Assignment 1
Image matching

ACV Autumn
Sep 28, 2020

國立交通大學
National Chiao Tung University

Quick Overview

image1

[1]SIFT

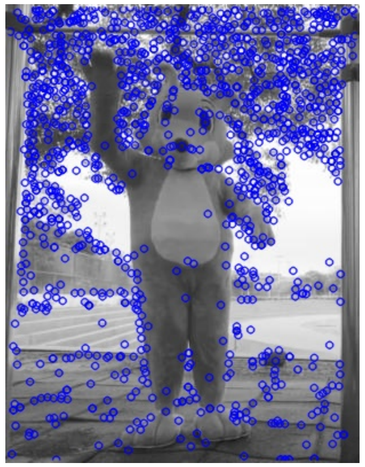
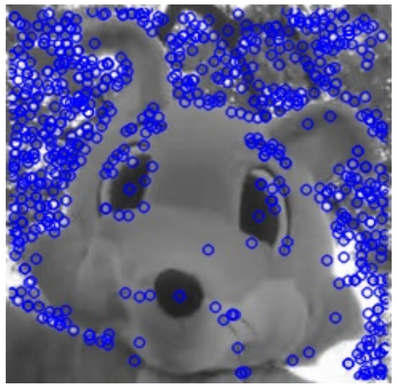
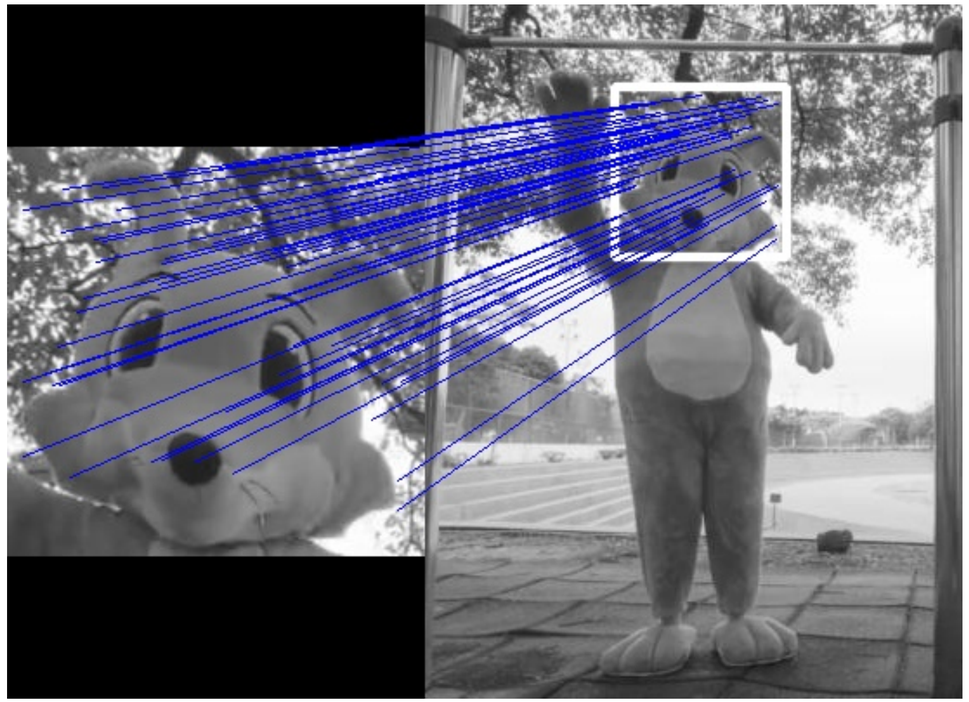


image2

[1]SIFT



[2]matching



Requirement

- Implement **SIFT** and **matching** using python
- Input : given 2 images as a set, 3 sets in total
- Output : **keypoints of the images and the matching results**

- **Write a report** :
  - Explain all parts of your code
  - Show what you have done in your experiments (e.g., parameters setting)
  - **Result images**

- **Result images** should include :
  - Image pyramid (as shown in page 5)
  - The DoG of any octave you like (page 7)
  - The images with keypoints (page 8)
  - The matching results (page 9)

Quickly go through the requirement

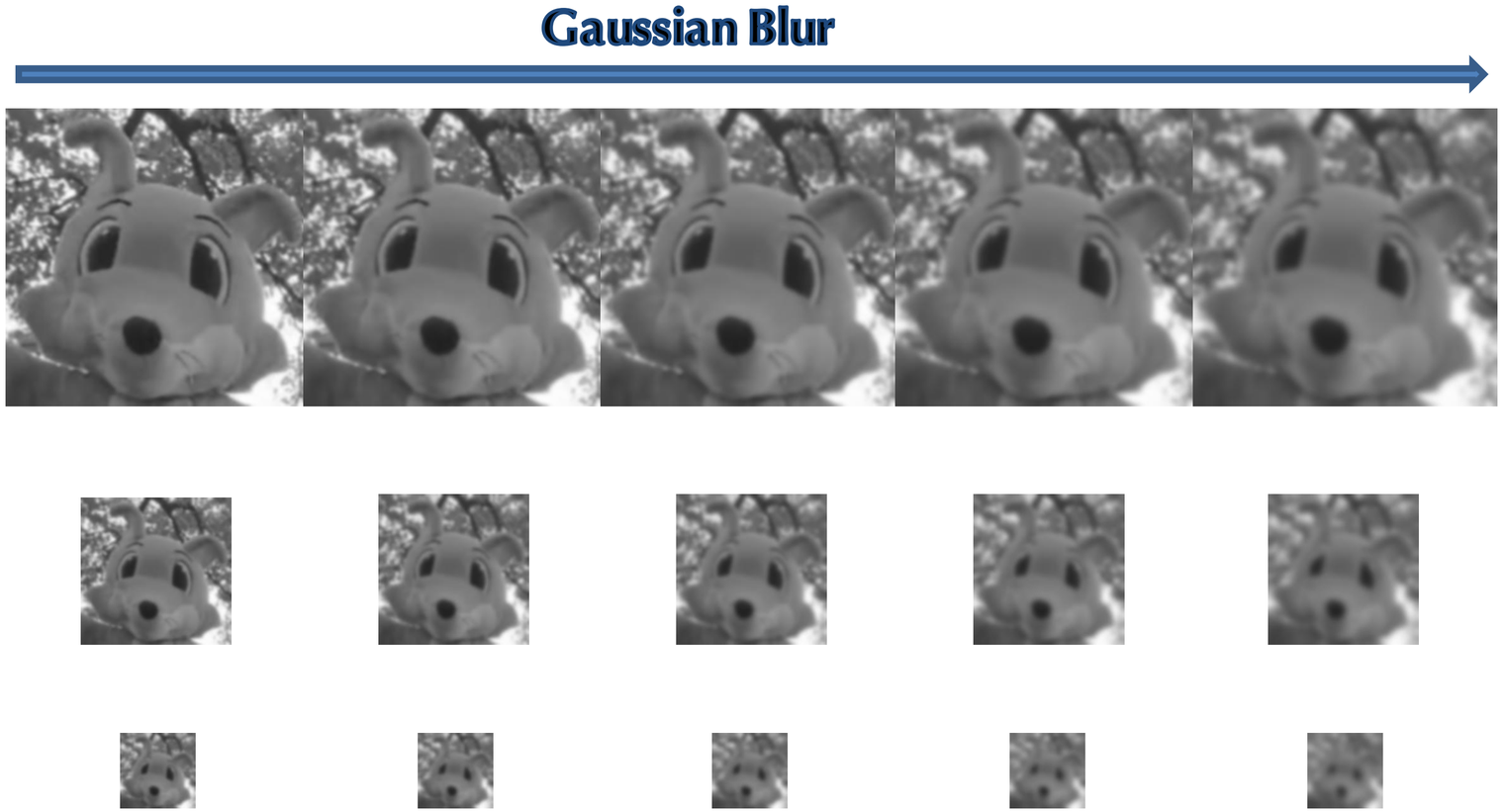
Implement SIFT

- Construct scale space (image pyramid, difference of gaussian)
- Find the local extrema (a.k.a. keypoints)
- Generate descriptors

Quickly go through the requirement

- Image pyramid for reducing the noise

Warning:
To contruct image pyramid, you **cannot** use high level function in any well-developed library.
E.g. cv2.pyrDown(), cv2.pyrUp()

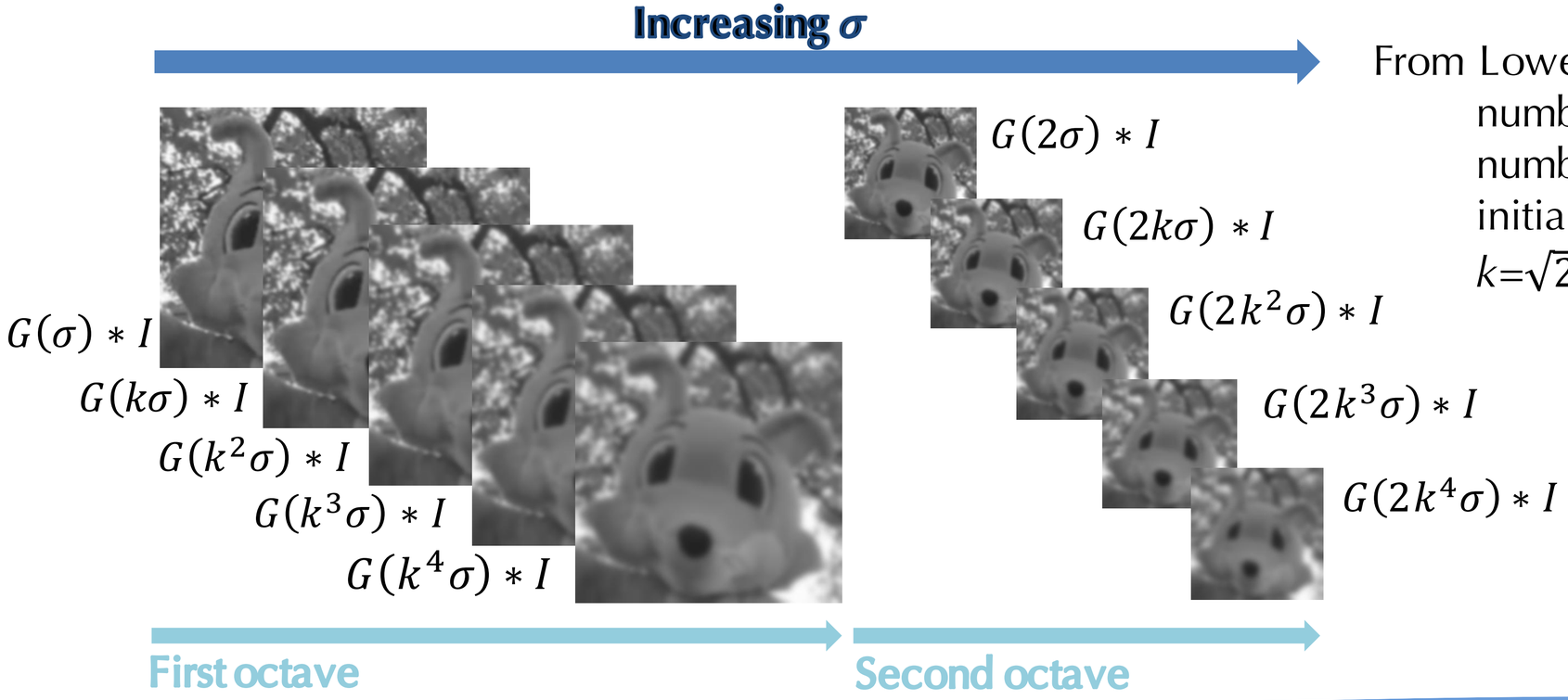


Quickly go through the requirement

s: # of scale levels in the i^{th} octave

- Image pyramid for reducing the noise : $2^{i-1}(\sigma, k\sigma, k^2\sigma, \cdots, k^{n-1}\sigma), \quad k = 2^{1/s}$

  an octave is actually a set of images were the blur of the last image is **double** the blur of the first image



From Lowe's paper :

number of octaves = 4

number of scale levels = 5

initial σ =1.6

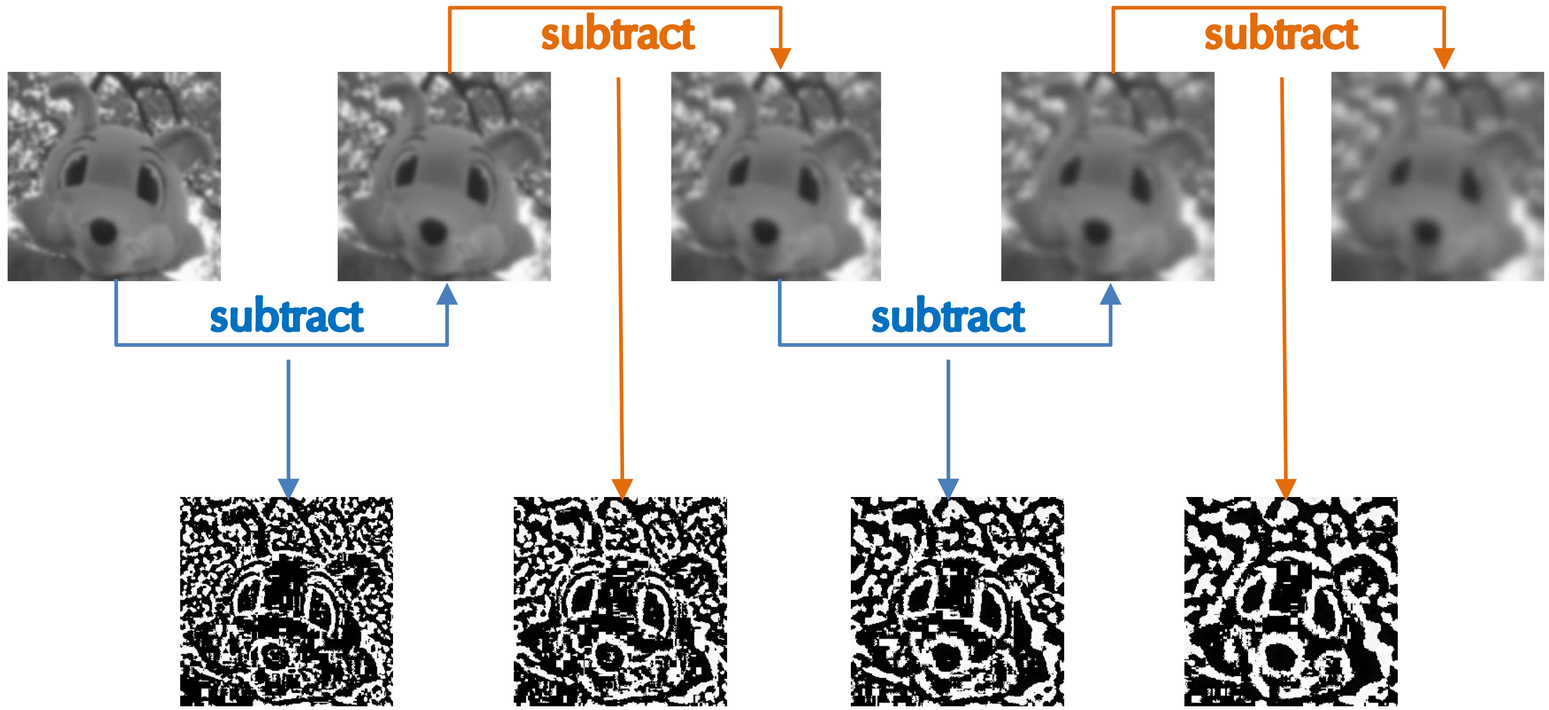
k=$\sqrt{2}$ as optimal values

Quickly go through the requirement

- Difference of Gaussian for reinforcing features

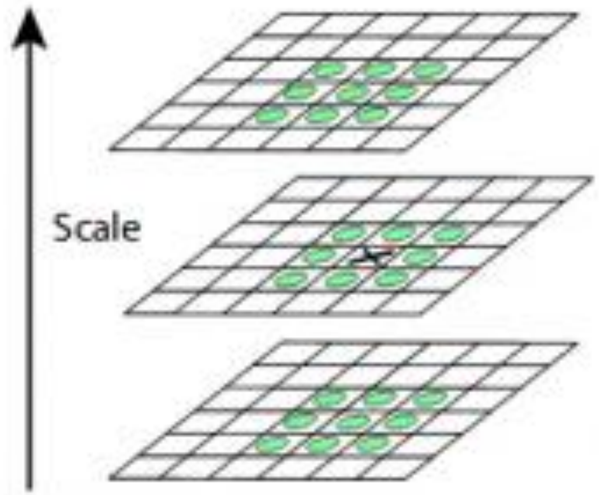
Second octave

DoG

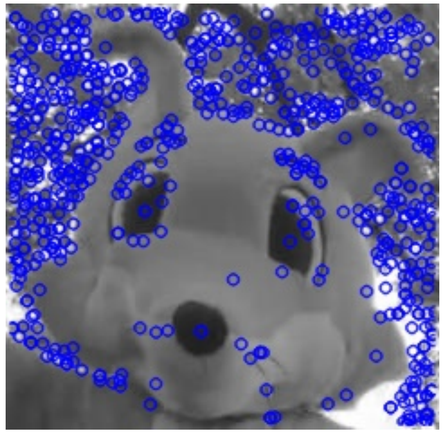


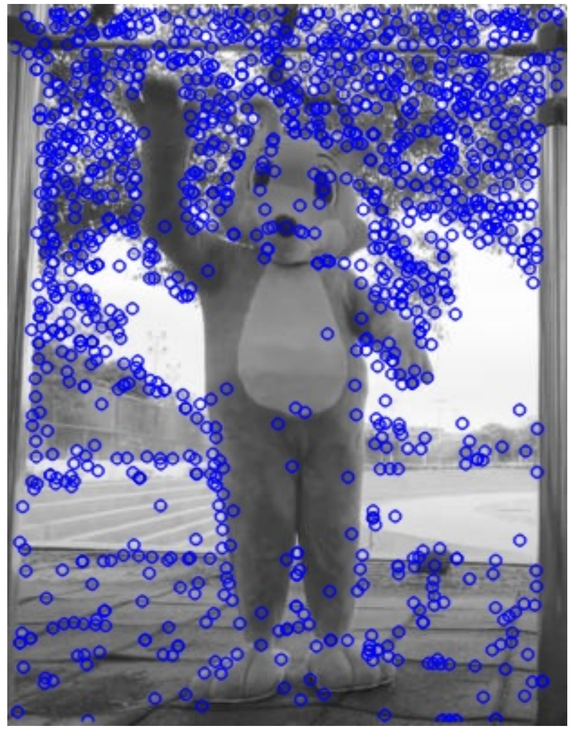
Quickly go through the requirement

- Find the local extrema



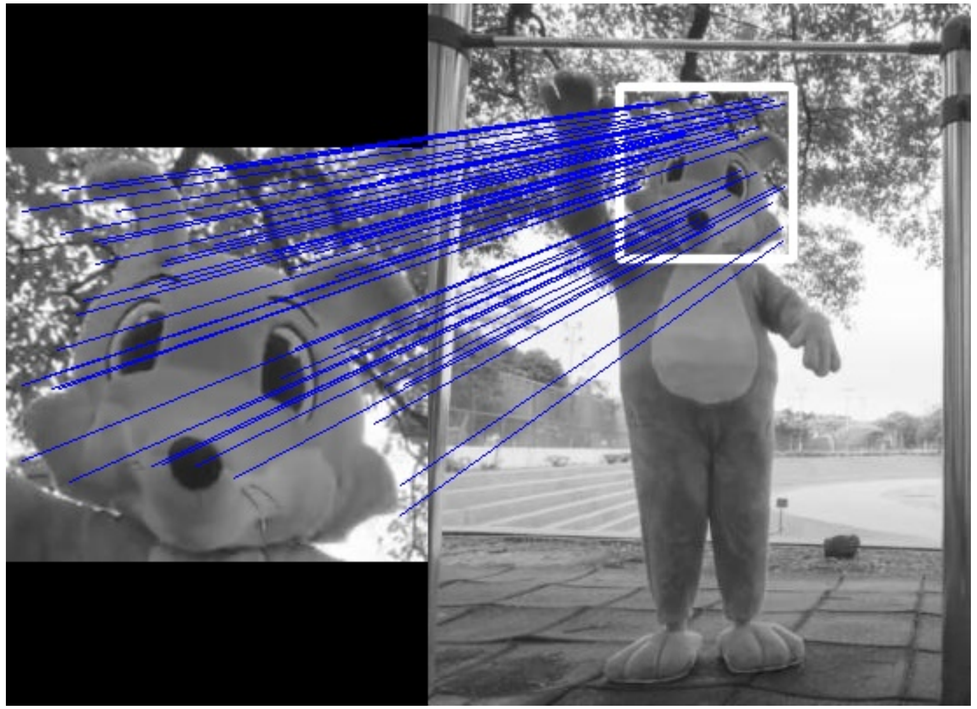
- Generate the descriptors





Quickly go through the requirement

- Match features between two images



For this step, you can use whatever function you want. For example,

- RANSAC
- Brute-Force Matcher
- BLANN

Rules

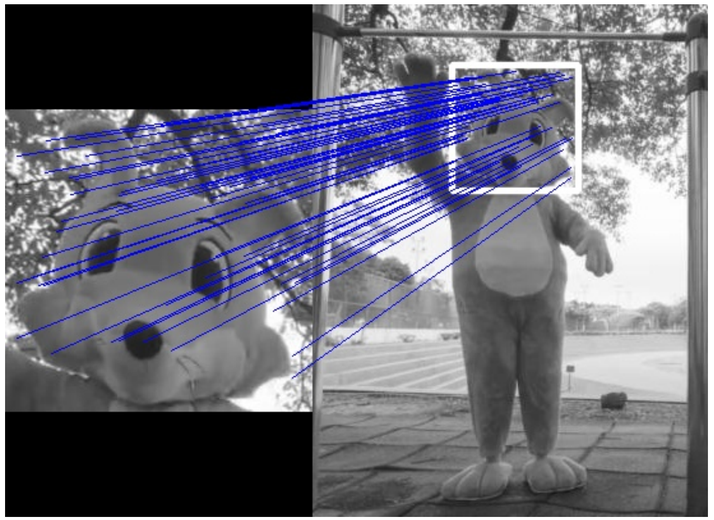
- Implement in **python3**
- Do not directly call the high level SIFT function in any well-developed library. e.g. cv2.xfeatures2d.SIFT_create()
- Do not copy/paste other's code
- Report no more than 10 pages

Score

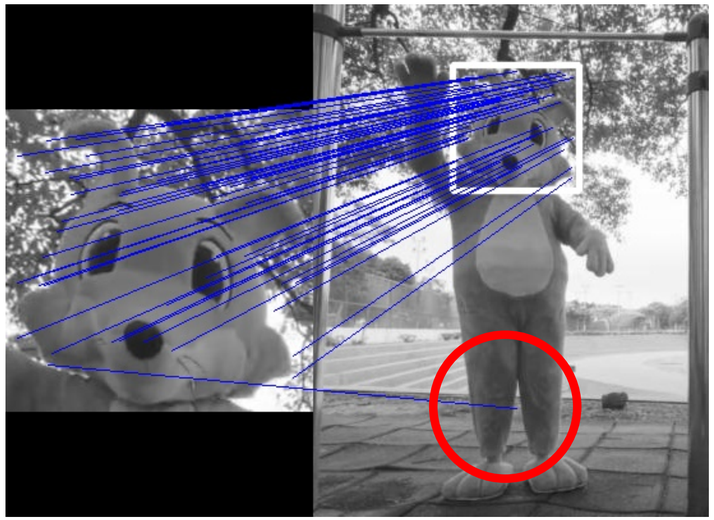
- Implement SIFT (45%)
- Implement matching (10%)
- Report (45%)
  - We have a scoring example for your matching output (as shown in page 12)
- Bonus (9%)
  - Implement SIFT and matching with your own images
  - Each set of images would add 3 points at most
  - 3 sets of images at most

Quality example

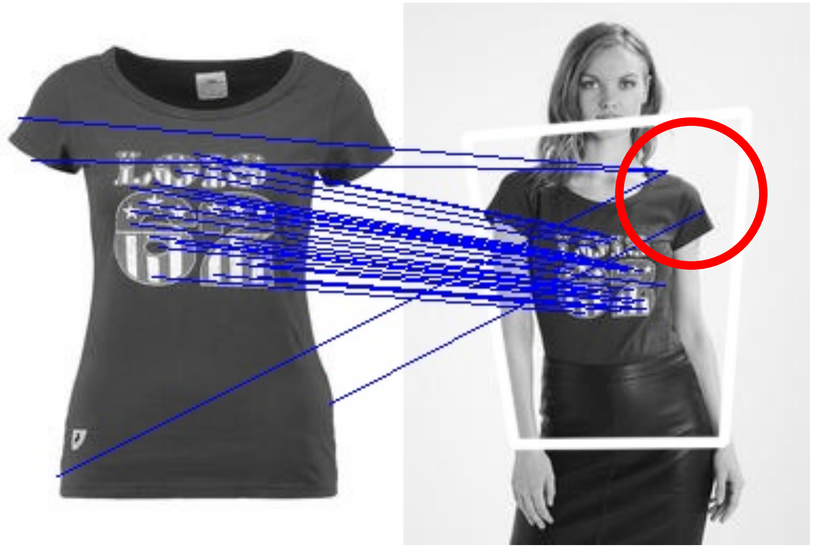
3 points



2 points



1 points



Reminder

- You should work on all the given images (3 sets)
- Feel free to modify any code provided from TA

Submission

- Every submission should consist of the followings:
  - Your code (student_id.py)
  - A readme.txt file describing how to run your code
  - A report (in pdf format)
- Please clip all your files into <student_id>.zip and submit through New e3
- Due on Oct **19**, 2020 **23:55:00** PM

- If you have any question about this homework, please e-mail to TAs
  - Chieh-Yun Chen (陳婕云) cychen.ee09g@nctu.edu.tw
  - Ching-Hao Wang (王敬豪) billywang.ee08g@nctu.edu.tw