CMPE 565- Assignment #2

Due: 17.11.2019, 23:59

Note: Homework is to be done individually. You need to submit you reports and codes to Moodle. Please do not insert your codes to your reports. Codes should come separately. In your report just put the input images and output results and explicitly stated parameters being used (like sigma parameter for Gaussian).
The images are provided in "data" folder.

1) **[20] Harris Corner Detection**

   Implement the Harris corner detector. To do this, you'll need to write a function:

   [cornerPts, numCornerPts]=HarrisCornerDetector (*image*, *sigma*, double *thres*)

   image: the input image,
   sigma: the standard deviation for the Gaussian,
   thres: the threshold for detection corners,
   cornerPts: the returned corner points,
   numCornersPts: the number of points returned,

   Compute the Harris corner detector using the following steps:

   i. Compute x and y derivatives of the image, use them to produce 3 images (I_x^2, I_y^2 and $I_x I_y$) and smooth each of them with the Gaussian, or, first apply the Gaussian and then compute the derivatives.
   ii. Compute the Harris matrix H for each pixel.
   iii. Compute corner response function $R = \frac{Det(H)}{Tr(H)}$, and threshold R.
   iv. Find local maxima of the response function using nonmaximum supression [use 5x5 window]

Required: Open Boxes.png in data folder. Run your HarrisCornerDetector function and show the results on this object. Save the resulting image as "1a.png". Show the result in your report.

2) **[10] Describe and match features in two images.** For this part you are going to use already existing routines in Matlab (this functions are valid in Matlab2019)

   i. Compute the descriptors for each corner point using following function:

      [features, validPoints]=extractFeatures(image,corners);

      Here, corners are the feature points that you have detected in step (1).

   ii. For each corner point in image 1, find its best match in image 2 using indexPairs=matchFeatures(features1, features2)

iii. Display the matches using

showMatchedFeatures(image1,image2,matchedPoints1,matchedPoints2);

Note: Look up for the explanation of this functions in Mathworks or in Matlab documentation!!

Required: Open "Rainier1.png" and "Rainier2.png", compute the Harris corner detector and find matches. Save the images "2a.png" and "2b.png" respectively. Show results in you reports.

3) **[45] Compute the homography between the images using RANSAC.** Following these steps:

i. **[10]** Write the prjection function:

[x2, y2]= ***Project***(*x1*, *y1*, *h*)

This should project point (*x1*, *y1*) using the homography *h*. Return the projected point (*x2*, *y2*). See the slides for details on how to project using homogeneous coordinates.

ii. **[10]** Write the function to count inliers

ComputeInlierCount(*h*, *matches*, *numMatches*, *inlierThreshold*).

This is a helper function for RANSAC that computes the number of inlying points given a homography *h*. That is, project the first point in each match using the function "Project". If the projected point is less than the distance *inlierThreshold* (default is 5) from the second point, it is an inlier. Return the total number of inliers.

iii. **[25]** Write the main Ransac function

[*hom, homInv*]= ***RANSAC*** (*matches*, *numMatches*, *Iterations*, *inlierThreshold*)

matches: a set of *numMatches* matches,
Iterations: the number of times to iterate,
inlierThreshold: a real number so that the distance from a projected point to the match is less than its square,
hom: the homography and *homInv* its inverse,

This function takes a list of potentially matching points between two images and returns the homography transformation that relates them. To do this follow these steps:

a) Iteratively do the following for "Iterations" times: (try 200)

   1. Randomly select 4 pairs of potentially matching points from "matches".
   2. Compute the homography relating the four selected matches. Look up the slides for the math.
   3. Using the computed homography, compute the number of inliers using "ComputeInlierCount".
   4. If this homography produces the highest number of inliers, store it as the best homography.

b) Given the highest scoring homography, once again find all the inliers. Compute a new refined homography using all of the inliers (not just using four points as you did previously.) Compute an inverse homography as well (inv() function in Matlab should work), and return their values in "hom" and "homInv".

c) Display the inlier matches using " showMatchedFeatures ". Show in your reports.

4) [25] Stitch the images together using the computed homography. Write image stitching function:

[stitchedImage]= ***Stitch*** (*image1*, *image2*, *hom*, *homInv*)

image1, image2: the input images,
hom: the homography and *homInv* its inverse,
stitchedImage: result.

Follow these steps:

i. Compute the size of "stitchedImage. " To do this project the four corners of "image2" onto "image1" using Project and "homInv". Allocate the image.
ii. Copy "image1" onto the "stitchedImage" at the right location.
iii. For each pixel in "stitchedImage", project the point onto "image2". If it lies within image2's boundaries, add or blend the pixel's value to "stitchedImage. " When finding the value of image2's pixel use BilinearInterpolation (you can use interp2 function in Matlab).

Required: For all image pairs in data folder run the algorithm from 1-4 and save the images. Show results on your reports.