Homework 4

Theory Question

Let the Gaussian function be denoted by:

$$g(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

And the Laplacian function:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

We apply the Gaussian function to the image (denoted as f), resulting in a Gaussian smoothed image, which we will denote with ff:

$$ff(x,y,\sigma) = \iint_{-\infty}^{\infty} f(x',y')g(x-x',y-y')dx'dy'$$

The Gaussian function and the Laplacian can be combined to form a new operator, this is the LoG operator:

$$h(x,y) = \frac{-1}{2\pi\sigma^4}\left(2 - \frac{x^2+y^2}{\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Follows from this that:

$$\nabla^2 ff(x,y,\sigma) = f(x,y) * h(x,y,\sigma)$$

By taking the derivative of ff w.r.t σ results in:

$$\frac{\partial}{\partial\sigma}ff(x,y,\sigma) = -\frac{\sigma}{2\pi\sigma^4}\iint f(x',y')\left(2 - \frac{(x-x')^2+(y-y')^2}{\sigma^2}\right)\exp\left(-\frac{(x-x')^2+(y-y')^2}{2\sigma^2}\right)dxdy$$

$$\frac{\partial}{\partial\sigma}ff(x,y,\sigma) = \sigma f(x,y) * h(x,y)$$

The LoG of an image is defined as:

$$\text{LoG}(f(x,y)) = \nabla^2 ff(x,y,\sigma)$$

So we can write the following

$$\text{LoG}(f(x,y)) = \frac{\partial}{\partial\sigma}ff(x,y,\sigma)$$

This partial derivative can be expressed as a discretized version using finite differences:

$$\frac{\partial}{\partial\sigma}ff(x,y,\sigma) = \frac{ff(x,y,\sigma+\Delta\sigma) - ff(x,y,\sigma)}{\Delta\sigma}$$

Since both ff are Gaussians, the difference between them, expressed in the equation above, is called Difference-of-Gaussians. This is a less compute intensive task, since the Gaussian operator can be decomposed into x and y parts. This means that instead of applying a 2D smoothing, we can apply two 1D smoothings, one over the x-axis and the other over the y-axis. DoG also needs a smaller operator for the convolution as compared to LoG.

Task 1

Harris Corner Detection

From this point on, we need to operate with the gray levels in an image. Since we have RGB images, we first begin by getting their corresponding grayscale images by using this formula (taken from OpenCV color conversion documentation):

$$\text{grayscale} = 0.299 \times \text{red} + 0.587 \times \text{green} + 0.114 \times \text{blue}$$

To start, we define Haar filters at a scale σ. This means that the size of our Haar filters will be the smallest even integer bigger than 4σ. For example, for $\sigma = 0.8$, our Haar filter (along x) can be represented by a 4×4 matrix like this

$$\begin{pmatrix} -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$$

And along y as this:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{pmatrix}$$

Our input image is convolved with both Haar filters, along x and along y, resulting in d_x and d_y respectively. The define the matrix $\mathbf{C}$, which is constructed in a $5\sigma \times 5\sigma$ neighborhood of the pixel where we want to identify the presence or absence of a corner:

$$\mathbf{C} = \begin{pmatrix} \sum d_x^2 & \sum d_x d_y \\ \sum d_x d_y & \sum d_y^2 \end{pmatrix}$$

Where d_x^2 is the result after the convolution of d_x and the x-Haar filter. Likewise, d_y^2 is the result after convolving d_y and the y-Haar filter. The other number, $d_x d_y$ is the the result of the convolution of the x-Haar filter and d_y.
Let λ_1 and λ_2 be the two eigenvalues of $\mathbf{C}$. Recognize that $\text{Tr}(\mathbf{C}) = \sum d_x^2 + \sum d_y^2 = \lambda_1 + \lambda_2$ and $\det(\mathbf{C}) = \sum d_x^2 \sum d_y^2 - \left(\sum d_x d_y\right)^2 = \lambda_1 \lambda_2$. This means that

$$\frac{\det(\mathbf{C})}{(\text{Tr}(\mathbf{C}))^2} = \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)^2}$$

We introduce the Harris response, defined by:

$$R = \det(\mathbf{C}) - k(\text{Tr}(\mathbf{C}))^2$$

In our case, k will be 0.05, but it can take any value between 0.04 and 0.06. We found that $k = 0.05$ generated better results. The threshold we will use for the Harris response is 80% of the maximum value of R in the image. Any pixel with an R value higher than the threshold we will consider as a corner.

After thresholding and finding the possible corners in the image, we need to apply non-maximum suppression to only keep the the corner with the highest Harris response in a window. For this we just loop through the Harris resposne array and keep only the highest value corner in a window. For the size of the window, we use one double the size of 5σ, so 10σ.

SSD: Sum of squared differences We can establish correspondences between corners by directly comparing the gray levels in a $(m+1) \times (m+1)$ window around the corner pixel in an image, and the gray levels in a window of similar size around the corner in the other image. One such metric is SSD, or sum of squared differences. It is defined by

$$\text{SSD} = \sum_i \sum_j |f_1(i,j) - f_2(i,j)|^2$$

Where f_1, f_2 are the first and second image, respectively. The lower the SSD, the closer the corners are to being the same. We want to find the minimum SSD between a corner from image 1 and a corner from image 2 to establish that correspondence.

NCC: Normalized cross-correlation Aside from SSD, there is another metric we can use to find correspondences between corners in 2 images in a $(m+1) \times (m+1)$ window. This NCC is defined by:

$$\text{NCC} = \frac{\sum_i \sum_j \left((f_1(i,j) - \mu_1)(f_2(i,j) - \mu_2)\right)}{\sqrt{\left[\sum_i \sum_j \left(f_1(i,j) - \mu_1\right)^2\right] \left[\sum_i \sum_j \left(f_2(i,j) - \mu_2\right)^2\right]}}$$

Where μ_1, μ_2 are the grayscale means of the first and second image respectively. Given that this is a cross-correlation, the higher the NCC, the closer those corners are to being the same. For this, we want to find the maximum value for that pair to be a correspondence. In order to simplify the code, we will consider a different value, $1 - \text{NCC}$, which will be minimized for a corner pair. This way both NCC and SSD are minimized.
For both NCC and SSD we tested a few window sizes, more on that in the results.

SIFT: Scale invariant feature transform

The SIFT algorithm consists of 6 steps.

1. An image is smoothed with scale σ. A DoG pyramid is constructed via repeated smoothing and downscaling of the image and substracting it to the previous one.

2. We identify local extrema by comparing the pixel and its 8 neighbors with the corresponding 9 pixels in the DoG that is higher and 9 pixels in the lower DoG (creates a $3 \times 3 \times 3$ region).

3. The localization of the extrema is refined with sub-pixel accuracy by using the Taylor series expansion in the vicinity of the previously found location ($\mathbf{x}_0$), resulting in the true location of the extremum being:

$$\mathbf{x} = -\mathbf{H}^{-1}(\mathbf{x}_0) \cdot \mathbf{J}(\mathbf{x}_0)$$

Where **H** is the Hessian matrix and **J** is the Jacobian matrix.

4. We threshold $s|\mathbf{D}(\mathbf{x})|$ to only take values ≥ 0.03. Additionally, we apply Harris corner detection and reject the extrema that draws their support from an edge.

5. We find the dominant local orientation by calculating the gradient vector magnitude and orientation of the Gaussian smoothed image at scale σ of the extremum.

6. Finally, we create a SIFT descriptor with each retained extremum in the DoG pyramid. Each extremum is surrounded by 16×16 pixels in its neighborhood, which are now divided into 4×4 cells, for each of these cells a 8-bin histogram is calculated from the weighted gradient magnitude. This results in a 128 element descriptor for each of the extrema.

Results

Task 1: Harris Corner Detection and SSD/NCC correspondence

(a) Temple image 1.

(b) Temple image 2.

(c) Hovde Hall image 2.

(d) Hovde Hall image 3.

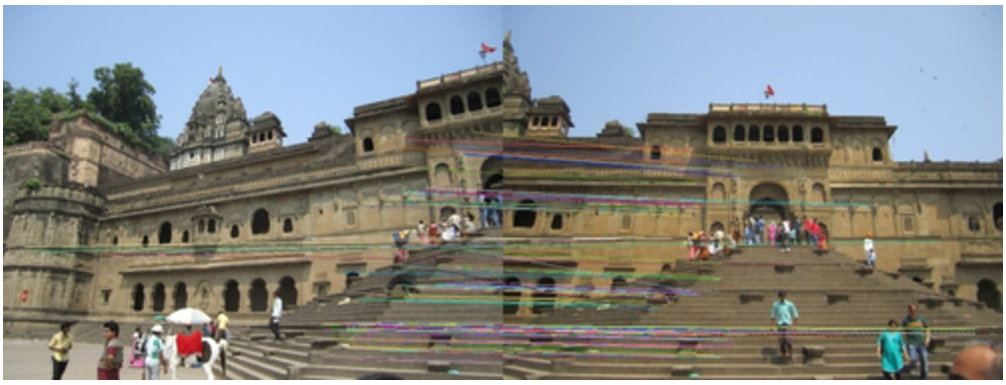
Figure 1: Input images.

Only the 100 corners with the highest Harris response are shown in the images. The following results show that as the scale (σ) increases, the number of corners detected decrease. Most

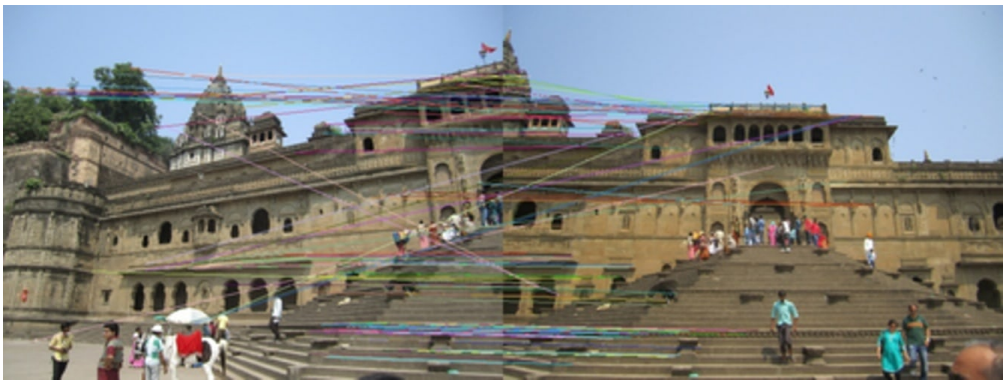
of the corners that have been detected are not actual corners at all, and this can be seen for all scales tested. This is interesting since one would expect the sharp corners of the buildings with the sky as backdrop to have the highest Harris response, yet this is not what happens. In both the Hovde hall and the temple images, we can see that at $\sigma < 1.6$ we have some corners detected in the stairs area, and these are also within the highest 100 values, yet at $\sigma \geq 1.6$, these start to disappear, and are almost completely gone at $\sigma = 2.0$.

(a) *temple_1* image corners.

(b) *temple_2* image corners.



(c) SSD Correspondence between *temple_1* and *temple_2*

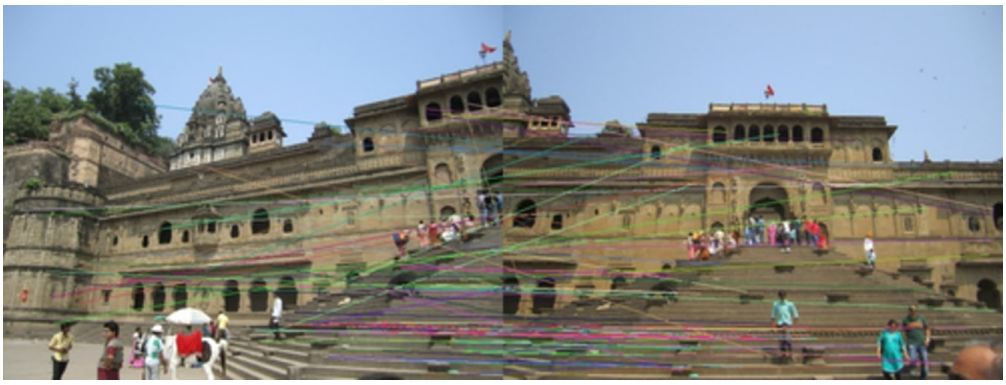


(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 2: Harris corner detection and correspondences with $\sigma = 0.8$.

As for matching, we use a window of size 41 around the detected corner. The reason for

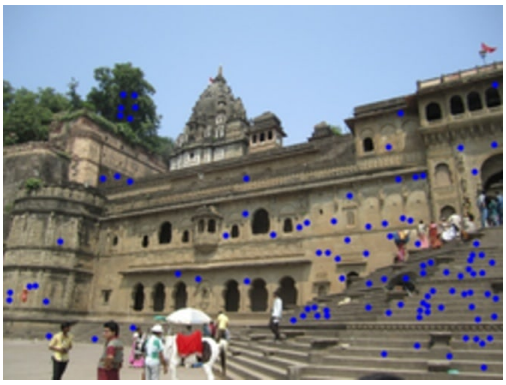
this is because there is far less mismatches with a window size of 41 compared to a lower one. In Fig. 3a and Fig. 3b we can see the difference in matching quality when using different window sizes. The bigger the window size, the better the matches become. Look at how there is a significant increase in mismatches with a window size of 9 (for the matching comparison). So in all the correspondence images we only use window sizes of 41. When getting the first few results for matching, we noticed that there were many corners that got matched with more than one corner in the other image and viceversa. To solve this, we made it so that a corner from image 1 can only be matched with one corner from image 2. If a corner from image 1 makes a match with a corner in image 2 that is already taken then we ignore it.
It seems like SSD matches are of higher quality compared to NCC. This happens at all scales tested, where NCC will create worse matchings, but it is more notorious on low scales such as $\sigma < 1.6$. Even with SSD matches being better than NCC, there are still noticeable mismatches, and these are more apparent on higher values of σ, as there are less keypoints. In other words, more keypoints allows for better matching, even when not all keypoints are of good quality.



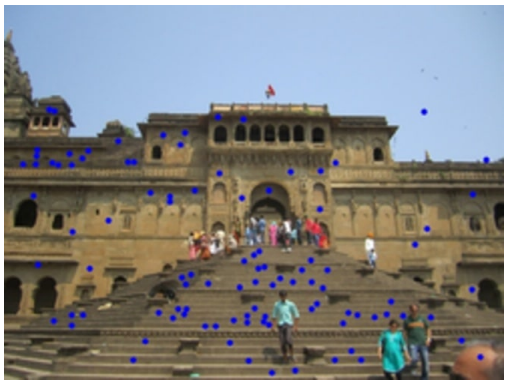
(a) SSD matching with window of size 9.

(b) SSD matching with window of size 41.

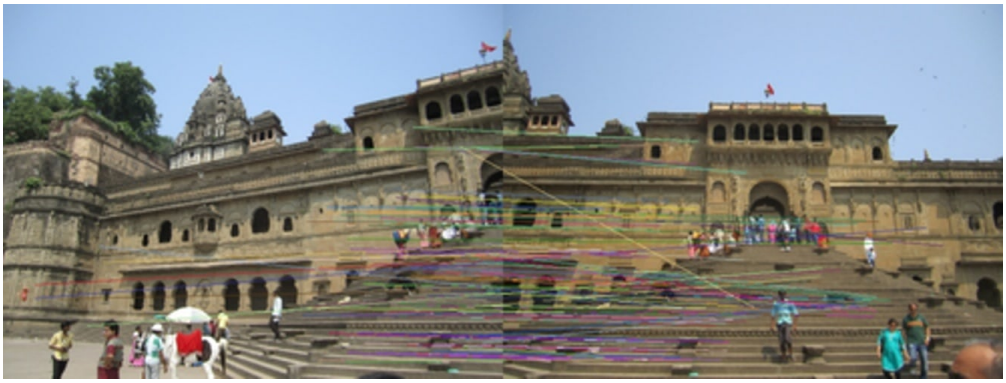
Figure 3: SSD matches with different window sizes.



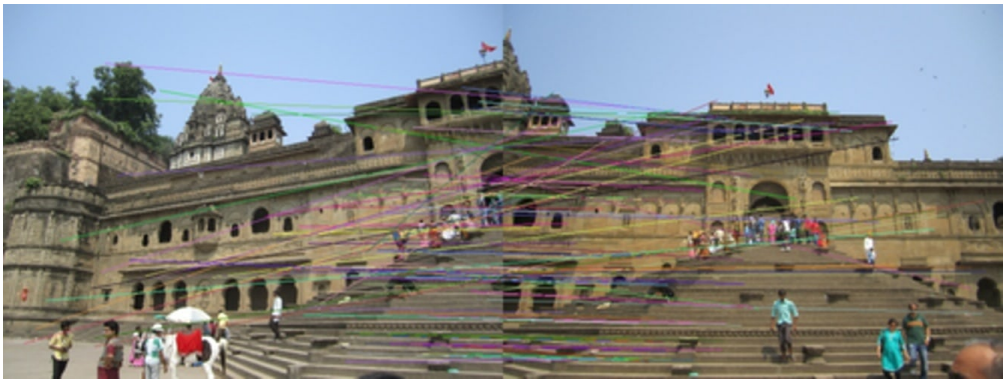
(a) *temple_1* image corners.



(b) *temple_2* image corners.



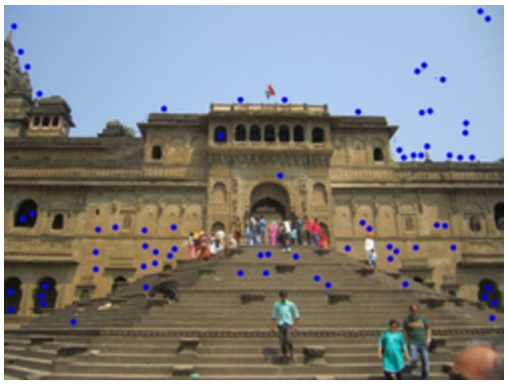
(c) SSD Correspondence between *temple_1* and *temple_2*



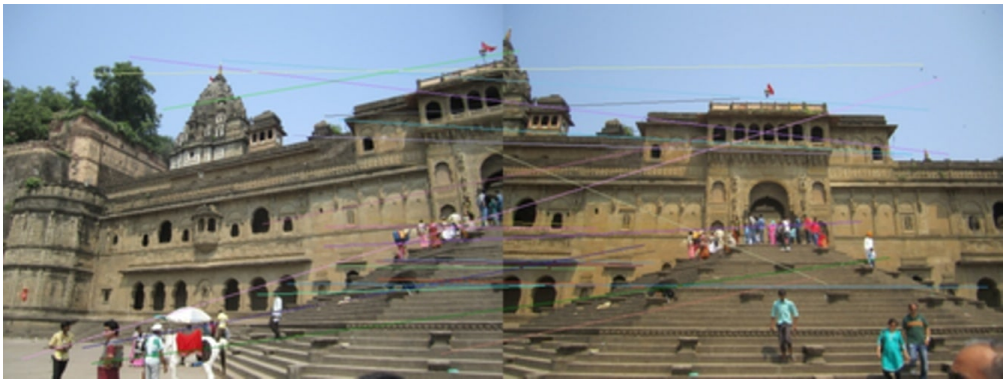
(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 4: Harris corner detection and correspondences with $\sigma = 1.2$.

(a) *temple_1* image corners.



(b) *temple_2* image corners.



(c) SSD Correspondence between *temple_1* and *temple_2*

(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 5: Harris corner detection and correspondences with $\sigma = 1.6$.

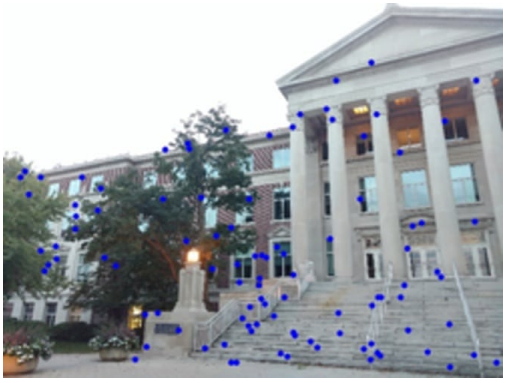
(a) *temple_1* image corners.

(b) *temple_2* image corners.

(c) SSD Correspondence between *temple_1* and *temple_2*

(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 6: Harris corner detection and correspondences with $\sigma = 2.0$.



(a) *hovde_2* image corners.

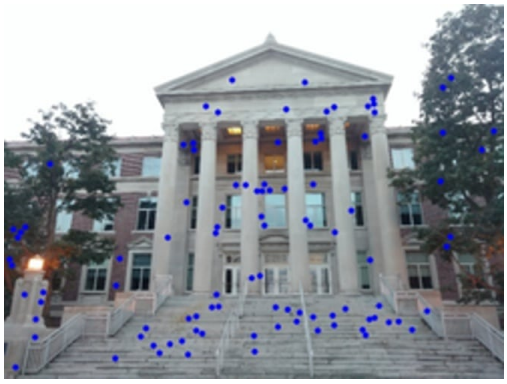
(b) *hovde_3* image corners.

(c) SSD Correspondence between *hovde_2* and *hovde_3*

(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 7: Harris corner detection and correspondences with $\sigma = 0.8$.

(a) *hovde_2* image corners.



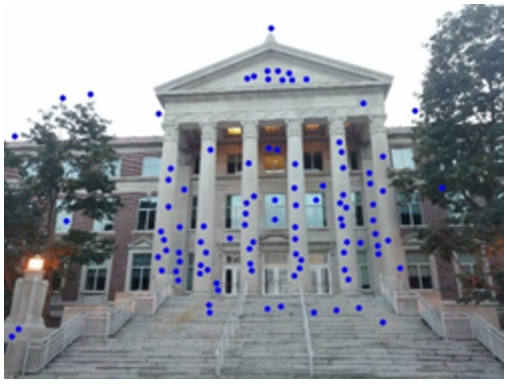
(b) *hovde_3* image corners.

(c) SSD Correspondence between *hovde_2* and *hovde_3*

(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 8: Harris corner detection and correspondences with $\sigma = 1.2$.

(a) *hovde_2* image corners.



(b) *hovde_3* image corners.

(c) SSD Correspondence between *hovde_2* and *hovde_3*

(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 9: Harris corner detection and correspondences with $\sigma = 1.6$.

(a) *hovde_2* image corners.

(b) *hovde_3* image corners.

(c) SSD Correspondence between *hovde_2* and *hovde_3*

(d) NCC Correspondence between *temple_1* and *temple_2*

Figure 10: Harris corner detection and correspondences with $\sigma = 2.0$.

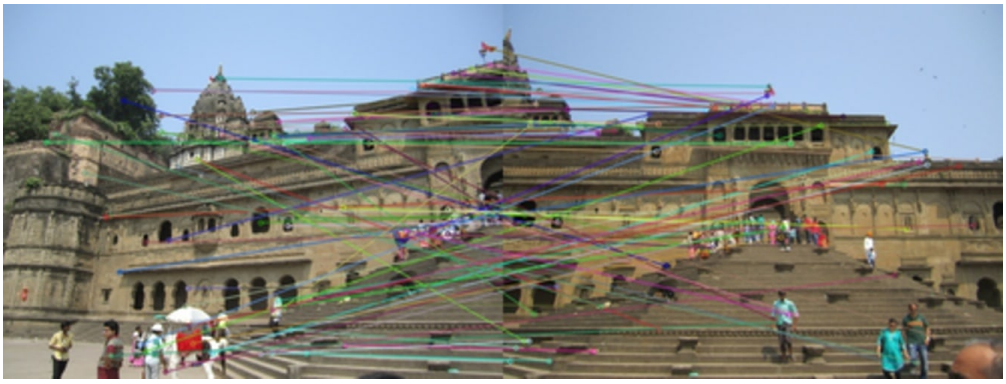
Task 1: SIFT Keypoints and Matching

The keypoints generated by SIFT look better than with our Harris corner detector implementation. However, there is some issues, mainly with, in the case of the temple images, a lot of the keypoints are detected where there is a lot of people, rather than the background, which ultimately is what we want to match. In the case of the images of Hovde hall, something similar happens, there is a large portion of the keypoints that are in the trees, which

do not show up on both images.
As for the keypoint matching, this seems to do better than our implementation. Notice how most of the matches are within the area of the background that is common to both images, more noticeable in the Hovde Hall images. There are still a few mismatches, as observed with a sharp diagonal line.

(a) *temple_1* SIFT keypoints.

(b) *temple_2* SIFT keypoints.



(c) SIFT matches

Figure 11: SIFT keypoints and matches on the temple images.

(a) *hovde_1* SIFT keypoints.

(b) *hovde_2* SIFT keypoints.

(c) SIFT matches

Figure 12: SIFT keypoints and matches on the Hovde hall images.

Task 1: SuperPoint and SuperGlue

The deep learning based keypoint detection and matching have generated the best results out of everything tested. We can see how only the areas that are common to both images have matching keypoints, while the rest don't. It has also generated the most amount of keypoints, and they do seem like they are on corners, or other areas of interest, compared to SIFT or Harris corner detection. It also doesn't generate wrong matchings, we can observe that by seeing some keypoints that are not matched but are close to others that do have a match (Hovde Hall near the top, there is a red dot close to several matches).

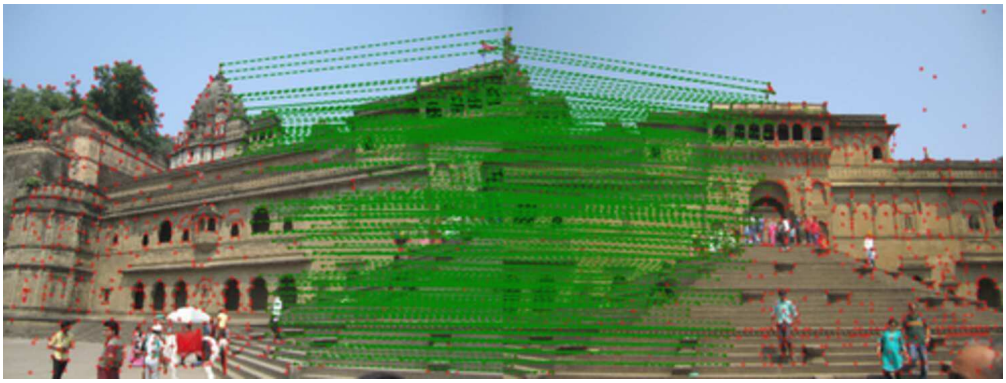


Figure 13: Temple image matching with SuperGlue

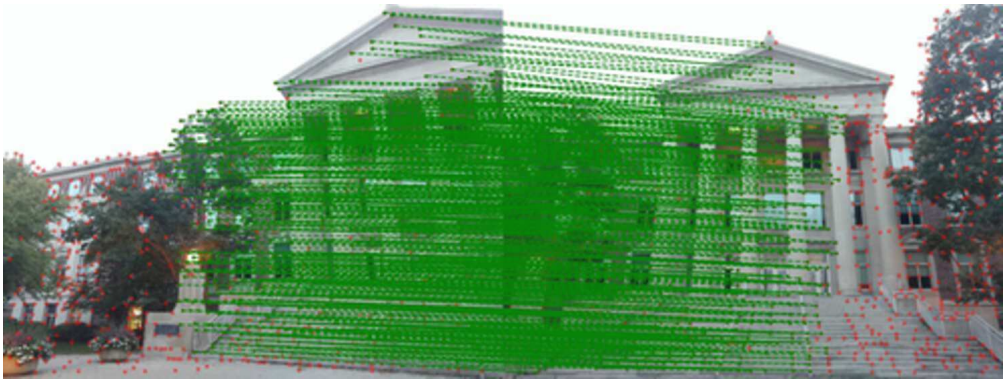


Figure 14: Hovde Hall image matching with SuperGlue

All in all, SuperPoint and SuperGlue have generated the best results out of our implementation of Harris corner detection, SIFT and SuperPoint+SuperGlue.

Task 2: Harris Corner Detection and SSD/NCC correspondence

(a) Keyboard image 1.

(b) Keyboard image 2.

(c) Building image 2.

(d) Building image 3.

Figure 15: Input images.

One interesting thing that happened here with the keyboard images is that NCC did matched very few keypoints in the dark areas under the keyboard, compared to SSD. It did eventually start to match some of them but at higher values of σ. Both NCC and SSD seem to create many mismatches at all scales. So they did not work that well on these images.

(a) *keyboard_1* image corners.

(b) *keyboard_2* image corners.

(c) SSD Correspondence between *keyboard_1* and *keyboard_2*

(d) NCC Correspondence between *keyboard_1* and *keyboard_2*

Figure 16: Harris corner detection and correspondences with $\sigma = 0.8$.

(a) *keyboard_1* image corners.

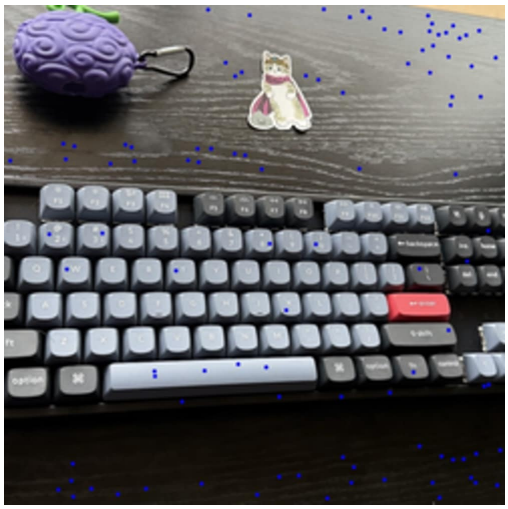
(b) *temple_2* image corners.

(c) SSD Correspondence between *keyboard_1* and *keyboard_2*

(d) NCC Correspondence between *keyboard_1* and *keyboard_2*

Figure 17: Harris corner detection and correspondences with $\sigma = 1.2$.

(a) *keyboard_1* image corners.



(b) *keyboard_2* image corners.

(c) SSD Correspondence between *keyboard_1* and *keyboard_2*

(d) NCC Correspondence between *keyboard_1* and *keyboard_2*

Figure 18: Harris corner detection and correspondences with $\sigma = 1.6$.

(a) *keyboard_1* image corners.

(b) *keyboard_2* image corners.

(c) SSD Correspondence between *keyboard_1* and *keyboard_2*

(d) NCC Correspondence between *keyboard_1* and *keyboard_2*

Figure 19: Harris corner detection and correspondences with $\sigma = 2.0$.

(a) *building_1* image corners.

(b) *building_2* image corners.

(c) SSD Correspondence between *building_1* and *building_2*

(d) NCC Correspondence between *building_1* and *building_2*

Figure 20: Harris corner detection and correspondences with $\sigma = 0.8$.

(a) *building_1* image corners.

(b) *temple_2* image corners.

(c) SSD Correspondence between *building_1* and *building_2*

(d) NCC Correspondence between *building_1* and *building_2*

Figure 21: Harris corner detection and correspondences with $\sigma = 1.2$.

(a) *building_1* image corners.

(b) *building_2* image corners.

(c) SSD Correspondence between *building_1* and *building_2*

(d) NCC Correspondence between *building_1* and *building_2*

Figure 22: Harris corner detection and correspondences with $\sigma = 1.6$.

(a) *building_1* image corners.

(b) *building_2* image corners.

(c) SSD Correspondence between *building_1* and *building_2*

(d) NCC Correspondence between *building_1* and *building_2*

Figure 23: Harris corner detection and correspondences with $\sigma = 2.0$.

Task 2: SIFT Keypoints and Matching

It is a similar story with SIFT. While the keypoints look a bit better than with our Harris corner detector, it still mismatched many of them.

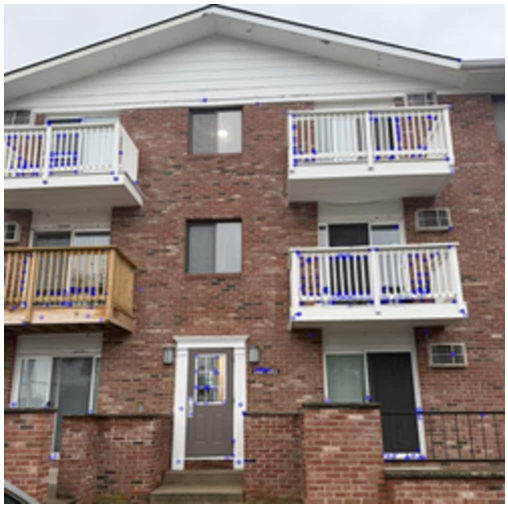
(a) *keyboard_1* SIFT keypoints.

(b) *keyboard_2* SIFT keypoints.

(c) SIFT matches

Figure 24: SIFT keypoints and matches on the keyboard images.

(a) *building_1* SIFT keypoints.



(b) *building_2* SIFT keypoints.

(c) SIFT matches

Figure 25: SIFT keypoints and matches on the building images.

Task 2: SuperPoint and SuperGlue

By far and away the best one out of the 3. Not only did it correctly match all the areas in the image, it also seemed to ignore the darker areas due to the difference in lighting in the keyboard image. It has the highest number of correct matches, in such a way that it is visibly much better.

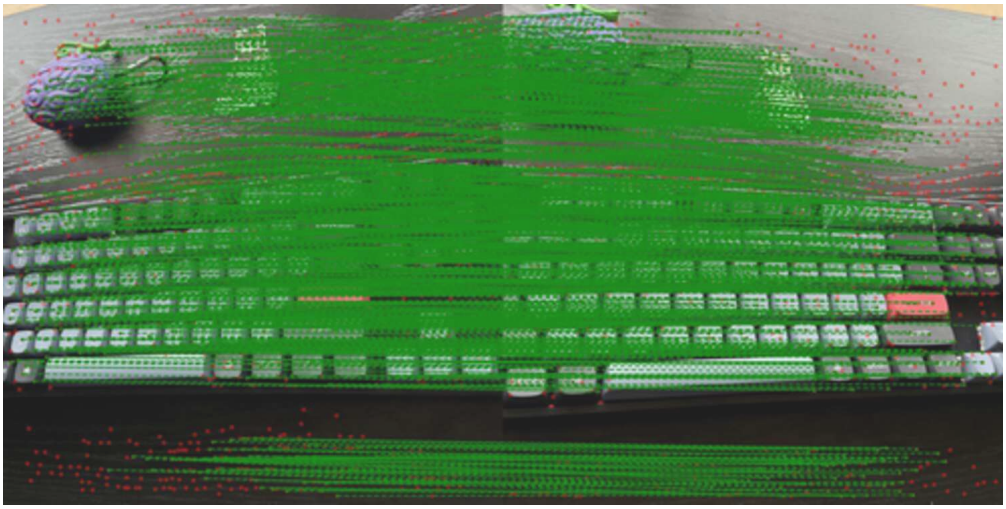


Figure 26: Keyboard image matching with SuperGlue

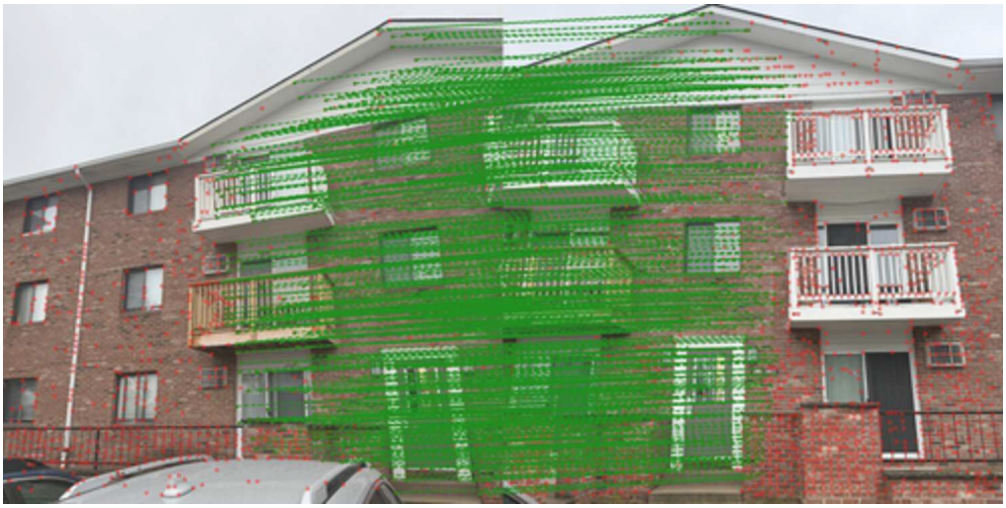


Figure 27: Building image matching with SuperGlue