

Homework 3
M1522.001000 Computer Vision (2020 Fall)
Due: Tuesday Nov 3, 11:59PM

The goal of this homework is to explore Lens System, Homographies and Image warping. There are 5 Questions in this assignment, 2 theory and 3 programming questions.

1 Submitting Your Assignment

Your submission for this assignment must include answers to theory questions, the code for your **Python** 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 all your files into a directory called "(studentid)-(yourname)-HW3" and pack it into a **zip** named "(studentid)-(yourname)-HW3.zip". For example, 202012345-gildonghong-HW3.zip. Upload your **zip** file to **ETL** until due date. Refer to the **ETL** page for the policies regarding collaboration, due dates, extensions, and late days.

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

- `theory.txt`(`theory.doc`) or `theory.pdf` : answers to the theory questions
- Folder result
 - `porto1_warped.png` : png image showing the result of warping applied to `porto1.png`
 - `porto_merged.png` : png image showing the result of combining `porto1.png` and `porto2.png`
 - `iphone_rectified.png` : png image showing the result of rectifying `iphone.png`
 - `writeup.txt`(`writeup.doc`) or `writeup.pdf` : the writeup describing your experiments
- Folder code
 - `main.py`

Refer to Section 3 and Section 4 for the details of the above files. Your zip file should be sent in before the due. Later than that, only one late day is allowed. Good Luck!

2 Warm Up

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

Let us start with the toys image (*toys.jpg*) and perform some simple image warping. 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 (*i.e.*, $T_h P_{i,j}$, where $P_{i,j} = [i, j, 1]^T$ is a point in the image expressed in homogeneous coordinates).

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

The transformation matrix T_h will stretch the image in the horizontal direction by a factor of 2. Properly generate the warped image without "holes" in the new image by mapping every point/pixel in the new image from a corresponding value in the old image and not the reverse. (This means that your code should iterate for every pixel in the warped image which has twice as much pixels as the original image in the horizontal direction and the same number of pixels in the vertical 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 vertical direction.

Note that while we use the (horizontal, vertical) format to each point/pixel (*i.e.*, $P_{i,j} = [i, j, 1]^T$), image matrix is referenced using the (row, column) format. You can easily check it in the `main.py` code. (Compare variable `igs_in` to variable `p_in`.) Therefore, pay careful attention to which format you are using while mapping the points to the warped image. If done properly, both sets of points should appear in the same location in their corresponding images.

3 Theory Questions

Question1: Lens System (30 points)

(a) Given Figure. 1, derive a Gaussian lens formula.

$$\frac{1}{f_1} = \frac{1}{L_1} + \frac{1}{L_3} \quad (1)$$

(b) Given Figure. 1, what is the blur circle diameter b ? Use A, L_1 and L_2 .

(c) Given Figure. 1, what is the circle of confusion diameter c ? Use A, L_1, L_2 and f_1 . (Assume that $L_1 > f_1$)

(d) Given Figure. 2, what is the distance L ? Use f_1, f_2, m_1 , and m_2 where m_1 and m_2 are the magnifications of each lens. (Assume that $m_1 > 0$ and $m_2 > 0$.)

Question2: Homography (15 points)

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

$$\mathbf{p} \equiv \mathbf{S}\mathbf{q} \quad (2)$$

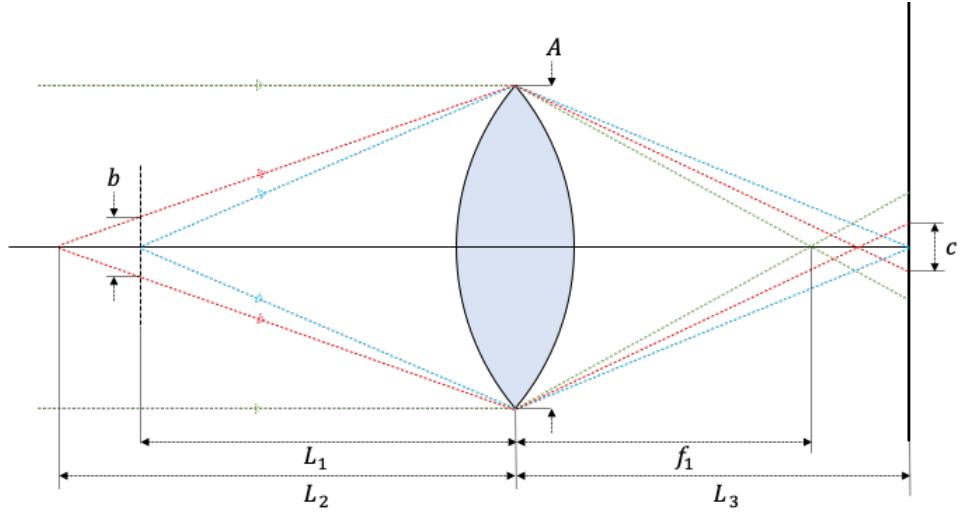


Figure 1: Diagram of circle of confusion diameter c for an out-of-focus subject at distance L_2 when the camera is focused at L_1 ($L_2 > L_1$). f_1 is the focal length of the lens, A is the aperture diameter of the lens, b is a blur circle diameter, and L_3 ($L_3 > f_1$) is a distance between the lens and the camera (i.e., image).

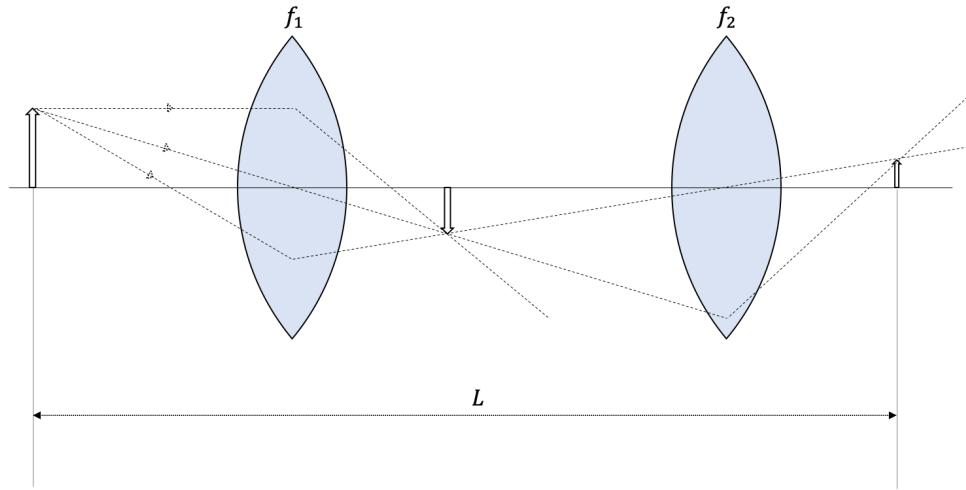


Figure 2: Diagram of two lenses system. f_1 and f_2 are the focal lengths of each lens and L is the distance between an image plane and an object for which to be properly focused.

Answer the following questions regarding \mathbf{S} :

- What is the size $rows \times columns$ of the matrix \mathbf{S} ? (Remember to use homogeneous coordinates.)
- What is the rank of the matrix \mathbf{S} ? Explain.
- What is the minimum number of point correspondences (\mathbf{p}, \mathbf{q}) required to estimate \mathbf{S} ? Why?

We can use homographies to create a panoramic image from multiple views of the same scene. This is possible 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:

$$\mathbf{p}_1^i \equiv \mathbf{H}\mathbf{p}_2^i \quad (3)$$

where \mathbf{p}_1^i and \mathbf{p}_2^i denote the homogeneous coordinates (e.g., $\mathbf{p}_1^i \equiv (x, y, 1)^T$) 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 be needed to solve for \mathbf{h} ?

4 Programming

First of all, make sure you have Python 3.6+ installed. You can check your version with command `python3 -V`. To run the program, you will also need Numpy and Pillow packages. To install them with the package manager for Python, run: `pip3 install numpy pillow`. Usage of third-party image processing library including **OpenCV** for this part is **prohibited**.

Part 1 Estimating transformations from the image points (15 points)

Write a Python function and explain your code in the writeup.

```
H = compute_h(p1, p2)
```

Inputs: \mathbf{p}_1 and \mathbf{p}_2 should be $N \times 2$ matrices of corresponded $(x, y)^T$ coordinates between two images. Outputs: \mathbf{H} should be a 3×3 matrix encoding the homography that best matches the linear equation derived above (from \mathbf{p}_2 to \mathbf{p}_1). Hint: Remember that \mathbf{H} will only be determined up to scale, and the solution will involve an SVD. You are allowed to use existing SVD function including `numpy.linalg.svd`, `scipy.linalg.svd`, etc.

```
H = compute_h_norm(p1, p2)
```

This version should normalize the coordinates \mathbf{p}_1 and \mathbf{p}_2 (e.g., from 0 to 1). Normalization improves numerical stability of the solution. Note that the resulting \mathbf{H} should still follow Eq. (2). Hint: Express the normalization as a matrix.

Part 2 Mosaicing (25 points)

Write a Python function and explain your code in the writeup.

```
p_in, p_ref = set_cor_mosaic()
```

which returns \mathbf{p}_{in} and \mathbf{p}_{ref} , $N \times 2$ matrices of corresponded $(x, y)^T$ coordinates between two images. Set a criterion and select the correspondences from given images. From there, you will be able to calculate \mathbf{H} for `warp_image` function. Explain the criterion and the number of correspondences used.

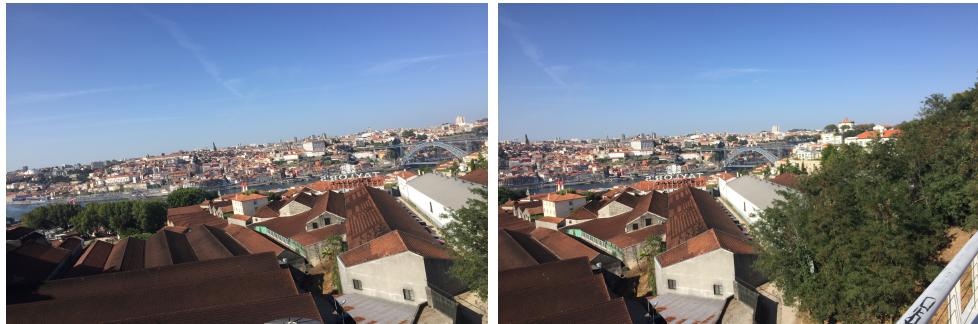
```
igs_warp, igs_merge = warp_image(igs_in, igs_ref, H)
```

which takes an input as image igs_in , a reference image igs_ref , and a 3×3 homography, and returns 2 images (igs_warp and igs_merge) as outputs.

The first output image is igs_warp , which is the image igs_in warped according to \mathbf{H} to be in the frame of the reference image igs_ref . \mathbf{H} must be derived by you from the given image.

The second output image is igs_merge , 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 different dimensions.



Part 3 Rectification (15 points)

The rectification is to make the new image plane parallel to the wall as best as possible.

Write a Python function and explain your code in the writeup.

```
c_in, c_ref = set_cor_rec()
```

which returns c_in and c_ref , $N \times 2$ matrices of corresponded $(x, y)^T$ coordinates between two images. You can select the four corner points (the intersections of the four boundary straight lines; i.e. c_in) and target corner locations(i.e. c_ref) to compute the homography matrix. From there, you will be able to calculate \mathbf{H} for `rectify` function.

```
igs_rec = rectify(igs, p1, p2)
```

Compute the 3×3 homography matrix to rectify the following image of an iphone.

Finally, note that we will use a code similarity checker to detect plagiarism. You are expected to work on the assignment individually. I firmly believe that every students can do his or her own work. For your sake, please do not copy and paste others code! SWEET AFTER BITTER!

