# Project # 2: Local Feature Extraction, Detection and Matching
## CSC 391: Introduction to Computer Vision | Dr. Pauca
**Swaran Ponshunmugam**

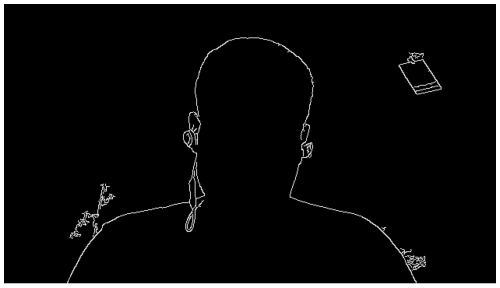
## PART ONE: EDGE AND CORNER DETECTION

### Detecting Edges with Canny Edge Detector

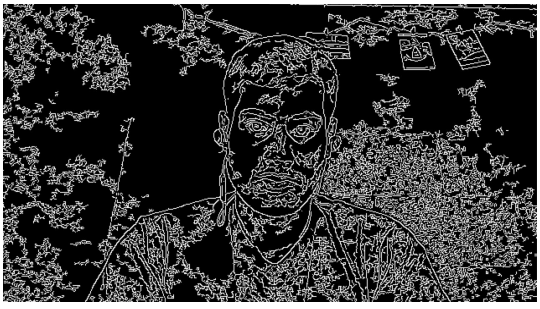
Background

The Canny Edge detection method relies on the logic of the sobel operator to find pixels at which there is a relative change in the gradients of neighboring pixels. In other words, if there a change in the gradient along an image and there is a uniform lining of this change, computers perceive this as an edge. There are benefits and drawbacks to this method. One benefit is that this methodology similarly resembles our cognitive faculty; when humans perceive an image changes in the gradient is used, alongside depth perception for seeing edges. However, this method may fail for a computer because there is a lot of change within the image, thereby making it difficult to pick apart what a human would consider a "good" edge from a "bad" edge. For this reason, Canny implements hysteresis thresholding (instead of streaking). Hysteresis thresholding allows users to specify the ranges for an edge (at what change in gradient intensity is it appropriate for classification of an edge). The range that is equal to or greater than the specified maximum threshold is preserved. The range that is less than or equal to the specified minimum is discarded. The range that is in between is only preserved if and only if it is in connection with a range that is greater than or equal to maximum range. My experimentation first looks at thresholding manipulation. It also includes light (low-light, nature, bright-light), depth (close, medium and far ranges), and movement(none, slow, fast) manipulation.

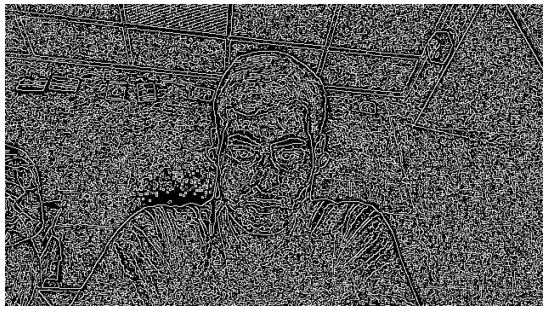
Hysteresis Thresholding Experimentation



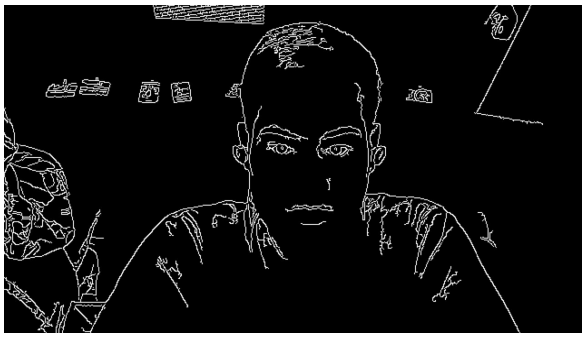
*Hysteresis Thresholding Manipulation Canny (0,1000)*



*Hysteresis Thresholding Manipulation Canny (0,100)*



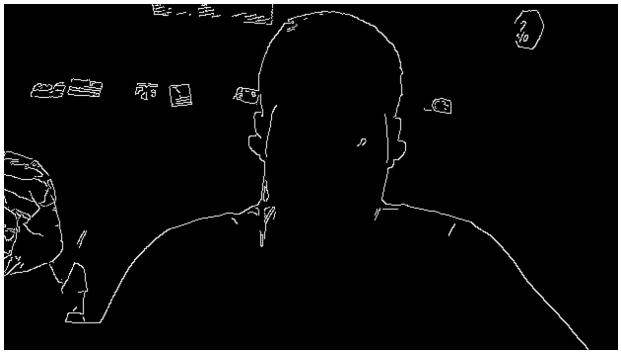
*Hysteresis Thresholding Manipulation Canny (0,1)*



*Hysteresis Thresholding Manipulation Canny (50,200)*

*Hysteresis Thresholding Manipulation Canny (100,150)*

*Hysteresis Thresholding Manipulation Canny 100,200)*



*Hysteresis Thresholding Manipulation Canny (200,300)*

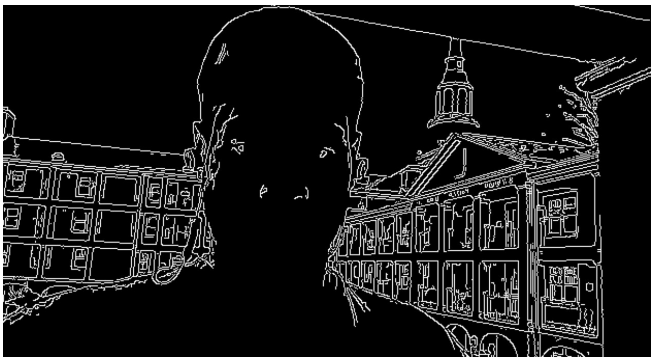
Hysteresis Thresholding Manipulation Analysis

The thresholding proved to be very important when utilizing Canny. When the lower ranges are set to 100, the image is not only easy to make out, but only the very important edges are included. What is even more interesting is how changing the distance between the lower and higher range matters to Canny; when experimenting with the (100,150) and the (100,200) ranges, the (100,200) range seems visibly better. The reason is that although (100,150) gives more detail, a lot of it is facial features and unnecessary noise. With ranges above (100,200), a lot of the edges disappear and almost work to create a silhouette of the image. The lower ranges are far too noise and it almost becomes a tool to see what is not an edge. So, to detect where there is not a lot of gradient change, one should use lower ranges and analyze the black portions of the image (the non-edges), thereby utilizing Canny for the opposite of what it was intended for.

Light, Distance, Motion Experimentation (with (100,200) Hysteresis thresholding range)

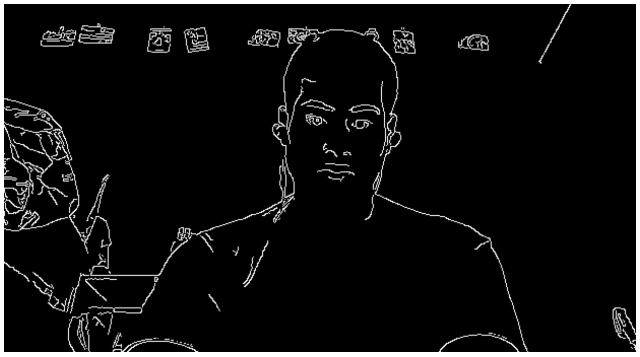
*Bright Lighting Canny*

*Dark Lighting Canny*



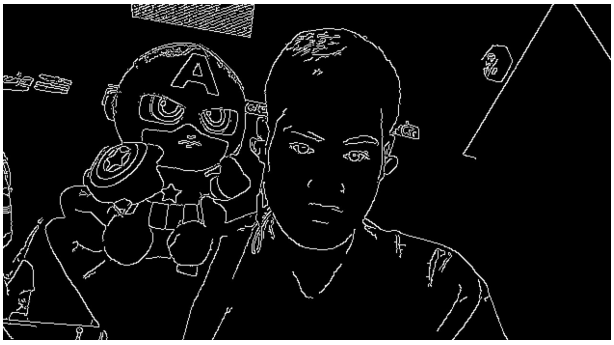
*Outdoor (natural) lighting Canny*

*Close up Canny*

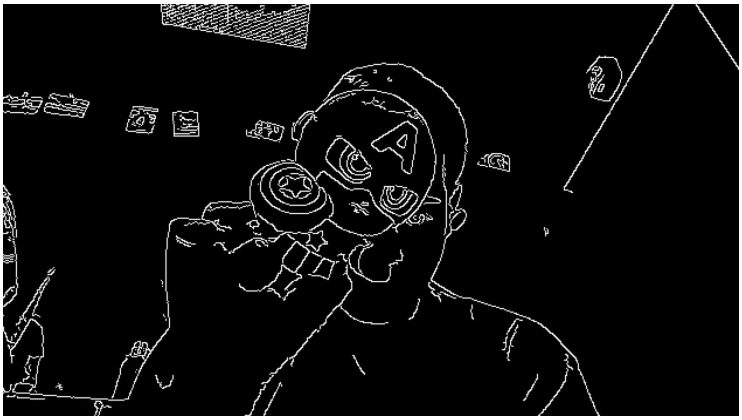


*Medium Distance Canny*

*Far Distance Canny*



*No motion Canny*



*Medium Motion Canny*

*Fast Motion Canny*

Light, Distance, Motion Analysis

Canny proves itself to be useful only in good lighting. That is, when used in little to not lighting, it is not quite able to detect what is a large change in gradient and thereby making it ineffective in areas where gradient differences are well lit. In natural, outdoor lighting, canny did a very good job at detecting far away images if they were large enough. In the natural lighting canny image, one can observe Poteat Residence Hall, 70 feet away.

In distance manipulation, Canny works very well in close up imaging, but as distance increases, detail naturally decreases. However, there is still a good amount of quality retained in the farther images.

With motion, Canny responds well with little to no motion. However, with fast motion, especially motion that oscillates back and forth (Captain America doll moved left to right at very fast pace), Canny is confused what is an edge and mistakenly thinks that the motion is an edge (even though this motion is in fact contributing to a change in gradients).

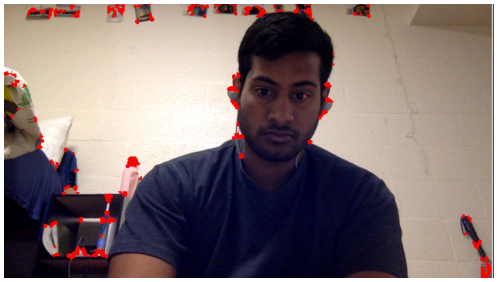
**Corner Detection with Harris Corner Detector**

Background

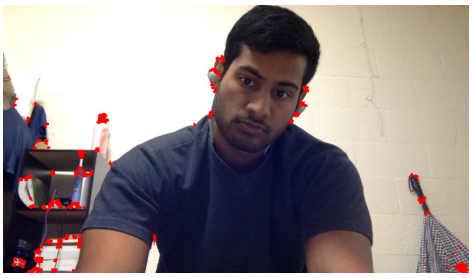
A corner is essentially a change in a portion of an image in all directions. The Harris Corner Detection Algorithm eloquently uses derivatives and matrices to trace points at which there is a change in all directions of a pixel and its neighbors. The implementation used was cv2.cornerHarris.

Experimenting With Harris Corner Detector (Range, Translation, Rotation, Scale)

*Blocksize = 3, kSize = 3*



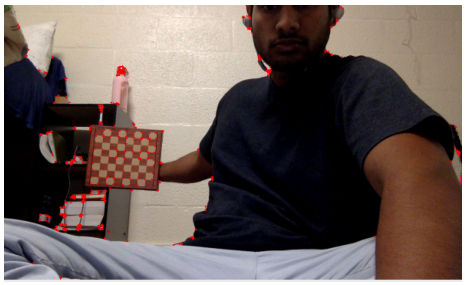
*Blocksize = 3, kSize = 5*



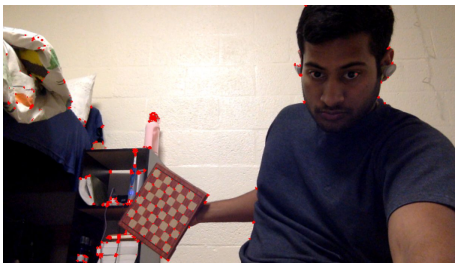
*Blocksize = 5, kSize = 5*

COMPUTER VISION PROJECT 2: LOCAL FEATURE EXTRACTION, MANIPULATION, AND DETECTION

Swaran Ponshunmugam

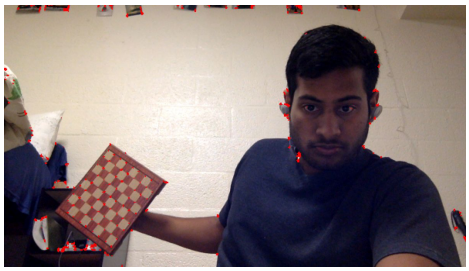


*Translation Harris 1*



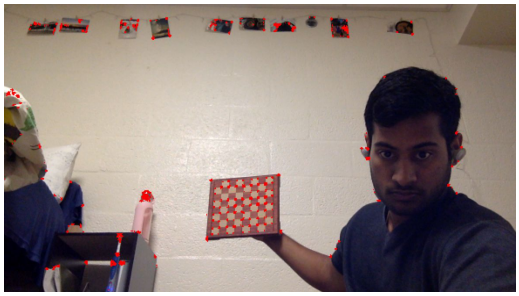
*Translation Harris 2*

*Rotation Harris 1*



*Rotation Harris 2*

*Scale (Near) Harris*



*Scale (Far) Harris*

Harris Corner Analysis

To initially find an ideal range, the second and third parameters of the Harris corner Detector algorithm were manipulated and experimented with. They are the blockSize (neighborhood size) and kSize (aperture of Sobel derivative), respectively. Using (3,5) and (5,5) for these parameters brought in points that were not corners. A block size of 3 and ksize of 3 was found to be ideal.

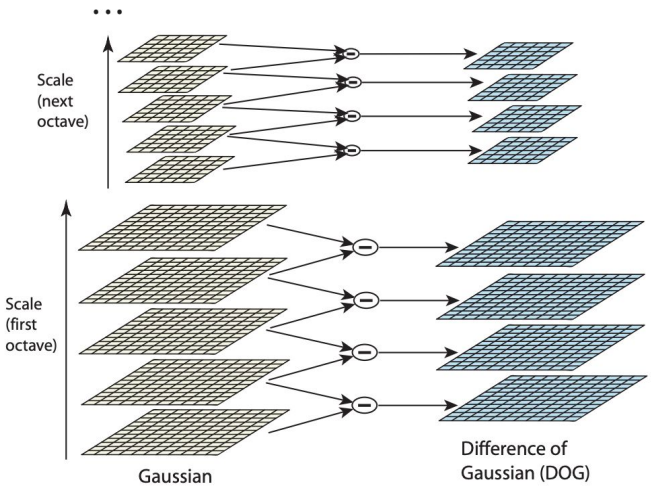
Further experimentation included translation, rotation, and, most importantly, scale. A translation of a chess board did not challenge the math-intense Harris detector. Surprisingly, it was able to follow the chessboard across the video screen. Similarly, the Harris detector was consistent across object rotations. The chessboard was rotated 45 degrees, yet, the Harris corner detector did not show many different corners.

Lastly, scale was manipulated. The Harris Corner detector did very poorly in this aspect. When the chessboard was near, or even at a medium distance, there was no indication of any corners. As soon as the chessboard was far away, the algorithm perfectly detected the corners. One may observe this also when they see the image of the pictures in the top on the wall in the back. Their corners are mapped out when the corners of my body was barely found.

## PART TWO: SIFT DESCRIPTORS AND SCALING

Background

Scale-Invariant Feature Transform (SIFT) Descriptors use an image and covolve repeatedly with Gaussians, subtracting adjacent Gaussians and then producing the difference of the Gaussian, downsampling by a factor of 2 and repeating this process. Then, SIFT extracts key points in analyzing pixels at which the direction of the gradient is different from that of its neighbors.[1]




*Finding the difference of Gaussian*[2]

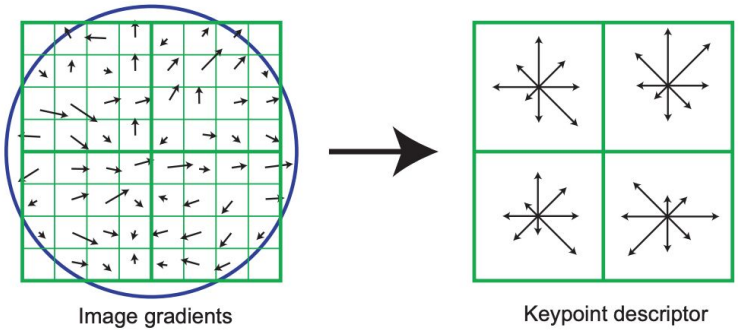
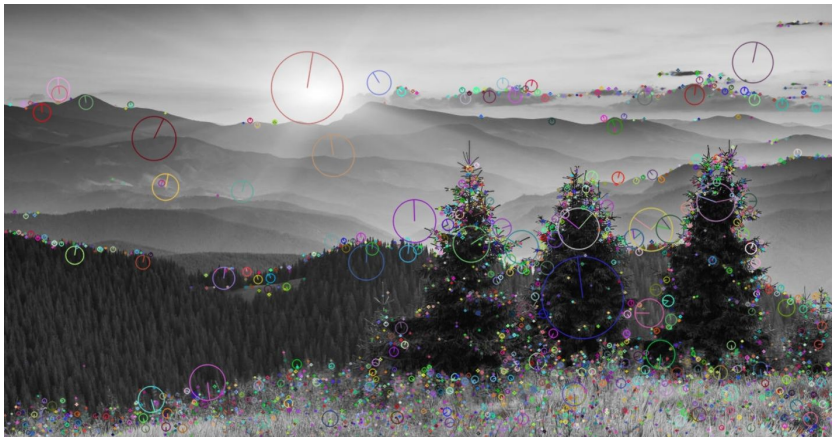



*Image gradients are compared alongside their neighborhood, then used for keypoint detection*[3]

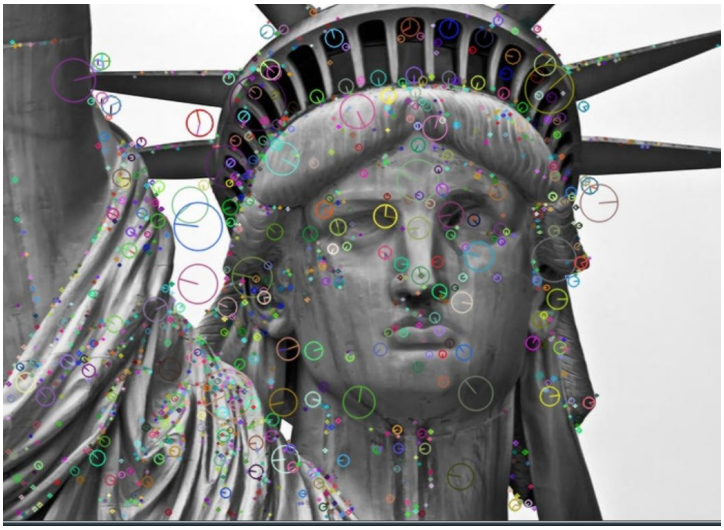
[1] https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf (Seminal paper of SIFT)
[2] ibid
[3] ibid

## SIFT Experimentation with translation, rotation, scale

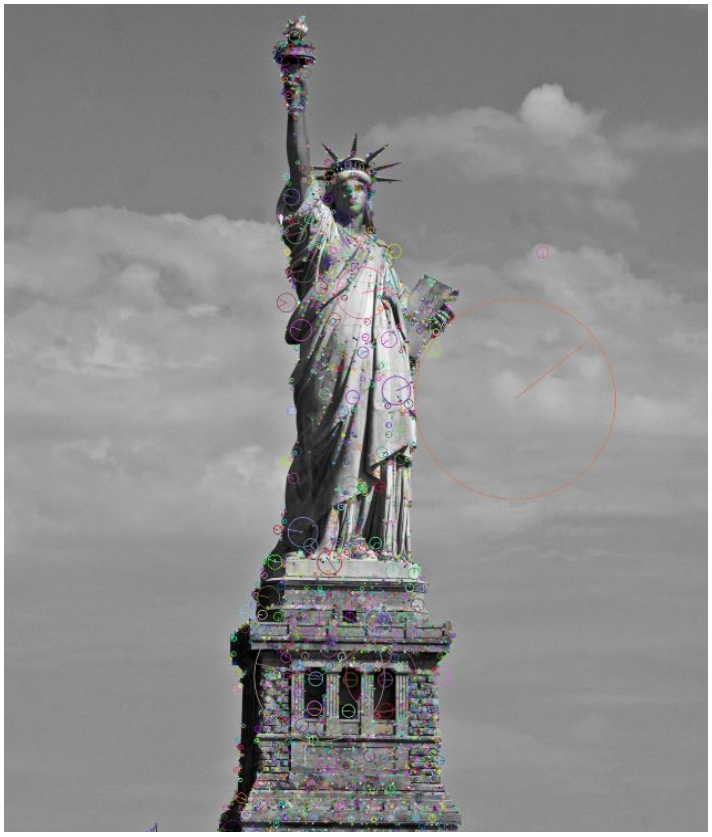


*Three, very similar trees, adjacent to each other, representing transformation*

*Five bananas are rotated with similar keypoint results*

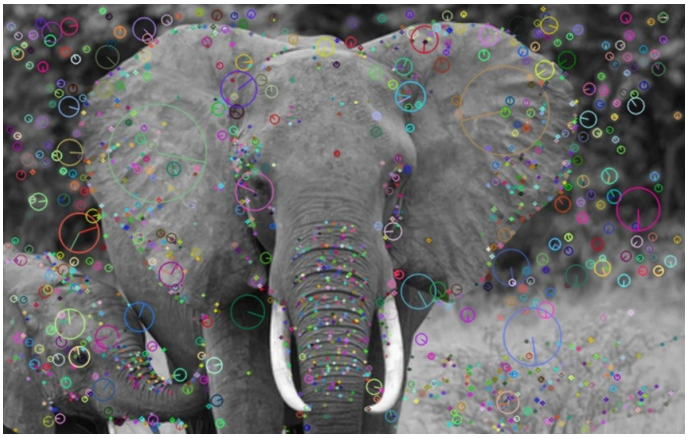


*A closeup image of the Statue of Liberty (Different image from below)*



*Far-away image of Statue of Liberty (Different image from above)*

*Far away image of elephant*



*Close up image of elephant (Same image as above)*

SIFT Analysis

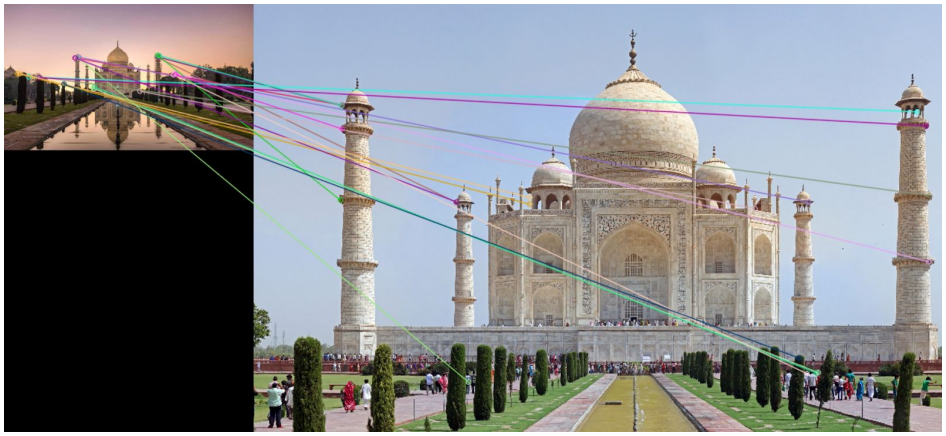
Experimenting with translation, rotation and scaling, I found the SIFT algorithm to be very consistent and reliable in identifying key features despite different changes in the image. As far as the transformation, the algorithm found key features along the lining of the three similar trees. In the image of the rotated bananas, the SIFT detector returned key points that were heavily concentrated towards the tip of the banana and around its core, despite the bananas being at very different angles. Most importantly, SIFT reacts very well across all scaling ranges. What is interesting is that even in the scaling comparison for the Statue of Liberty, two different images were used, but SIFT still detected keypoints that were similar in the face region of Lady Liberty. For the same elephant image at different scalings, SIFT picked up near exactly the same features without fail.

## PART THREE: KEYPOINTS AND MATCHING

Background and Analysis

The SIFT and Harris Corner algorithms work to give us insight into several informative features including keypoints. These keypoints can be used as descriptors to match across two separate images. Harris corner matching is not as effective as SIFT mostly, due to scale variance. However, Harris corner matching is useful when dealing with similar-scale. Harris corners have a strong ability to detect patterns so, when there are images with cornered patterns (like chessboard, or bricks), matching with Harris corners is useful. SIFT is all around better, however because of its ability to take scale into consideration. For example, using Harris corners, the matcher mixed up the pillars of the Taj Mahal, whereas in SIFT, the pillars were matched almost fully correctly

Experimentation



*Harris corner matching*

*Sift corner Matching*