Homework 4
M1522.001000 Computer Vision (2021 Fall)

Due: Tuesday Nov 23, 11:59PM

The goal of this homework is to explore Stereo Camera Models, Optical Flow, and Lucas-Kanade Method.

There are 6 theory questions and 2 **Python** programming questions in this assignment, and 100 points in total.

Put your **code and writeup** into a directory called "`(studentid)-(yourname)-HW4`" and pack it into a `zip` named "`(studentid)-(yourname)-HW4.zip`".

For example, `202112345-gildonghong-HW4.zip`.

Your writeup should be **typed** with **English**. Please do **not** attach your code to writeup. 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 homework should be formatted as following:

```
(studentid)-(yourname)-HW4
├─ writeup.pdf
├─ run_motion.py
├─ data
│  ├─ 0.jpg
│  ├─ ...
│  └─ 49.jpg
└─ motion.mp4
```

Points will be deducted when your submission does not follow the above structure. Your `zip` file should be sent in before the due. Later than that, only one late day is allowed. Finally, note that we will use a code similarity checker to detect plagiarism. You are expected to work on the assignment individually. We firmly believe that every student can do his or her own work. For your sake, please do not copy and paste others code. Good Luck!

1 Stereo

Question 1

[10 points] Two cameras are said to form a *rectified pair* if their camera coordinate systems differ only by a translation of their origins (*i.e.*, the camera centers) along a direction that is parallel to either the x or y axis of the coordinate systems.

1. Derive an expression for the essential matrix $\mathbf{E}$ of a rectified pair.
2. Prove that the epipolar lines of a rectified pair are parallel to the axis of translation.

Question 2

[15 points] Consider three images I_1, I_2 and I_3 that have been captured by a system of three cameras, and suppose the fundamental matrices $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$ are known. (Notation: the matrix $\mathbf{F}_{ij}$ satisfies the equation $\mathbf{x}_j^\top \mathbf{F}_{ij} \mathbf{x}_i = 0$ for any correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}_j$ between images I_i and I_j.) In general, given a point $\mathbf{x}_1$ in I_1 and a corresponding point $\mathbf{x}_2$ in I_2, the corresponding point in $\mathbf{x}_3$ in I_3 is uniquely determined by the fundamental matrices $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$.

1. Write an expression for $\mathbf{x}_3$ in terms of $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$.
2. Describe a degenerate configuration of three cameras for which the point $\mathbf{x}_3$ cannot be uniquely determined by this expression.

Hint: Consider the epipolar geometry of the situation. Draw a picture!

Question 3

[15 points] Suppose two cameras fixate on a point P (Figure 1) in space such that their optical axes intersect at that point. Show that if the image coordinates are normalized so that the coordinate system origin $(0, 0)$ coincides with the principal point, the $\mathbf{F}_{33}$ element of the fundamental matrix $\mathbf{F}$ is zero.

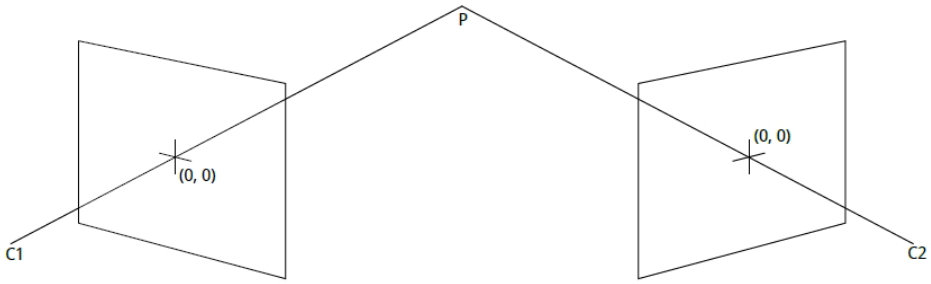

Figure 1: $C1$ and $C2$ are the optical centers. The principal axes intersect at point P.

2 Optical Flow

Question 4

[5 points] As we discussed in class, the "aperture problem" shows our inability to perceive the motion correctly. Discuss what does the aperture problem imply in terms of spatial information. (1) Why do we perceive the motion ambiguously? (2) Why is the *aperture* necessary in the Aperture problem?

Hint: Consider why the aperture is needed in the first place.

Question 5

[5 points] Consider a situation where the motion of the camera is in the forward direction. Does the optical flow equation still hold? Find a simple way to figure out which scene point the object is moving towards.

Question 6

[10 points] As we discussed in class, it is hard to estimate the optical flow with Lucas-kanade flow method in some situations. Suppose you are trying to estimate the flow inside the aperture in Figure 2, discuss whether this case will be difficult or not.

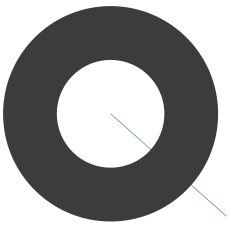

Figure 2: Q

Justify your answer in terms of the following matrix's characteristic:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

3 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, SciPy , Scikit-Image and OpenCV packages. To install them with the package manager for Python, run `pip3 install numpy scikit-image scipy opencv-python`. Then, you should complete the code. Specifically, you have to fill in two blocks starting with `### START CODE HERE ###` and ending with `### END CODE HERE ###`. For this, usage of third-party image processing library other than `numpy` & `RectBivariateSpline` is **prohibited**. Never use **OpenCV**.

In this section, you will implement a tracker for estimating dominant affine motion in a sequence of images, and subsequently identify pixels corresponding to moving objects in the scene. You will be using the images in the directory `data`, which consists aerial views of moving vehicles from a non-stationary camera.

3.1 Dominant Motion Estimation

To estimate dominant motion, the entire image $I(t)$ will serve as the template to be tracked in image $I(t+1)$, i.e. $I(t+1)$ is assumed to be approximately an affine warped version of $I(t)$. This approach is reasonable under the assumption that a majority of the pixels correspond to stationary objects in the scene whose depth variation is small relative to their distance from the camera. Using the affine flow equations, you can recover the 6-vector $p = [p_1\ p_2\ p_3\ p_4\ p_5\ p_6]^T$ of affine flow parameters. They relate to the equivalent affine transformation matrix as:

$$M = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

relating the homogenous image coordinates of $I(t)$ to $I(t+1)$ according to $\mathbf{x}_{t+1} = M\mathbf{x}_t$ (where $\mathbf{x} = [x\ y\ 1]^T$). For the next pair of consecutive images in the sequence, image $I(t+1)$ will serve as the template to be tracked in image $I(t+2)$, and so on through the rest of the sequence. Note that M will differ between successive image pairs.

Each update of the affine parameter vector, Δp is computed via a least squares method using the pseudo-inverse as described in class. `Note`: Image $I(t)$ will almost always not be contained fully in the warped version of $I(t+1)$. Hence the matrix of image derivatives and the temporal derivatives must be computed only on the pixels lying in the region common to $I(t)$ and the warped version of $I(t+1)$ to be meaningful.

For better results, Sobel opterators may be used for computing image gradients. So, we have provided a code for them.

Part 1: Lucas–Kanade Method [20 points]

Complete the following function and explain your code in the writeup:

```
dp = lukas_kanade_affine(img1, img2, p, Gx, Gy)
```

where `dp` is Δp, and `img1`, `img2`, `p`, `Gx` and `Gy` are $I(t)$, $I(t+1)$, p, I_x and I_y, respectively. Do not attach your code to the writeup. The explanation in the writeup should include high level explanations of your algorithm design and an itemized list of your assumptions and parameter choices.

A naive implementation may not work well. Then, refer to the following tips:

- Scaling image coordinates can be crucial for this type of problem. You should scale coordinates of image gradients ∇I to be between 0 and 1. But be sure to undo that scaling whenever you need to use coordinates to actually reference pixels in an image.
- This function will probably take too long. Then, try to subsample pixels when computing the Hessian matrix H and Δp.

3.2 Moving Object Detection

Once you are able to compute the transformation matrix M relating an image pair $I(t)$ and $I(t+1)$, a naive way of determining pixels lying on moving objects is as follows.

Figure 3: Sample results of affine motion subtraction

Warp the image $I(t)$ using M so that it is registered to $I(t+1)$, and subtract it from $I(t+1)$. The locations where the absolute difference exceeds a threshold can then be regarded as moving objects.

For better results, hysteresis thresholding may be used. So we have provided a code for hysteresis thresholding.

Part 2: Affine Motion Subtraction [20 points]

Complete the following function and explain your code in the writeup:

```
mask = subtract_dominant_motion(img1, img2)
```

where `img1` and `img2` form the input image pair, and `mask` is a binary image of the same size that dictates which pixels are considered to be corresponding to moving objects. You should invoke `lukas_kanade_affine` in this function to derive Δp, and produce the aforementioned binary mask accordingly. Do not attach your code to the writeup. The explanation in the writeup should include high level explanations of your algorithm design and an itemized list of your assumptions and parameter choices. Any experimental try will be gracefully accepted. Also, discuss any findings you find interesting.

Your implementation has to yield similar or better results in Figure 3. In addition, its total running time should be less than 30 minutes. TA's implementation runs within 10 minutes in Google Colab environment.

Finally, note that we will use a code similarity checker to detect plagiarism. You are expected to work on the assignment individually. We 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!