EEL 4930/5934: Autonomous Robots

HH3: Hands-on Homework #3 (Spring 2023)

Tasks Overview:

A. Augmented visuals by homography estimation
B. Camera calibration
C. SfM (Structure from Motion) pipeline

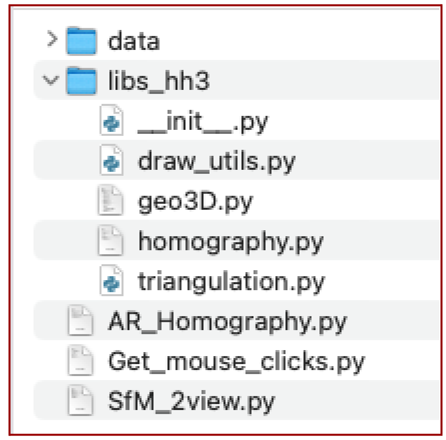
Grading Breakdown (Both Sections)

- Part A: 30% (Bonus point option: +5%)
- Part B: 20%
- Part C: 50% (10% + 5% + 15% + 5% + 5% + 10%)

Instructions

- Download the **HH3_Blank** folder from canvas
- Complete the functions asked for (see below)
- Generate the outputs and write in report (PDF)

References: Lecture 6 contents



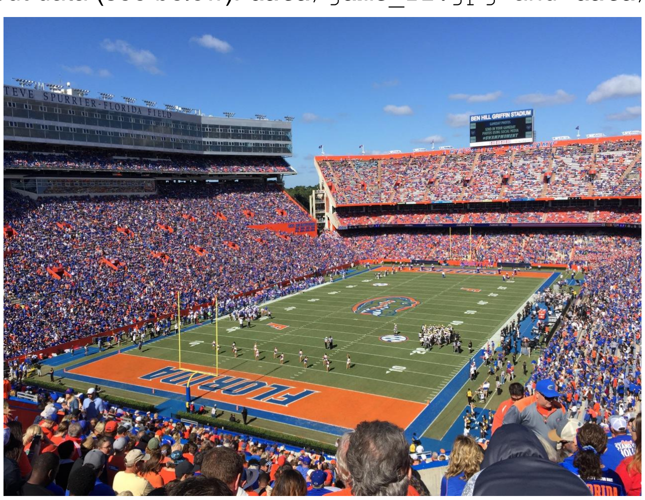
Submission: [Through Canvas only; Due: April 4, 2023 by 11.59pm]

- A single zip file with a folder (code) and PDF (report)
  - Your completed code: do not add any more python files, just complete the functions
  - A PDF report: the generated outputs only (things that are asked in **blue color**; see below)
- Assignments more than 20 MB file size will get negative penalty (-10% to -50%)

Part A: Augmented Visuals by Homography Estimation

Refer to Lecture-6: slide 16-24 and the following files

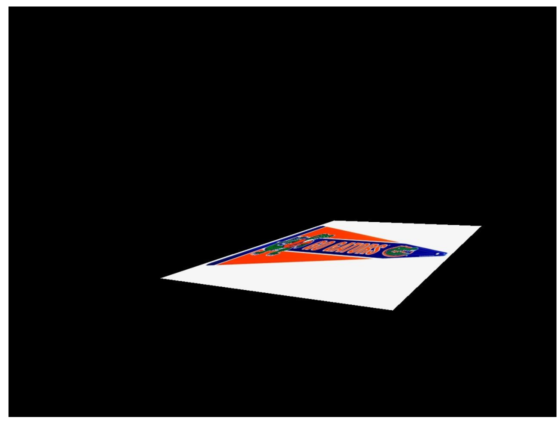
- Driver script: `AR_Homography.py`
- Library: `libs_hh3/homography.py`
- Input data (see below): `data/game_fl.jpg` and `data/logo_Gators.png`



Your target is to augment the logo into the destination image by following the homography perspective transformation. To achieve this, please follow these steps

- Get the four points on the destination image using the `Get_mouse_clicks.py`
- The driver script `AR_Homography.py` is already completed for you
- You need to complete the following library function (in `libs_hh3/homography.py`)

  ```
  def computeHomography(Us, Vs)
  ```
- You can check your outputs compared to using cv2.findHomography function
- **There will be two outputs as shown below; you should generate AR_out and AR_warped figures and show them in your report.**

AR_out	AR_warped
	

Bonus point (+5): There is a basic warping function implemented for you (in `libs_hh3/Homography.py`)

```
def warp_and_augment(im_logo, im_dst, H)
```

You will see that it generates somewhat noisy output in some pixels. There are a few better ways to implement this function for much accurate warping. You will get bonus points (max: 5) if you can implement a better one! Please write the main ideas/intuitions of your algorithm briefly and show the comparative results in your report.

Part B: Camera Calibration

In this part, you will calibrate a camera: cell-phone or any other camera that you have access to. Refer to Lecture-6: slide 25-27 and find the **intrinsic camera parameters** as follows:

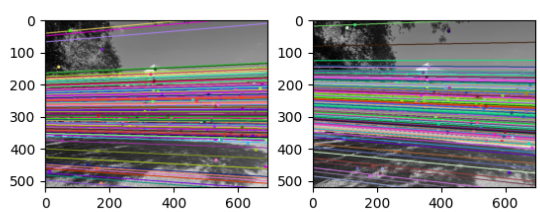
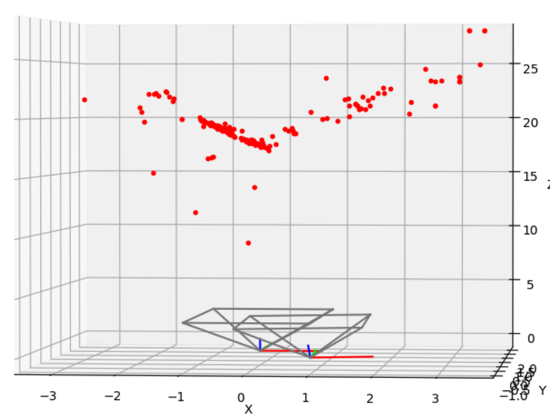
- Print a checkerboard and place it on a wall (a sample `data/pattern.pdf` is provided for you)
- Use any of the following libraries to calibrate your camera:
  - ROS, OpenCV, CalTech Matlab code
  - Or any other library of your choice
- Provide the intrinsic matrix **K** and your camera model in your report.
- Make sure your calibration is accurate! We will not have access to your camera for a comprehensive evaluation. However, if the **K** is inaccurate, your Part C results will be totally wrong.

Part C: SfM (Structure from Motion) pipeline

Refer to the Lecture-6 contents and the following files

- Driver script: `SfM_2view.py`
- Library: `libs_hh3/geo3D.py`
- Utility functions:
    - Some drawing functions are provided for you in `libs_hh3/draw_utils.py`
    - A few 3D triangulation algorithms are provided for you in `libs_hh3/triangulation.p`
- Input data (for initial testing):
    - Images: `data/uf_left.png` and `data/right.png`
    - Intrinsic matrix **K**: `data/K_iphone_reduced.txt`
- Data collection:
    - Use the same camera (you used in part B) to collect a 2-view image pair as the given input. Please try to choose a scene with a clear structure and identifiable object shapes.
    - Also use the intrinsic matrix **K** from your part B.

With the given inputs, the existing code will generate the following outputs

Input: left image	Input: right image
	
Output: Epipolar lines 	Output Camera pose and 3D reconstruction 

As you will see in the driver script: `SfM_2view.py`, the following steps are followed for the SfM

1. `load_image_pair()`	6. `_find_camera_matrices_rt()`
2. `_extract_keypoints_sift()`	7. `_find_projection_matrices()`
3. `_estimate_fundamental_matrix()`	8. `_triangulate_3d_points()`
4. `draw_epipolar_lines()`	9. `plot_point_cloud()`
5. `_estimate_essential_matrix()`	

You will need to implement parts of the colored functions #3, #5, #6, and #7. (see `libs_hh3/geo3D.py`)

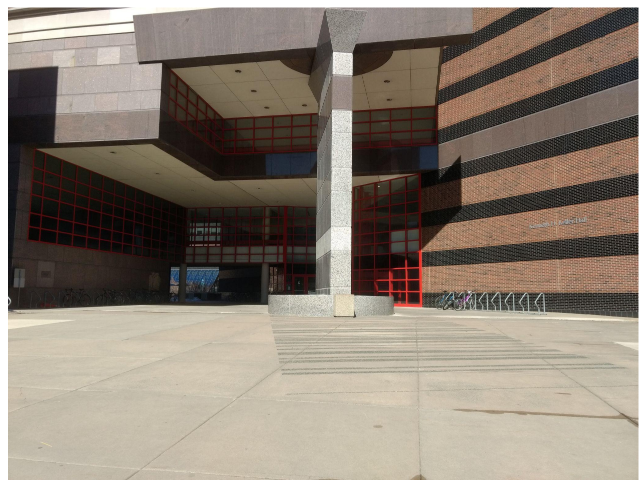
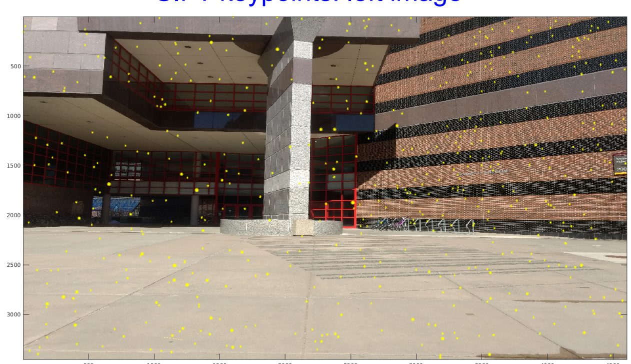
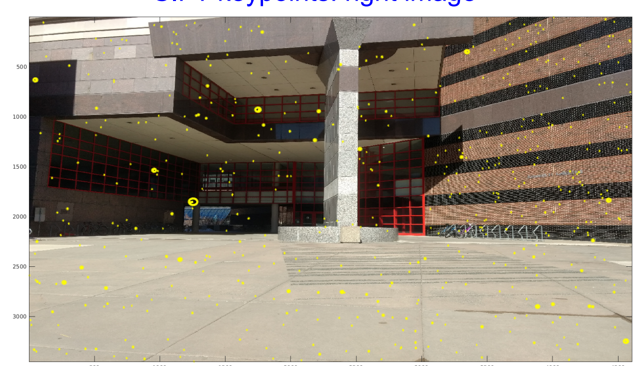
The ToDos are the following

- Go through the implementations provided for you in `libs_hh3/geo3D.py`
- Implement the `_estimate_fundamental_matrix()` function [10 points]
- Implement the `_estimate_essential_matrix()` function [5 points]
- Complete the `_find_camera_matrices_rt()` function [15 points]
- Implement the `_find_projection_matrices()` function [5 points]

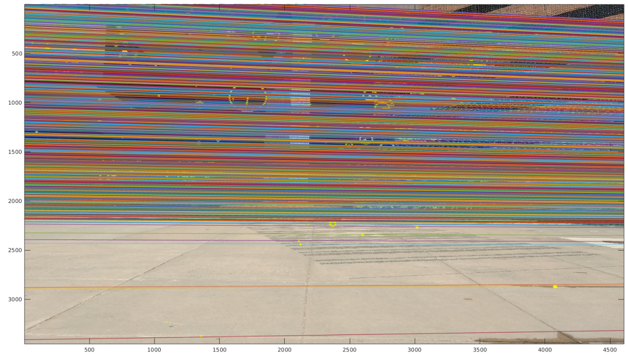
In addition, you will need to write a couple of functions to visualize

- The SIFT keypoints in each input image [5 points]
- SIFT keypoints matched between two images, before and after the 'ratio test' [**Bonus:** +5 points]
- The 4 camera poses estimated by the `_find_camera_matrices_rt` function [10 points]

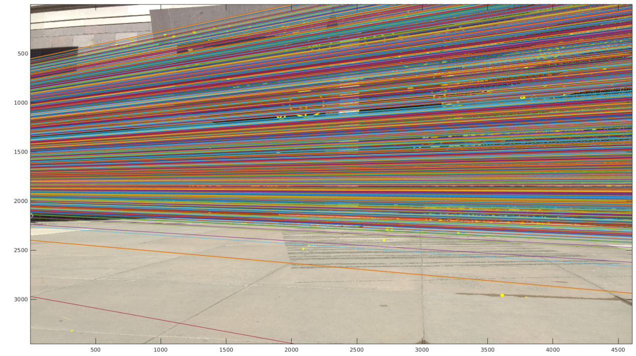
For the report, show the generated outputs for the aforementioned cases. A sample example is given below:

Left image	Right Image
	
SIFT keypoints: left image	SIFT keypoints: right image
	

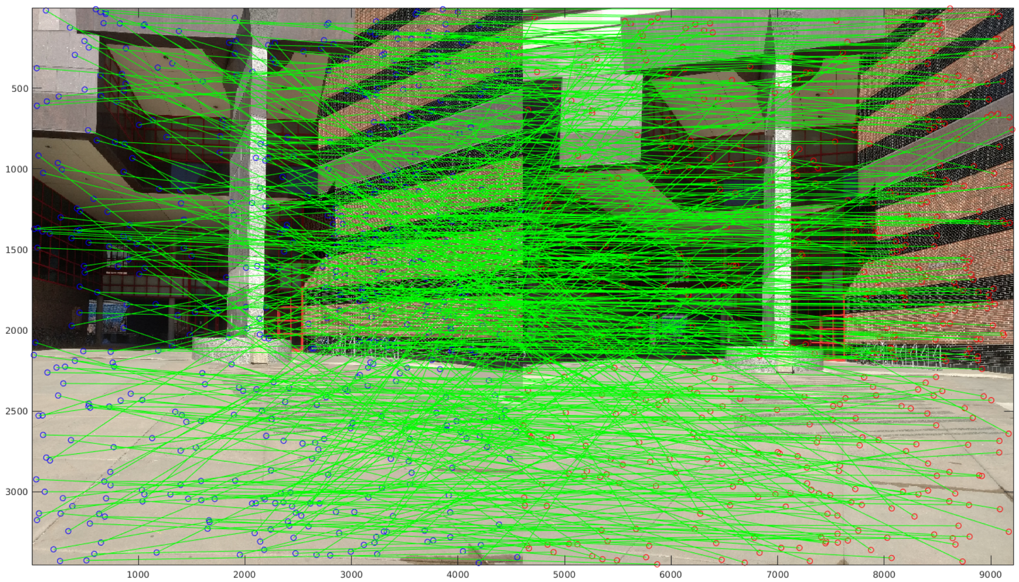
Epipolar lines: left image



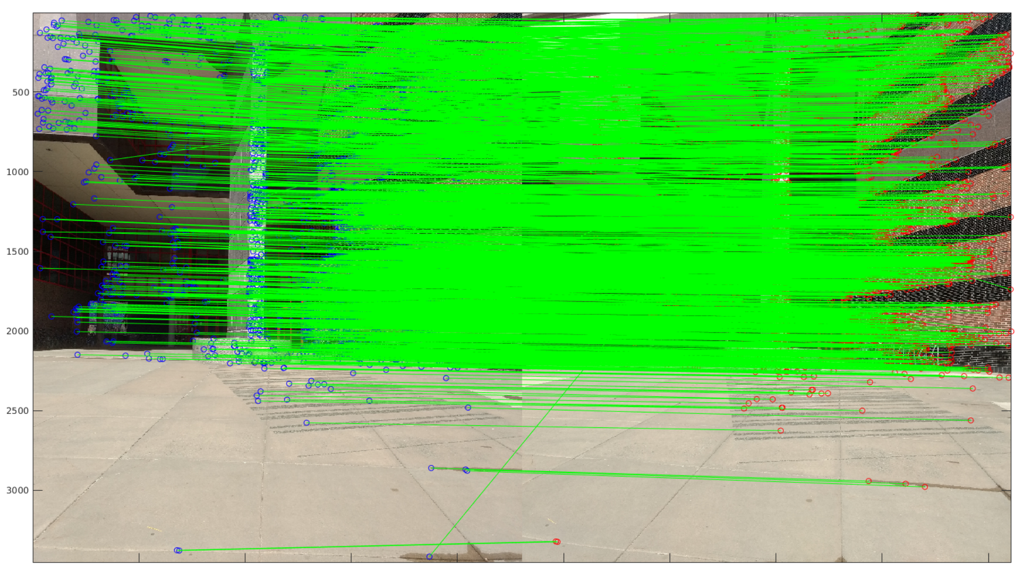
Epipolar lines: right image



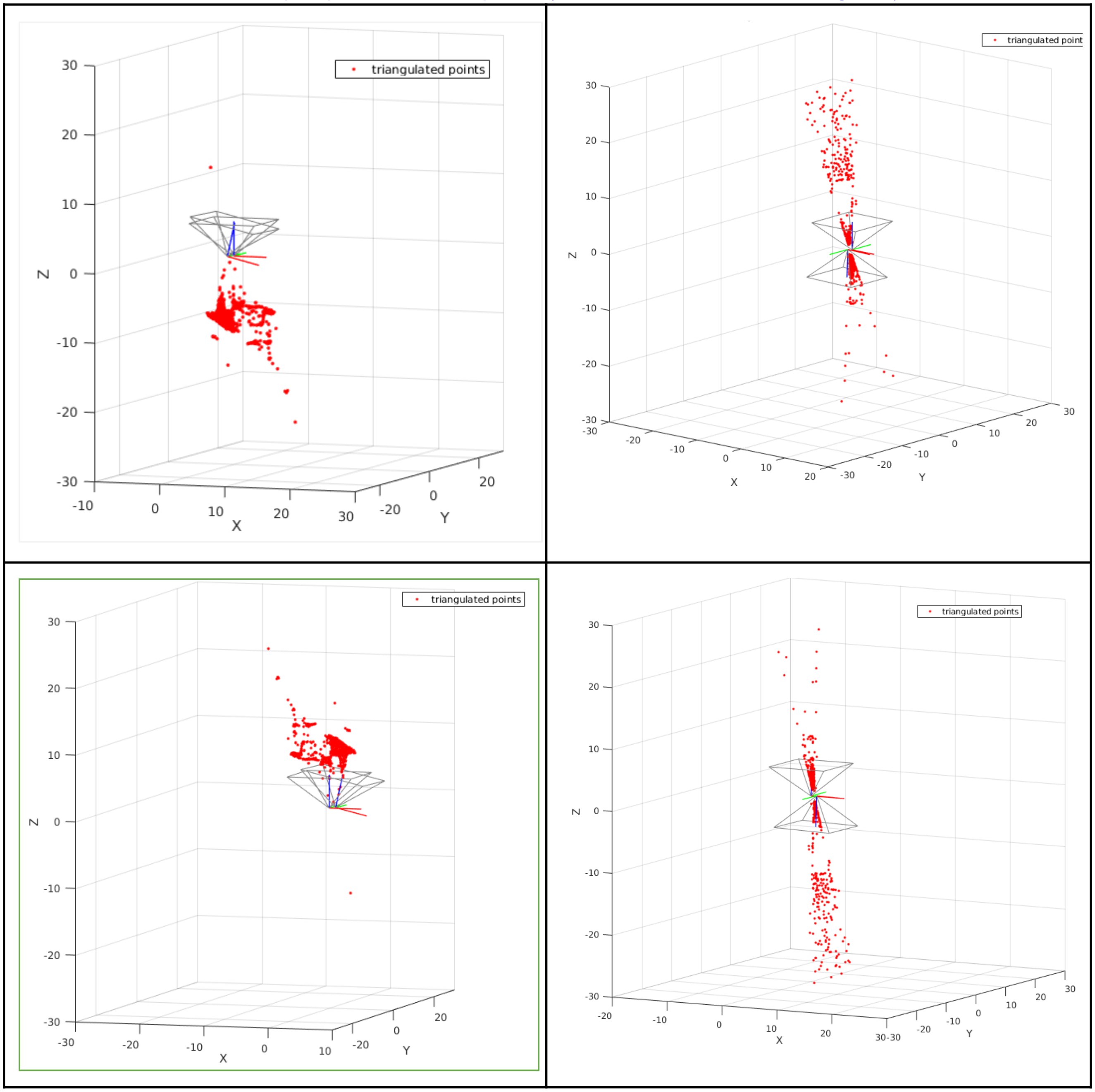
SIFT feature matching between left and right image before ratio test (bonus part)



SIFT feature matching between left and right image after ratio test (bonus part)



The four prospective camera poses (the correct one is outlined in green)



Finally, you will **show the final 3D reconstruction** of your scene. Please adjust the scale and orientation of the plot so that the camera poses and the structure can be seen clearly. Also report the following calculations:

- Left projection matrix: **P1**
- Right projection matrix: **P2**
- Fundamental matrix: **F**
- Essential matrix: **E**

Remember, the assignment is due: April 4, 2023 by 11.59pm
Similar to the previous homework, your zipped submission folder (HH3_GatorID.zip) should contain the completed code folder and a report PDF.