CSCI 3290: Computational Imaging and Vision **(Due: 23:59 17/03/2020)**

Assignment 2: Image Stitching

1 Assignment description

Image stitching is a technique to combine a set of images into a larger image by registering, warping, resampling and blending them together. A popular application for image stitching is creation of panoramas. Generally speaking, there are two classes of methods for image stitching, direct methods and feature-based methods. In this assignment, we will implement a feature-based method. It contains three parts: i) feature detection and matching, ii) homography estimation and iii) blending.

2 Assignment details

2.1 Feature detection and matching

Given two input images, detect SIFT features from them. And then establish the feature correspondence between SIFT features in the two input images using Brute-Force matcher. In OpenCV, there is a tutorial for SIFT https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html, and a tutorial for Brute-Force matcher https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html.

This part is corresponding to *extract_and_match_feature* function in the provided skeleton code:

```
def extract_and_match_feature(img_1, img_2, ratio_test=0.7):
    """
    :param img_1: input image 1
    :param img_2: input image 2
    :param ratio_test: ratio for the robustness test
    :return list_pairs_matched_keypoints: a list of pairs of matched points:
    [[[p1x,p1y],[p2x,p2y]]]
    """
    list_pairs_matched_keypoints = []

    # to be completed ....

    return list_pairs_matched_keypoints
```

You need to do the following things in this function:

1) extract SIFT feature from image 1 and image 2,

2) use a bruteforce search to find pairs of matched features: for each feature point in img_1, find its best matched feature point in img_2,

3) apply ratio test to select the set of robust matched points.

2.2 Homography estimation

The 2D image transformations (translation, rotation, scale, affine and perspective) can be represented as homography. Therefore, we can estimate the best homography by the matched pairs of features. Also, we employ RANSAC algorithm to eliminate the "bad" matches. This can make our program more robust. In this section,you need to implement the RANSAC algorithm to find a robust homography between two input images using the feature correspondence.

This part is corresponding to *find_homography_ransac* function in the provided skeleton code:

```
def find_homography_ransac(list_pairs_matched_keypoints,
                           threshold_ratio_inliers=0.85,
                           threshold_reprojection_error=3,
                           max_num_trial=1000):
    """
    :param list_pairs_matched_keypoints:
        a list of pairs of matched points: [[[p1x,p1y],[p2x,p2y]],...]
    :param threshold_ratio_inliers:
        threshold on the ratio of inliers over the total number of samples,
        accept the estimated homography if ratio is higher than the threshold
    :param threshold_reprojection_error:
        threshold of reprojection error (measured as euclidean distance, in
        pixels) to determine whether a sample is inlier or outlier
    :param max_num_trial:
        the maximum number of trials to take sample and do testing to find
        the best homography matrix
    :return best_H: the best found homography matrix
    """
    best_H = None

    # to be completed ...

    return best_H
```

2.3 Blending

Having the estimated homography, we can warp the second image to align with the first image and then blend them together to get a a single panorama image. For warping, we employ inverse warping and bilinear resampling.

This part is corresponding to *warp_blend_image* function in the provided skeleton code:

```
def warp_blend_image(img_1, H, img_2):
    """
    :param img_1: the original first image
    :param H: estimated homography
    :param img_2: the original second image
    :return img_panorama: resulting panorama image
    """
    img_panorama = None

    # to be completed ...

    return img_panorama
```

You need to do the following things in this function:

1) warp image img_2 using the homography H to align it with image img_1 (using inverse warping and bilinear resampling),

2) stitch image img_2 to image img_1 and apply average blending to blend the two images into a single panorama image.

3 Submission guidelines

3.1 Marks

* extract_and_match_feature: 40 points
* find_homography_ransac: 30 points
* warp_blend_image: 30 points

3.2 Put personal information in your source code

In all your source files, type your full name and student ID, just like:

```
#
# CSCI3290 Computational Imaging and Vision *
# --- Declaration --- *
# I declare that the assignment here submitted is original except for source
# material explicitly acknowledged. I also acknowledge that I am aware of
# University policy and regulations on honesty in academic work, and of the
# disciplinary guidelines and procedures applicable to breaches of such policy
# and regulations, as contained in the website
# http://www.cuhk.edu.hk/policy/academichonesty/ *
# Assignment 2
# Name :
# Student ID :
# Email Addr :
#
```

3.3 Late submission penalty

If you submit your solution after the due date, 10 marks will be deducted per day, and the maximal deduction is 30 marks even you delay more than 3 days. But there are hard deadlines as we need time to grade and submit grade. The hard deadline is 29 April 2020. After this day, we will not grade your assignment.