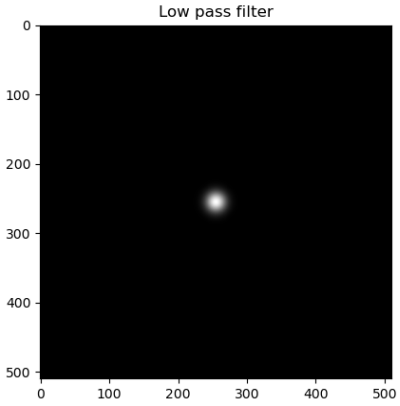
Computer Vision - Assignment 2

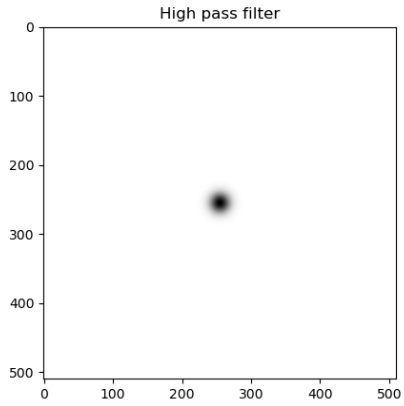
Saurabh Burewar (B18CSE050)

Q1: Image filtering in Fourier

Algorithm -

1. Read the image and convert it to grayscale.
2. Get the fourier transform of the image.
3. Create a low pass filter by making a mask. The mask is basically a white circle of given diameter (here we have taken 10). The less the diameter, the blurrier the image becomes.
4. For a high pass filter, the mask is a black circle with white background. The parameters are the same.
5. We multiply the fourier of the image with the respective filter masks.
6. Then, we do an inverse fourier to get the resulting image.
7. We also take the absolute value of all values in our array.

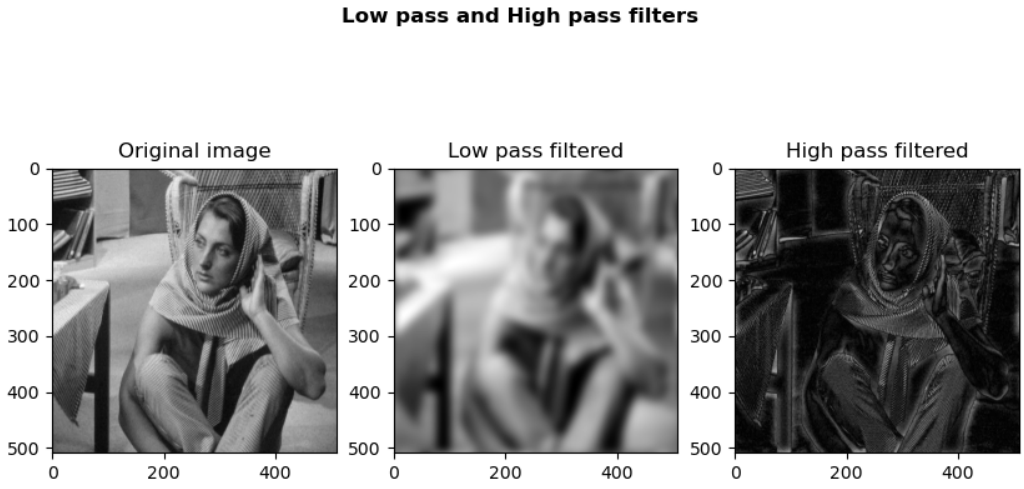




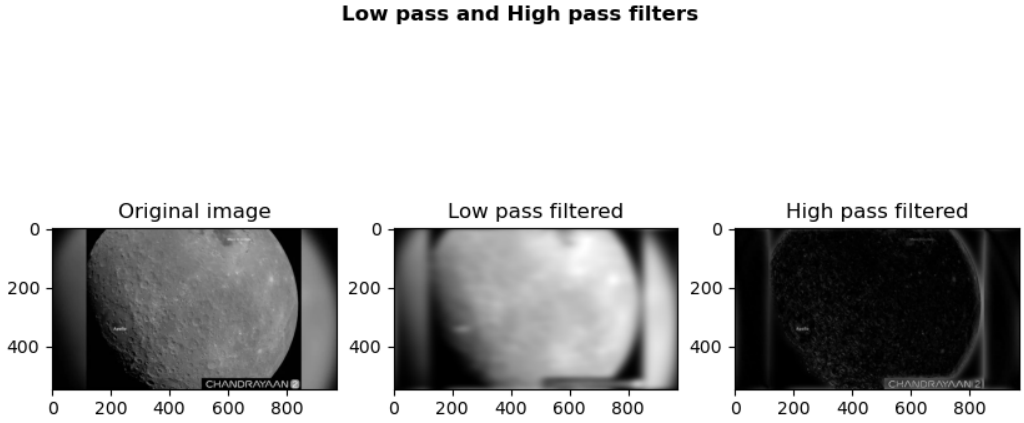
Results

For the low pass filter, we get a blurred image. By changing the diameter of the circle of our mask, we control how blurry the filtered image is. The less the diameter, the more blurry the image gets. For diameter=10, we get the results shown below.

For the high pass filter, we get edges in the image. By changing the diameter of the circle of our mask, we control how sharp the edges are in the filtered image. The less the diameter, the sharper the image gets. For diameter=10, we get the results shown below.



For our second image, we get the following results. In the low pass everything is blurred and in the high pass, the edges of the moon, and the craters are enhanced. The letters are enhanced as well.



Q2: Image filtering in Wavelet

The wavelet transform of the image can be found using pywt.
Algorithm -

1. We find the DWT.
2. Shrink the transform coefficients in finer scales to reduce the effect of noise.
3. Emphasize features within a certain range using a nonlinear mapping function.
4. Perform IDWT to reconstruct the image.

Q3: Image Inpainting

We use two algorithms for inpainting the given images, telea and ns. We get the following results for those algorithms -

From telea, we get this image -

From ns, we get this image -

Both the algorithms give very similar results and there aren't any noticeable differences in the both results.

Q4: Image Stitching

Algorithm -

1. Read the images, resize them (to save memory) and make a list of all images.
2. We work with 2 images at a time. First we stitch the 1st and 2nd image, then the resultant is stitched with the 3rd image and so on. So, we call our stitching function for every two images.
3. The stitching function takes two images as input and outputs the stitched image.

Stitching function -

1. Takes two images and finds keypoints using SIFT.
2. Uses keypoints to create the homography matrix using RANSAC.
3. Makes a new blank image and warps the first image on it according to the homography matrix.
4. Fits the second image on our new image.

The five images we are working with are the following -

The algorithm needs images to have distinct features to make finding matching keypoints easier but since the architecture in the campus all looks the same, it makes it harder for SIFT to find right matching points. This can be seen in the correspondences found in the images below.

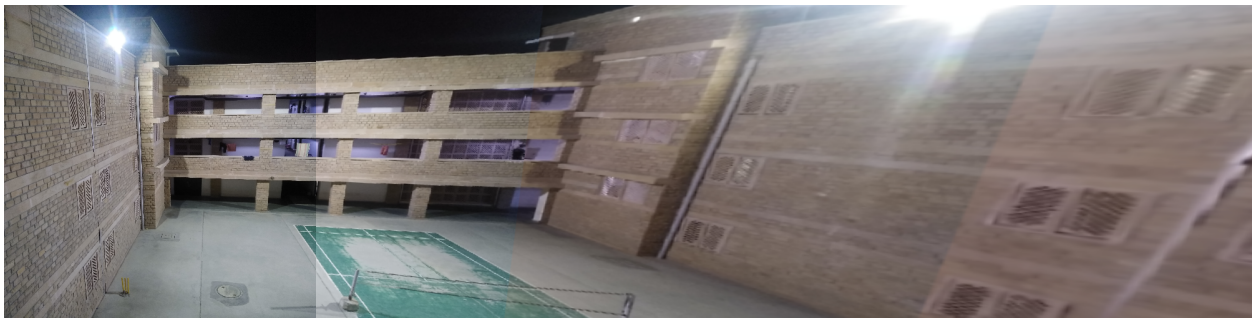
This phenomenon results in bad stitched images. Due to this the images need to have a fair amount of common ground to work with, so we get good results.

Results

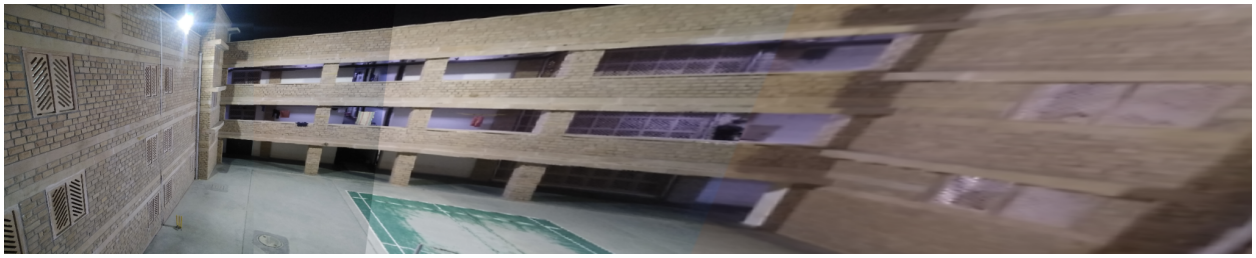
1. Starting from rightmost image, stitching the first and second image we get -

2. Next, stitching the above image with the 3rd image -

3. Next, stitching the above image with the 4th image -



4. Next, stitching the above image with the 5th image -



References

- https://pythonnumericalmethods.berkeley.edu/notebooks/chapter24.02-Discrete-Fourier-Transform.html
- https://stackoverflow.com/questions/54641616/low-pass-filter-for-blurring-an-image
- https://towardsdatascience.com/inpainting-with-ai-get-back-your-images-pytorch-a68f689128e5
- https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/
- https://kushalvyas.github.io/stitching.html
- https://medium.com/@navekshasood/image-stitching-to-create-a-panorama-5e030ecc8f7
- https://learnopencv.com/seamless-cloning-using-opencv-python-cpp/

Code

The code for this report is available in the zip folder named “B18CSE050assignment2”.