Homework 1

Due: 2024/10/7 (一) 11:59 AM

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

GitHub Classroom: https://classroom.github.com/a/i0rw692v

GitHub Registration: https://forms.gle/ucH5A2fsANX9MPzS7

Outline

- Problem1: Homography estimation
- Problem2: Homograohy warping
- 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.

Problem 1: Homography Estimation

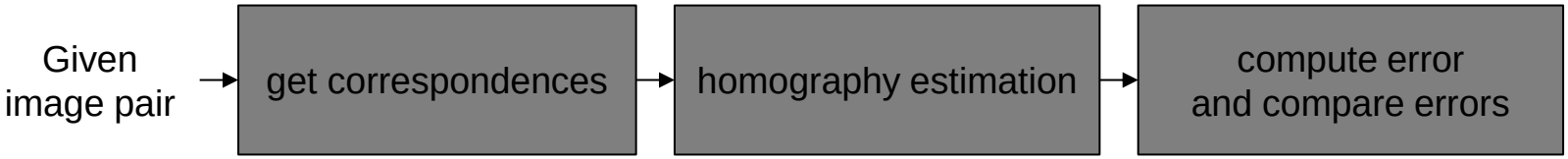


image pair #1

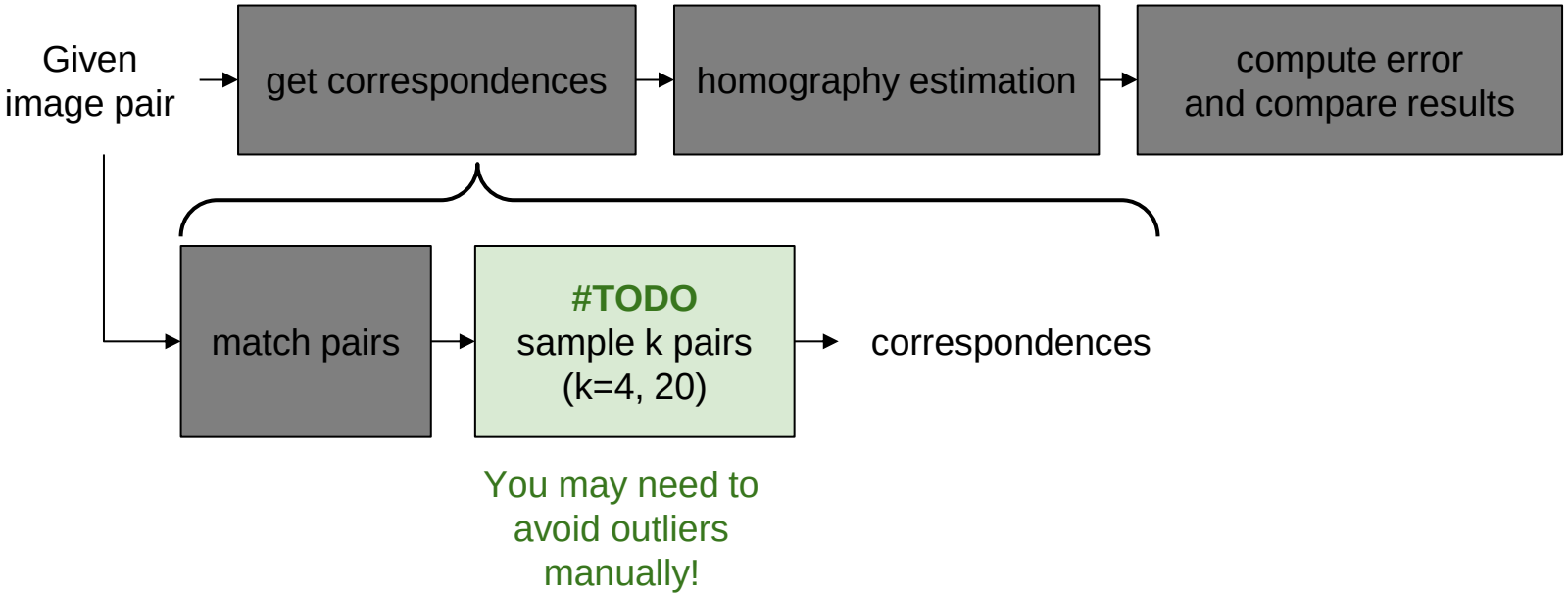
image pair #2

Problem 1: Homography Estimation

Feature Matching

- Perfom local feature detection on each image.
- Find the correspondence between anchor image and target images by descriptor matching.
- Reject some the outliers by ratio test and manual select k pairs from the matching result, where k = 4, 20.
- Please select the point on the floor

Problem 1: Homography Estimation



Problem 1: Homography Estimation

Normalize Direct Linear Transform

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

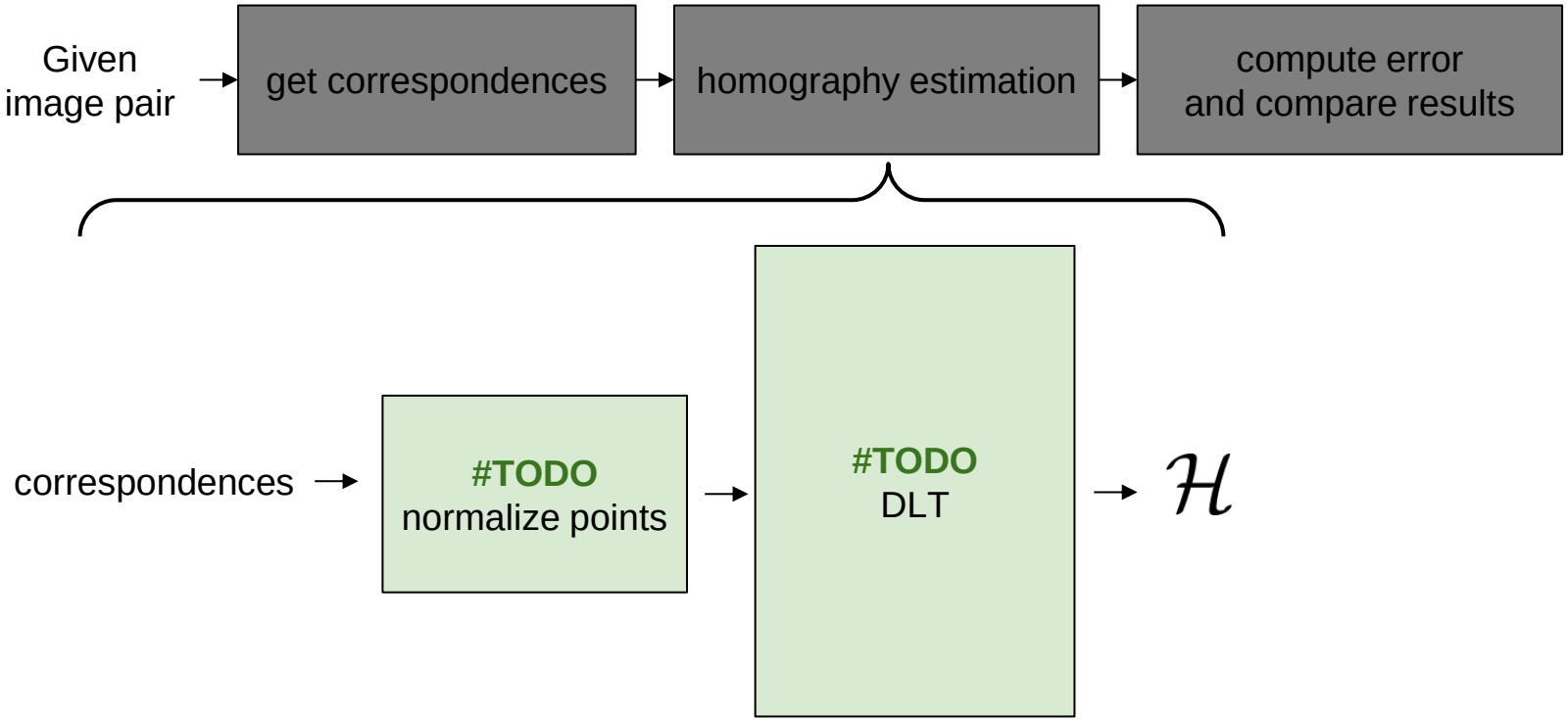
$$\hat{p}_t \approx \mathcal{H} p_s$$

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

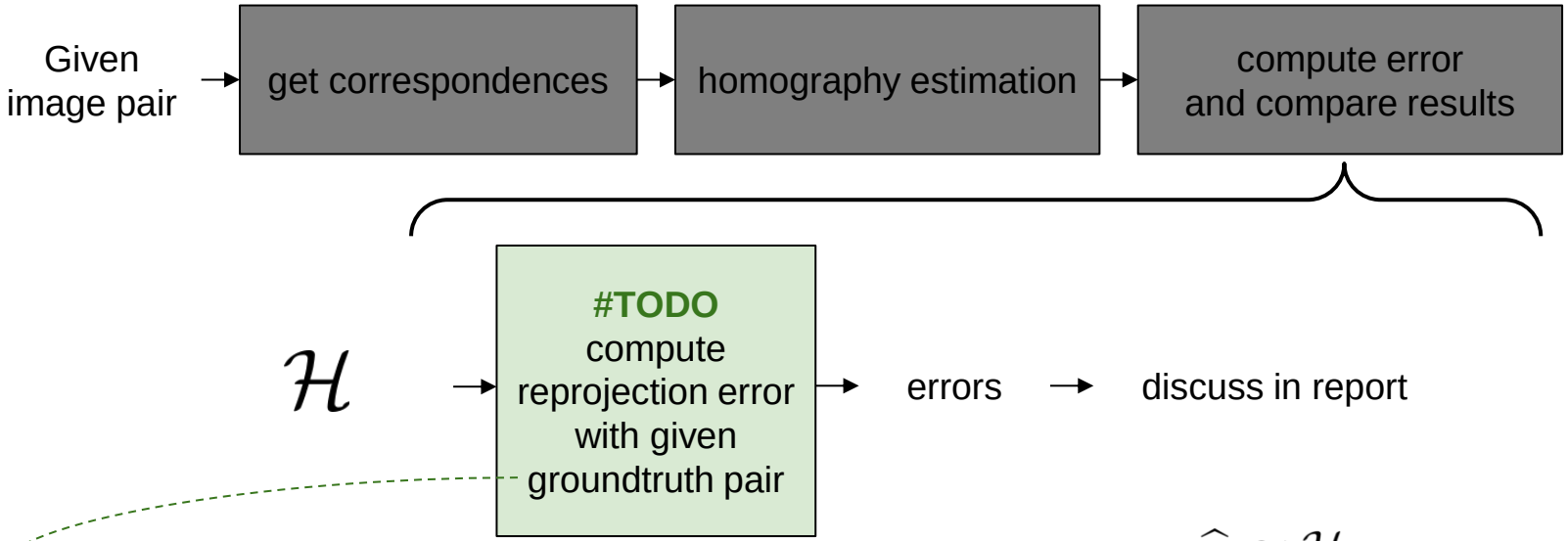
groundtruth:
correspondence_01.npy: from A to B
correspondence_02.npy: from A to C

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

Problem 1: Homography Estimation



Problem 1: Homography Estimation



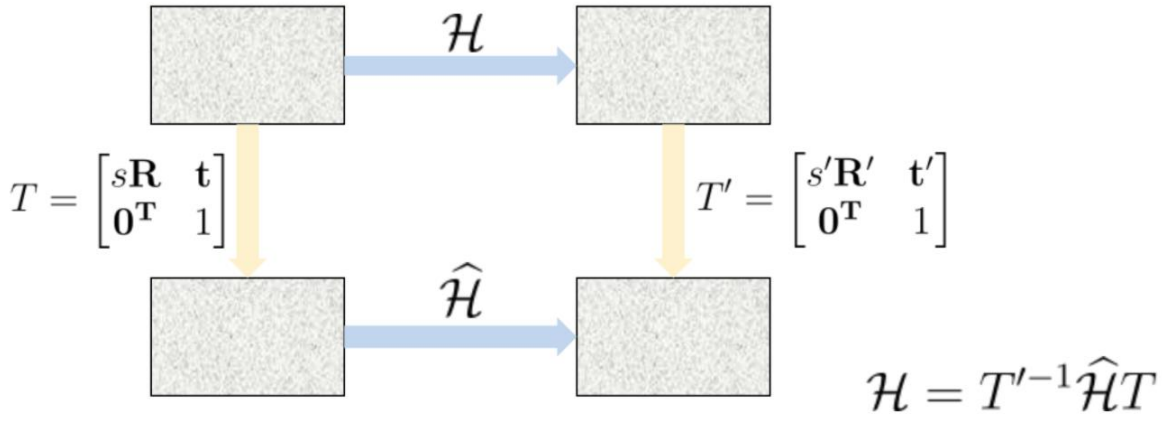

```
>>> import numpy as np
>>> gt_pairs = np.load("groundtruth_correspondences/correspondence_01.npy")
>>> gt_pairs.shape
(2, 631, 2)
>>> p_s = gt_pairs[0]
>>> p_t = gt_pairs[1]
>>>
```

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

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

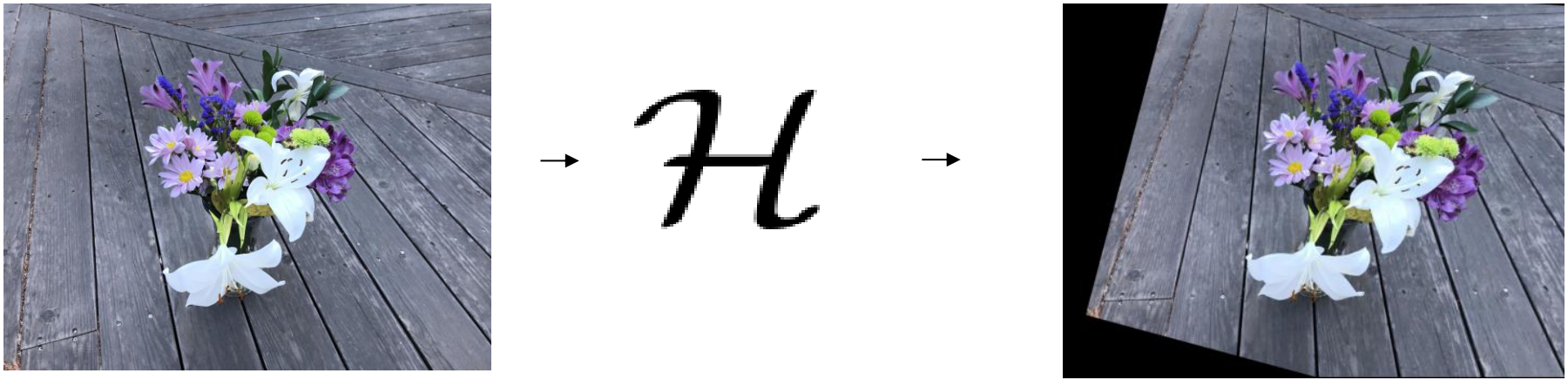
Problem 1: Homography Estimation

Normalized Direct Linear Transform



Problem 2: Homography Warpping

Homography warping refers to the process of transforming an image or a set of points from one plane to another using a homography matrix.



Problem 2: Homography Warpping

Warp the Image A according to the Homography you estimation

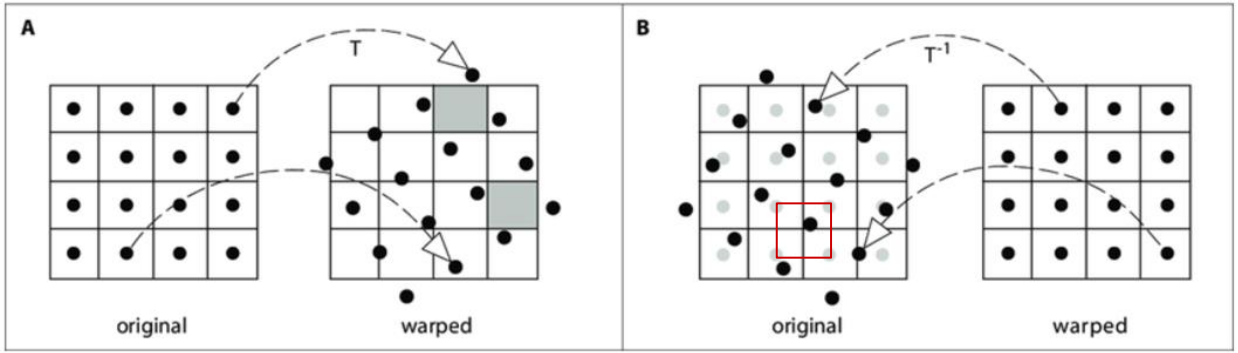
- Implement bilinear interpolation for image warping
- Warping the image A according to the Homography you estimation previously

Problem 2: Homography Warpping

Given image and homography matrix → **#TODO** warping → Warping image

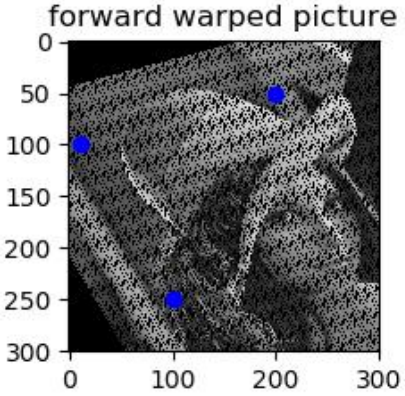
→ $\mathcal{H}$ →

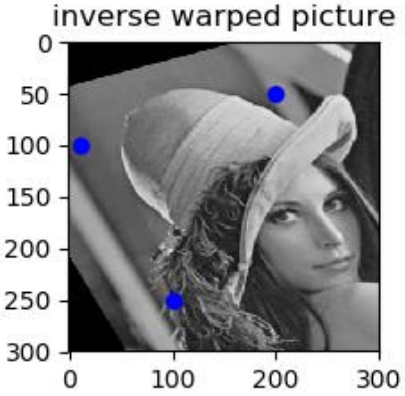
Problem 2: Homography Warpping

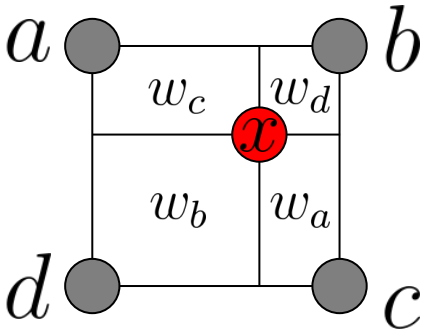


forward warping

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, 20)
  - Compare the errors:
    - Sample k = 4, 20
    - Your method (if you have other try)
    - Experimental comparisons
  - Discussion
    (interesting finding, difficulties you encountered, insights you observe)

Report

- Problem 2: Homography Warpping
  - Warping results (show with concatenation of Image A and Image B/C)
  - Briefly explain your method (warping efficiency)

- [*] Youtube link
  - You should record your demonstration, including the start time and the GitHub clone action
  - Example :
    - https://youtu.be/-VnjVda7c8o?si=zowfe7vjvCMMFrOk

- Please tell us **how to execute** your codes, including the package used and the environment.

Submission

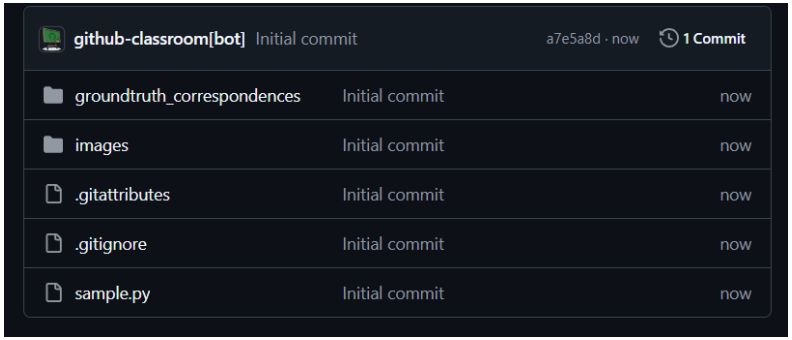
- **Due**: **2024/10/7 (一) 11:59 AM**
- Github classroom: https://classroom.github.com/a/i0rw692v

  please fill your ID and github username in the spreadsheet

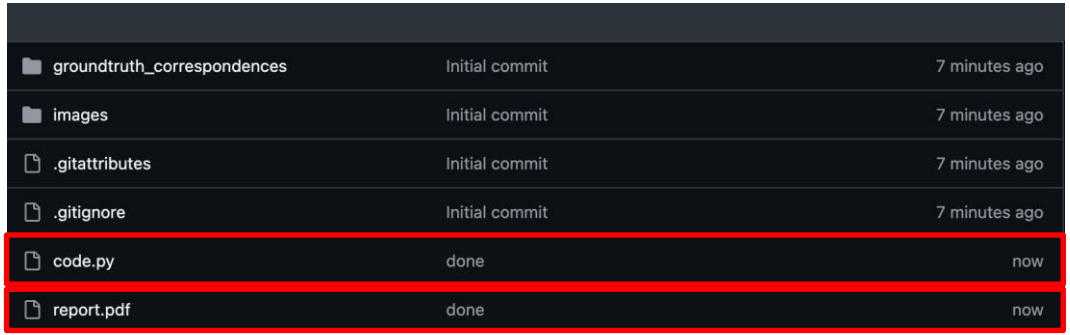
Python Submission

- **code.py**
- **report.md (or report.pdf)**

Example



Initial repository



Python Submission example

API policy

- The APIs you may use:
  - OpenCV:

    File IO, e.g., imread, etc.

    The libraries you use for bonus
  - Numpy:

    Linear algebra: numpy.linalg
  - Matplotlib:

    Display image
- 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
  - Matplotlib

Remind

- OpenCV may conflict with Matplotlib
  - https://github.com/opencv/opencv-python/issues/386

```
QObject::moveToThread: Current thread (0x199fcb0) is not the object's thread (0x1ae4c70).
Cannot move to target thread (0x199fcb0)

qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "/home/taha/.local/lib/python3.8/site-packages/cv
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the applicati

Available platform plugins are: xcb, eglfs, linuxfb, minimal, minimalegl, offscreen, vnc, wayland-egl, wayland,

Aborted (core dumped)
```

- If you face such problem you can install opencv-python-headless instead of opencv-python

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
- 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**.