The Hong Kong University of Science and Technology
Department of Computer Science and Engineering
CSIT 5410 (Spring 2020)

Assignment 3

Total = 100 marks
Due: 11:55pm, May 04, 2020
Assignments must be submitted via Canvas
Late Policy: 10% reduction; only one day late is allowed, i.e. 11:55pm, May 05

Overview

This assignment consists of a major topic: image stitching. You need to follow the sub-steps below and complete the stitching process. No skeleton codes will be provided except for the csit5410_assign3.m script. The basic requirement is that your program should be able to generate the correct stitching images under the csit5410_assign3.m. All the work should be done by yourself. No third-party library or code and the built-in functions relating to the main steps are allowed to use. All the work should be submitted via Canvas system.

Programming assignment specifics (70%)

M-file: *csit5410_assign3*
The routine `csit5410_assign3` complete the following tasks:
Task 1: Cylindrical projection
Task 2: Corner point detection
Task 3: Distance Matching
Task 4: Random Sample Consensus (RANSAC)
Task 5: Merge

(Task 1) Task 1: Cylindrical Projection – Image Warping
Generally, in order to obtain a natural stitching result, we need to warp the input images to the cylindrical surface as a preprocessing step for image stitching.
The following formulations are the forward transformation that converts flat coordinates to cylindrical coordinates:

$$\begin{cases} x' = f * \tan^{-1}\left(\frac{x-\frac{W}{2}}{f}\right) + f * \tan^{-1}\left(\frac{W}{2*f}\right) \\ y' = \frac{f*(y-\frac{H}{2})}{\sqrt{\left(x-\frac{W}{2}\right)^2+f^2}} + \frac{H}{2} \end{cases},$$

Where (x', y') is the coordinate on the cylindrical surface, f is the locus of the camera, which is equal to,

$$f = \frac{W}{2 * \tan\left(\frac{\varphi}{2}\right)},$$

$\varphi \in (0, \pi)$ is a parameter controlling the degree of deformation.
However, in the usual practice, the reverse transformation, which converts cylindrical coordinates to original coordinates, is more useful since it can avoid "black dot" effects. Please apply the **reverse transformation** to implement your cylindrical projection,

Algorithm 1. Cylindrical Projection

Input: Image I, degree of deformation φ

1. Compute the new height H' and new width W' of the projected image I'
2. **for** (x', y') in I' **do**
   $x = reverse\_transform(x', \varphi)$
   $y = reverse\_transform(x', y', \varphi)$
   $I'(x', y') = I(round(x), round(y))$
   end for
3. **Output**: Projected image I'

The above pseudo-codes use the simplest way to map the intensity value of the original image to the projected image, which is called nearest interpolation. You may consider to use bilinear interpolation or bicubic interpolation to obtain a better projection result.

(Task 2) Task 2: Corner Point Detection – Feature Extraction

In order to match two images, we need to extract some features that can be used to represent some significant parts of the images. In this task, you are required to implement your own Corner Point Detector based on what you have learned in the lectures. However, using the intensity values of the corner points detected by your detector only may not be sufficient enough to measure the similarity of the corresponding positions of two images. A local region centered at the detected corner points should be considered and then extract some statistical values from it to generate a descriptor for each corner point.

The function prototype of the corner point detection is as follows,

```
[Points,V] = harris(Img, threshold),
```

Where 'Img' is the input image, 'threshold' is the thresholding value, 'Points' are the coordinates of the result corner points and 'V' are the intensity values of the corresponding corner points. For example, V(i) == Img(Points(i, 1), Points(i, 2)).

The function prototype of the local feature calculation is as follows,

```
descriptors = featureDescribe(img, Points),
```

where 'img' is the input image, 'Points' are the corner points detected by the corner point detector and 'decriptors' is a matrix with row equals to the number of the corner points detected and column equals to the number of the descriptors (e.g. statistical values) you use for each corner points.

(Task 3) Task 3: Distance Matching – Similarity Measurement

Euclidean distance can be employed to match the corresponding corner points among two images.

The function prototype of the distance matching is as follows,

```
[matchA,matchB] = distanceMatch(describeA,describeB,pointsA,pointsB,
                                threshold),
```

where 'describeA' and 'describeB' are the matrices obtained by the function `featureDescribe` for image A and image B respectively, 'pointsA' and 'pointsB' are the coordinates of the corner points in image A and image B respectively and 'threshold' is a thresholding value of the Euclidean distance for the decision of matching. 'matchA' and 'matchB' are the coordinate matrices of the corner points that are matched with column equals to the number of the matched points.

NOTE: in order to compute the geometric transformation in the RANSAC function, we should use 3D homogenous coordinates here. That is, the size of 'matchA' should be 3*M, where M is the number of corner points matched. For each matched points, the coordinate should be (x,y,1).

(Task 4) Task 4: Random Sample Consensus (RANSAC)
We use RANSAC to find a best transformation that contains the largest number of inliers.

The function prototype of the distance matching is as follows,

```
[H, inliers] = RANSAC(pointsA,pointsB,threshold,iteration),
```

where 'pointsA' and 'pointsB' are the coordinate matrices of the corner points that are matched with column equals to the number of the matched points (the outputs of the function `distanceMatch)`. 'threshold' is the parameter for evaluating the error during iterations and 'iteration' indicates the number of iterations. The output parameters are the transformation matrix and the points of inliers.

Workflow of the provided RANSAC function:
*Step 1: Select **8** pairs of point randomly and calculate the transformation matrix H*
Step 2: Calculate the error = ||Hu - v|| and the number of point n that has error smaller than threshold
Step 3: if n > 95% or reach the maximum number of iterations, return H. Else, repeat the algorithm from Step 1 and update H if n' > n
Step 4: re-compute H by estimating on all inliers (Least square fits)

The transformation matrix in the image stitching is the homography. The definition is given as follows,

- A **Homography** is a transformation (a 3×3 matrix) that maps the points in one image to the corresponding points in the other image.
- Define the homography $\boldsymbol{H}$ as,

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

- Let us consider the first set of corresponding points — (x_1, y_1) in the first image and (x_2, y_2) in the second image. Then, the homography $\boldsymbol{H}$ maps them in the following way,

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

- We can force the last component of $\boldsymbol{H}$ to be 1 so that we can solve $\boldsymbol{H}$ by an inhomogeneous linear system. That is, $h_{33} = 1$.
- **You need to use 8 pairs of corner points to compute the homography in this assignment.**

(Task 5) Task 5: Merge

Here is the last step for image stitching. You should use the transformation matrix *H* obtained from RANSAC to merge the two images.
After the homograph is computed, you should wrap the image and composite them together. The warping should be done using the inverse of the homograph since the forward warping will create black dots as what we have talked about in the cylindrical projection.
The program should firstly compute the bounding rectangle of the resulting image and then use the inverse homograph matrix to do the wrapping.
After the wrapping, the program then composites the overlapping part of the two images by interpolating the color intensity. This is called "blending".

The function prototype of the merging is as follows,

```
result = merge(imgA,imgB,H),
```

where 'imgA', 'imgB' are the two input images, 'H' is the transformation matrix obtained from RANSAC and 'result' is the resulting image.

Sample runs of the programming assignment

You are supposed to obtain the outputs on the screen when you run the following command in the MATLAB/OCTAVE environment:
>> csit5410_assign3

Assignment Submission and Marking

Your submitted programs should be *cylindricalProjection*, *harris*, *featureDescribe*, *distanceMatch*, *RANSAC* and *merge* plus all the other related M-files. You must compress all your files with the following filename format: [your 8-digit student ID]_assign3.rar (or zip), e.g: 07654321_assign3.rar (or zip), into one file.

Please also attach your stitching results for different groups of images when you submit the codes. If there are more than two images in the group, then your result image should be the stitching results for **all** the images within the group.

Note that we take plagiarism seriously. You are allowed to discuss or share your idea to your classmate, but **you are not allowed to share your code/pseudocode of your assignment**. Please also follow the referencing skills at https://libguides.ust.hk/referencing/plagiarism to avoid plagiarism.

If your assignment compressed file has been submitted multiple times before the due date (including late submission date), the newer version will replace the old version in marking.

~~ End of Assignment 3 ~~