McGill University
ECSE 415 - Intro to Computer Vision
Assignment #2 - Due 11:59pm - October 26th, 2017

Preface

The purpose of this assignment is to give you an introduction to image features (i.e. computing and matching). Please submit your assignment solutions electronically via the myCourse assignment dropbox. The solutions should be in **Jupyter Notebooks** format, see tutorial #2 for a reference. The complete assignment submission should be a zip file containing your Jupyter code (.ipynb) in addition to all output images. Attempt all parts of this assignment. The assignment is out of a total of **30 points**. The student is expected to write his/her own code. Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be marked.

Problem 1 - Harris Corner Detection [10 points]

For this question, you will take your own image and perform the Harris Corner Detection algorithm. It will be helpful to resize your input image and convert it to grayscale using the built-in openCV functions. A step-by-step guide of implementing this algorithm is provided in Lecture 6 – Slide 46. All portions of this algorithm need to be implemented by hand using nested for-loops. Similar to assignment #1, zero-padding the image will prove beneficial. The output of this algorithm should consist of the input image and a method to display the Harris Corners (e.g. using openCV's cv2.circle() function).

Problem 2 - Panoramic Images [20 points]

For this question you will be taking two new images of a scene to create a panoramic image. Note: You should take these pictures by rotating the camera in place, although the homography matrix can account for camera translation it is easier to only estimate for rotation. The procedure to create a panoramic image are as follows:

1. Detect keypoints and extract local invariant descriptors from the two input images. For this step you will be using SIFT or SURF features. You may use openCV's SIFT or SURF object to detect these features (cv2.xfeatures2d.SIFT_create() or cv2.xfeatures2d.SUFT_create()). You should display the detected keypoints on the two input images (you may use cv2.drawKeypoints()). **[2 pts.]**

2. Match the feature descriptors between the two input images. You are allowed to use the BFMatcher() class. Again, you should display the matched keypoints (you may use cv2.drawMatches()). **[2 pts.]**

3. Use the RANSAC algorithm to estimate a homography matrix using the matched feature vectors. You will be implementing the RANSAC algorithm yourself but you may call on openCV's cv2.findHomography() to find the homography matrix (set method = 0). You should reference the steps in Lecture 9 - slide 50. The number of samples must be set to $N = 10$ and you should try different threshold values and choose which you think is most optimal. Once you find the best RANSAC solution (highest number of inliers) you

must display the results. On the original image pair you should color the inliers yellow and outliers blue (see Lecture 9 - slide 51 as a reference). **[8 pts.]**

4. Apply the warping transformation using the matrix found in step 3). **[2 pts.]**
5. Blend the two images together. You can use openCV's blend function for blending the resultant images. You will have to crop your images accordingly to use this function. The final output of this algorithm must display your two initial input images and the final panoramic image.**[5 pts.]**
6. Lastly, you should comment on any artifacts you notice from the resulting overlap. **[1 pt.]**

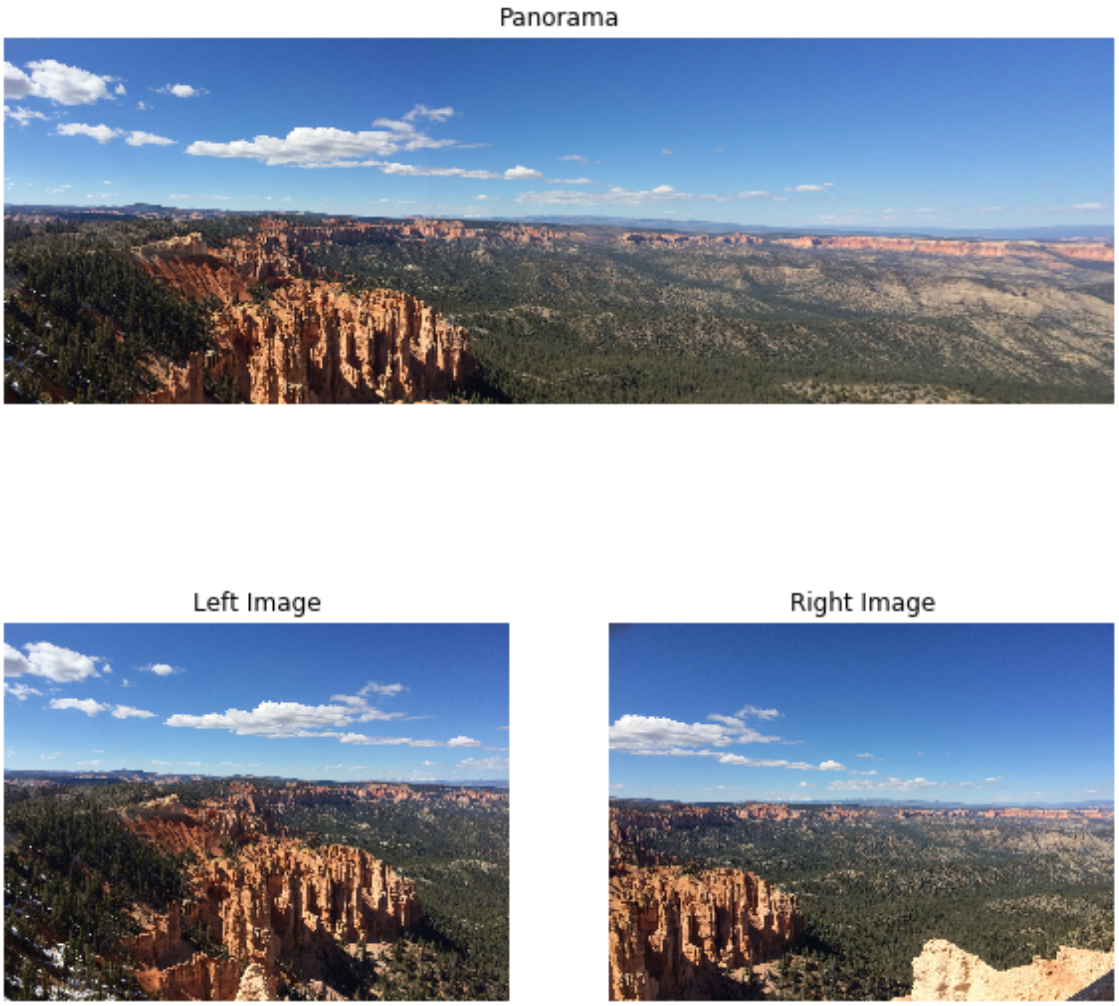

Fig. 1 - Panoramic Sample