CSE 473/573 - Computer Vision and Image Processing

Spring 2021

Project #2

Name: Karan Bali
UBIT Name: kbali
UBIT Person Number: 50381691

NOTE : RUN-TIME is extremely HIGH
Note: I've pasted the resulting images at the end.

In this project, we were supposed to perform homographic warping, stitching & blending of 2 or more images to get a panoramic image. There were 3 tasks:

Task-1: Background stitching:

For this task:

1. At first, I get the dimensions of both images. If an image is too large, i have the option to resize it by maintaining the 'Aspect ratio'. This helps during task-2 when the in-situ panorama image might get too large & the required number of SIFT features for Ransac based Homography won't be possible. (** I've commented out this section as of playing SAFE because i wasn't aware of the test data resolution)

2. I calculated the SIFT Features (keypoints & descriptors) for 2 images. **The appropriate number of SIFT features required to perform task-1 & task-2 will depend on the nature of the particular images & other limiting factors (i.e. Resolution of images, expected run time, etc). So ideally, we'll like to compare all possible image's SIFT features, but this will exponentially increase the run time. So, I decided to assume that the images will be comparable in resolution to the ones given to us for task-2 (i.e. t2_{n}.png).**

3. Then I compare the descriptors using the norm of descriptors. I perform the ratio test between the 2 best matches and subsequently get the best match for a particular feature in both images. I store the indexes & keypoints of best matches in a variable.

4. I return a 'Bad Match' if the total number of good matches is below a threshold.

5. Then I find the Homography between 2 images using the best matches from the previous step & cv2.findHomography.

6. To get a perfect warp of both images (without cut-out), I use cv2.perspectiveTransform to get transformation between the corner of 2 images & using the result from cv2.perspectiveTransform & previously computed homography matrix, I wrap both images perfectly. Finally, I save & return the final warped image.

Task-2: Image Panorama:

For this task:

1. I'll copy & use the code from task-1 into the task-2 (i.e. function stitch_background()). It performs the same task as it did in task-2 (i.e. warping & stitching two images).

2. For the main part of task-2, I initialize the overlap-array. Then I Loop through all images twice to get the overlap-array (with flag=1, i.e. just want to know whether 2 images are good/bad match).

3. I find the image with the most number of matches & consider it as my main anchor image.

4. Then I loop through the overlap-array, to compute a new stitched image using the anchor image & j'th image with flag=0 (i.e. want to get a new warped image). I perform this step where overal[i,j] == 1. I also skip the part where i==j.

5. Finally, I save the final anchor image & return the overlap-array.

Task-3: Image Panorama Example:

For this task:

1. **I perform this task as same as task-2. The only difference is the images used for task-2 & task-3 relatively. As mentioned above, many SIFT & Homography-related calculations depend on the properties of images & the nature of the task. (i.e. Resolution of images, number of images to join together, aspect ratio, expected run time). Ideally, we should gather & compare all SIFT features. But this will**

increase our run time exponentially. So I assumed the images under consideration to have similar resolutions & the number of images to join. If this assumption is not held, we can still perform the task by not providing any input to 'cv2.SIFT_create('number of features'). But this might increase run time a lot.

2. **Although I do resize the image (by maintaining the aspect ratio), this only helps to an extent. Also, the resulting images might be too 'Zoomed out', if input images have a large & asymmetric resolution.**

Resulting Images:

Task-1:

Output Image: ** Scroll Down**

Task-2: ** Scroll Down**

Output image: Scroll Down****

Task-3: ** Scroll Down**

Output Image:

Input Images: ** Scroll Down**