In this assignment, your task is to implement an image stitching method in order to create a single panoramic image by computing homography and blending multiple corresponding pictures (examples are shown in below). You will use **python3** to implement this assignment.

Images

You are going to work on 3 different image sets: Paris, North Campus and CMPE Building

1. Paris (3 Images)

2. North Campus (5 Images)

3. CMPE Building (5 Images)

Image Stitching

For the assignment, you will have to follow several steps to stitch images.

1. Selecting corresponding points

In order to stitch two images successfully, first you will need a good selection of corresponding points between your input images. For this purpose, you are allowed to use *ginput* function (under ***matplotlib.pyplot***) which will operate on images and provide you to select these point pairs *(x,y)* with mouse clicks. You can use any other function if you know a good replacement as well.

Also, keep in mind that you are going to try out different correpondence point configurations. Thus, you don't want to waste too much time marking points. Mark lots of corresponding point pairs for each of the images sets, write it to a file and use it in your main code (you can use *numpy.save* and *numpy.load* for this process).

Important Note: Using any automated interest point detecting functionality such as **SIFT, SURF** or **Harris** is **forbidden**.

2. Homography estimation

Write a function that can calculate the homogpraphy matrix between two images using the corresponding point pairs which were selected before. For your solution, the signature of this function should be:

`homography = computeH(points_im1, points_im2)`

points_im1 and **points_im2** are the matrices which consists of corresponding point pairs.

To calculate the homography, you are allowed to use svd function (in ***numpy.linalg***).

3. Image warping

After you have successfully estimated the homography between two images, you should be able to warp these images using ***backward transform***. For your solution, the signature of this function should be:

`image_warped = warp(image, homography)`

For this step, you have to write your own warping function. You are not allowed to use any functionality such as ***PIL.imtransform***. However, you can use any interpolation function (in numpy, scipy and opencv) to interpolate the results of backward transform (you will need this process at some point in your solution).

4. Blending images

Once you have the warped images, you can merge them into a single panoramic/mosaic/stitched image. If you directly put images on top of each other, every pixel you write will overwrite pixels of the previous image. Hence, you can use the image pixel that has the maximum intensity for overlapping areas.

Tasks

To see the effects of the base image stitching (mosaicing) process, you will work on several tasks using these image sets that are shown above.

1. Stitch 3 images of Paris by taking the **paris_b** image as the base image.

2. Stitch 5 images of **North Campus** and **CMPE Building** using methods;

    - **left-to-right**
        - Stitch *left_2* and *left_1* by taking *left_2* as base image (call the result *mosaic_1*)
        - Then stitch *mosaic_1* and *middle* by taking *mosaic_1* as base image (call the result *mosaic_2*)
        - Then stitch *mosaic_2* and *right_1* by taking *mosaic_2* as base image (call the result *mosaic_3*)
        - Then stitch *mosaic_3* and *right_2* by taking *mosaic_3* as base image (call the result *mosaic_final*)

    - **middle-out**
        - Stitch *left_1* and *right_1* by taking *middle* as base image (call the result *mosaic_1*)
        - Stitch *left_2* and *right_2* by taking *mosaic_1* as base image (call the result *mosaic_final*)

    - **first-out-then-middle**
        - Stitch *left_1* and *left_2* by taking *left_1* as base image (call the result *mosaic_left*)
        - Stitch *right_1* and *right_2* by taking *right_1* as base image (call the result *mosaic_right*)
        - Stitch *mosaic_left* and *mosaic_right* by taking *middle* as base image (call the result *mosaic_final*)

Effects of Corresponding Points

For this part of the assignment, you will examine the effects of different corresponding points (number, selection, noise and normalization). In order to see these effects, you will only use the **Paris** image set (you don't have to experiment on all image sets).

- **Number of corresponding point pairs**
    - Use only 5 corresponding point pairs
    - Use 8+ correpondence point pairs (maximum 15 points will suffice)
- **Point selection**
    - Add at least 5 wrong corresponding point matches to points that you have selected correctly.
- **Noisy Points**
    - Add pixel noise to your corresponding point pairs. Experiment on different variances (keep variance in a reasonable range, it shouldn't be too high or too low)

- **Normalize corresponding point pairs**
    - Normalize all points before calculating the homography by the process below;
        - Move the average to (0,0) and make the average length square root 2 for corresponding points of each image separately.

For each effect, comment on what can be done differently to obtain a good selection of points. How would you decide if a set of points is sufficient enough to perform the stitching process successfully and efficiently?

Bonus

- Implementing better and clever blending techniques will be rewarded. Comment on what can we done additionally to improve the quality of stitched images.
- Try to avoid using **for** loops while writing your code. **Vectorized** codes will be rewarded.
- Reports written in LaTeX will be rewarded additional points. Also, attach the required *.tex files to your submission.

Deliverables

- **Project report (.pdf & .tex):** Given the tasks above, describe what you have implemented and experimented on. Comment on the effects of corresponding points. Why did they produce such results? Include resulting images in to your report and comment on them.
- **Source codes (.py files):** Do not print the source codes or include them into your report. Code files should be submitted separately. Reproducibility of your code is also important. If you use JupyterNotebook in your assignment, please include a main **.py** file along with it in your submission as well (consist of functionality wanted above).
- **Stitched images (.jpg or .png):** Provide stitched image of given pictures and the ones you have shot. Since the stitching operating may produce images with high resolution, you can compress (or resize them) before you put them into your submisssions.
- **Corresponding point pairs (.npy file):** Provide the point pairs that you have selected **only for the Paris image set**.

WARNING! Submit all files as one zip file. Please send files in correct format. Use zip for packaging, do not use rar, 7z etc. If you are not be able to upload your submission files to

Moodle, submit only a **Dropbox** or **Google Drive** link of the compressed package (not the zip file) with the name; "**[cmpe537-fall24-02][Your Student Id][Your Name].zip**".