First Assignment (Image Stitching)

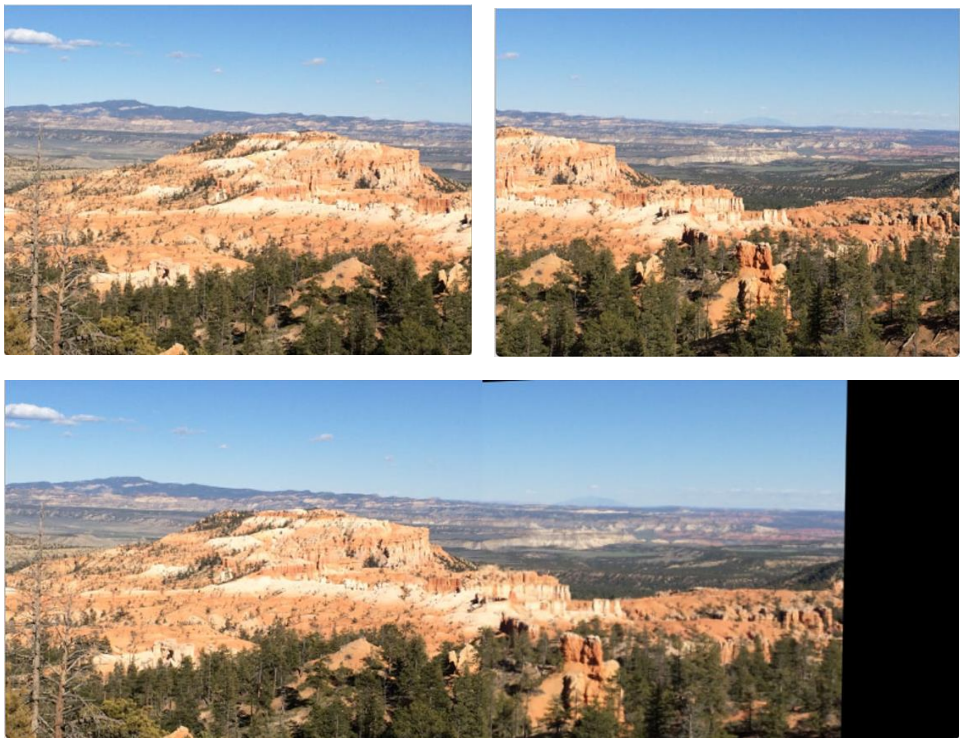
TUTORS: MIRELA POPA

TEACHING ASSISTANT: BULAT KHAERTDINOV

DEADLINE: 09/05/2022 AT 11.59 PM

DESCRIPTION

In this assignment, you will develop your own panorama application by stitching together pairs of images, like in the example below. Use your own pairs of pictures (show analytically reported results on one image pair but also visualize how the algorithm works on 2 or 3 more images). Your images should be 'parallel' to each other to avoid perspective distortions.



STEPS

1. Use feature points that you can detect using the Harris corner detection (check detectHarrrisFeatures in Matlab or corner_harris from skimage.feature library in python, or other functions available in opencv – cv2 library in python). Describe every key-point by extracting fixed-size patches around it (try different sizes and report a short sensitivity analysis of the final result).
2. Optionally, compute SIFT descriptors (check cv2.FeatureDetector_create() in opencv v2.x or cv2.xfeatures2d.SIFT_create() in opencv v3.x, version in which you will need the opencv_contrib package) on the resulted key-points from Harris corner detector. You can utilize for Matlab the VLFeat library by using the custom frames (key-points). For this purpose, check vl_sift.
3. Compute the distances between every descriptor in image 1 with every descriptor in image 2. For this, use: a) Normalized correlation and b) Euclidean distance after normalizing each descriptor.
4. Select the best matches based on a threshold or by considering the top few hundred pairs of descriptors. Also, make a sensitivity analysis based on these parameters.
5. Make a simple implementation of RANSAC to estimate an Affine transformation mapping one image onto the other. For this, you need 3-5 matches of points to initialize the estimation of the transformation. Evaluate the obtained transformation by fitting the other points against the model and compute the ratio of inliers/outliers. For finding a good transformation, you need to consider a number of iterations, with different random initializations of the algorithm. Report the number of inliers and outliers and the average residual for the inliers (the squared distance between the point coordinates in one image, and the transformed coordinates of the matching points in the other image). Display the location of the inliers matches on both images.
6. Warp image 2 onto image 1 using the best estimated transformation with the highest number of inliers. You will need to learn about affine2d in Matlab or AffineTransform in python. Then, you can check functions for applying geometric transformations on images, such as imwarp in Matlab and warpPerspective from cv2 library in python).
7. Define as a score of accuracy the Euclidean distance between chosen key-point coordinates in image 1 and the corresponding transformed ones in image 2. Base every sensitivity analysis (see steps above) based on this score and plot the results for every parameter you use.
8. Perform experiments with various images and include the results in the report. Optionally, you can try also difficult cases like “look into the past image stitching” (e.g. like here, where part of the photo is a modern aspect of a city/landscape and part of it

comes from the past). You can do the same with images showing a location at different times of the year, etc.

IMPLEMENTATION TIPS:

- For RANSAC, a very simple implementation is sufficient. Use three to five matches to initialize the Affine transformation in each iteration. You should output a single transformation that gets the most inliers in the course of all the iterations. For the various RANSAC parameters (number of iterations, inlier threshold), play around with a few "reasonable" values and pick the ones that work best. For randomly sampling matches, you can use the randperm or randsample functions in Matlab and numpy.random.permutation() or random.sample() functions in python.
- In MATLAB, the solution to a nonhomogeneous linear least squares system AX=B is given by X = A\B, and for python check numpy.linalg.lstsq for computing the solution of the equation.
- For more details about feature extraction, matching, image fitting and aligning, and different transformations, please be referred to the following lectures in Canvas: Lecture 3a – Key features and motion estimation (feature_detection_matching.pdf) and Lecture 3b: Hough, RANSAC, ICP (Hough-Ransac-ICP-Transformations_master.pdf).

DELIVERABLES:

1. A report of 500-1000 words with plots and figures. Do not forget to display everything (key-points used in every step, etc.). Provide experimental results for the aforementioned information in the implementation steps. For example, describe your implementation, and the interesting parameters for RANSAC, Affine Transformation, feature description and matching, your challenges, observations extracted from the performed experiments (you should not repeat the steps described above, but write about what you noticed, what was difficult and especially which set of parameters/methodology led to a good output).

2. Well documented code in python (a script or a jupyter notebook). Your code will be tested with five image pairs of ours.

3. Upload everything on Canvas as a zip file, using the following format: Name_Surname_First_Assignment_CV2022.zip.

Grading

A) **Code – 1.5 Points**
B) **Report– 1 Point**