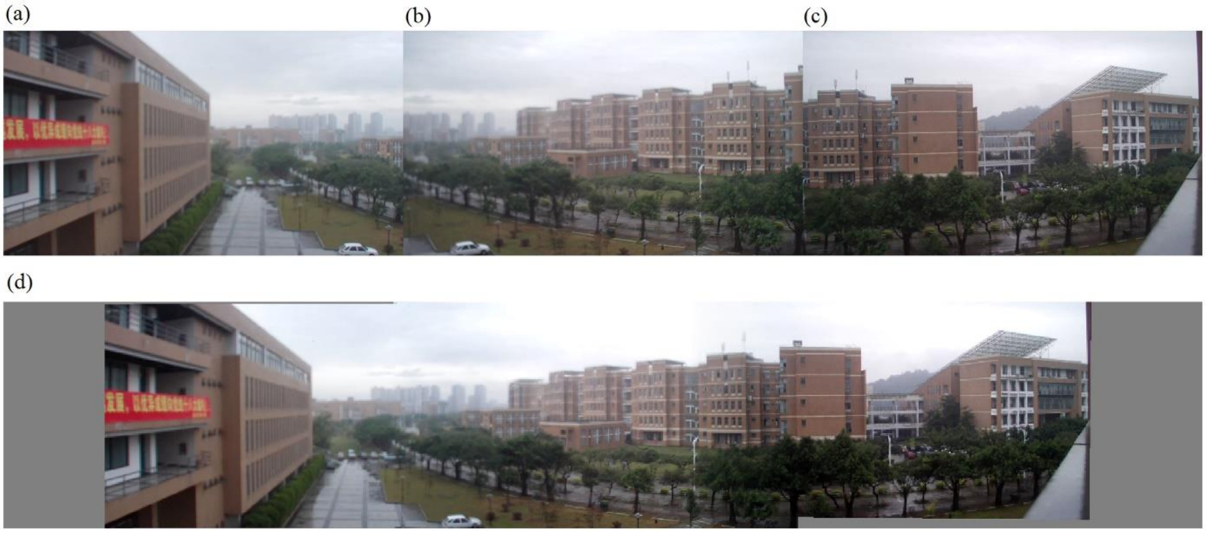
In this assignment, we generate a panoramic image by stitching multiple images.



Explanation of Algorithm

Generally we merge images through finding common keypoints and blending them but I want to explain step by step.

Firstly we read the image directory which have the input images.

Declare count of the images as variable.

Read all input images on the directory with for loop and importing os.

Resize all images.

Create an array and store the input images.

Find the final image size with the input images shapes.

Then match features of first image and second image with matchfeatures method.

On matchfeatures method, we firstly use ORB (explain later) as feature detector

We found keypoints and descriptors of first image and second image through detectandcompute.

We match them through brute force matcher and sort them.

We take them and append to list.

Then we take source points and destination points.

We calculate Homography Matrix through using findHomography.

We also use RANSAC (explain later) on findHomography function.

Then we store the Homography Matrix in a list.

After that, We find alpha mask and final image of two images through warp method.

Warp method warps the image according to the homography matrix and places the warped image at (y_offset,x_offset) and returns the final image.

We make this processes for right input so we going to make this processes again for left side input.

So, we get the images and masks from our warp method and we store them on different arrays.

Then we blend the arrays through blend method we use the arrays as inputs for the method.

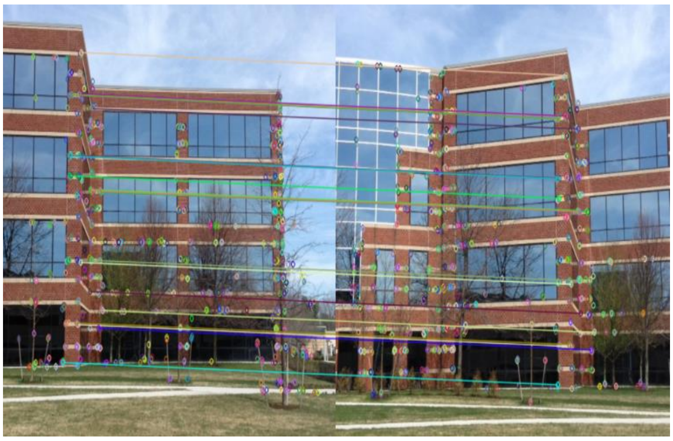
Blend method blending image using Image Pyramids. We calculate Gaussian Pyramids using OpenCV.add() Once we have the Gaussian Pyramids, we take their differences to find Laplacian Pyramids or Difference of Gaussians. Then we add all the Laplacian Pyramids according to the edge of the overlapping image. Finally we upscale all the Laplasian Pyramids to reconstruct the final image.

Then we find appropriate bounding box through mask.

We find default value and max and min values on mask value is 1 (True) to crop

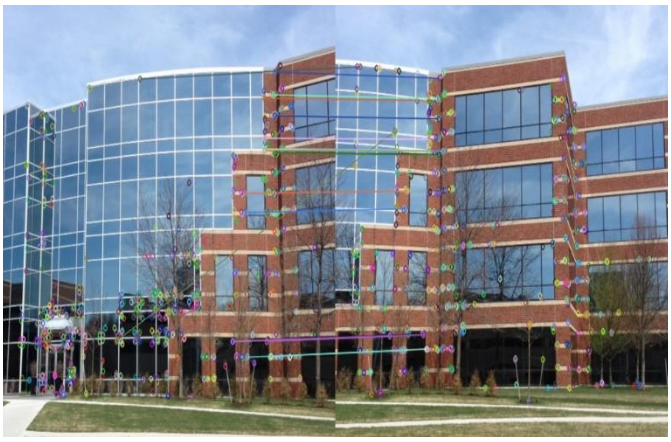
Finally, we get cropped image through the values which we find to crop.

Save the final image using imwrite (opencv).



Firstly we going to compare image 2 and 3. Find them keypoints.

After that, we going to compare image 1 and 2. Find them keypoints.



SIFT, or Scale Invariant Feature Transform, is a feature detection algorithm in Computer Vision.

SIFT helps locate the local features in an image, commonly known as the '*keypoints*' of the image. These keypoints are scale & rotation invariant that can be used for various computer vision applications, like image matching, object detection, scene detection, etc.

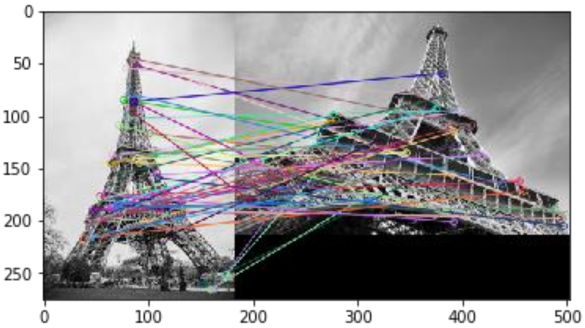
We can also use the keypoints generated using SIFT as features for the image during model training. **The major advantage of SIFT features, over edge features or hog features, is that they are not affected by the size or orientation of the image.**

SIFT (Scale Invariant Fourier Transform) Detector is used in the detection of interest points on an input image. It allows identification of localized features in images which is essential in applications such as:

- Object Recognition in Images

- Path detection and obstacle avoidance algorithms

- Gesture recognition, Mosaic generation, etc



However, I use ORB detector.

ORB: An efficient alternative to SIFT or SURF. It is a good alternative to SIFT and SURF in computation cost, matching performance and mainly the patents.

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. ORB is a fusion of FAST keypoint detector and BRIEF descriptor with some added features to improve the performance. FAST is Features from Accelerated Segment Test used to detect features from the provided image. It also uses a pyramid to produce multiscale-features.

ORB uses BRIEF descriptors but as the BRIEF performs poorly with rotation. So what ORB does is to rotate the BRIEF according to the orientation of keypoints. Using the orientation of the patch, its rotation matrix is found and rotates the BRIEF to get the rotated version. ORB is an efficient alternative to SIFT or SURF algorithms used for feature extraction, in computation cost, matching performance, and mainly the patents. SIFT and SURF are patented and you are supposed to pay them for its use. But ORB is not patented.

I used RANSAC to get the best homographic matrix. the findHomography method utilizes a robust estimation technique called Random Sample Consensus (RANSAC) which produces the right result even in the presence of large number of bad matches. We have seen that there can be some possible errors while matching which may affect the result. To solve this problem, algorithm uses RANSAC.

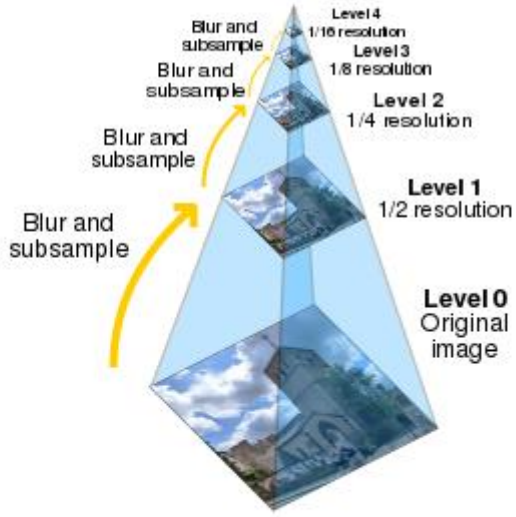
We can use normal stitching or some PIL lib implementations instead Laplacian blending. But Laplacian blending has some advantages.

One of the advantage of Laplacian Pyramid is that it preserves the color of the source image. For the chipmunk composite, the Laplacian Pyramid method has a significantly better effect than the gradient domain method.

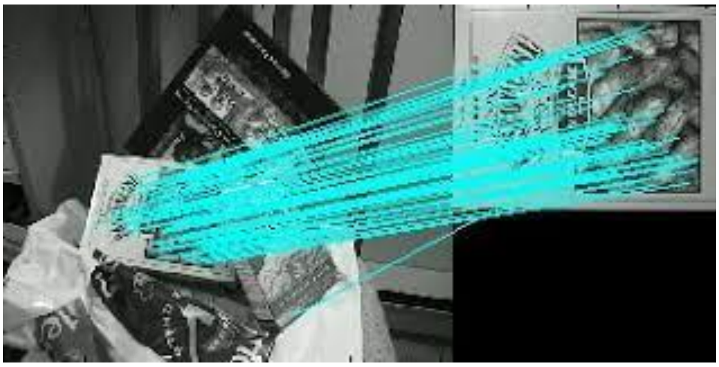
Another property of Laplacian Pyramid method is that the effect depends on the size of the Gaussian filter, but not monotonically. If the size of the filter is too small, you might get clear boundary between the source and the target image. If the size of the filter is too large, you might get artifacts at the positions of the boundary of the source image.

Laplacian pyramid provides an extra level of analysis as compared to Gaussian pyramid by breaking the image into different isotropic spatial frequency bands.

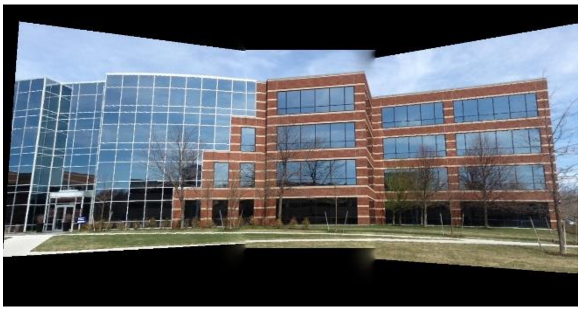
Since Laplacian is a high pass filter, so at each level of this pyramid, we will get an edge image as an output. So, here we will take advantage of this fact and obtain the Laplacian pyramid by subtracting the Gaussian pyramid levels. Thus the Laplacian of a level is obtained by subtracting that level in Gaussian Pyramid and expanded version of its upper level in Gaussian Pyramid.

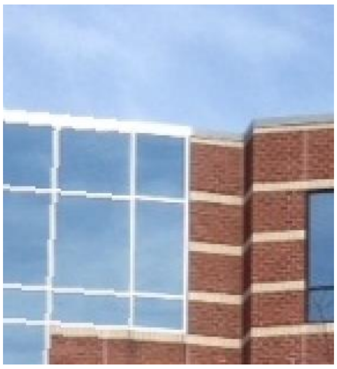


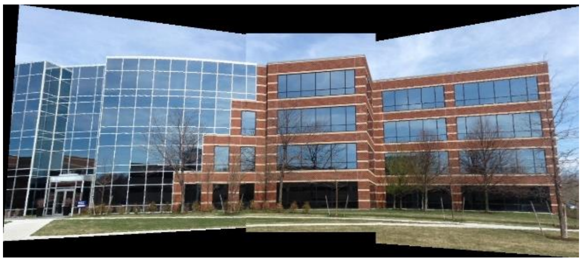
To improve results, you should set image pyramids level maximum. You should pay attention to size of the input images, if they are all at same size it will give better results than different sizes or you can resize the images at your code to have many overlapped areas and keypoints. You can also make your features detector matching points count high, It should give so much better results.

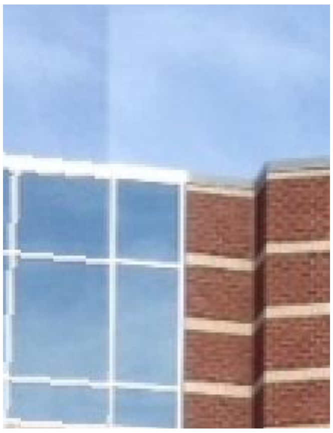


You should make image pyramids level maximum otherwise, the final image seems so clearly it makes from 3 different images.



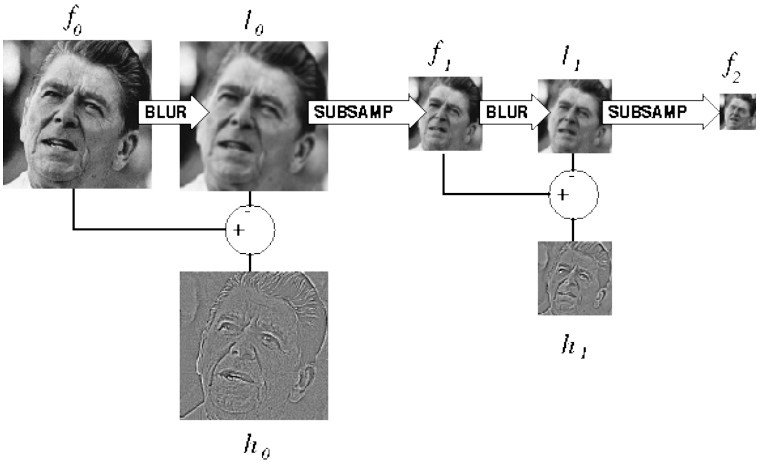




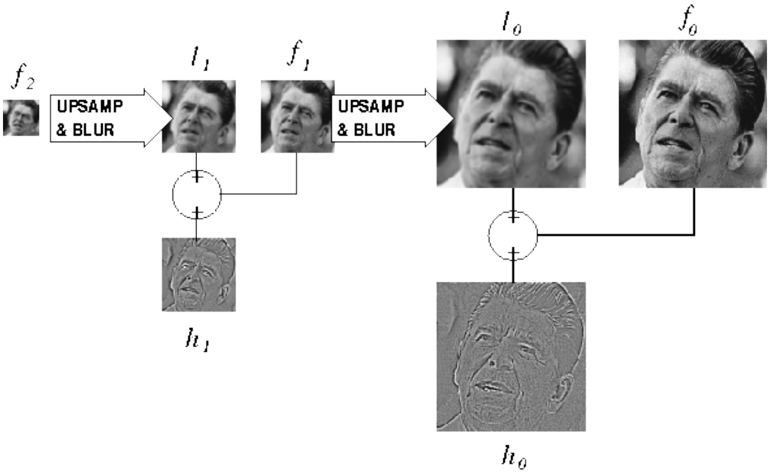


As you can see on the first image I make the image pyramids level 5 and transition of between two different images is smoother than the below which has image pyramids level 1.

A Laplacian pyramid is very similar to a Gaussian pyramid but saves the difference image of the blurred versions between each levels. Only the smallest level is not a difference image to enable reconstruction of the high resolution image using the difference images on higher levels. This technique can be used in image compression.



Decomposition step for two-level Laplacian Pyramid. The finished pyramid consists of the two ``highpass" bands, $h_0$ and $h_1$, and the ``lowpass" band, f_2.



Reconstruction step for two-level Laplacian Pyramid. The process begins with the two ``highpass" bands, $h_0$ and $h_1$, and the ``lowpass" band, f_2, and then perfectly reconstructs the starting image, f_0.

RESOURCES

1- http://sepwww.stanford.edu/data/media/public/sep/morgan/texturematch/paper_html/node3.html

2- https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/

3- https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/

4- https://www.geeksforgeeks.org/feature-matching-using-orb-algorithm-in-python-opencv/

5- http://cs.brown.edu/courses/cs129/results/proj2/huag/

6- https://web.cs.hacettepe.edu.tr/~erkut/bbm413.f15/slides/11-pyramids-4pp.pdf

7- https://en.wikipedia.org/wiki/Pyramid_(image_processing)

8- https://theailearner.com/tag/laplacian-pyramid-opencv/