CS4277/CS5477 ASSIGNMENT 1: LINE MATCHING USING CROSS RATIO

1. OVERVIEW

Welcome to the course! In this first assignment, you will implement the Line-Sweep [1] algorithm to find point correspondences between pairs of images with wide baselines.

References: Lecture 1; Reference paper [1].

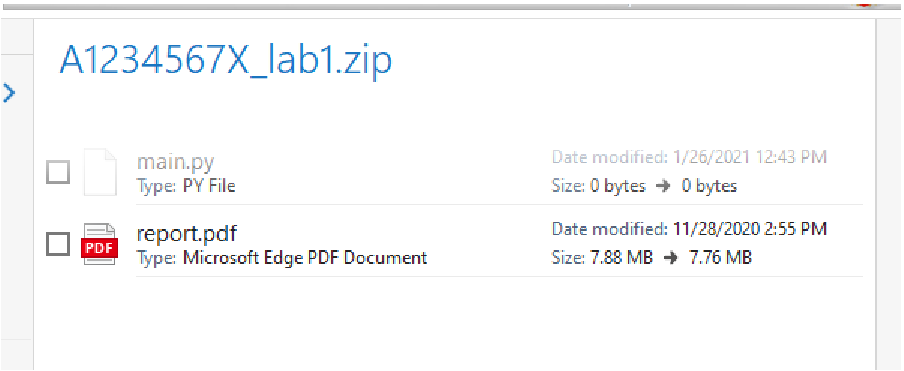
Honour Code. This coding assignment constitutes **10%** of your final grade in CS4277/CS5477. Note that plagiarism will not be condoned! You may discuss with your classmates and check the internet for references, but you MUST NOT submit code/report that is copied directly from other sources!

2. SUBMISSION INSTRUCTIONS

Items to be submitted:

- **Source code (main.py).** This is where you fill in all your code.
- **Report (report.pdf).** This should describe your implementation and be no more than one page.

Please clearly indicate your name and student number (the one that looks like A1234567X) in the report as well as the top of your source code. Zip the two files together and name it in the following format: **A1234567X_lab1.zip** (replace with your student number). Opening the zip file should show:



A1234567X_lab1.zip

□ main.py
Type: PY File
Date modified: 1/26/2021 12:43 PM
Size: 0 bytes → 0 bytes

□ report.pdf
Type: Microsoft Edge PDF Document
Date modified: 11/28/2020 2:55 PM
Size: 7.88 MB → 7.76 MB

Submit your assignment by **2 February 2021, 2359HRS** to LumiNUS. 25% of the total score will be deducted for each day of late submission. The folder for submission will be Assignments → Assignment 1 → Assignment 1 Submissions.

3. GETTING STARTED

This assignment as well as the subsequent ones require Python 3.7, or later. You need certain python packages, which can be installed using the following command:

```
pip install -r requirements.txt
```

If you have any issues with the installation, please post them in the forum, so that other students or the instructors can help accordingly.

4. Problem Statement

In 3D Computer Vision, finding correspondences between images e.g., Figure 1, is useful for downstream tasks such as finding extrinsic camera parameters via robust homography.

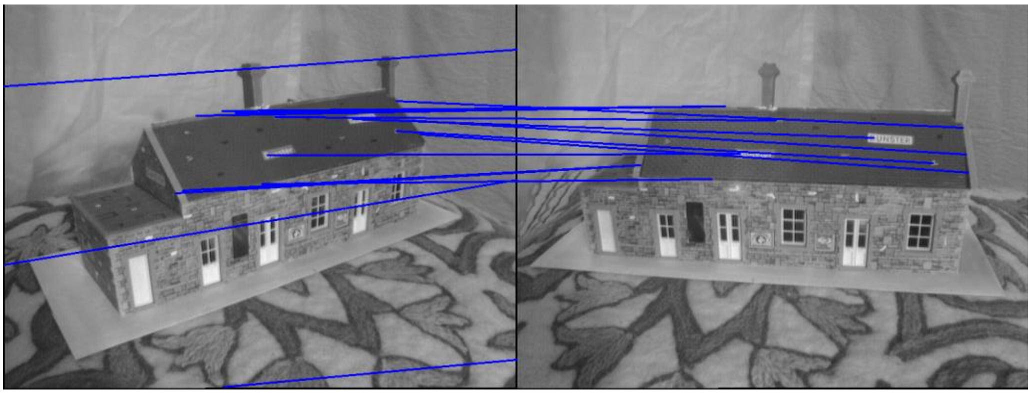


Figure 1: Point correspondences between images

Typically, feature matching is done to identify keypoint matches between pairs of images. Traditionally, we use SIFT features as seen in the figure below, and select the k keypoint matches with the largest affinity (or smallest distance), e.g., Figure 2:

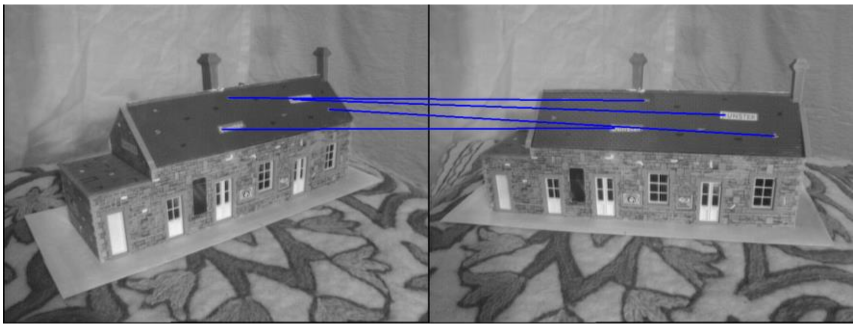


Figure 2: $k = 4$ keypoint matches.

Unfortunately, using large values of k e.g., Figure 3, creates additional erroneous matches. Hence, to find good keypoint matches, the size of k and the total possible number of keypoint matches found via feature matching is usually limited e.g., $k = 4$. To find additional point correspondences, we will implement Line-Sweep [1].

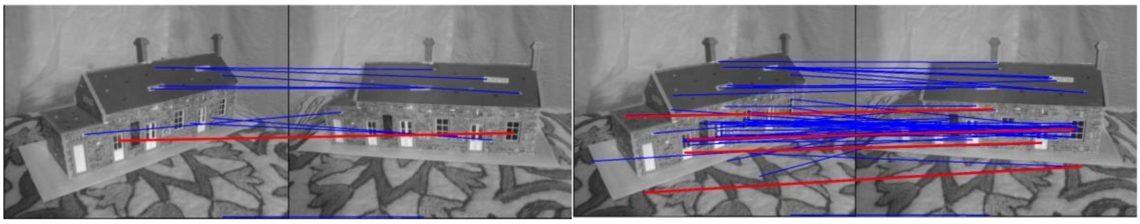
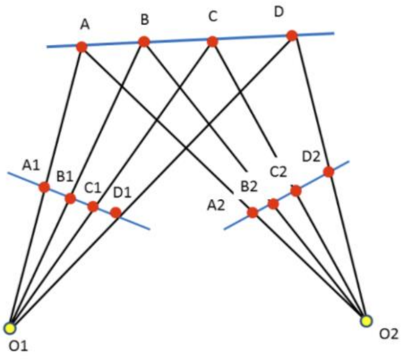


Figure 3: k=10 (left) and k=50 (right)

5. LINE-SWEEP

Line-Sweep finds additional point correspondences using the cross-ratio of line segments, which is invariant under projective transformation i.e., Figure 4.



$$\text{Cross}(A1, B1, C1, D1) = \text{Cross}(A2, B2, C2, D2)$$

$$\text{Cross}(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \bar{\mathbf{x}}_3, \bar{\mathbf{x}}_4) = \frac{|\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2|\,|\bar{\mathbf{x}}_3\bar{\mathbf{x}}_4|}{|\bar{\mathbf{x}}_1\bar{\mathbf{x}}_3|\,|\bar{\mathbf{x}}_2\bar{\mathbf{x}}_4|}$$

$$\text{where} \quad |\bar{\mathbf{x}}_i\bar{\mathbf{x}}_j| = \det\begin{bmatrix} x_{i1} & x_{j1} \\ x_{i2} & x_{j2} \end{bmatrix}$$

Figure 4: Cross-ratio example. O1 and O2 are camera centers. A1, B1, C1, D1 lie on the same image plane.

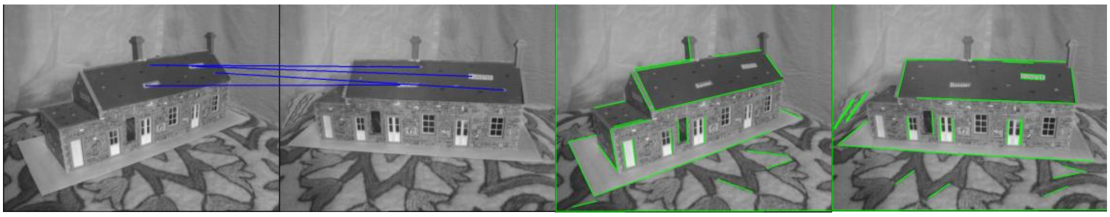


Figure 5: Initial correspondences (left); Detected lines (right)

From initial keypoint matches found via SIFT feature matching and sets of detected lines in both images i.e., Figure 5, Line-Sweep finds additional keypoint matches through the following steps:

1. We form virtual lines between each pair of initial keypoints e.g., Figure 6.

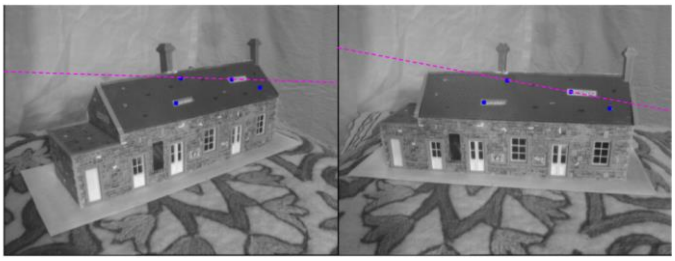


Figure 6: Example of virtual lines (in pink) from keypoints.

2. For each virtual line, we find line crossings i.e., intersections between the virtual line and the detected (real) lines e.g., Figure 7.

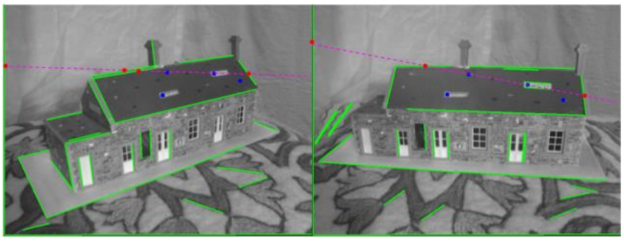


Figure 7: Line crossings in red

3. For each pair of line crossings in each image, we compute the cross ratios with the original keypoints forming the virtual lines.
4. If the pair of line crossings in the first image has the same cross-ratio with the pair of keypoints in the second image, the line crossings correspond to each other i.e., Figure 8.

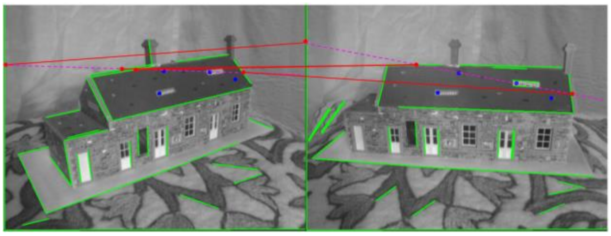


Figure 8: Additional point correspondences.

5. We repeat the algorithm for each virtual line to obtain new point correspondences i.e., Figure 9:

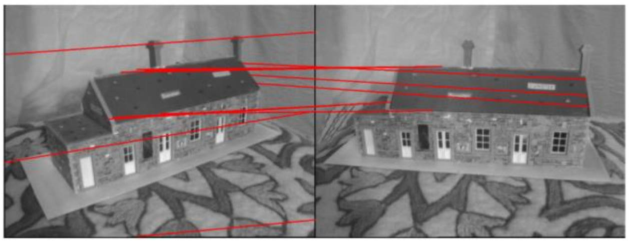


Figure 9: Line-Sweep point correspondences

6. ASSIGNMENT TASK

Your task in this assignment is to implement the following helper functions for the Line-Sweep algorithm. We will provide the skeleton file with missing code. You are to implement the following in `main.py`:

- `get_keypoints [1]`: use the cv2.SIFT_create to perform find SIFT keypoints within each image. More information is provided in the comments within the source code.
- `get_matches [1]`: given the SIFT descriptors from both images, use feature matching i.e., cv2.BFMatcher and order the keypoint matches in ascending order of distance.
- `convert_line_pts_to_lines [2]`: Given line points (x, y, x', y'), convert the line points into the line representation (a, b,1) given in the lecture notes where ax + by + 1 = ax' + by' + 1 = 0. You are expected to use material from the lecture notes.
- `get_line_intersections [2]`: Given lines (a, b, 1) and (a', b', 1), find the intersection between the lines i.e. (u, v, 1). You are expected to use material from the lecture notes.
- `get_line_crossings [3]`: Given the virtual line and the detected (real) lines, find the line crossings between the virtual line and real lines.
- `get_cross_ratios [1]`: Compute the cross ratio of line segments given collinear points (A, B, C, D).

Write your code with the block (“”” YOUR CODE HERE “””) and (“”” END YOUR CODE HERE “””) for each function. Students that do not follow submission instructions will incur a 1-point penalty. Students are strongly encouraged to seek help via the forums. We also provide the following outputs from our implementation in `ta_impl_results` for error checking:

- initial-keypoint-matches: initial keypoint matches from SIFT feature matching, available as .png and .npy files.
- new-keypoint-matches: new keypoint matches from Line-Sweep, available as .png and .npy files.
- final-keypoint-matches: keypoint matches from feature matching and Line-Sweep, available as .png file.

Students can post their questions on the forums. You are strongly encouraged to post on the forums and check the forums before sending emails.

7. REFERENCES

[1] S. Ramalingam, M. Antunes, D. Snow, G. H. Lee and S. Pillai, "Line-sweep: Cross-ratio for wide-baseline matching and 3D reconstruction," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1238-1246, doi: 10.1109/CVPR.2015.7298728.