AVD624 - Computer Vision

Programming Assignment - 1

Image Stitching

Submitted by

Vatsalya Gupta
SC19B098
B.Tech. ECE VI Semester

Department of Avionics
Indian Institute of Space Science and Technology
Thiruvananthapuram - 695 547
February 2022

1 2D Convolution

The main script is *q1_kernels.m*. The Sobel and Gaussian kernel are generated using the user-defined functions *sobel_kernel.m* and *gaussian_kernel.m* respectively. The 5 basic Haar-like masks are defined in the main script. Sobel vertical and horizonal kernel are 3x3 and 5x5 for different cases. Gaussian kernel is built with the help of meshgrid. Haar-like masks are scalable. The user-defined function *norm_upto_255.m* normalises the matrix to form a *uint8* image for display. The following results were obtained.

1.1 Gaussian filtering

By enlarging the size of Gaussian kernel, it covers a boarder area on the image, making the image more blurry. Also, due to zero-padding, the edge becomes darker when we set a large Gaussian kernel, since it takes more zeros from the padding area outside the image. Stronger blur can be achieved by increasing the value of σ. It can also be seen that the filtering through a Gaussian kernel of smaller scale repeatedly, is same as filtering through a Gaussian kernel of larger scale.

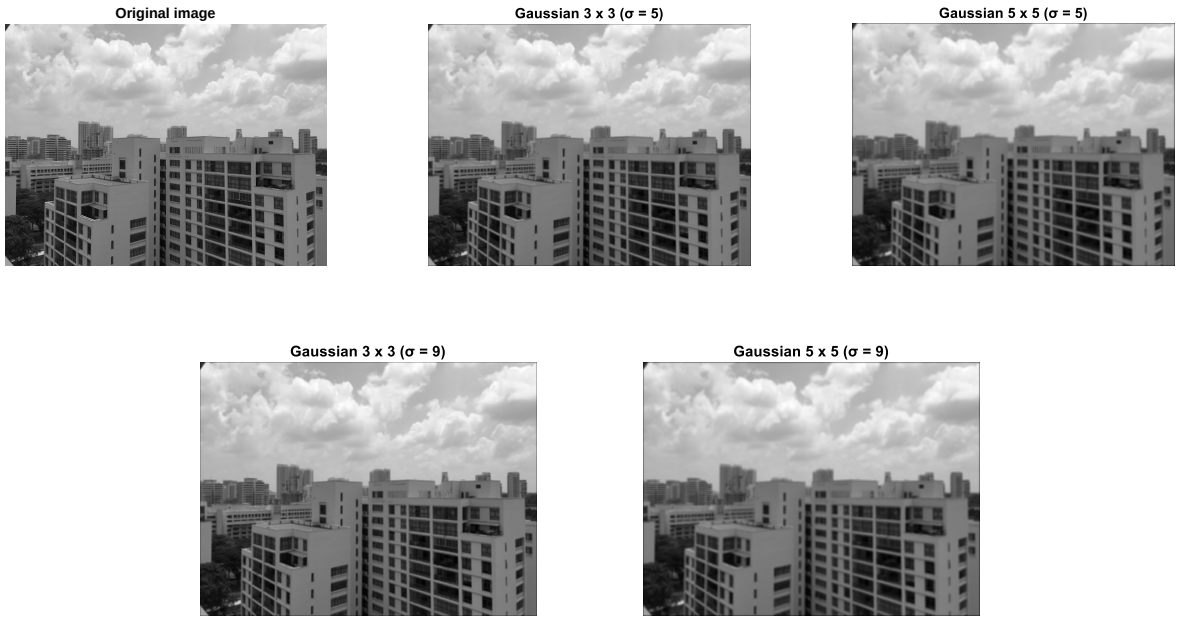

Figure 1: Gaussian filtering output for different kernel sizes and scales

1.2 Sobel filtering

Sobel kernel is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the resulting vertical and horizontal gradient approximations can be combined to give the gradient magnitude. For the chosen image, a small kernel size gives the best performance because actual edges are in the image are also in this size range. To achieve better segmentation 3x3 kernels are modified to 5x5 kernels which show linear regions more clearly than in previous case. But a larger kernel size overlooks small edges and they are not detected. Larger edges in the image that are also in that size range are highlighted more.

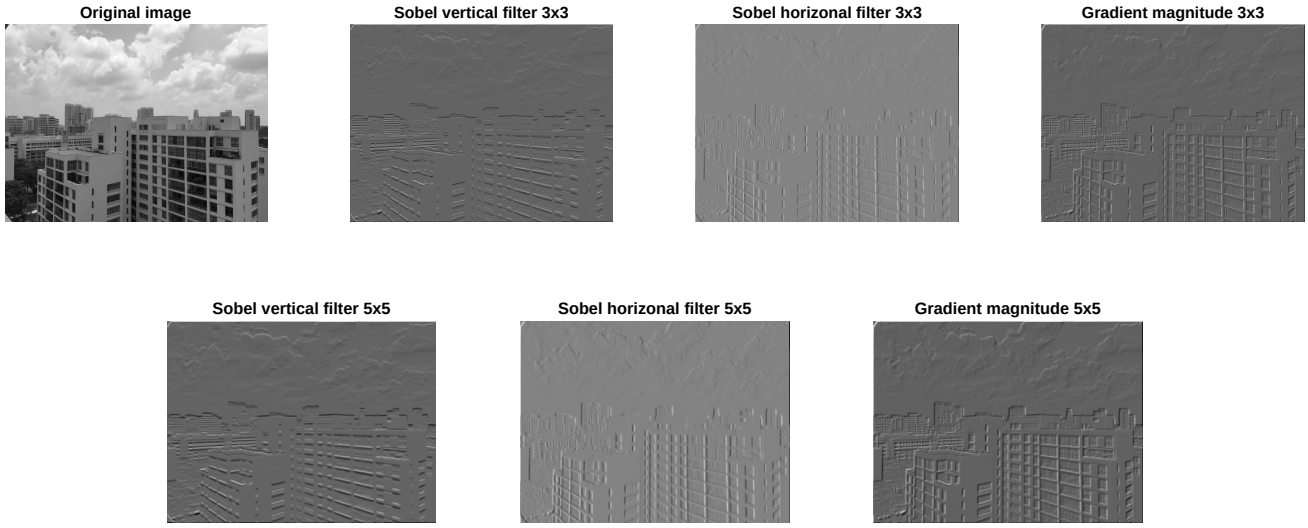

Figure 2: Sobel filtering output for different kernel sizes

1.3 Harr - like masks

The Haar features find horizontal and vertical edges, transitioning edges and double corners. The effect is similar to size changes of sobel kernel. Larger kernel size tries to capture features at the larger scales.

- Edge features: Harr 1, Harr 2
- Line features: Harr 3, Harr 4
- Four-rectangle (double corner) features: Harr 5

By changing the scale of Haar-like masks, we can notice the difference of ability to find out the features (edge, line) in the image. The smaller masks can indicate small and subtle features, while the larger ones can spot the wider edges and lines. Corners on the other hand can still be detected for larger kernel sizes.

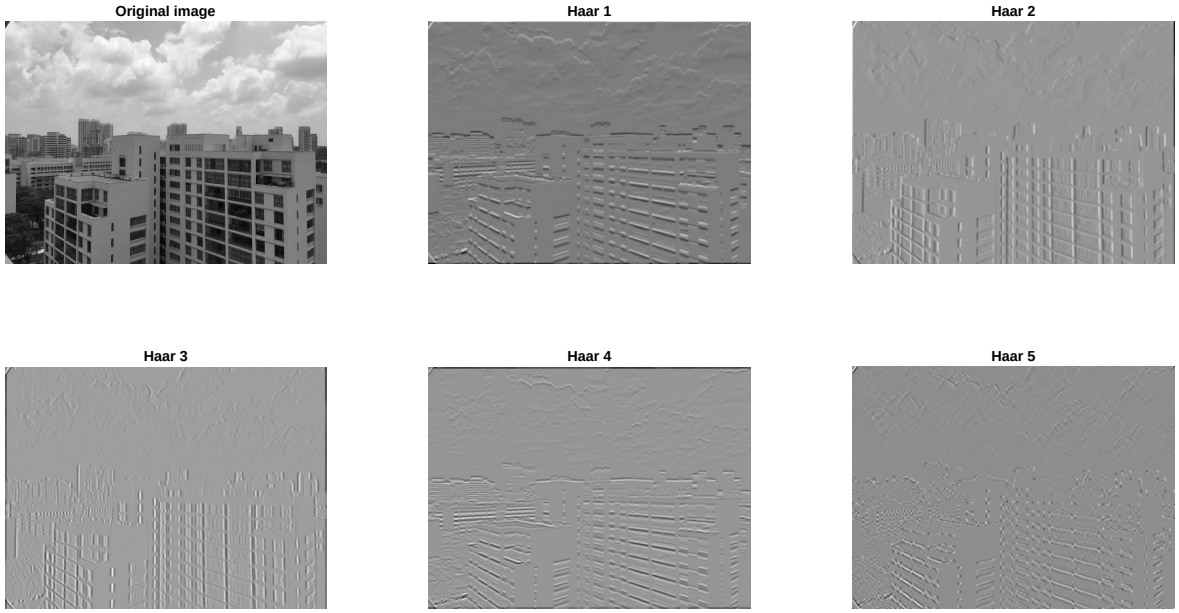

Figure 3: Harr - like masks (scale = 3)

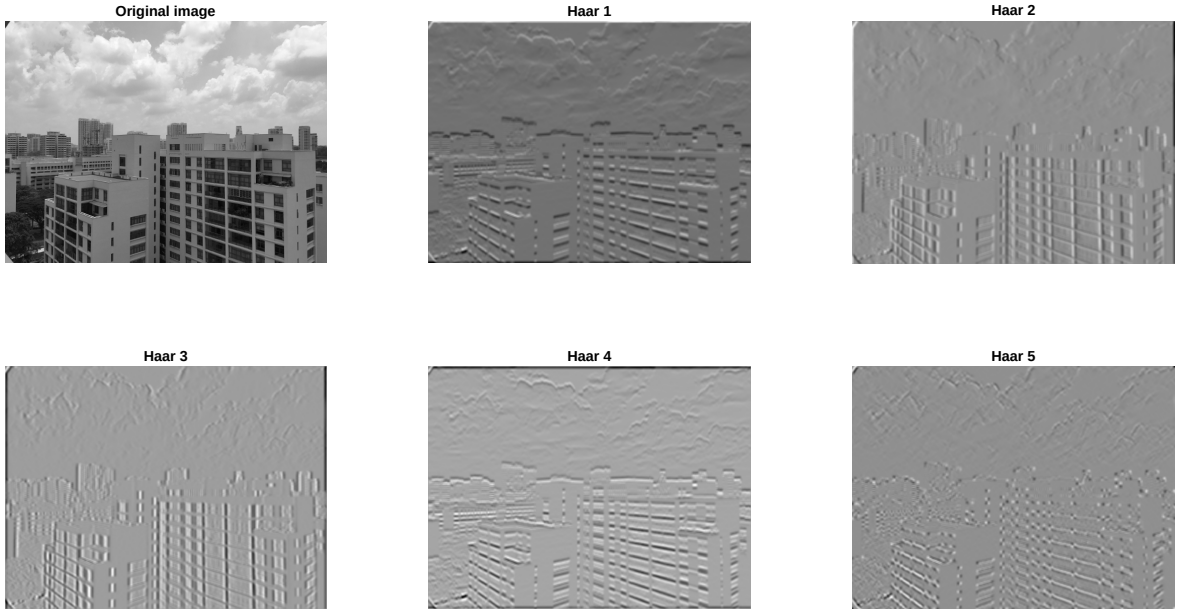

Figure 4: Harr - like masks (scale = 5)

2 Homography and Transformation

The main script is *q2_homography_transformation.m*. A GUI is created to felicitate the user for selecting 4 correspondence points. The function *q2_gui.m* defines the working of the interface as well as the image paths, while *q2_gui.fig* defines its aesthetics. The homography matrix is obtained using SVD. The user-defined function *homography_transform.m* transforms one image into another's space by corresponding location on input image for every pixel of the output image.

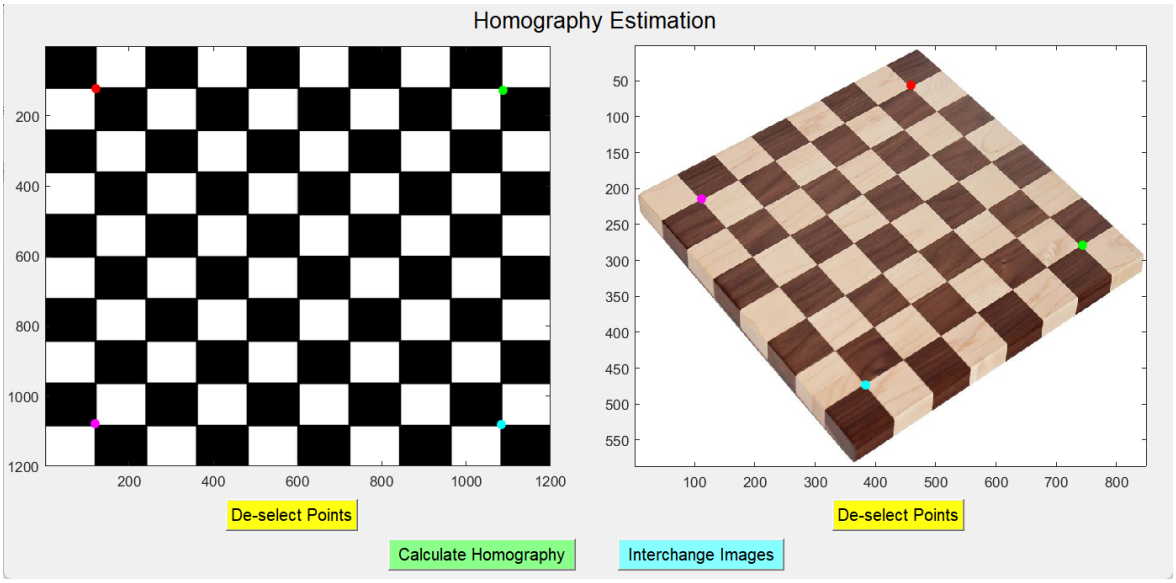

Figure 5: Selection of 4 correspondence points using GUI

Homography matrix $\mathbf{H}_{12}$ obtained for image 1 with respect to image 2,

$$\mathbf{H}_{12} = \begin{bmatrix} 4.7275 \times 10^{-4} & -7.7831 \times 10^{-4} & 0.9996 \\ 4.2554 \times 10^{-4} & 3.1221 \times 10^{-4} & 0.0280 \\ -1.9907 \times 10^{-7} & -1.6397 \times 10^{-7} & 0.0021 \end{bmatrix}$$

Homography matrix $\mathbf{H}_{21}$ obtained for image 2 with respect to image 1,

$$\mathbf{H}_{21} = \mathbf{H}_{12}^{-1} = \begin{bmatrix} 657.2653 & 1.4672 \times 10^{3} & -3.2620 \times 10^{5} \\ -895.1790 \times 10^{-4} & 1.1828 \times 10^{3} & 4.0264 \times 10^{5} \\ -0.0075 & 0.2271 & 467.7756 \end{bmatrix}$$

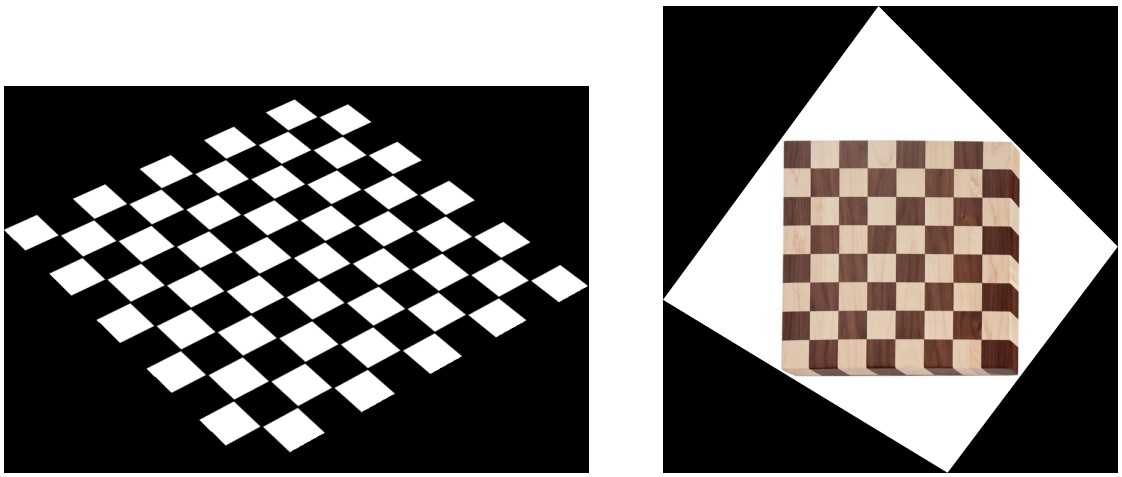

Figure 6: Image 1 transformed into image 2 (left); and image 2 transformed into image 1 (right)

3 Image Stitching

The main script is *q3_image_stitching.m*. A GUI is created to felicitate the user for selecting 4 correspondence points. The function *q3_gui.m* defines the working of the interface as well as the image paths, while *q3_gui.fig* defines its aesthetics. The homography matrix is obtained using SVD. The user-defined function *homography_transform.m* transforms one image into another's space by corresponding location on input image for every pixel of the output image. *stitch_two_images.m* is a user-defined function which draws the transformed image over the other image, pixel by pixel as per the homography estimated.

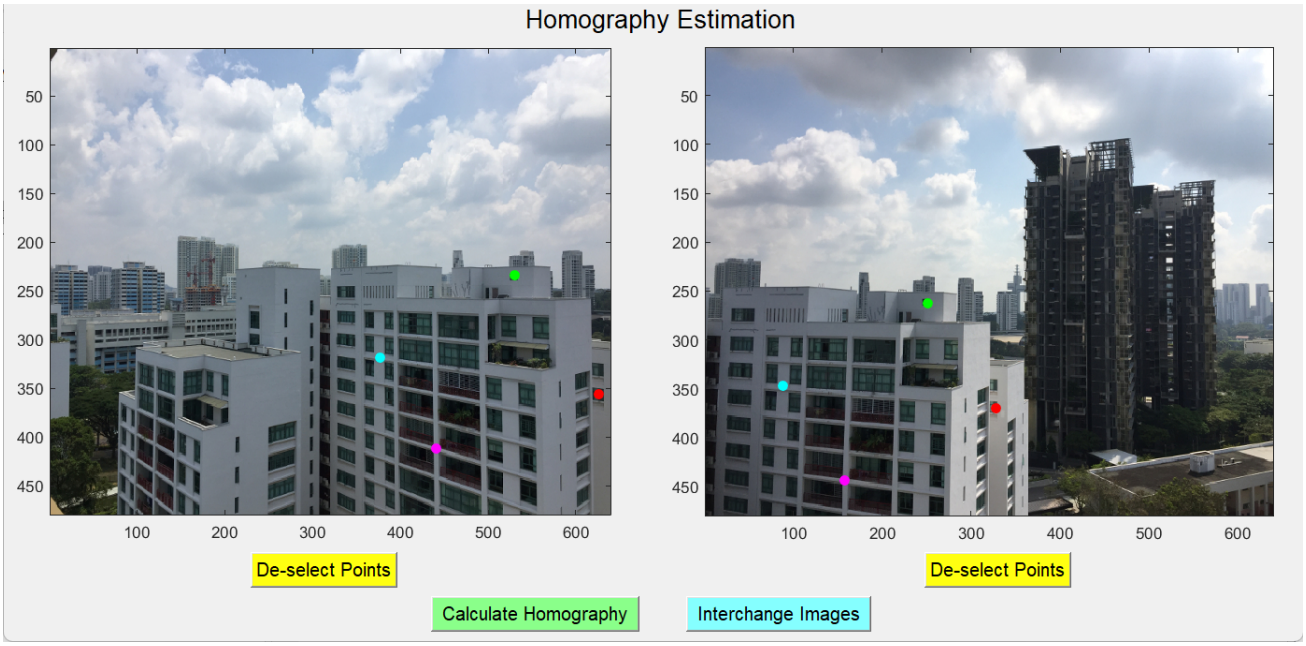

Figure 7: Selection of 4 correspondence points using GUI

Homography matrix $\mathbf{H}_{12}$ obtained for image 1 with respect to image 2,

$$\mathbf{H}_{12} = \begin{bmatrix} -435.1084 & 17.5976 & 1.3022 \times 10^5 \\ -110.0010 & -345.1244 & 3.9275 \times 10^4 \\ -0.4026 & 0.0431 & -0.0033 \end{bmatrix}$$

Homography matrix $\mathbf{H}_{21}$ obtained for image 2 with respect to image 1,

$$\mathbf{H}_{21} = \mathbf{H}_{12}^{-1} = \begin{bmatrix} -0.0013 & -1.9227 \times 10^{-4} & -0.9982 \\ 7.8031 \times 10^{-4} & -0.0029 & -0.0605 \\ 3.1426 \times 10^{-6} & -2.5523 \times 10^{-7} & -0.0033 \end{bmatrix}$$

Figure 8: Image 1 transformed into image 2 (left); and image 2 transformed into image 1 (right)

Figure 9: Image 1 transformed & stitched to image 2 (left); and image 2 transformed & stitched to image 1 (right)

In some cases there will be 'double edges' effect in the overlapping area. Usually this is caused by the error when matching the two images. There will be some offset. When we shift the first image to the transformed location on top of the second image there will be repeated scenes located near each other as they do not overlap exactly. This can be corrected using mean of all the differences between keypoints in image 2 and matching transformed keypoints of image 1. We can also use some techniques like gain compensation and blending to make the overlapping parts merge with each other.

■ ■ ■