Yale CS 479/579 Problem Set 5 2014

Due Saturday, April 19, 2014, 6:00 pm
Submit by uploading to classesv2.

Below you will find the problems for Problem Set 5 Please note the following:

1. All problems should be uploaded to ClassesV2 together in a single zip folder.
2. Each problem should be in its own folder of the format "problem*n*". For example the solution to the first problem should be placed in a folder called "problem1a"
3. All assignments will be graded in the Zoo, which is a Linux environment. This means that all file and folder names are case-sensitive! It also means that you should test your solution on the Zoo computers.
4. You can start Matlab on the Zoo by opening a terminal and typing in *matlab*.
5. An automatic grader may be used, which may not be forgiving of formatting errors.
6. Check ClassesV2 from time to time, as we will post updates and correct any errors in the assignment there.
7. All Matlab functions should have its own m file with the same name as the function (as per Matlab's convention)

Problem 1 (Matlab, 479, 579)

Write a version of the Harris corner detector as follows (note that M and R are defined as in Lecture 19 – slides are available under "Resources").

1) Compute the image derivatives at each point in the image using the mask (-2, -1 , 0 , 1, 2) in the x and y directions.
2) Compute the matrix M at each pixel by doing a simple average of the values I_x^2, I_y^2, I_{xy} in a 15x15 pixel window centered on the pixel.
3) Compute the value R= det(M) - .04 trace(M)^2, and display a color coded result of the resulting values for the two images given with this problem.
4) Choose an appropriate threshold to identify potential corners and display the result as a binary image.
5) Select pixels as feature points if they have the maximum value of R in a 15 x 15 window around the pixel. Display the result as a binary image.
6) Compare the results of the detector you built in 1) through 5) with the vision.CornerDetector available in the Computer Vision toolbox in Matlab.

Place your code in a Matlab function

```
[feature_rows, feature_cols] = harris(image);
```

The function should take an image as input, in the format returned by imread, and produce two 1d arrays as output. Each element of the arrays should be the row and column of the detected feature.

(579 only) Compare the quality of the feature matches between points using the version of the corner detector coded in steps 1-5 and using a (15x15) pixel window as the feature descriptor, and using the descriptors provided by the Matlab toolbox. You can place your code in a function harris_512 with the same parameters and output as the function above.

Problem 2 (Matlab, 479 and 579)

For the simple binary image given with this problem, compute the direction of the image gradient at the corners of the boxes using a method that is reminiscent of SIFT (see slides 23-29 in Lecture 20). First, compute I_x and I_y using the mask (-1, 0, 1). Next, compute the histogram of gradient directions in a 5x5 window around each corner. Your histogram should use 10 bins for the directions 0 to 360. Display the image with arrows pointing in the gradient direction computed.

Include a readme.pdf that includes both your images and your histogram for each corner. You can use the matlab quiver function to draw arrows.

(**579 only**) Compute the result of the gradients for the same image rotated 30 degrees. Comment on how consistent the directions are, and whether more bins would be needed to use these directions in rotating patches to examine feature matches. Include your result, and any data you need to back it up, in the readme.pdf file listed above.

Problem 3 (Matlab, 479 and 579)

For the two images given with this problem, there is a list of possible point matches provided in match_points.mat. The mat file contains two variables: 'points_image_1' and 'points_image_2' with corresponding columns. That is, the location of the points of the *i*th match in the first image is in the *i*th column in points_image_1. The corresponding point in the second image is in *i*th column in points_image_2.

Assuming translation only is needed to find the transformation that will properly relate the two images, use the RANSAC technique with 5 random samples to estimate the translation. You are to implement the RANSAC technique yourself in a function called RANDSAC and not use Matlab's implementation. Provide a function

```
[t_row,t_col,inlyers] = RANDSAC(image1, points_image_1, image2,points_image_2)
```

where image1 and image2 are in the format provided by imread and the points inputs are described above. The output should be change in row and change in column for the translation and an array containing the indexes of inlyers. **Do not assume that both inputs are the same size**.

Provide a readme.txt which includes the translation your program found in the input images and a sorted list of inlier indexes that agrees with that translation.

Problem 4 (Matlab, 479 and 579)

In this problem you are going to implement an image blending function that works on two levels of a pyramid. Create your pyramid by using the original image as your base and a Gaussian blur of the image as the higher level. You can then calculate the details of the base level by finding the difference between it and the higher level.

After you create your pyramid, you need to blend each level independently. First blend the higher level using a mask. Afterwards, blend the details of the lower level using a different mask. You should then be able to combine the higher and lower levels of your blended image into a single output.

Create a matlab function

```
blend = multiresolution_blend(left, right)
```

that takes two equally sized images and blends them together. The transition between them should be in the center of the image. That is, the left part of the blended image should come from `left` and the right part of the blended image should come from `right`.