Homework Set 3: Camera, Stereo and Motion Estimation

Instructions

- This homework contains two parts, i.e., a set of questions (15 pts, in Section 1) and a programming project (85 pts, in Section 2).
- Upload a zipped file named as `id_yourname.zip` via the Beihang SPOC platform, including
  - (1) an electronic version (saved as `id_name.pdf`, can be printed or handwritten) of your report, it includes the answers of the questions in Section 1.
  - (2) a folder (named as `id_name`) includes the codes and results. You should use Jupyter Notebook in this homework.

    For more details about installing Jupyter Notebook, please check the Appendix.
- The deadline is 23:59 pm, June 26th, 2024.

1 Questions (15 pts)

1.1 (5 pts) Explain how a stereo camera setup is used to estimate depth. In your explanation, include the basic principles and concepts such as disparity and rectification.

1.2 (5 pts) Describe the main steps of the Lucas-Kanade method for optical flow estimation.

1.3 (5 pts) Explain the role of pyramid representation in handling large motion estimation in Lucas-Kanade method.

2 Programming Projects (85 pts + 15 extra points)

2.1 Task-1: Stereo Matching (30 pts + 15 extra pts)

Now that we have learned the algorithms behind stereo vision we can begin to tackle real-world examples. This exercise gives you a brief introduction to using Python to implement what you've recently learned in order to compute the depth map according to a pair of stereo images. You are required to implement the key functions in `student_stereo.ipynb`. The Jupyter Notebook file contains the starting code, and you need to accomplish the following items:

(i) **(20 pts)** Create a disparity map. Use the Block Matching alogorithm to try and build a disparity map. Note that you need to detect and mask out occluded and dis-occluded area in the disparity map, in which the estimated disparity values are not reliable.

(ii) **(10 pts)** Compute the depth map. Use camera parameters and the disparity map to build a depth map.

(iii) **(15 pts)** (Optional) Using the Energy Minimization method with Dynamic Programming to build a disparity map.

Note: Applying off-the-shelf implementations of stereo algorithms, e.g., `cv2.StereoBM_create` and `cv2.StereoSGBM_create` functions is not encouraged. You may just earn half of the points if you do so.

2.2 Task-2: Motion Estimation (55 pts)

We have done some cool stuff with static images in past assignments. Now, let's turn our attention to videos! For this assignment, the videos are provided as time series of images. We also provide utility functions to load the image frames and visualize them as a short video clip. You are required to implement the key functions in `motion.py` and call these functions to run the corresponding notebook cells in `student_motion.ipynb`.

Note: You may need to install video codec like FFmpeg. If you have `conda`, you can generally install it with `conda install -c conda-forge ffmpeg`. For Linux/Mac, you will also be able to install ffmpeg using `apt-get` or `brew`. For Windows, you can find the installation instructions here.

(a) We are going to implement basic Lucas-Kanade method for sparse feature tracking. In order to do so, we first need to find keypoints to track. Harris corner detector is commonly used to initialize the keypoints to track with Lucas-Kanade method. For this assignment, we are going to use `skimage` implementation of Harris corner detector. You can see `student_motion.ipynb` for details.

   (i) **(15 pts)** Implement `lucas_kanade()` in `motion.py`.

   (ii) **(10 pts)** Now we can use Lucas-Kanade method to track keypoints across multiple frames. To filter 'bad' tracks, implement the `compute_error()` in `motion.py`.

   (iii) **(5 pts)** Describe the difference after using `compute_error()`.

(b) One limitation of the naïve Lucas-Kanade method is that it cannot track large motions between frames. You might have noticed that the resulting flow vectors (blue arrows) in the previous section are too small that the tracked keypoints are slightly off from where they should be. In order to address this problem, we can iteratively refine the estimated optical flow vectors. You can see `student_motion.ipynb` for details.

   (i) **(15 pts)** Implement `iterative_lucas_kanade()` in `motion.py`.

   (ii) **(10 pts)** To track both large and small motions, we can compute flow vectors from coarse to fine scale. Implement `pyramid_lucas_kanade()` in `motion.py`.

Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!

Appendix: Installing and Setting Up Jupyter Notebook

- We recommend using `conda` to manage the python environment. You can download `miniconda` installer from here.

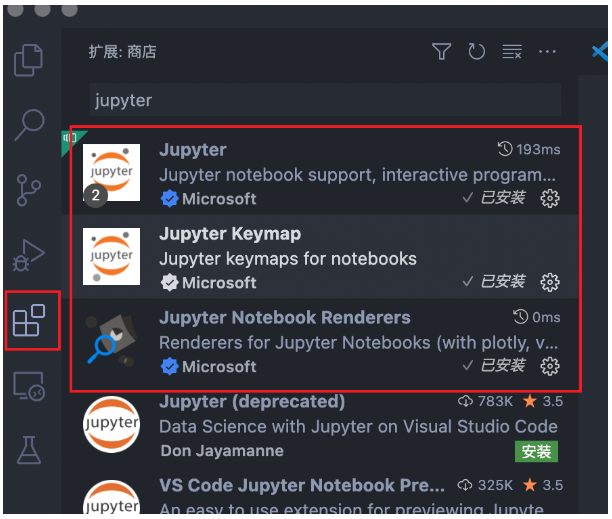
  * If you want to use customized mirror in `conda`, you can run the following commands:

```
conda config --set show_channel_urls yes
conda config --add channels \
    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
conda config --add channels \
    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels \
    https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
```

- Install packages via `conda` (or `pip`).

```
conda install -c conda-forge jupyterlab ffmpeg
conda install numpy matplotlib scikit-image
```

- We recommend using `vscode` to run the notebook. You can install `vscode` from here. Then install `Python`, `Jupyter`, `Jupyter Keymap` and `Jupyter Notebook Renderers` extension in `vscode`, see following figure.



扩展: 商店

jupyter

Jupyter 193ms
Jupyter notebook support, interactive program...
Microsoft ✓ 已安装

Jupyter Keymap
Jupyter keymaps for notebooks
Microsoft ✓ 已安装

Jupyter Notebook Renderers 0ms
Renderers for Jupyter Notebooks (with plotly, v...
Microsoft ✓ 已安装

Jupyter (deprecated) 783K ★ 3.5
Data Science with Jupyter on Visual Studio Code
Don Jayamanne 安装

VS Code Jupyter Notebook Pre... 325K ★ 3.5
An easy to use extension for previewing Jupyte

- Open the `.ipynb` file, and select the corresponding kernel.