pair, leading to a poor match score close to 0, whereas the Notre Dame and Mt. Rushmore pairs likely have more consistent lighting conditions, resulting in better performance with match scores between 70-95%. This suggests that while SIFT is robust in many scenarios, its performance can degrade with substantial changes in brightness, highlighting the importance of consistent illumination for feature matching algorithms.

Part 4: SIFT feature descriptor

It begins by converting the rotation angle from degrees to radians for mathematical operations and calculates the center of the image, which serves as the pivot point for rotation. The rotation matrix is constructed based on the standard rotation formula, taking into account the translation needed to keep the image center stationary. The function then creates a new, empty array to hold the rotated image, ensuring that the dimensions are adjusted to accommodate the rotation if necessary. Pixel-by-pixel, it calculates the corresponding source coordinates in the original image for each point in the rotated image's frame using the inverse of the rotation matrix. If these source coordinates fall within the bounds of the original image, the pixel value is transferred to the corresponding location in the rotated image; otherwise, it's set to zero, representing an empty (black) pixel. This method of looping over each pixel and applying the inverse transformation is computationally intensive but illustrates the fundamental principles of image rotation in computer graphics.



Part4: SIFT feature descriptor

Our version of SIFT features may not be rotation- or scale-invariant since the implementation does not include the orientation assignment and scale-space extrema detection steps that are fundamental to the SIFT algorithm's invariance properties. To achieve rotation invariance, each keypoint must be assigned a consistent orientation based on the dominant direction of the gradient in its local patch. This is typically done by creating a histogram of gradient orientations and selecting the peak direction. To achieve scale invariance, the keypoints need to be detected across multiple scales using a Gaussian pyramid, and the features must be extracted at the scale at which each keypoint was detected. If these steps are not properly implemented, the resulting feature descriptors will not be robust to changes in orientation and scale. To make them so, we would need to integrate the orientation assignment for rotation invariance and ensure that keypoint detection and descriptor extraction are performed in a scale-invariant manner by properly utilizing the Gaussian pyramid and identifying the appropriate scale for each keypoint.

Part 4: Intensity-based Matching

The implementation of intensity-based feature matching provided here operates by comparing fixed-size patches from two images and identifying the pairs with the highest correlation coefficient. The process begins by ensuring both images are of the same size, cropping if necessary. It then normalizes the pixel intensities of both images to a common scale, which aids in comparing images with differing brightness or contrast. The algorithm steps through the first image in strides, extracting patches and comparing each with patches from the second image, also in strides. For each patch in image1, it computes the correlation coefficient against all patches in image2, and records the location of the highest matching patch. This results in a list of coordinates for matched patches, suggesting regions of similarity between the two images.

The normalization of intensities before matching is crucial because it allows for a fair comparison between corresponding patches by mitigating the effects of varying lighting conditions or camera exposure settings. Without normalization, patches with similar structural content but different intensity scales may not match well, leading to poor performance of the matching algorithm. By normalizing, the algorithm can focus on the structural content of the patches rather than their absolute brightness, which increases the robustness of the feature matching.

Part 4: Intensity-based Matching

To enhance the intensity-based matching approach, one could introduce a flexible window size that adapts to the image content, employing larger windows for uniform regions and smaller ones for detailed areas to capture the most relevant features. Adjusting the stride to be smaller or dynamically changing it based on the level of detail could result in finer and potentially more accurate matching, despite a higher computational cost. Incorporating transformation invariance by estimating and compensating for changes in scale or rotation before matching, using more robust correlation metrics like mutual information, and adopting a multi-scale strategy would make the approach more resilient to variations in scale and orientation. Preprocessing steps, such as applying filters to emphasize edges or interest points, followed by a post-processing consistency check like RANSAC to eliminate outliers, could further refine the matching results. Finally, leveraging machine learning models trained on datasets of known matches could significantly improve the system's ability to deal with complex variations in images, pushing the boundaries of traditional intensity-based matching.

Part 4: SIFT vs Intensity-based Matching

It appears that the SIFT-based feature matching results in more consistent and coherent line mappings between the two views of Notre Dame compared to intensity-based. The lines in the SIFT image are predominantly horizontal, indicating that corresponding points between the two images are at similar vertical positions, which is expected given the structure of the building and the likely camera movement. In contrast, the intensity-based matching results in many crisscrossing lines, suggesting a high number of false matches. This indicates that SIFT, with its scale and rotation invariance and distinctiveness, outperforms intensity-based matching, which is more susceptible to changes in lighting, scale, and rotation. These characteristics make SIFT a more robust choice for feature matching in varied imaging conditions, as evidenced by the cleaner, more structured correspondence pattern.

Part 5: SIFT Descriptor Exploration

Tweaking the size of the frame we focus on for feature matching in the algorithm, we see different results. When the window size was decreased, the algorithm excelled in environments with better texture and minimal viewpoint shifts, delivering higher precision in feature matching. However, this smaller window made the algorithm more prone to noise and less capable of handling significant scale and rotation changes. On the other hand, increasing the window size enhanced the robustness of the descriptors, allowing for more effective matching over varied scales and rotations, as well as better noise suppression.

Part 5: SIFT Descriptor Exploration

Adjusting cell density in the feature window had an immense effect on our algorithm’s feature matching performance. More cells sharpen the feature’s portrayal, capturing those subtle textural shifts that allow for sharper matching in intricate scenes. This fineness comes at a cost, though: bulkier feature descriptors that demand more processing power and tend to pick up noise. On the flip side, fewer cells yielded a broader, more generalized descriptor. Quicker to process and less fazed by blockages or mess, these descriptors, with their lower complexity, gather information more broadly.

Part 5: SIFT Descriptor Exploration

Altering the number of orientation bins per histogram in the SIFT descriptor has a notable effect on feature matching performance. Initially, each 4x4 cell in our 16x16 patch used 8 orientation bins, forming a 128-dimensional vector after concatenation. Increasing the number of orientation bins enhanced the descriptor's sensitivity to gradient directions, capturing more detailed orientation information. This increase in granularity improved the matching accuracy for images with complex textures and distinct orientation patterns, as it provided a more nuanced description of the gradient distribution within each cell. However, it also made the algorithm more sensitive to noise and slight variations in orientation, potentially leading to mismatches. On the other hand, decreasing the number of bins resulted in a coarser feature description, which could generalize better over different images and be less sensitive to noise or minor variations in orientation. This could be beneficial for more consistent matching in less textured images or where exact orientation detail was less critical. In our experiments, there was a sweet spot where the number of bins provided sufficient detail for accurate matching while maintaining robustness against noise and minor variations, leading to optimal performance across our dataset.

Part 5: SIFT Descriptor Exploration