Homework 1

Due: 2022/10/4 11:59 AM

Email: 3dcv@csie.ntu.edu.tw

GitHub Classroom: https://classroom.github.com/a/5IGz7BXw

GitHub Registration: https://forms.gle/ikNyfxXJUw7krcG88

Outline

- Problem1: Homography estimation (Q1-1 ~ Q1-3)
- Problem2: Document rectification (Q2-1 ~ Q2-2)
- Report and submission

Problem 1: Homography Estimation

Given three color images A (1-0.png), B (1-1.png), and C (1-2.png),

please follow the instruction to compute the homographies that warps the anchor image A to target image B and C.

1-0.png

1-1.png

1-2.png

Problem 1: Homography Estimation

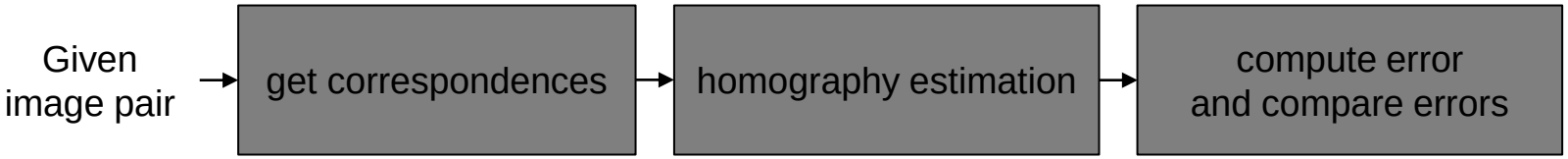


image pair #1

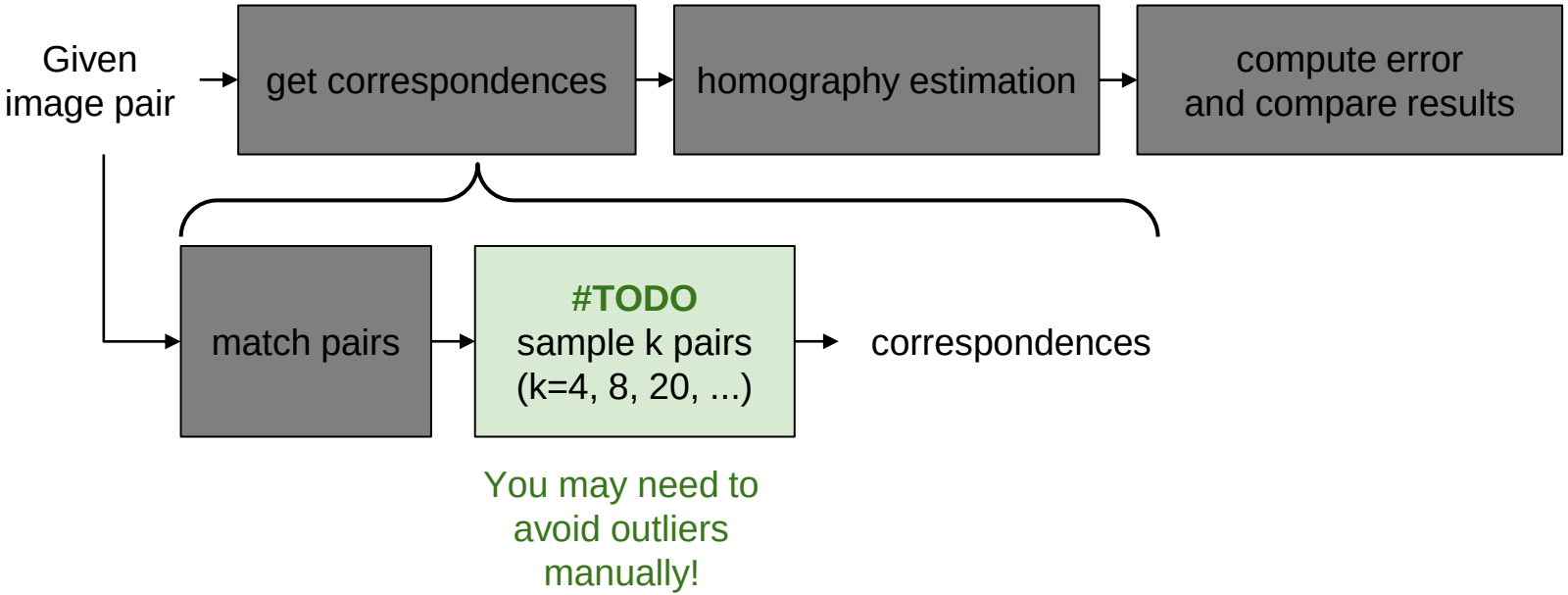
image pair #2

Problem 1: Homography Estimation

Q1-1 Feature Matching

- Perfom local feature detection on each image.
- Find the correspondence between anchor image and target images by descriptor matching.
- Reject the outliers by ratio test or manual comparison and select top k pairs from the matching result, where k = 4, 8, 20.
- (BONUS) Try other local features, e.g., SuperPoint

Problem 1: Homography Estimation



Problem 1: Homography Estimation

Q1-2 Direct Linear Transform

- For each k value, estimate the homography between anchor image and target images with direct linear transform.
- Compute the reprojection error with the ground truth matching pairs.

$$\hat{p}_t \approx \mathcal{H} p_s$$

$$\text{error} = \frac{1}{N} \sum^{N} \| p_t - \hat{p}_t \|_2$$

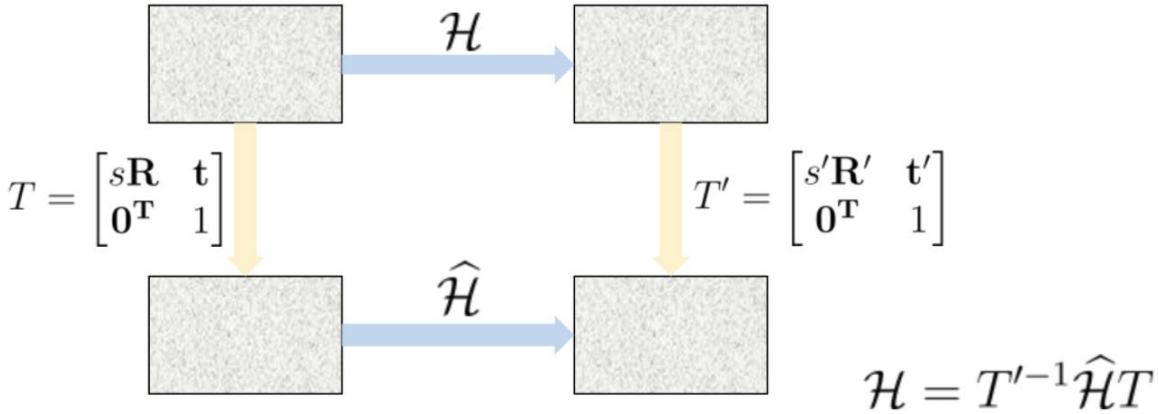
groundtruth:
correspondence_01.npy: from A to B
correspondence_02.npy: from A to C

Each contains a NumPy array (2 x 100 x 2):
(image, number of points, xy coordinates)

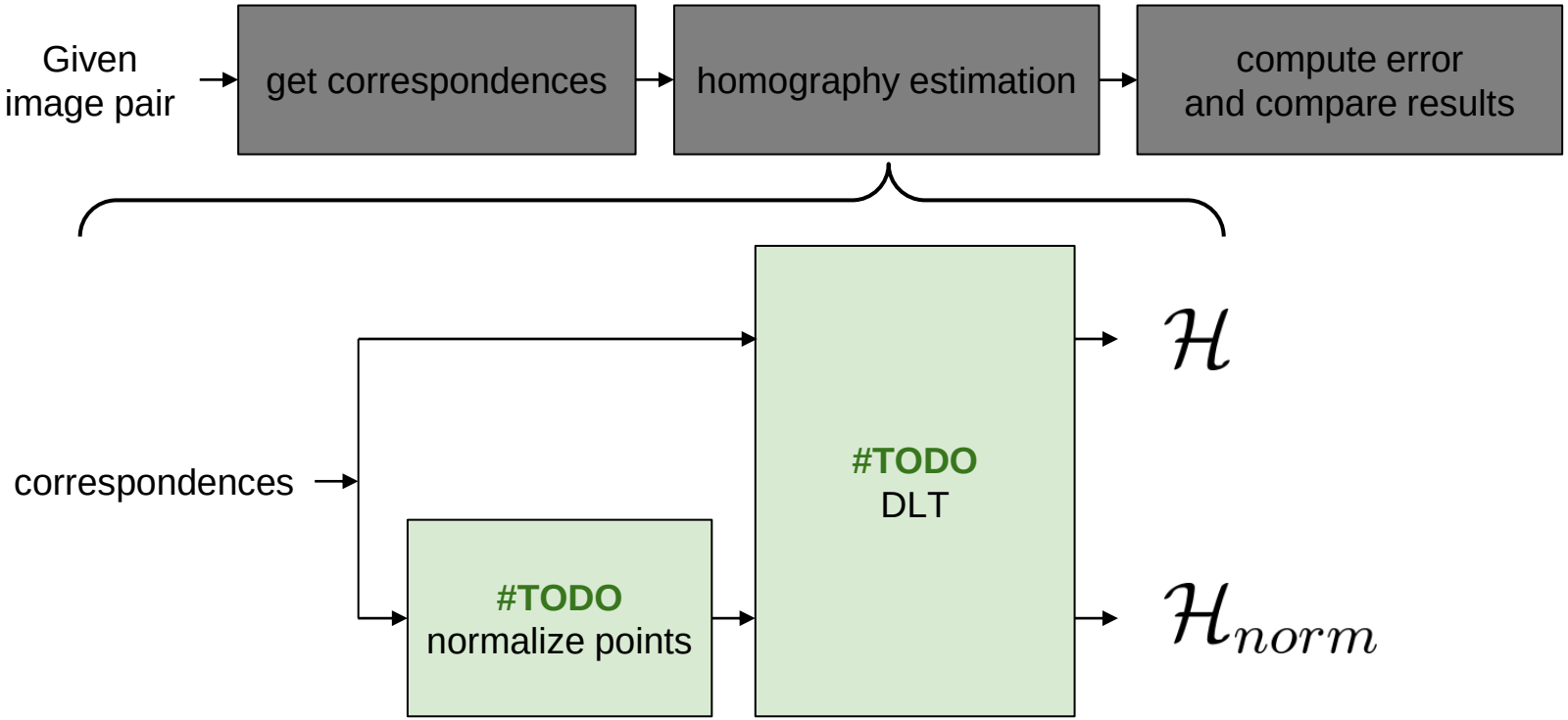
Problem 1: Homography Estimation

Q1-3 Normalized Direct Linear Transform

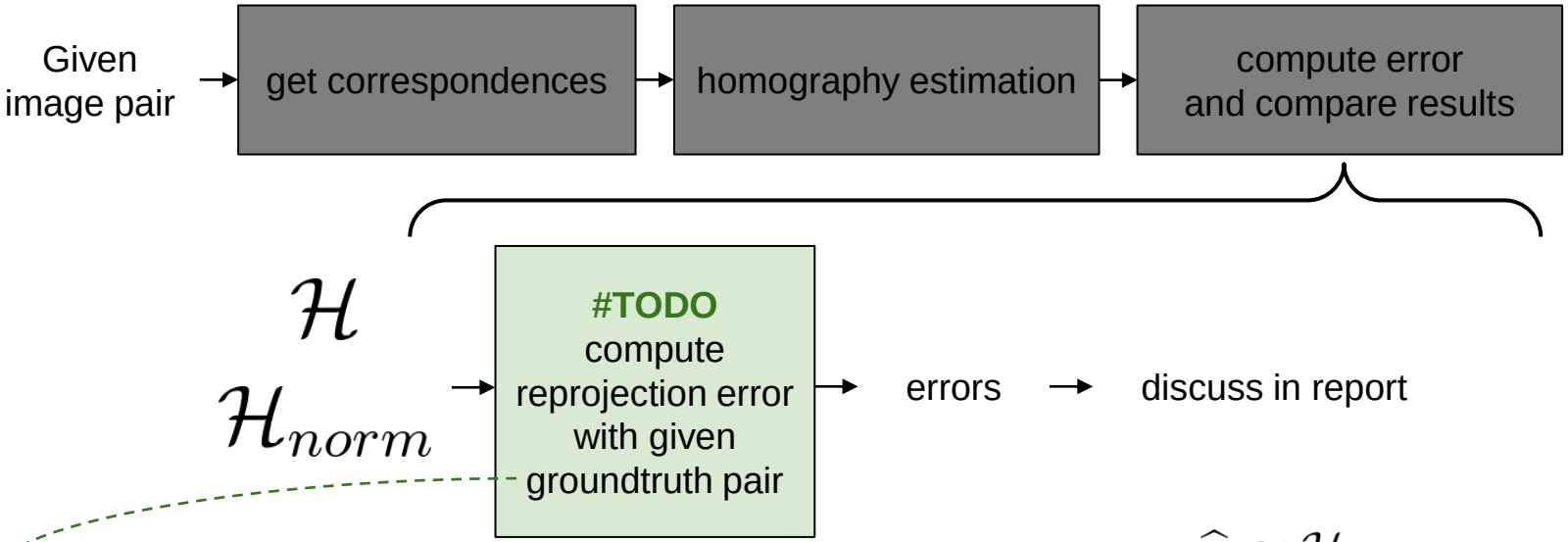
- Similar to Q1-2, but use **normalized** direct linear transform instead.
- Compute the reprojection error and compare the results with Q1-2.
- (BONUS) Try other methods or tricks that may improve DLT or Normalized DLT.



Problem 1: Homography Estimation



Problem 1: Homography Estimation




```
In [1]: import numpy as np

In [2]: gt_pairs = np.load('correspondence_01.npy')

In [3]: gt_pairs.shape
Out[3]: (2, 100, 2)

In [4]: p_s = gt_pairs[0]

In [5]: p_t = gt_pairs[1]
```

* for cpp version: correspondence_*.txt are saved as flattened list of (2, 100, 2) array

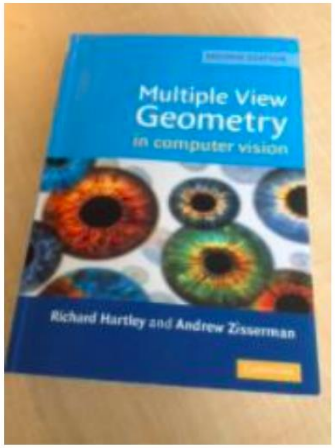
$$\widehat{p}_t \approx \mathcal{H} p_s \quad \leftarrow \text{reprojection}$$

$$error = \frac{1}{N} \sum^{N} \|p_t - \widehat{p}_t\|_2$$

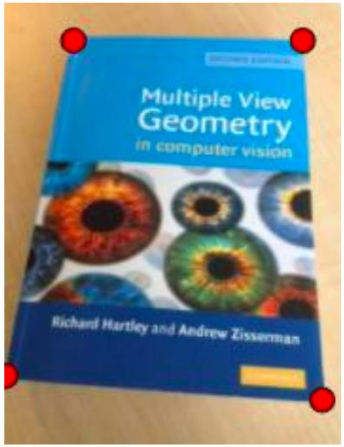
Problem 2: Document Rectification

Rectification is one of the most fundamental techniques when digitizing documents.

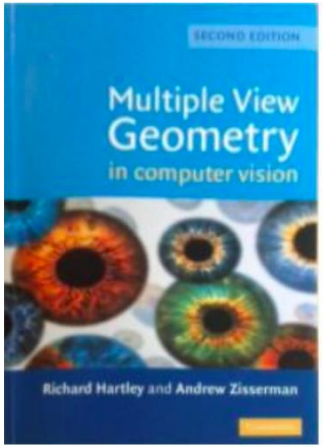
Given an image of a document captured by the camera, please recover its original geometric property which is lost after perspective transformation.



Source Image



Marked Image



Rectified Image

Problem 2: Document Rectification

Q2-1 Capture Document & Mark 4 Corner Points

- Find an interesting document that your want to rectify. Note that the image must be captured by yourself.
- Automatically or manually mark the corner points on the image.

Problem 2: Document Rectification

Q2-2 Homography Estimation & Warp Image

- Compute the homography for image transformation.
- Implement bilinear interpolation for image warping.

Problem 2: Document Rectification

#TODO
Capture paper/document on your own

→

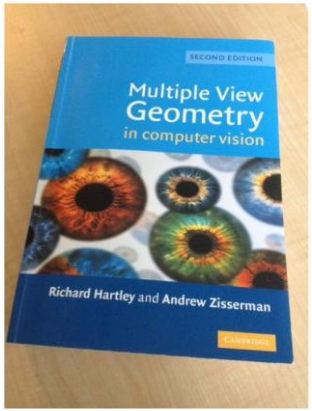
#TODO
select the 4 corners of the captured image

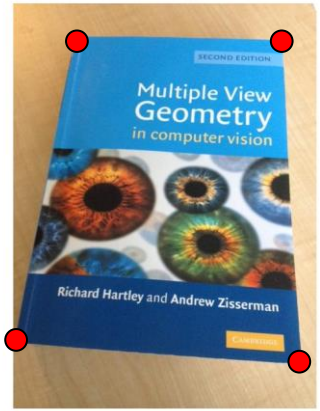
→

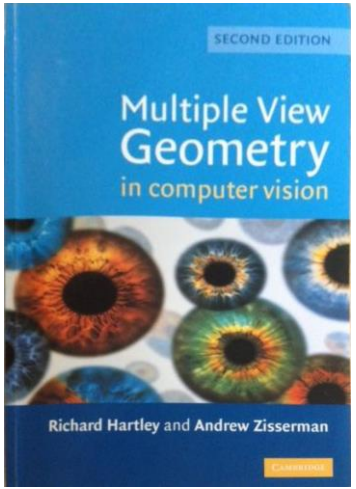
#TODO
warping

→

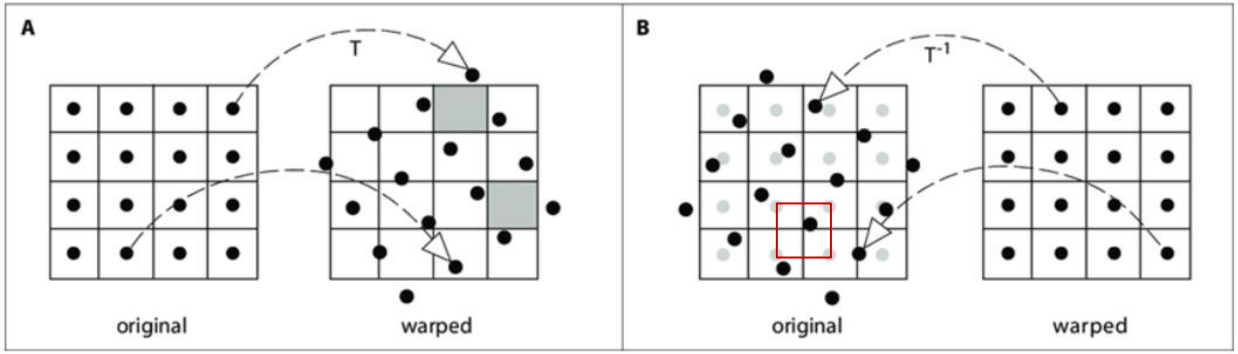
rectified result





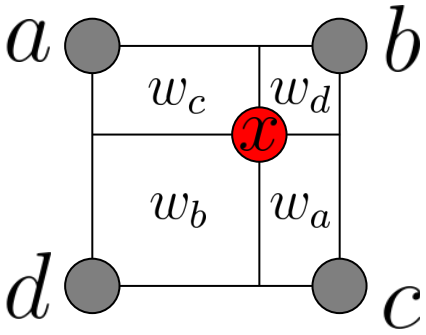


Problem 2: Document Rectification



forward warping

backward warping



bilinear interpolation

$$x = w_a a + w_b b + w_c c + w_d d$$

Report

- Problem 1: Homography estimation
  - Screenshots:
    - Sample k correspondences (k = 4, 8, 20 or even more)
  - Compare the errors:
    - DLT vs. normalized DLT
    - Sample k = 4, 8, 20 or even more
  - (Bonus)
    - Your method
    - Screenshot: correspondences of other local features
    - Experimental comparisons
  - Discussion
    (interesting finding, difficulties you encountered, insights you observe)

Report

- Problem 2: Document rectification
  - The input document image (must be captured by yourself)
  - Rectified results
  - Briefly explain your method (how you choose the corners, warping efficiency)
- Please tell us **how to execute** your codes, including the package used and the environment.

Submission

- **Due**: **2022/10/4 11:59 AM**
- Github classroom: https://classroom.github.com/a/5IGz7BXw

  please fill your ID and github username in the spreadsheet

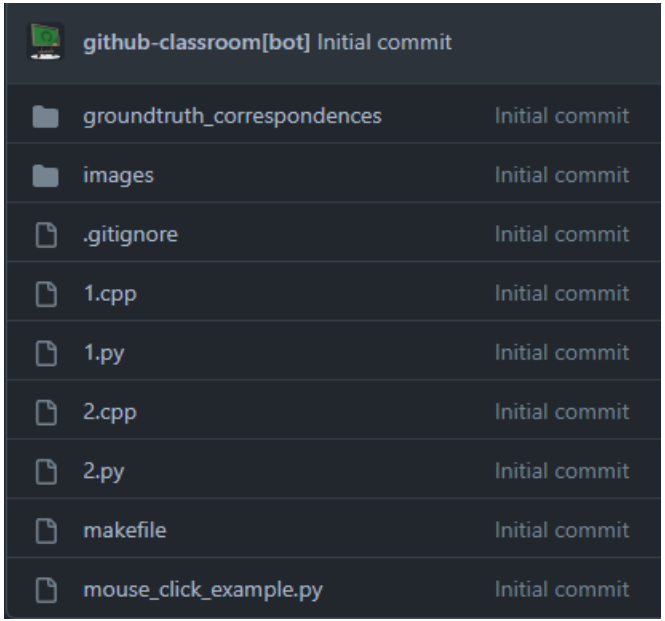
Python Submission

- **1.py 2.py, image you captured for part 2**
- report.md (or report.pdf)

C++ Submission

- **1.cpp, 2.cpp, makefile, image you captured for part 2**
- report.md (or report.pdf)

Example



Initial repository

Python Submission example

API policy

- The APIs you may use:
  - OpenCV:

    File IO, UI, e.g., imread, imshow, waitKey, setMouseCallback, etc.

    Linear algebra (c++)

    The libraries you use for bonus
  - Numpy:

    Linear algebra: numpy.linalg
- The following APIs are **forbidden**:
  - OpenCV: findHomography, warpPerspective

Environment

- TA will run your code with following environment:
  - Python >= 3.6
  - OpenCV == 4.5.1.48
  - Numpy >= 1.19.5

Grading Rubrics

- We will evaluate both **the functionality of the code** and **the quality of the report**.
- **Functionality**: Can it run? How's the performance?
- **Quality**: theoretical/experimental analysis, observation, discussion, …
- Note that it **might be curved** based on overall performance of students.
- Grade
  - Meet the basic requirement (programming & report) → A
  - Basic requirement + advanced studies (programming & report) → A+

General Policies

- Programming Languages: Python, C++
- Report Format: PDF or Markdown

  (Warning for Markdown users: Latex equations cannot be rendered properly in GitHub)
- Late Submission: **-10% from your score** / day
- Plagiarism: You have to **write your own codes**.
- Discussion: We encourage you to discuss with your classmates, but remember to **mention their names and contributions in the report**.