Assignment 4:

Stereo Matching

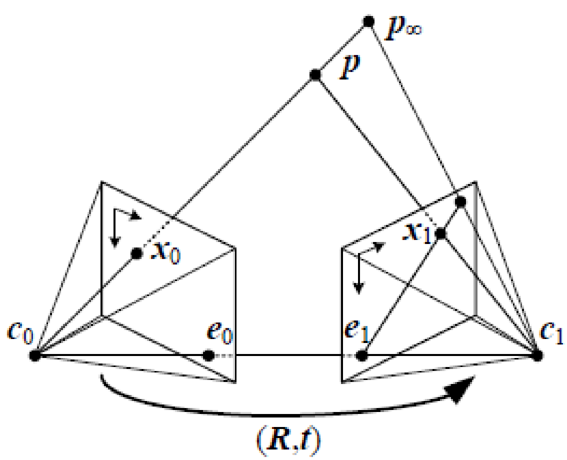
Computer Vision

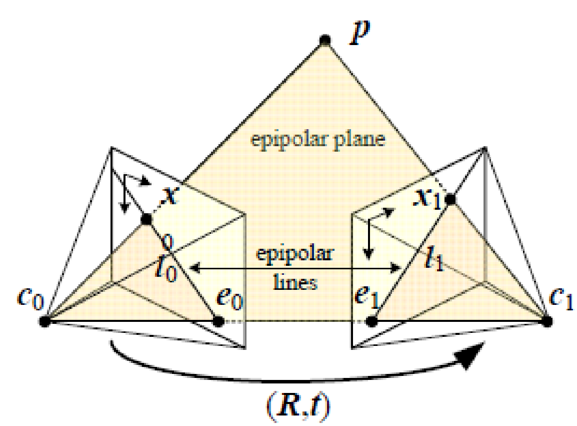
National Taiwan University

Spring 2022

Stereo Matching

- For pixel x_0 in one image, where is the corresponding point x_1 in another image?
  - Stereo: two or more input views
- Based on the epipolar geometry, corresponding points lie on the epipolar lines (next lectures…)
  - A matching problem



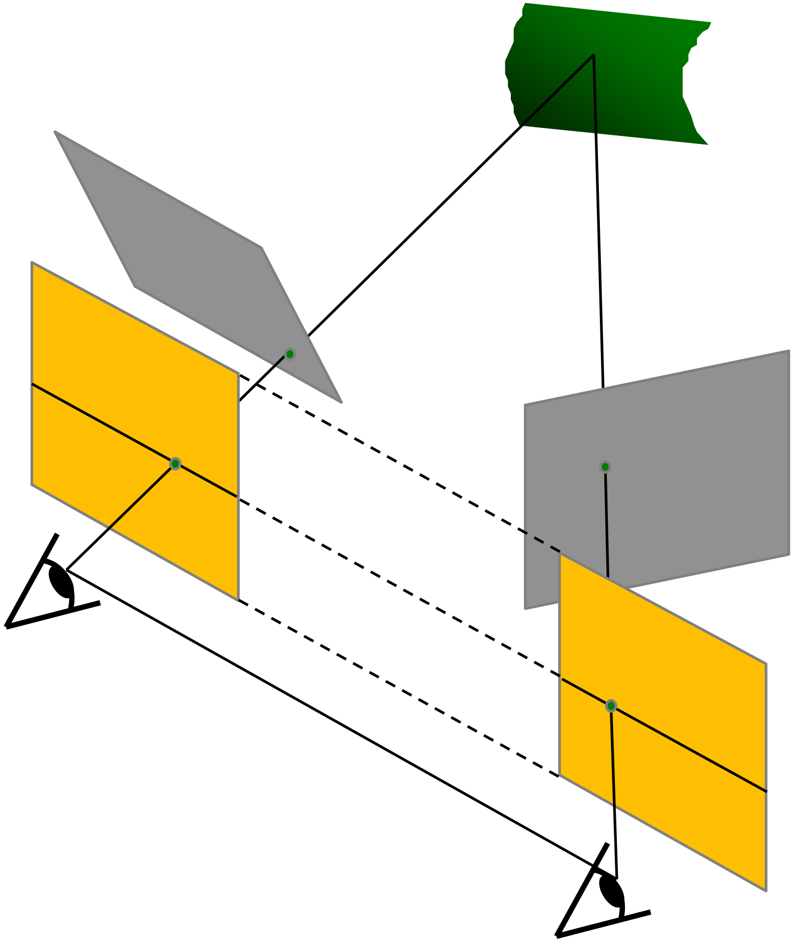


Components of a Stereo Vision System

- Calibrated cameras
- Rectified images
- Compute disparity
- Estimate depth

Image Rectification

- Re-project image planes onto a common plane parallel to the line between optical centers.
- Pixel motion is horizontal after this transformation.



(The testing images in this assignment have been rectified.)

Image Rectification

- [Loop and Zhang 1999]

Original image pair overlaid with several epipolar lines.

Images transformed so that epipolar lines are parallel.

Images rectified so that epipolar lines are horizontal and aligned in vertical.

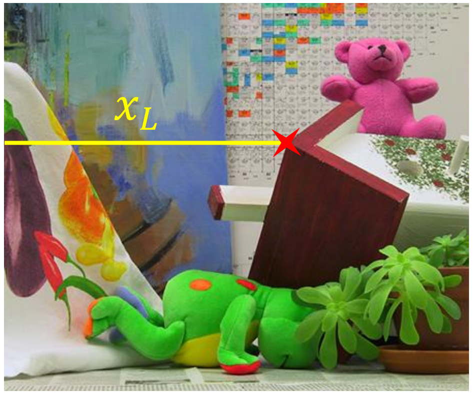
Final rectification that minimizes horizontal distortions. (Shearing)

Loop and Zhang. Computing Rectifying Homographies for Stereo Vision. In CVPR 1999.

Disparity Estimation

- After rectification, stereo matching becomes the disparity estimation problem.
- Disparity = horizontal displacement of corresponding points in the two images
    - Disparity of = $x_L - x_R$

$Image_L$



$Image_R$

- You need to implement Disparity Estimation in hw4.

Disparity Estimation

- “Hello world” algorithm: block matching
  - Consider SSD (Sum of Squared Distance) as matching cost

d	0	1	2	3	…	33	…	59	60
SSD	100	90	88	88	…	12	…	77	85

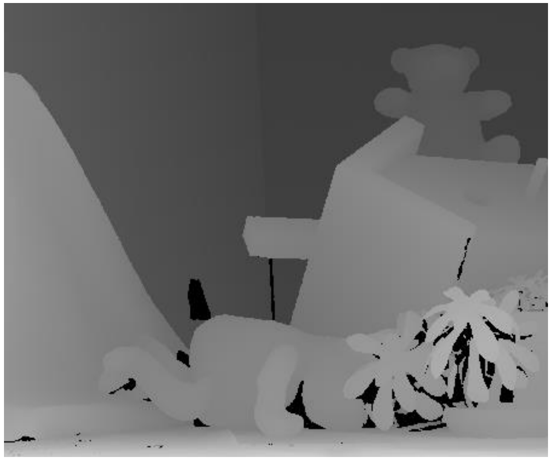
Minimal cost [Winner-take-all (WTA)]

$Image_L$

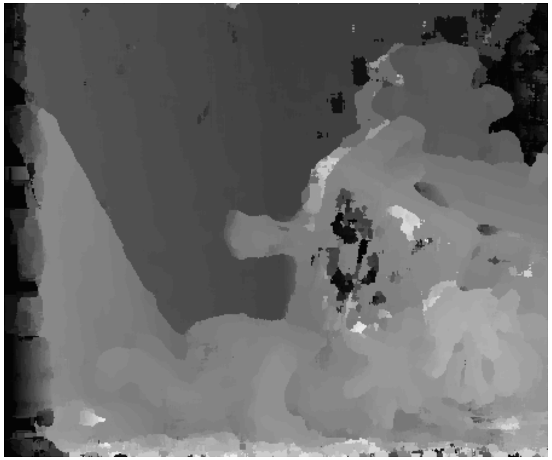
$Image_R$

Disparity Estimation

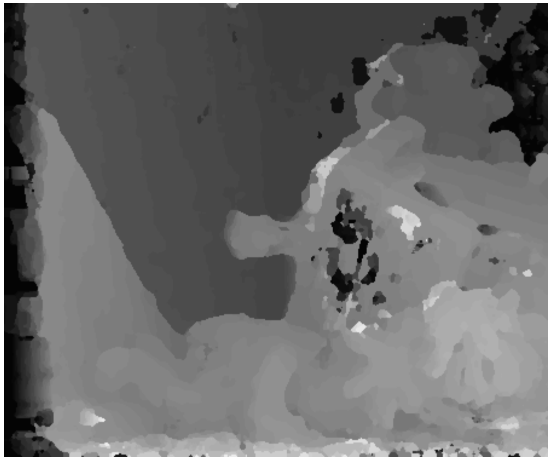
- Block matching result



Ground-truth



Window 5x5



After 3x3 median filter

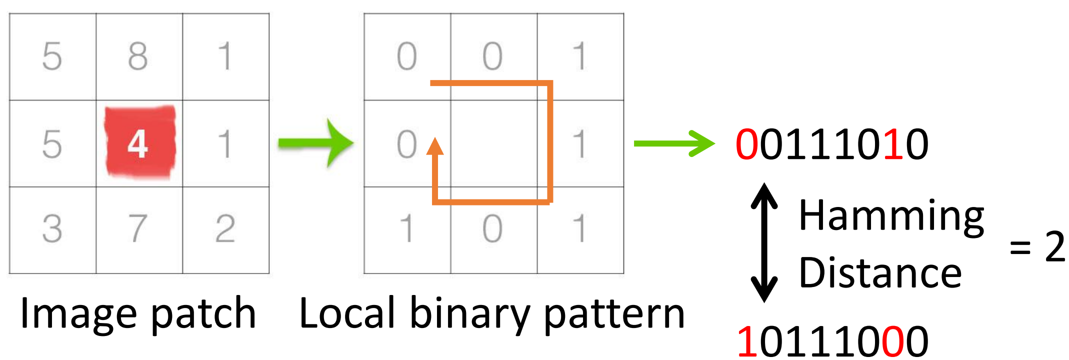
Typical Stereo Pipeline

- It consists of 4 steps:
  - Cost computation
  - Cost aggregation
  - Disparity optimization
  - Disparity refinement

D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV 2002.

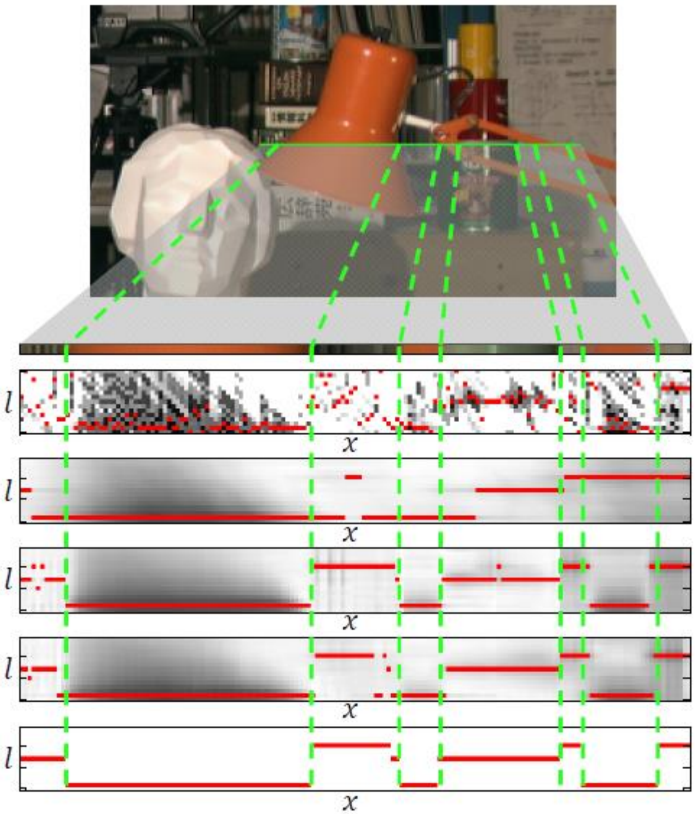
Step 1: Cost Computation

- Matching cost
  - Squared difference (SD): $(I_p - I_q)^2$
  - Absolute difference (AD): $|I_p - I_q|$
  - Normalized cross-correlation (NCC)
  - Zero-mean NCC (ZNCC)
  - Hierarchical mutual information (HMI)
  - Census cost
  - Truncated cost



Step 2: Cost Aggregation

- Illustration of the matching cost



Raw cost

Smoothed by box filter

Smoothed by bilateral filter

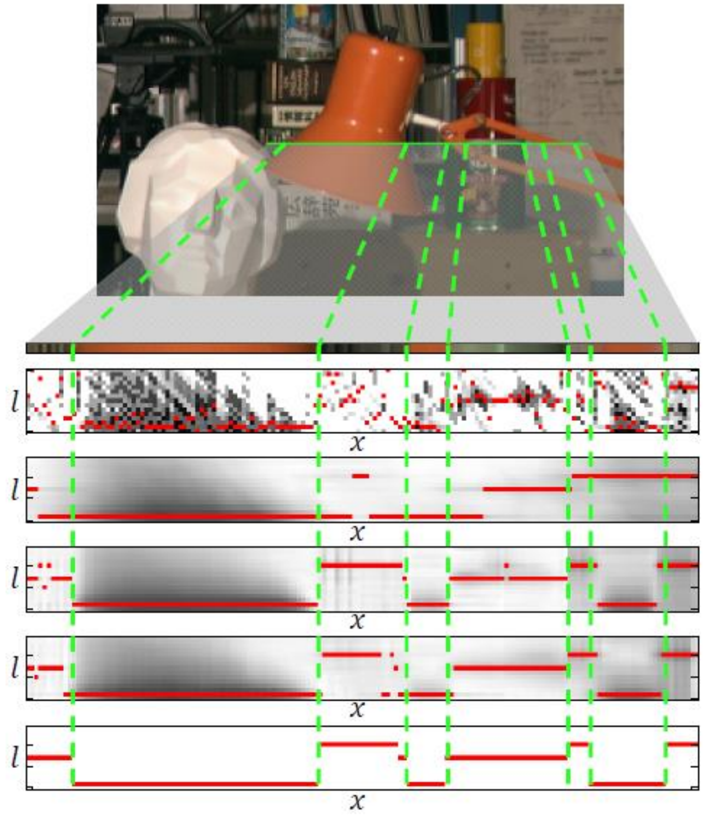
Smoothed by guided filter

Ground-truth

The disparity with min cost at certain location

Step 3: Disparity optimization

- Winner-take-all



Step 4: Disparity Refinement

- Left-right consistency check
  - Compute disparity map D_L for left image
  - Compute disparity map D_R for right image
  - Check if $D_L(x, y) = D_R(x - D_L(x, y), y)$
    - If Yes, keep the computed disparity
    - If No, mark hole (invalid disparity)

Note: D_R are only used in this step!!
Only need to keep D_L for the next step.

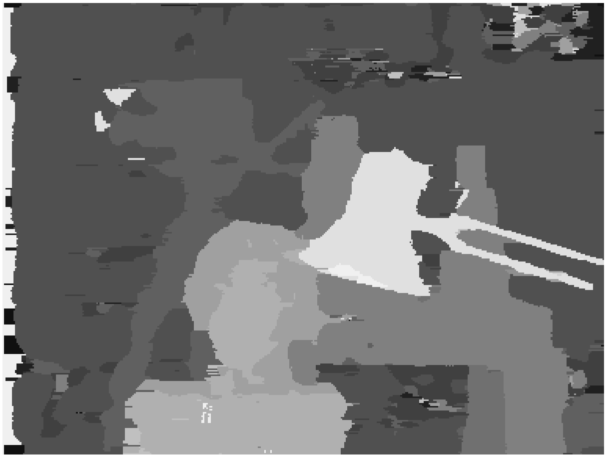
$(x\text{-}D_L(x,y),\ y)$

$(x,\ y)$

Two corresponding positions in images

Step 4: Disparity Refinement

- Hole filling
  - F_L, the disparity map filled by closest valid disparity from left
  - F_R, the disparity map filled by closest valid disparity from right
  - Final filled disparity map $D = \min(F_L, F_R)$ (pixel-wise minimum)
  - Tips: pad maximum disparity for the holes in boundary



Step 4: Disparity Refinement

- Weighted median filtering

(a) left image (b) box agg.+LR check+hole-filling (c) median filtering of (b) (d) weighted median filtering of (b) (e) zoom-in

Ma et al. Constant time weighted median filtering for stereo matching and beyond. In ICCV 2013.

Assignment Description

- computeDisp.py (TODO)

```python
import numpy as np
import cv2.ximgproc as xip


def computeDisp(Il, Ir, max_disp):
    h, w, ch = Il.shape
    labels = np.zeros((h, w), dtype=np.float32)
    Il = Il.astype(np.float32)
    Ir = Ir.astype(np.float32)

    # >>> Cost Computation
    # TODO: Compute matching cost
    # [Tips] Census cost = Local binary pattern -> Hamming distance
    # [Tips] Set costs of out-of-bound pixels = cost of closest valid pixel
    # [Tips] Compute cost both Il to Ir and Ir to Il for later left-right consistency


    # >>> Cost Aggregation
    # TODO: Refine the cost according to nearby costs
    # [Tips] Joint bilateral filter (for the cost of each disparty)


    # >>> Disparity Optimization
    # TODO: Determine disparity based on estimated cost.
    # [Tips] Winner-take-all


    # >>> Disparity Refinement
    # TODO: Do whatever to enhance the disparity map
    # [Tips] Left-right consistency check -> Hole filling -> Weighted median filtering


    return labels.astype(np.uint8)
```

Good News:
you CAN use cv2.ximgproc package with plenty of filtering operations

Maximum possible disparity
(do not need to search the disparity larger than it)

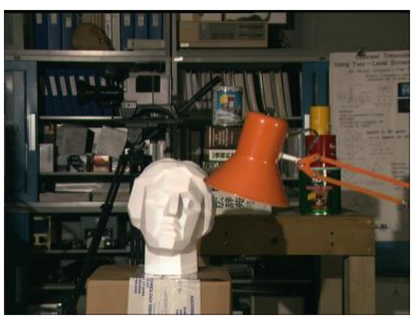
You are not forced or limited to those tips. *But, they are good for you to improve your algorithm.*

CANNOT use deep learning based methods.

Assignment Description

- main.py
  - Read image, execute stereo matching, and visualize disparity map.
  - Usage: *python3 main.py --image {input_image}*
- eval.py (DO NOT EDIT this file)
  - Compute disparity maps of the **left image** for the four standard test pairs from [Middlebury v2]

Tsukuba



with gt

Venus

without gt

Teddy

with gt

Cones

without gt

  - Evaluation metric: bad pixel ratio (error threshold = 1)

Package

- Python: 3.6+
- Python standard library
  - https://docs.python.org/3.6/library/
- Opencv-contrib-python: 3.4.2.16
  - pip3 install opencv-contrib-python==3.4.2.16

Report

- Your student ID, name
- Visualize the disparity map for all 4 testing images.
- Report the bad pixel ratio for 2 testing images with given gt.
- Describe your algorithm in terms of the standard 4-step pipeline.

Submission (1/2)

- Directory architecture:
  + R09876543/
    - **computeDisp.py**
    - **report.pdf**
- Put above files in a directory (named StudentID) and compress the directory into zip file (named StudentID.zip)
  - e.g. R09876543.zip
  - After TAs run “unzip R09876543.zip”, it should generate one directory named “R09876543”.
  - If any of the file format is wrong, you will get zero point.
- Do NOT copy homeworks (code and report) from others
- Submit to NTU COOL
- Deadline: 5/19 23:59
  - Late policy: http://media.ee.ntu.edu.tw/courses/cv/22S/hw/delay_policy.pdf

Submission (2/2)

- If we can not execute your code, you’ll get 0 points. But you’ll have a chance to modify your code.

- Your code has to be finished in 10 mins.
  - Otherwise, you’ll only get 70% points.
  - Intel Core i7-6800K CPU + 128GB RAM

- We will execute your code on Linux system, so make sure your code can be executed on Linux system before submitting homework.

Grading (Total 15%)

- Code: 60% (15% for each testing image)

Score	Tsukuba	Venus	Teddy	Cones
15	< 8	< 5	< 18	< 15
12	>= 8	>= 5	>= 18	>= 15
5	>= 9	>= 7	>= 24	>= 20
0	>= 10	>= 10	>= 30	>= 25

- Report : 30%
- Ranking: 10% (on average score of all testing images)
    - 10%, Top ~30%
    - 7%, Top ~60%
    - 5%, Top ~80%
    - 0%, others

TA information

- Yu-Kai Chen (陳昱愷)

  E-mail: chenyukai@media.ee.ntu.edu.tw

  TA time: Wed. 13:00 - 15:00

  Location: 線上

- Chih-Ting Liu (劉致廷)

  E-mail: jackieliu@media.ee.ntu.edu.tw

  Location: 線上