University at Buffalo
Department of Computer Science and Engineering
CSE 473/573 - Computer Vision and Image Processing

Project #2
Due Date: 4/9/22, 11:59PM
Recent Updates in Yellow

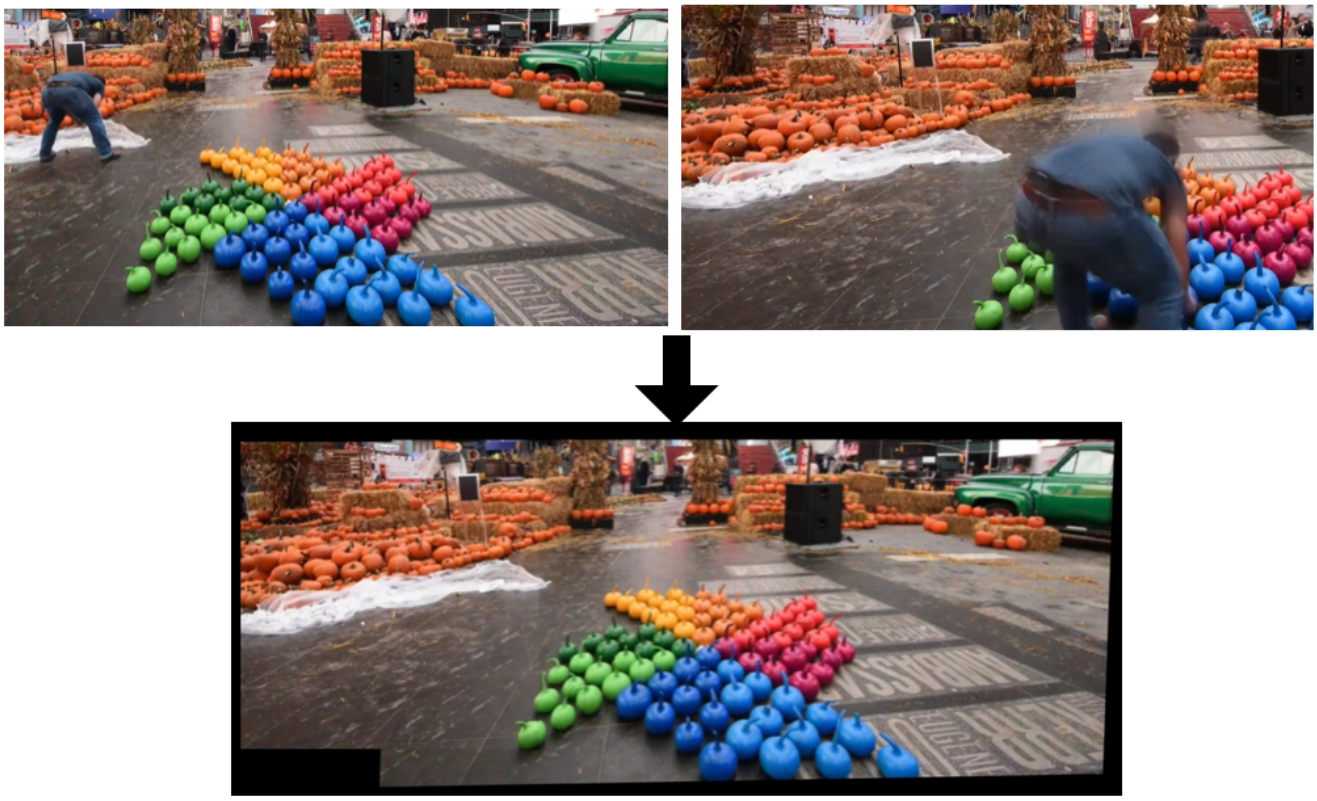

Figure 1: Example of background stitching.

1 Background Stitching (50pts)

The goal of this task is to experiment with image stitching methods. Two images may have the same background but different foreground. For example, a moving person may be moving in the scene. You need to stitch the two images into one image eliminating foreground objects that move

in the scene. You may assume that the region where a foreground object covers the background has similar characteristics to the rest of the background and is visible in two or more images. See the example in Fig 1. The shape of the output image after stitching might be irregular since you cannot crop either of the transformed images. There are no restrictions regarding the method you use to stitch photos into one photo.
Expected Steps:

- Extract a set of key points for each image.
- Extract features from each key point.
- Match features and use matches to determine if there is overlap between given pairs of images.
- Compute the homography between the overlapping pairs as needed. Use RANSAC to optimize your result.
- Transform the images and stitch the two images into one mosaic, eliminating the foreground as described above, but do NOT crop your image.
- Save the resulting mosaic according to the instructions specified in the script. (50 points)

2 Image Panorama (50pts)

This task aims to stitch multiple images into one panoramic photo. As shown in Fig. 2, the given images might be non-overlapping or multiply-overlapped (overlapping two or more other images). The shape of the output image after stitching might be irregular since you cannot crop either of the transformed images. There are no restrictions regarding the method you use to stitch photos into a panoramic photo. For this project, you can assume the following:

- Your code will need to be able to stitch together four or more images, and you will not know that in advance.
- You can assume that IF an image is to be part of the panorama, it will overlap at least one other image by at least 20%.
- Images that do not overlap with any other image can be ignored.
- Images can overlap with multiple images.
- Although the Figure below shows horizontal panoramas, your five images can be stitched together in any way.
- You are only expected to produce one overall image.
- While some of the most modern techniques may use a spherical projection for better panoramas, you are free to assume that basic 2D planer transformations are sufficient for this project.

Steps:

- Extract features for each image, match features, and determine the spatial overlaps of the images automatically. For instance, among four images, if image1, image3 and image4 overlap with each other, then your overlap array should be $\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$. Follow the instructions in the provided script and save the overlapping result as a $N * N$ one-hot array in the JSON file. (20 points)
- Conduct image transformation and stitch into one panoramic photo. Save the photo as the instructions specified in the script. (30 points)

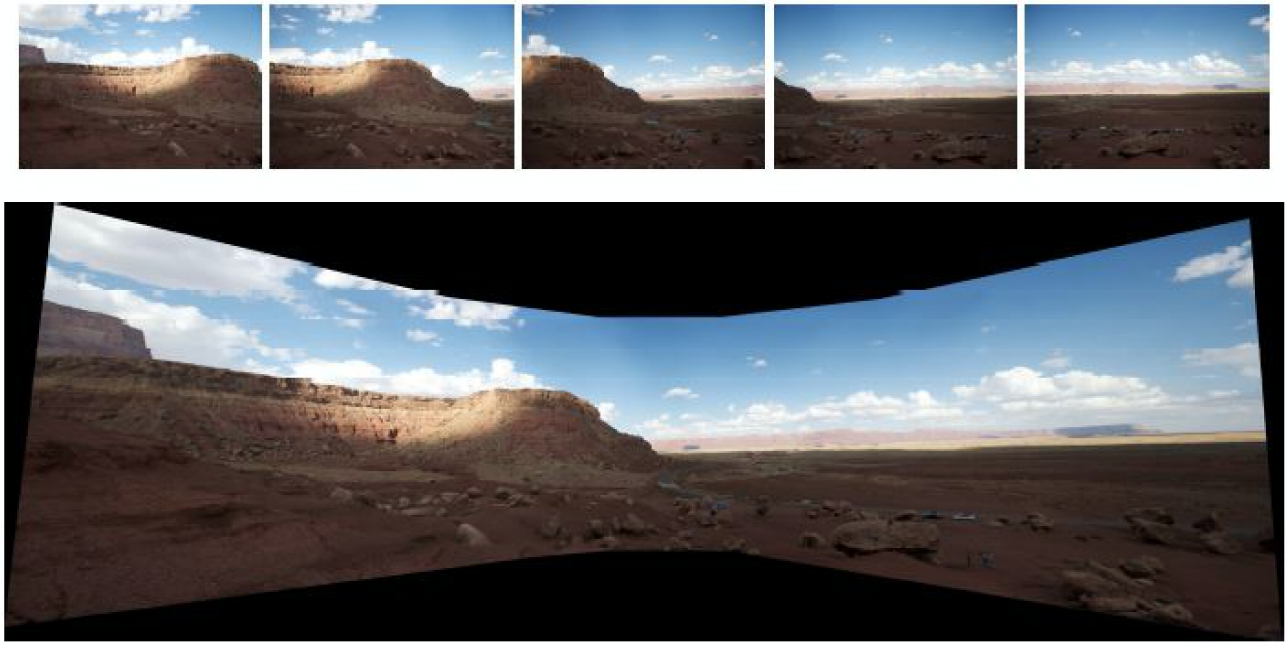

Figure 2: Example of image panoramas.

3 Image Panorama Example (10pt Bonus)

Find something that is Buffalo or UB-related (suggested by not required), photograph it, and demonstrate that your code works on regions where at least four images overlap IN A VERTICAL DIRECTION. Set the number of your images N, and name your input images as specified in the script.

4 Project Guidelines and Submission

- Do not modify the code provided.

- You can use the basic Python and OpenCV libraries for feature extraction and homography computation (cv2.findHomography()), but you may NOT use any libraries related to establishing candidate matches (APIs that have "match" in their names) or stitching (APIs that have "stitch" or "mosaic" in their names)
- You should map and blend pixels across images based on the homography computation to obtain the final mosaiced images for all parts of this project.
- You may NOT copy code or code snippets from other projects on the internet.
- If you decide to use SIFT for feature extraction, please note that it has been patented and removed from OpenCV3, but it is included in OpenCV2.
- All work should be your own. You cannot copy or plagiarize code from other students or online discussion boards, websites, or archives. Any attempt to do so will fail.
- Compress the python files, i.e., t1.py, t2.py, images folder, results.txt, "result.png" into a zip file, name it as "UBID.zip" (replace "UBID" with your eight-digit UBID, e.g., 50305566) and upload it to UBLearns before the due date. Not following the name rules "UBID.zip" will result in a reduction in grade.
- Late submission guidelines apply for this project.
- Not following the project guidelines will result in a reduction in grade.

5 Checking your submission with autograder

6 Hints for installing OpenCV

6.1 OpenCV SIFT Link

You might find the following link helpful if you plan to use SIFT feature from OpenCV API.
`https://docs.opencv.org/3.4.2/d5/d3c/classcv_1_1xfeatures2d_1_1SIFT.html`

6.2 Installation

The latest version of OpenCV will no longer include this API in its standard package, and fees might apply if you want to use it. The older version of OpenCV still has it. Below are a few steps to help you get the older compatible version.

- Remove installed OpenCV (Skip this step if you don't have it installed)

```
$ pip uninstall OpenCV-python
```

- Install opencv 3.4.2.17

```
$ pip install opencv-python==3.4.2.17
```

- Install opencv contribution library 3.4.2.17

```
$ pip install opencv-contrib-python==3.4.2.17
```

OpenCV(3.4.2.17) has been tested and it does provide a fully functional API like

```
cv2.xfeatures2d.SIFT_create()
```

Note: You may need **administrative privilege** to execute the installation on your machine.