# CS 764: Lab 05

## Questions on Manual Mosaicing

## Answers

### 1

We chose 12 point correspondences between the query and train images using mouse-click. We have found the perspective transformation using cv2.warpPerspective() of OpenCV library.This is used because the images captured from the camera can be at a different perspective and thus directly can't be mosaiced.They have to be transformed in a single view in order to perform mosaicing.

### 2

The image which appears to the left of the mosaic is the query image and the image to the right is the train image. For the images in 'campus' folder, 'campus1.jpg' is the train image and 'campus2.jpg' is the query image.
The stitched image of these images will be as follows:

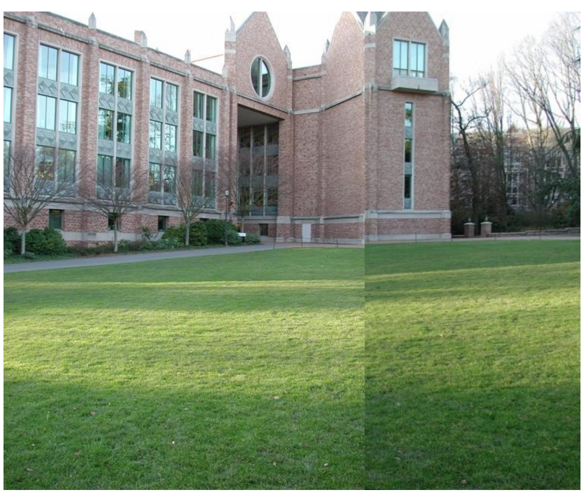

Figure 1: Campus Stitched Panorama

If suppose the train and query images are interchanged, i.e., if 'campus2.jpg' is the train image and 'campus1.jpg' is the query image, we get the stitched image as follows:

Figure 2: Campus Stitched Panorama Interchanged

Clearly the stitching is not proper. The resulting image is the same as the train image. The common area between the two images is appearing but the area present in query image is not appearing.

## 3

The image which appears to the left of the mosaic is the query image and the image to the right is the train image. Given 1-D images J1 has correspondences in range of [10,20] and J2 has correspondences in the range of [85,95].Clearly the image J2 is query image and image J1 is the train image. For image mosaic, the new dimensions of the stitched image of 2 images, will have a width less than or equal to the sum of width of each image. This is because there will be a common area in the images and thus the maximum possible width will be their sum of widths and according to the amount of overlapped area, the new image appears, with extra pixel to be black. So for the given images since each image has 100 pixels, the width of each image will be (0-99) and thus the stitched image has a width of (0-199) with a few black pixels at the rightmost end(the amount of black pixels depends on overlapped region).

## 4

For normalizing the points, we multiply a matrix $T$ with points $x$. This matrix T involves translating and scaling such that the resulting normalised points have mean 0 and unit variance. The transformation matrix T is given as :

$$T = \begin{bmatrix} s^{-1} & 0 & -s^{-1}m_1 \\ 0 & s^{-1} & -s^{-1}m_2 \\ 0 & 0 & 1 \end{bmatrix}$$

where $(m_1, m_2)$ are the centroids pixels of image and computated by averaging the x and y coordinates of all selected points in the images. $s$ is the scalar quantity measuring the root mean squared distance of each pixel from the centroids which acts as the scaling constant.
This is the affine transform that we have used during normalization.

## 5

Here let the point correspondences of query image(i.e left image) be $x$, and that of train image(i.e right image) be $x'$ such that $x'$ is transformed into $x$ i.e $x' = Hx$. The points normalization involves the following:

- The points are translated so that their centroid is at the origin (0, 0).
- The points are then scaled so that the average distance from the origin is equal to $\sqrt(2)$ .
- This transformation is applied to each of the two images independently

Let the normalized points of $x, x'$ be $\tilde{x}, \tilde{x'}$ with transformation $T, T'$ respectively such that $\tilde{x} = Tx$ and $\tilde{x'} = T'x'$.Let $H_{norm}$ be the transformation between the normalised points $\tilde{x}, \tilde{x'}$, then $\tilde{x'} = H_{norm}\tilde{x}$. This transformations $T, T'$ are defined as follows:

$$T = \begin{bmatrix} s^{-1} & 0 & -s^{-1}m_1 \\ 0 & s^{-1} & -s^{-1}m_2 \\ 0 & 0 & 1 \end{bmatrix} T' = \begin{bmatrix} s'^{-1} & 0 & -s'^{-1}m'_1 \\ 0 & s'^{-1} & -s'^{-1}m'_2 \\ 0 & 0 & 1 \end{bmatrix}$$

where $(m_1, m_2)$,$(m'_1, m'_2)$ be the centroids pixels of domain and range image respectively and computated by averaging the x and y coordinates of all selected points in the images. $s, s'$ be scalar quantities measuring the root mean squared distance of each pixel from the centroids.
Thus,we have

- normalising the points
  $\tilde{x} = Tx$
  $\tilde{x'} = T'x'$

- finding transformations
  $x' = Hx$
  $\tilde{x'} = H_{norm}\tilde{x}$

And the relation between $H, H_{norm}$ is $H = T'^{-1}H_{norm}T$

## 6

Different evaluation metrics are available to measure the similarity between two images. Few of them are:

1) Root Mean Squared Error (RMSE)

2) Structural Similarity Index (SSIM)

These two metrics are computed between the normalised image "campus_with_normalization.jpg" and without normalised image "campus_no_normalization.jpg" of mosaiced image of 'campus' folder( the images can be found in 'results' folder). The metrics obtained are:
RMSE: 39.53336955985109
SSIM: 0.39611275154212433
In this case, the RMSE is more and the SSIM is less, implying that the images are less similar. This is because of the approximation of homography estimate during denormalisation.

# Questions on RanSaC subroutine

## Answers

### 1

The value of N can be obtained from the following formula:
$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}$
where
$p$=probability of point to be an inlier
$\varepsilon$=probability of point to be an outlier
$s$ = number of sample points
We chose $p = 0.99$ i.e there is 99% chance that at least one iteration is free of outliers. Since 50% of the points are outliers, $\varepsilon = 0.5$, and we chose 4 sample points, i.e., initially the algorithm picks 4 random points. So putting these values ($s = 4$) gives $N = 72$.

### 2

The value of T is the proportion of inliers among all points. So it is given as $T = (1 - \varepsilon)n$ where $n$= total number of points. In each iteration, we find the set of inliers, and if the inlier size $> T$, then we save those inliers. But the model can give better results if other 4 random points are chosen. So the value of N keeps updating with the number of outliers and all the inliers with size$> T$ are saved. Then the model with maximum number of inliers is chosen.
For $N = 72$ and $T = (1 - \varepsilon)n = 164.5$, we got inliers size in range of 250-330 in each iteration, where the threshold t keeps updating with the standard deviation of errors. Since inlier size $> T$ for each iteration, percentage of times this parameter T was used is 100%.
If we fix the threshold t to 1, and don't update it, then we got inliers size in range of 5-140 in each iteration and clearly, none of them is $> T$, so percentage of times this parameter T used here is 0%.

### 3

In each iteration, the errors are calculated for the matches and those points in the matches whose error lies within t, are inliers. If the threshold t is set to 1, then errors are computed and if they are less than 1, then the points are considered as inliers. In general, the errors are assumed to be Gaussian Distributed. So the value of t is chosen as $t = \sqrt{(3.84)}\sigma$ where $\sigma$ is the standard deviation of errors. And thus, for each iteration, t is computed using errors and inliers are found.

### 4

If we set the threshold t to be 1,with $N = 72$ and $T = (1 - \varepsilon)n = 164.5$, then the inliers size were in the range 5-140 in each iteration. The maximum inlier size, we achieved is 133.
If the threshold is updated using $t = \sqrt{(3.84)}\sigma$, then the largest inlier size achieved is 328 with the inlier size lying in the range 250-330 in each iteration.

# Questions on Mosaicing Implementation

## Answers

### 1

We have used SIFT Detector and descriptor to compute the matching correspondences between the two images of 'campus' folder. The number of matches obtained by SIFT are 329 and these matches are filterd using our custom ransac code and the number of filtered matches i.e inliers came out to be 326 and thus the percentage pruned is 99.08%.

### 2

We have taken the images(in the dataset folder in data folder) (Figure 3) and the resulting mosaic image using manual-mosaic (Figure 4) and auto-mosaic (Figure 5) are as follows:
Clearly, the auto mosaic was the same as that we achieved when selecting the point correspondences manually. So auto-mosaic works with this dataset. We could not think of any dataset where the auto-mosaic fails but manual-mosaic works.

(a) Train image

(b) Query Image

Figure 3: Our Dataset images

Figure 4: Manual Mosaic Panorama on our dataset

Figure 5: Auto mosaic Panorama on our dataset

# Questions on Let's generalize

## Answers

### 1

**Method Followed:**
1. Here for stitching multiple images, we are treating the next image to be stitched as the secondary image and the panoramic image formed by stitching previous images as the base image.
2. Firstly we are given the folder in which images are present with names as increasing order of integer values (1.jpg, 2.jpg, etc.). We are first reading all the image names and then storing them in a 2D array. After sorting the image names in increasing order, we are reading images and putting them in the list.
3. Now if we try to apply the method of stitching multiple images mentioned above directly for more than 2 images, we get to see a heavy distortion because if for the group of images the camera rotation is more than 120 degrees (approx), we cannot reproject them to a single plane. Hence for removing the distortion, we are projecting the images onto the surface of a cylinder, then unroll the cylinder to make a flat plane. This yields a single image we can display on a screen, or print on a piece of paper.
4. After rolling the image, the black region on the left and right is eliminated and these transformed images are used to create a panorama.
5. As our images are not rectangular. So for overlapping the images, we used mask of the secondary image to stitch it over the base image. Firstly, the secondary image is projected onto the cylinder and unrolled, and its mask image is also obtained using the data points obtained from the projection. Then the matches and the homography are found and also the new frame size is calculated and the homography is corrected. Afterward, the perspective transform of the secondary image and its mask image is done and a blank frame is also created of the same dimensions in which the base image is placed at the correct position. Then we overlap these two images using the mask properly and repeat further.

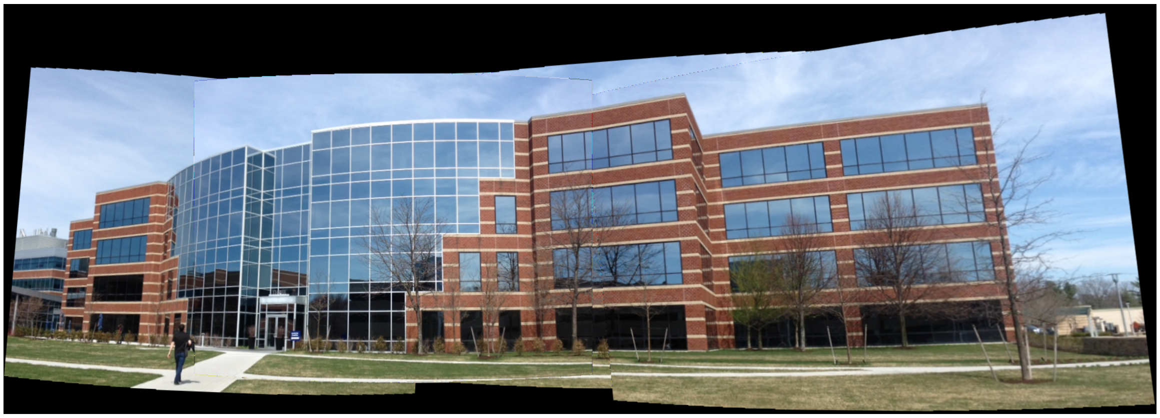

Figure 6: Dataset Stitched Panorama

## 2

Yes, there are cases when this code is not producing appropriate results.
Figure 7 is obtained by taking 3.jpg as reference (base) image for the Mountain images dataset given in the problem statement.
Figure 8 is obtained by taking 4.jpg as reference (base) image for the own selected dataset images.
For our code, it should be kept in mind that the numbered image should have something in common (overlapping region) with any of the previous images. Hence selection of reference images and even selecting the appropriate dataset is important for our code to produce proper panoramic image.

## 3

No, the result becomes distorted or the code can even break in some cases due to insufficient number of matching points if we do not consider the overlapping region while choosing the reference (base) image.
**Issues faced**
1. In our code, reference image is chosen by user by giving the index of the image as input. After that it is overlapped with the image whose index is least other than the reference image chosen. Hence if the reference image does not have considerable amount of matches with the next image it is being overlapped with and if it further continues. Then either our code breaks

Figure 7: Mountain Distorted Stitched Panorama

or we get the distorted image as shown in figure 7 and 8.

**Reason behind these issues**

Our dataset contains numbered images in a regular pattern (For eg - 2.jpg is having overlapping area with 1.jpg and 3.jpg but not considerable overlapping area with 4.jpg and 5.jpg). In the code after selecting the reference image we are overapping it with the first images, hence if we choose 4.jpg as

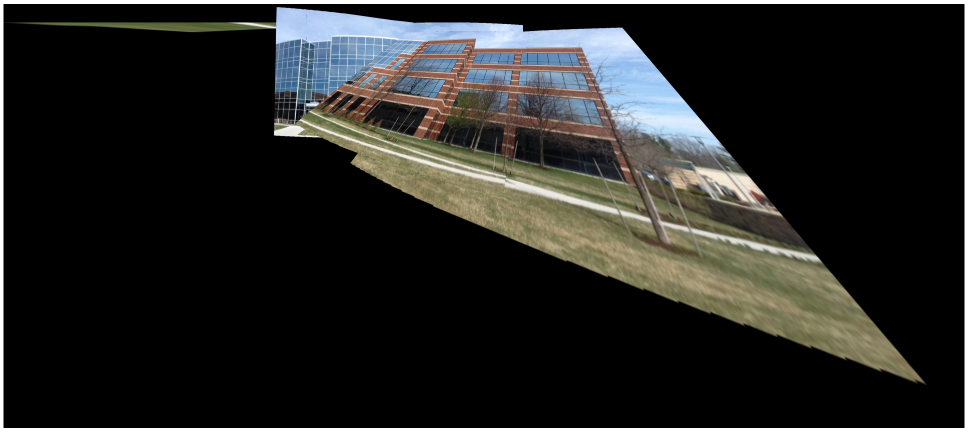

Figure 8: Dataset Distorted Stitched Panorama

reference image then the overlapping region between 4.jpg and 1.jpg will be very less and panorama will become distorted or the code can even break in some cases due to insufficient number of matching points.
Hence it should be kept in mind that the numbered image should have something in common (overlapping region) with any of the previous images.
Hence selection of reference images and even selecting the appropriate dataset is important for our code. This limitation is due to the selection of images based on their names in increasing order.

# 4

Here we know the sequence of images and it is becoming a limitation in the case where we are choosing any image of the database as reference image. If we don't know the sequence then we can modify the code as follows:
After selection of reference image, we can compare its matches with all the other images present and overlap it with the one which is having maximum number of matches with reference image. After this our reference image is updated and we can repeat same steps until all the images are combined.

## 5

Stitcher API of OpenCV provides sitching of multiple images. It uses homography model for creating photo panoramas captured by camera. In our approach, image names which are in ascending order are important component for overlapping images in a loop. Stitcher API can take images with any name and in any order and stitch them. It even provides us with the status flag which indicates success if status=0, otherwise the stitching failed.

## 6

**Items needed to be fixed:**

1. The condition of naming the images in increasing order wrt their adjacent images can be removed.
2. Projecting the image onto a sphere instead of cylinder can generate even better results.
3. During panorama generation, because of presence of seam some meaningful content from the images can be cropped or removed. This can be prevented by seam carving algorithm in which content aware resizing of images is done.

**Technique used to remove the seam (seam carving algorithm)**

1. Calculate energy map for image
2. Find minimum seam from top to bottom edge
3. Remove minimum seam from top to bottom edge
4. Repeat Steps 1 - 3 until desired number of seams are removed
5. Repeat Steps 1 - 4 for left to right edge