CS 763/764: Computer Vision

Answers

2. Questions on manual mosaicing

1. How many point-correspondences did you choose? Which OpenCV library function did you use to find the transformation in (2)? and why?

Solution.
Our implementation requires 8 point-correspondences to stitch the images properly. We used the **findHomography()** function (without the cv2.RANSAC parameter) of OpenCV to find the homography transformation matrix and used **warpPerspective()** for transforming the images. From Task02, we know that computation of the homography matrix is done using more than 4 points, thus we need to select atleast 4 points to get a decent stitch. Hence, we used **findHomography()** function of OpenCV. Also, **findHomography()** gives us the flexibility to internally implement RANSAC to remove the outliers while finding the homography matrix.

2. Consider a case in which we keep I2 as the source image. List the ways or properties in which output after mosaicing will change and, the ways in which it will remain the same.

Solution.

Consider the two stitched images where we have I1 as the source in Figure1a, and I2 as the source in Figure1b, respectively.

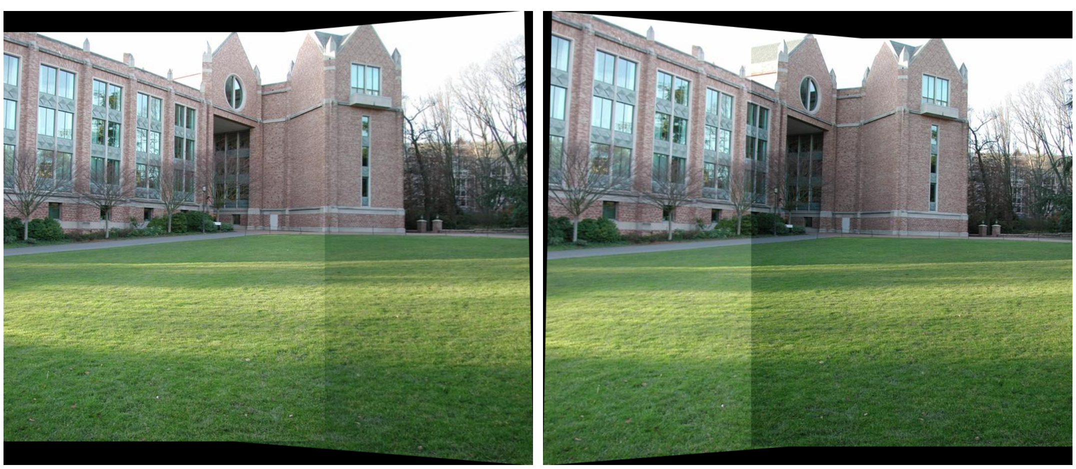

Figure 1. (a) I1 uncropped as source (b) I2 uncropped as source

Ways in which the outputs will be the same

- The stitched output after mosaicing will be similar (not same, different perspective) if we process the images to get rid of the seam at the interface of the two images.

Ways in which the outputs will change

- The notable difference is the placement of the seam in both the images. In the image with I1 as source, the seam is more towards the right, whereas in the case with I2 as source, it is more towards the left. The perspective of the stitched image in the former is in respect to the perspective of I1, now it will be in perspective of I2.

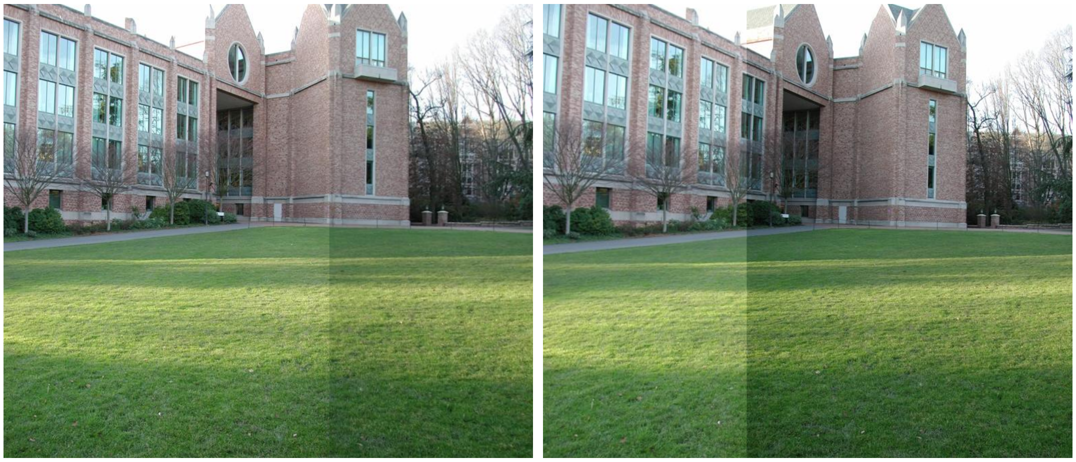

FIGURE 2. (a) I1 as source (b) I2 as source

- In the case when I2 is the reference, we see that the cropping is done with respect to I2, and thus no part of I2 is missing in the cropped image, this changes when I1 is reference image, such that the cropped portion contains all details of I1. The details of the images other than the reference image may get lost in the cropped versions.
- The intensity and contrast in both the images is different, this gives rise to a seam while stitching both the images. The position of this seam depends upon the interface between the reference image, and is different in both the cases.
- Also the details, such as warped regions and perspective distortions depend upon the choice of source and reference images.

3. Consider two 1-D images J1 and J2 of size 100 pixels each. Both share some common points (correspondences). J1 has correspondences in range of [10,20] and J2 has correspondences in the range of [85,95]. Will the size of the output image (after mosaicing J1 and J2) change if we keep J1 as source image vs if we keep J2 as the source image. Explain.

Solution.

Since, the projection lengths of the combined images will always be same irrespective of whichever of J1 or J2 used as source, thus the size of the output 1D image will always be the same.

4. Explain your choice of the affine transform.

Solution.

The affine transform matrices used in the case of normalization is given as follows. Let p_i be the `P2` points of reference image and p_i'' be the `P2` points of image to be warped. The notations are as follows.

μ_p : Mean of all the $p_i s = (\mu_{px}, \mu_{py})$

$\mu_{p'}$: Mean of all the $p_i' s = (\mu_{p'x}, \mu_{p'y})$

σ_p : Standard deviation of all the $p_i' s = (\sigma_{px}, \sigma_{py})$

$\sigma_{p'}$: Standard deviation of all the $p_i'' s = (\sigma_{p'x}, \sigma_{p'y})$

$$D = \begin{bmatrix} 1/\sigma_{px} & 0 & -\mu_{px}/\sigma_{px} \\ 0 & 1/\sigma_{py} & -\mu_{py}/\sigma_{py} \\ 0 & 0 & 1 \end{bmatrix}$$

$$D' = \begin{bmatrix} 1/\sigma_{p'x} & 0 & -\mu_{p'x}/\sigma_{p'x} \\ 0 & 1/\sigma_{p'y} & -\mu_{p'y}/\sigma_{p'y} \\ 0 & 0 & 1 \end{bmatrix}$$

The above affine transform matrices D and D' are multiplied with the $p_i s$ and $p_i' s$ to get the respective normalized coordinates.

5. Explain the relationship between H and H_n.
Solution.

From the discussion of Q.4, we can clearly derive the following equation.

$$D' = H_n \cdot (D \cdot x)$$

$$x' = D'^{-1} \cdot H_n \cdot D \cdot x$$

$$x' = H \cdot x$$

$$H = D'^{-1} \cdot H_n \cdot D$$

6. Design an error metric and report the percentage error while using normalization and not using normalization.
Solution.
Relative Mean Squared Error between the images transformed using the H and H_n matrices is a good error metric for computing the error in a single perspective. This is because the error will contribute only for the pixels which have been relocated differently during stitching in the normalized and non normalized case.

Percentage error calculation = $\frac{RMSE}{N}$

where, N = total number of pixels in the image
Percentage error obtained for campus images for I1: 0.0046

3.1 Questions on RanSaC subroutine

Assume threshold is fixed to 1 in the following questions, and we run the non-adaptive ("50%") version. Answer the following questions with respect to the campus dataset provided.

1. Explain your value of N.
Solution. We have taken p = 0.99 and initial estimate of outliers as per the question is fixed at 50%. Hence N can be estimated as :

$$N = \frac{log(1-p)}{1-w^s}$$

where p = 0.99, w = 0.5, s=4
On substituting the values, we get

$$N = 71.35 \approx 72$$

2. Explain your value of T. What is the percentage of times this parameter was used?
Solution.
We took 50 best key points from the SIFT algorithm. Among these 50 points, on setting initial T to 50%, we get initial estimate for T as 25. Whenever we get a match which has more than 25 inliers, the parameter T is updated. In the iteration we had, T was used 1.4% times.

3. Explain your value of t and how you used this for checking whether a sample is an inlier or not.
Solution. t here is set to 1.0, but since we wanted the transition between two images to be smooth and not abrupt, we need more key points, hence we needed to increase the threshold to 5.0 in order to get a larger set of inliers.

4. What was the size of the largest consensus set as a function of the inliers?
Solution. The largest consensus set we found for these values was 29 out of 50 (58% inliers, 42% outliers)

3.2 Questions on Mosaicing Implementation

1. Report the number of feature points used for campus and the percentage of pruning due to your method
Solution. We are initially detecting 800 keypoints in both the images to be stitched. Using the **cv2.BruteForce()** matcher with L2 norm as the metric for similarity of two keypoints, we are selecting the best 50 matched keypoints to obtain the homography matrix.

Percentage of keypoints pruned using RanSac: 26%

2. Consider the personalized ("group") dataset you used earlier for the manual case where you handtweaked the correspondences. Does auto-mosaic work with this dataset? Can you think of a dataset that breaks auto-mosaic but will (obviously) work with the manual case.
Solution. Yes auto mosaicing works for the group dataset.

Yes, auto mosaicing will break for a dataset where two images have repeatitive patterns that are common in both the images to be stitched. Example: stitching two images of the same chessboard taken at different perspective. In this case, only manual mosaicing will work.

4 Questions on Generalized Mosaicing

1. Explain the method you followed in accomplishing this task.
We have multiple images present with a reference image specified. The following steps were implemented to obtain the stitched image.

- Compute the homography matrices corresponding for all the images to warp them into the perspective of the reference image.
- Check the images for which we are getting negative coordinates for, by computing a correction term, we correct our homography matrix by adding translation terms t_x and t_y such that the warped image obtained with the corrected homography matrix lies within the image boundaries.
- Now, when we have the warped image in the perspective of the reference image, we create a mask corresponding to the warped image.
- The mask specifies where the warped image should be with respect to the reference image. Using **np.where()**, we fill the values in the mask with that of the warped image where the mask value is 1, and fill the remaining part with the source image.
- Once we obtain the stitched image of one image with respect to the reference, we stitch for the other images with respect to this stitched image. Repeating the above steps iteratively, we are able to stitch all the images.

2. Show the results for the given datasets as well as on your own datasets (of 5 images). Mention the results when your code is working and when it is not working for some reference images or some datasets (if that's the case).

Solution. Result for the given dataset (mountain) as shown in Figure(3) .

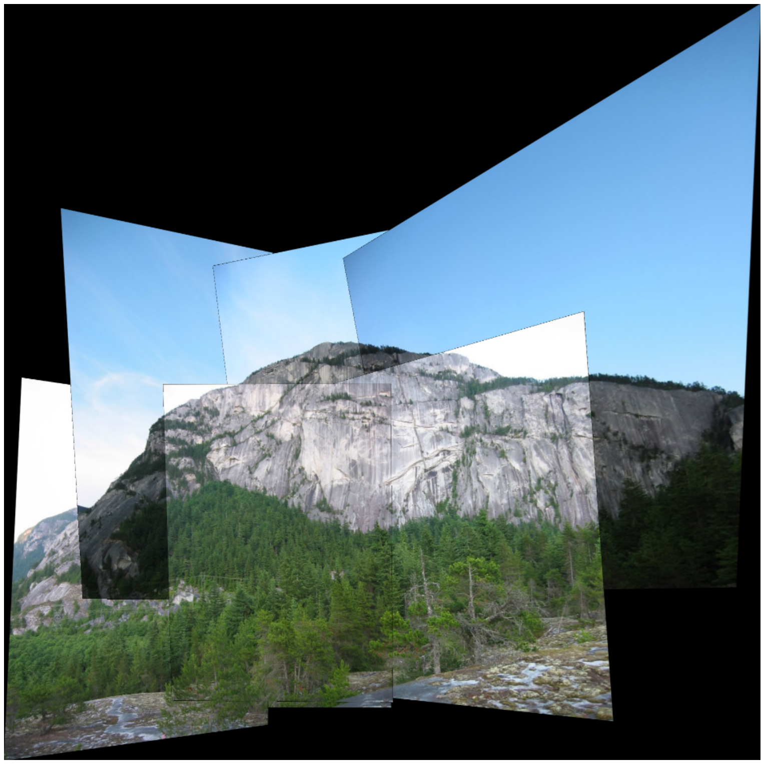

FIGURE 3. (a) Final Stitched Image

Custom stitched image is shown in the figure below Figure(4).

FIGURE 4. (a) Custom Stitched Image using 5 images

3. Do you get the result if you choose any image to be the reference image? If no, what issues are you facing? Give your opinion on reason behind these issues.
Solution. No, we get a distorted image where we chose rightmost or leftmost extreme images which are far apart from the images which are located around the center. The distortion comes because the farther the image is from the reference image, the more distortion it has to suffer to come into the perspective of the reference, choosing the image which is equidistant from all the images is better for stitching.

4. Here, we assumed that you know the sequence in which you have to stitch the images. How would you modify your approach if you don't know the sequence? Describe the approach briefly
Solution. In order to determine the order of stitching the images, we pick a reference image among the set of images and find the sum of distances between the matched key-points in the pair of images selected using the **cv2.BruteForce()** matcher with the L2 norm distance. We check the image which has the least sum of distances, this means that this image matches the most with the reference. Repeating this steps repeatedly for images we obtain the final stitched image. If it is very distorted, we change the reference image and repeat until we get and acceptable, good quality stitched image.

5. How does the functionality you are providing (or could provide) differ from the Stitcher API.
Solution. The stitcher API, implements image blending which tends to resolve the exposure related issues between the two images being stitched. This difference in exposure leads to intensity variations around the beam resulting in a seam at the interface. This seam can be removed using image blending, which uses a weighted matrix in combination with the images to be stitched such that the transition appears smooth.

6. List items that you think need fixing. For example, the mosaicing results show a "seam". What techniques can be used to remove the seam?
Solution. Items that need fixing:
1. seam
2. In our implementation we also saw a black line wherever the images are getting stitched.
3. There is a lot of non image region (blank space) in the output image. This can be removed by projecting the image on cylindrical coordinates or spherical coordinates in order to remove the distortion obtained due to stitching images one over the other directly.
4. If there is not enough matching features between two images then we might not get a proper output.
The seam can be resolved using the following techniques :

1. Blending both the images at the seam. 2. Image blending, which uses a weighted matrix in combination with the images to be stitched such that the transition appears smooth.The weighted matrix is such that the weight of the pixels reduce as we move towards the edge. 3. Due to intensity difference, this can happen, so we need to make sure that contrast of the images that we stitch are same, which can be accomplished by matching the histogram of reference image to that of all the other images before stitching.