Homework 3
M1522.001000 Computer Vision (2019 Spring)
Due: Wednesday April 25, 2:00PM

There are 6 Questions in this assignment, 2 theory and 4 programming questions.

1 Submitting Your Assignment

Your submission for this assignment consists of answers to a few theory questions, the code for your **MATLAB** implementation and a short writeup describing any thresholds and parameters used, interesting observations you made or things you did differently while implementing the assignment. Make sure all the files needed for your code to run (except data) are included. Your writeup should be **typed** with **English**

Put your files into a directory called "(studentid)-(yourname)-HW3" and pack it into a tar.gz or zip named "(studentid)-(yourname)-HW3". For example, 201721234-gildonghong-HW3.tar.gz or 201721234-gildonghong-HW3.zip and send it to TA's email (`ta.cv@vision.snu.ac.kr`) with title as "(studentid)-(yourname)-HW3".

Your submitted zip file should include the files arranged in this layout:

- `theory.txt` or `theory.pdf` : answers to the theory questions
- Folder `result`
  - `taj1r_warped.png` : png image showing the results of warping applied to taj1r.png
  - `taj_merged.png` : png image showing the results of combining taj1r.png and taj2r.png
  - `HTC_rectified.png` : png image showing the results of rectifying HTC.png
  - `writeup.doc` or `writeup.pdf` : the writeup describing your experiments
- Folder `matlab`
  - `ransac.m,computeH.m, computeHnorm.m, warpImage.m, rectify.m`
  - `main_ransac.m, main_warpImage.m, main_rectify.m`
  - `points.mat`, which contains lists of the corresponding points you used for main_warpImage.m and main_rectify.m.

Refer to Section 3 and Section 4 for the details of the above files.

Your zip file should be sent before the due.If it is later than that, you will use up one late day.

2 Warm Up

This section is meant to provide a short intro to get you used to working with images in Matlab. It is recommended that you do this sections early so that you will not be overwhelmed by what the rest of the assignment tasks ask you to do.

Let us start with the toys image (*toys.jpg*) and perform some simple image warping. Let us begin with a simple operation that will stretch the image in the horizontal direction. One possible method that can be used is to apply a transformation matrix to each point/pixel in the image. Apply the transform T_h to the image (ie. $T_h P_{i,j}$, where $P_{i,j} = [i, j, 1]^T$ is a point in the image expressed in homogeneous coordinates).

$$T_h = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The transformation matrix T_h will stretch the image in the vertical direction by a factor of 2. Take care to properly generate the warped image without holes in the new image by mapping every point/pixel in the new image to a corresponding value in the old image and not the reverse. (This means that your loop should iterate for every pixel in the warped image which has twice as much pixels as the original image in the vertical direction and the same number of pixels in the horizontal direction). It is important to observe the relationship between the transformation matrix T_h and the resultant image. Further explore this relationship by modifying the transformation matrix to stretch the image in the horizontal direction.

We can now check if the image warping worked as expected by displaying the two images (original and warped) in Matlab to see if their scales are correct. This can be done using one of many Matlab commands (try using imshow, image, etc. and identify the difference between each function). To ensure that the scaling was done properly you can use the command size on each image to check their dimensions. We can perform yet another check to ensure that the image warping was done correctly. Start by selecting points in the original image (try using the ginput command). Then transform and map these points onto the warped image and plot both sets of points (selected and transformed) on the corresponding images to check the accuracy of the warping. Matlab also provides a user interface which allows a user to select corresponding point in two images. Call *cpselect*, specifying the names of the image you want to register and the reference image. Try typing this at the command line:

h = cpselect('westconcordaerial.png','westconcordorthophoto.png');

3 Theory Questions [30 pts]

Question 1: Focal length (10 points)

Show that, for an object to be properly focused, the distance $\mathbf{d}$ from the lens to the image plane is equal to $(\mathbf{1}+\mathbf{m})\mathbf{f}$ where $\mathbf{f}$ is the focal length and $\mathbf{m}$ is the magnification. This distance is also called the effective focal length. Show that the distance between an image plane and an object must be $(\mathbf{m}+\mathbf{2}+\mathbf{1/m})\mathbf{f}$ for an object to be in focus.

Question 2: Homography (20 points)

Consider two points $\boldsymbol{p}$ and $\boldsymbol{q}$ on planes Π_1 and Π_2. Let us assume that the homography that maps point $\boldsymbol{p}$ to $\boldsymbol{q}$ is given by $\mathbf{S}$.

$$\boldsymbol{p} \equiv \boldsymbol{S}\boldsymbol{q} \tag{1}$$

Answer the following questions regarding $\mathbf{S}$:

(a) What is the size $rows \times columns$ of the matrix $\mathbf{S}$? (remember to use homogeneous coordinates)

(b) What is the rank of the matrix $\mathbf{S}$? Explain.

(c) What is the minimum number of point correspondences ($\boldsymbol{p}$,$\boldsymbol{q}$) required to estimate $\mathbf{S}$? Why?

We can use homographies to create a panorama image from multiple views of the same scene. This is possible for example when there is no camera translation between the views (e.g., only rotation about the camera center). In this case, corresponding points from two views of the same scene can be related by a homography:

$$\boldsymbol{p_1}^{\boldsymbol{i}} \equiv \boldsymbol{H}\boldsymbol{p_2}^{\boldsymbol{i}} \tag{2}$$

where $\boldsymbol{p_1}^{\boldsymbol{i}}$ and $\boldsymbol{p_1}^{\boldsymbol{i}}$ denote the homogeneous coordinates (e.g., $\boldsymbol{p_1}^{\boldsymbol{i}} \equiv (x, y, 1)^T$) of the 2D projection of the i-th point in images 1 and 2 respectively, and $\mathbf{H}$ is a 3×3 matrix representing the homography

(d) Given N point correspondences and using (2), derive a set of 2N independent linear equations in the form $\mathbf{Ah} = \mathbf{b}$ where $\mathbf{h}$ is a 9×1 vector containing the unknown entries of $\mathbf{H}$. What are the expressions for $\mathbf{A}$ and $\mathbf{b}$? How many correspondences will need to solve for $\mathbf{h}$?

4 Programming [30 pts]

Part1 Line detection using RANSAC (10 points)

Write the following Matlab functions and explain your code in the writeup.

```
function [line,inliers] = ransac(points,iter,thr,mininlier)
```

Inputs: *points* is a $2 \times N$ matrix with corresponding $(x, y)^T$ coordinates. *iter* gives the number of iterations, *thr* the threshold on the distance from a point to the line used to determine if the point is an inlier, and *mininlier* is the number of inliers needed to declare a good fit. Outputs: Set of *inliers* $\in 0, 1^P$, where $1 = inlier$ and $0 = outlier$.

Write a script `ransac_main.m` which does the following: Draw 100 points $\in [-1, 1] \times [-1, 1]$ passing through the line $y = x$ and 100 points uniformly at random $\in [-1, 1] \times [-1, 1]$. Add zero mean Gaussian noise to these points with $\sigma = 0.1$. Apply RANSAC to the noisy points. Plot 6 iteations of RANSAC

with the estimated line and the inliers to the estimated line. Provide your plots in the writeup.

Part 2 Estimating transformations from the image points (10 points)

Write a Matlab function and explain your code in the writeup

```
function H = computeH(p1, p2)
```

Inputs: p1 and p2 should be $2 \times N$ matrices with corresponding $(x, y)^T$ coordinates between two images. Outputs: **H** should be a 3×3 matrix encoding the homography that best matches the linear equation derived above (from p2 to p1). Hint: Remember that **H** will only be determined up to scale, and the solution will involve an SVD. You can use matlab svd function.

```
function H = computeHnorm(p1, p2)
```

This version should normalize the coordinates p1 and p2 (e.g., from -1 to 1). Normalization improves numerical stability of the solution. Note that the resulting **H** should still follow Eq. (2). Hint: Express the normalization as a matrix.

Part 3 Mosaicing (5 points)

Write a Matlab function and explain your code in the writeup

```
function [Iwarp, Imerge] = warpImage(Iin, Iref, H)
```

which takes as input an image *Iin*, a reference image *Iref*, and a 3×3 homography, and returns 2 images as outputs. The first image is *Iwarp*, which is the input image *Iin* warped according to **H** to be in the frame of the reference image *Iref*. The second output image is *Imerge*, a single mosaic image with a larger field of view containing both the input images.

In order to avoid aliasing and sub-sampling effects, consider producing the output by solving for each pixel of the destination image rather than mapping each pixel in the input image to a point in the *destination* image (which will leave holes). Also note that the input and output images will be of different dimensions.

Run `warpImage(Iin, Iref, H)` with (Iin=*taj1r.png*, Iref=*taj2r.png*) and save its output as (*taj1r_warped.png*, *taj_merged.png*)

Part 3 Rectification (5 points)

Write a Matlab function and explain your code in the writeup

```
function [H, result] = rectify(Img, p1, p2)
```

Compute the 3×3 homography matrix to rectify the following image of an HTC phone. The rectification is to make the new image plane parallel to the


wall as best as possible. You can select the four corner points (the intersections of the four boundary straight lines) and target corner locations to compute the homography matrix.

Run `rectify(Img, p1, p2)` with *HTC.png* and save its output as *HTC_rectified.png*

Revision History

Revision	Date	Author(s)	Description
1.0	2019/04/11	TA	HW2 out
1.1	2019/04/16	TA	Fix typos in problem 1.
1.2	2019/04/16	TA	Exchanged 'horizontal' with 'vertical' in WarmUp