Computer Vision

Assignment-2 (Mosaic)

G. Aravind (DESE, MTech AI) SR No. 18132

Output of dataset given in question:

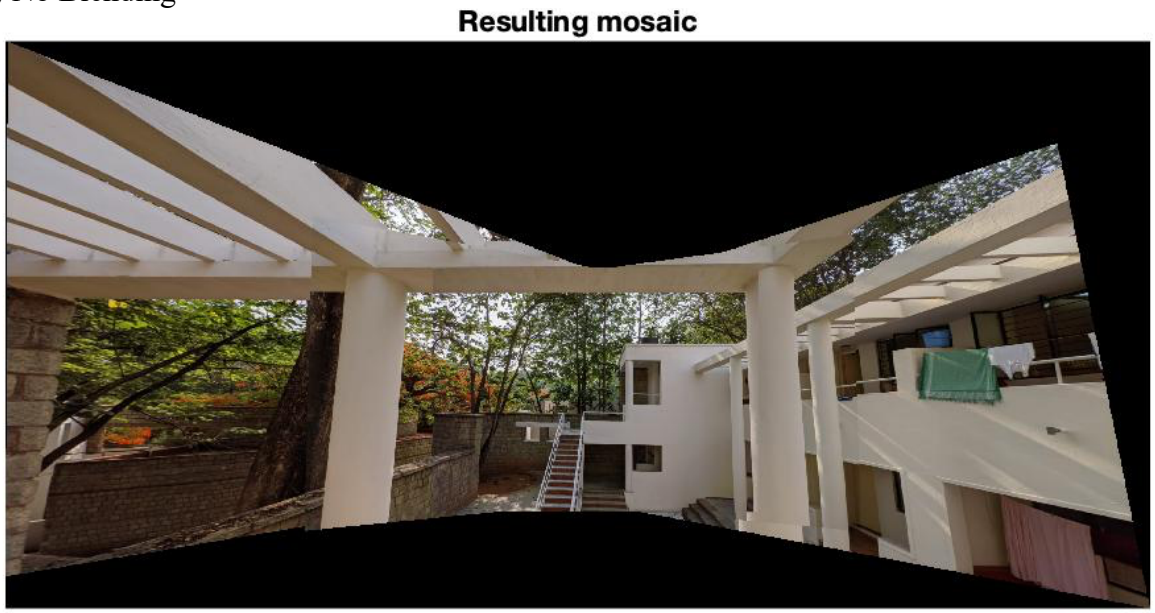
1) With average blending

2) Without Blending

Output of custom dataset (5 images): (Input images stored in same OneDrive folder)

1. Blending done by averaging the overlapping regions

2) No Blending



Observations:

It can be observed that in the first dataset, blending the images (averaging at overlapping points) produces a better looking output as compared to without blending. But on the other hand, in the second custom dataset, blending actually produces a worse output as compared to without using blending. But on the contrary, blending makes it difficult to distinguish the borders between different images in the mosaic.

Hence, it is not necessary that blending will always produce a better mosaic image and it depends on a lot of factors like difference in lighting conditions, amount of overlap, difference in orientation between images, etc. And when two images of the same object are taken in very different lighting conditions, then even after blending we may see very different contrast in different parts of the mosaic image. Also, when there are too many input images, non-blending may be a better option than blending.

Brief description of Mosaic algorithm

- Created a separate function called "merge" which takes two images as input and returns a panoramic mosaic image of both the images as output. Then from another driver function, called the merge function 4 times for 5 images in each dataset (One merge function call for two images).

- Used SIFT to compute features and their descriptors:

    - Normalized pixel values of both images from the range [0 … 255] to the range [0 … 1] so that vl_sift( ) function can be applied on them.
    - Converted both images to grayscale so that we can apply vl_sift( ) only one on each image, instead of calling it 3 times for each color channel in case of RGB images.
    - Obtained the important features in both images and their corresponding descriptors using vl_sift( ). Features are internally evaluated using Difference of Gaussians. Each feature vector contains 4 elements: x-coordinate, y-coordinate, scale and orientation.
    - Features from both images are matched and their corresponding scores are computed using the Euclidean distance between the feature descriptors. Then the image points corresponding to the best matches are selected and converted to homogenous coordinates.

- Use RANSAC to estimate the homography matrix H.

    - Run RANSAC loop for 100 iterations, and for each iteration, choose 4 random matching features and compute H matrix for those 4 points.
    - Let A be a matrix containing the 4 feature points and let H_{3x3} matrix be represented as H_{9x1} vector. Then the solution of H_{9x1} is the eigenvector of A^TA corresponding to its smallest eigenvalue. We compute this eigenvector of A^TA by computing the SVD of A, and pick the last eigenvector of the orthogonal matrix V^T.
    - For each iteration, the H matrix is evaluated and we find how many points satisfy the threshold (inlier points) and how many points do not.
    - Based on the number of points that satisfy threshold, a score is computed for each H matrix.
    - The H matrix with the highest score after 100 iterations (maximum number of inlier points) is chosen as the most accurate homography matrix and used further for computing the mosaic.

- Create a mosaic from the given features and homography matrix H (with respect to the coordinate system of image-1)

    - Create a black canvas (reference plane) where both the images will be warped and stored. Estimate the size of canvas required to fit both images using maximum and minimum x and y coordinates of image1 and image2.
    - In the code, "box2" contains the four extreme points (corners) of image-2 with respect to image-2's coordinate system. We multiply it by H^{-1} to get the extreme

points of image-2 with respect to image-1's coordinate system, and we further use it to estimate the size of canvas required.
    - Created a meshgrid using the endpoints on both x-axis and y-axis representing the pixels on which both images can be projected.
    - Using H matrix, performed backward warping and warped both image-1 and image-2 into the meshgrid.
    - Computed which points are overlapping between the two images after warping, and storing that information in a matrix called "mass".
    - For points in the meshgrid which correspond to neither image-1 nor image-2, we set them to 0 (black). And for overlapping points, we divide the sum of pixel values by 2 to achieve average blending of both images. The resulting image will be the panoramic mosaic of image-1 and image-2.

- We call this merge function 4 times for each dataset to pairwise merge all 5 images and obtain a mosaic.