3D Data Processing

Camera calibration

Hyoseok Hwang

Today

- Camera calibration
  - Display pattern images
  - Estimate intrinsic parameter of your phone
- 3D rendering
  - Open3D
  - Rendering an image on 3D space

Goal

- Prepare check board images
- Display on your notebook or tablet (pattern.png)
  - Check all rectangles are squares
  - Measure length of each rectangle edge

Process

- Capture two images
- Get K, extrinsic parameter (between two cameras)
- (Undistortion)
- Rectification
- Stereo matching
- Generate Point Clouds
- Visualization

Capture images

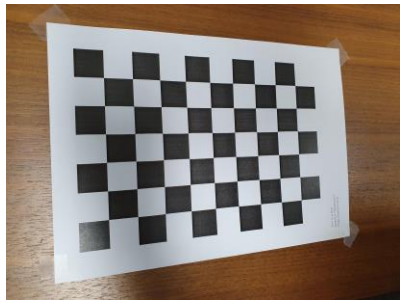
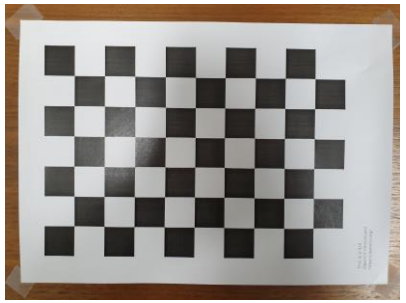
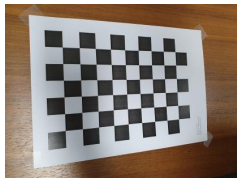
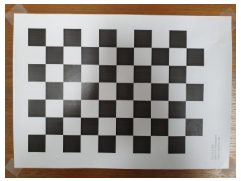
- Capture stereo image (left and right)
- Example

Capture images

- Move the camera horizontally
- Don't move too much.
  - This is because there are fewer overlapping parts.
  - Within 10cm is reasonable.
- If the pattern image is down-sampled, the stereo image must be resize to the same size.
- Save two images

Get K, extrinsic parameter (between two cameras)

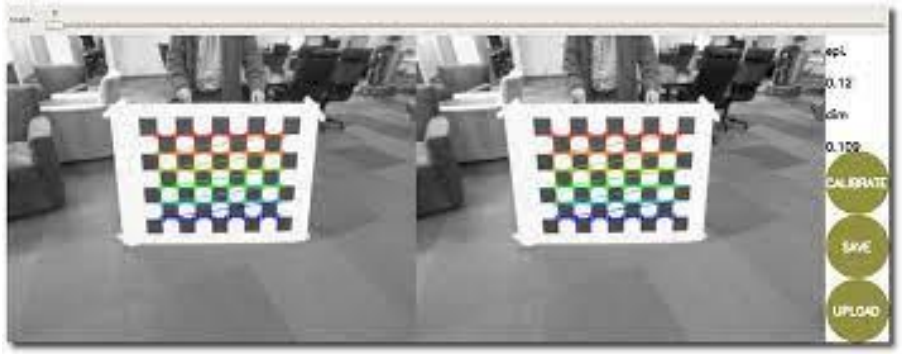
- Camera intrinsic parameter
  - Get your intrinsic parameter using previous practice
  - NOTE, If you want to use a smaller image, recalibrate by reducing the size of the images that captured the pattern.

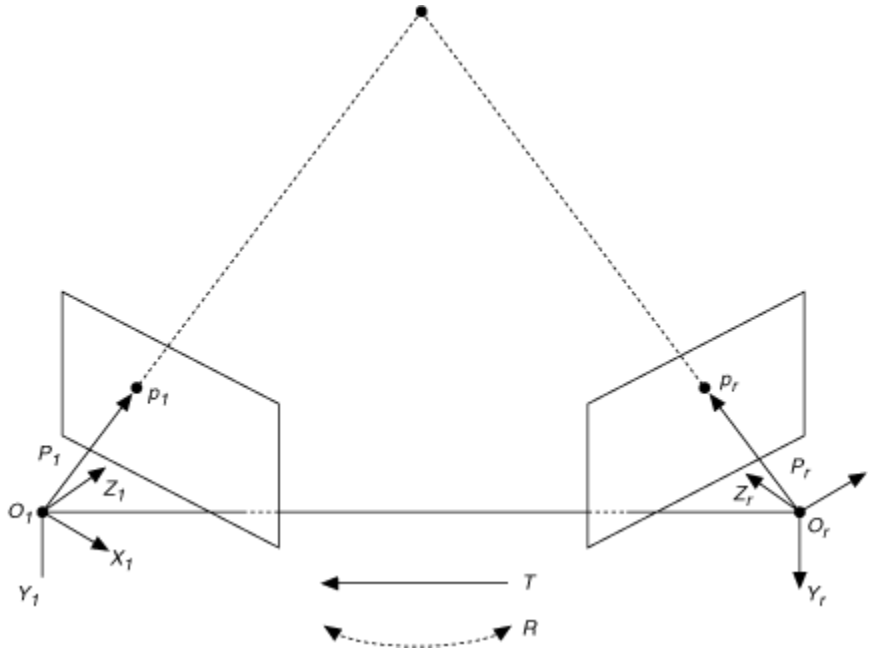





K ≠ K

Get K, extrinsic parameter (between two cameras)

- Extrinsic parameter
  - In principle, stereo images should be calibrated → impossible
  - We make extrinsic parameter by approximation
    - No rotation
    - Translation to x-axis only
    - $\begin{bmatrix} 1 & 0 & 0 & Move\ to\ Xmm \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$





(Undistortion)

- Undistortion
  - Projective geometry and Epipolar geometry assumed linear operation
  - The image needs to be restored to a pinhole model that removes the effect of the lens.
  - Rectification does this in parallel, so you don't have to do it in advance.

Python:

cv.undistort(src, cameraMatrix, distCoeffs[, dst[, newCameraMatrix]]) -> dst

Parameters

src	Input (distorted) image.
dst	Output (corrected) image that has the same size and type as src .
cameraMatrix	Input camera matrix $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$.
distCoeffs	Input vector of distortion coefficients $(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]])$ of 4, 5, 8, 12 or 14 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed.
newCameraMatrix	Camera matrix of the distorted image. By default, it is the same as cameraMatrix but you may additionally scale and shift the result by using a different matrix.

https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga69f2545a8b62a6b0fc2ee060dc30559d

Rectification

- Rectification
  - Make all epipolar lines of images are parallel and have same y-values.
- Two methods
  - Rectification of calibration parameters
    - Images, K, distortion coefficient, Extrinsic(R, T)
    - OpenCV: stereoRectify()
  - Uncalibrated rectification
    - Feature matching, fundamental matrix
    - OpenCV: stereoRectifyUncalibrated()

Rectification

- stereoRectify()

```
cv.stereoRectify( cameraMatrix1, distCoeffs1, cameraMatrix2, distCoeffs2, imageSize, R, T[, R1[, R2[, P1[, P2[, Q[, flags[, alpha[, newImageSize]]]]]]]] )
```

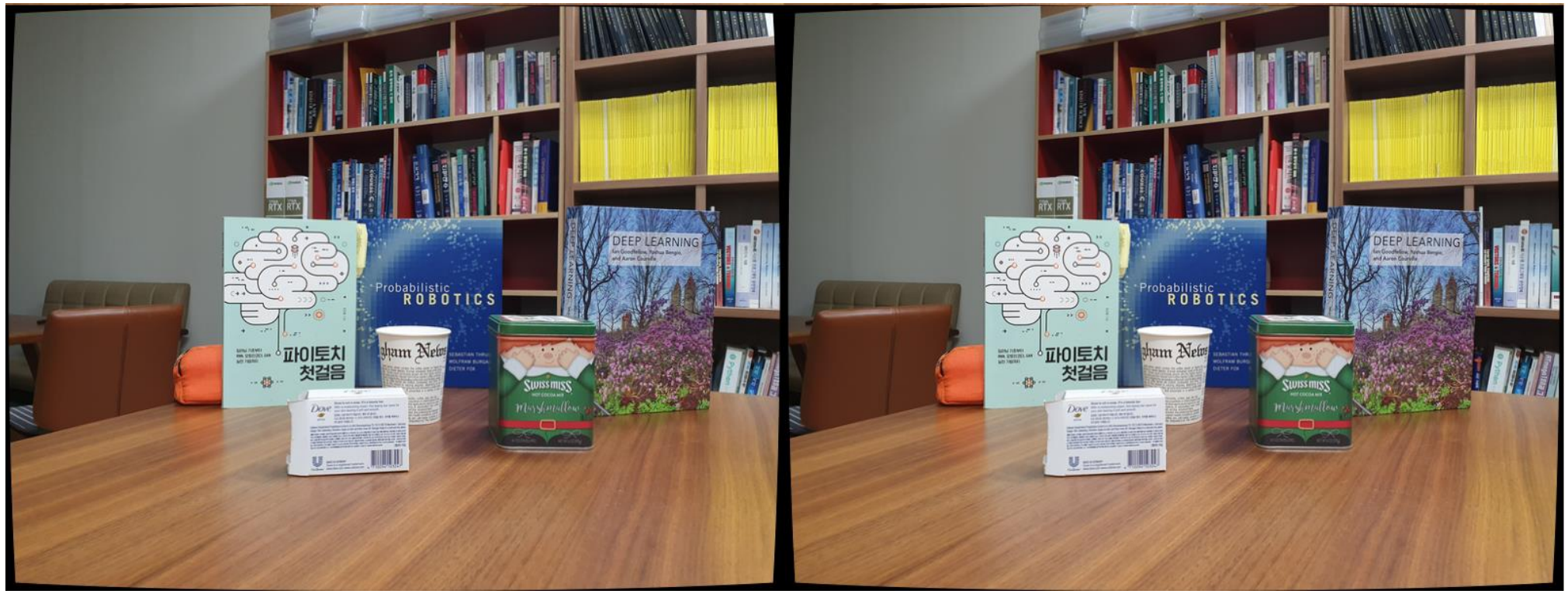
Parameters

cameraMatrix1	First camera intrinsic matrix.
distCoeffs1	First camera distortion parameters.
cameraMatrix2	Second camera intrinsic matrix.
distCoeffs2	Second camera distortion parameters.
imageSize	Size of the image used for stereo calibration.
R	Rotation matrix from the coordinate system of the first camera to the second camera, see **stereoCalibrate**.
T	Translation vector from the coordinate system of the first camera to the second camera, see **stereoCalibrate**.
R1	Output 3x3 rectification transform (rotation matrix) for the first camera. This matrix brings points given in the unrectified first camera's coordinate system to points in the rectified first camera's coordinate system. In more technical terms, it performs a change of basis from the unrectified first camera's coordinate system to the rectified first camera's coordinate system.
R2	Output 3x3 rectification transform (rotation matrix) for the second camera. This matrix brings points given in the unrectified second camera's coordinate system to points in the rectified second camera's coordinate system. In more technical terms, it performs a change of basis from the unrectified second camera's coordinate system to the rectified second camera's coordinate system.

https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga617b1685d4059c6040827800e72ad2b6

Rectification

- stereoRectify() result



Rectification

- Stereo matching
  - Dense matching: template matching
  - Convert rectified images to grayscale
  - OpenCV: stereoBM
    - Create stereoBM instance first:
      - stereo = cv2.StereoBM_create(numDisparities=00, blockSize=00)
      - numDisparities = Maximum distance
      - blockSize = template size
    - Compute disparity
      - disparity = stereo.compute(img1,img2)
      - Each pixel value is represented as int16

Rectification

- Stereo matching result

reference

Generate Point Clouds

- Disparity map is corresponding to left image
- For all pixels of left image, compute 3D position except
  - Disparity less than 0
  - Pixel RGB value is [0,0,0]
  - Z is more farther than threshold
- 3D position of each pixel,
  - $Z = \frac{Bf}{D(u,v)}$
  - $X = \frac{(u-c_x)Z}{f_x}$
  - $Y = \frac{(v-c_y)Z}{fy}$
- Set color using RGB
- Utilize the practice materials from last time (calibration).

Back projection

- Set direction

$$\begin{bmatrix} ku \\ kv \\ k \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$u = \frac{ku}{k} = \frac{f_x X + c_x Z}{Z} \qquad v = \frac{kv}{k} = \frac{f_y X + c_y Z}{Z}$$

$$f_x X + c_x Z = uZ \qquad f_y Y + c_x Z = vZ$$

$$X = \frac{(u - c_x)Z}{f_x} \qquad Y = \frac{(v - c_y)Z}{f_y}$$

Visualization

Homework2

- Stereo dense matching
  - 수업 자료를 활용하여, stereo matching을 수행하는 코드 작성
  - 본인의 카메라와 카메라로 촬영한 사진을 이용할 것
  - OpenCV, Open3D 사용 가능

- 제출물
  - Source code (ipynb)
    - 소스 코드 실행시 다음이 반드시 display 되도록 함 (채점 기준)
    - Rectification 결과
    - Stereo matching 결과 (Depthmap)
    - 3D point clouds Visualization
  - Stereo images (RGB 사진으로 이름을 left.png, right.png 로 할 것)
  - 보고서: 실행 방식, 결과 첨부

Homework2

- 채점 기준
  - 제출: 20
  - 제출한 소스코드 실행 (60)
    - Rectification 수행 : 20 (left, right 모두 display)
    - Displarity map 수행: 20
    - 3D Visualization 수행: 20
    - 위의 세 항목은 정확하게 수행이 되지 않는 경우 최대 10점씩 감점
    - 정확하게 수행되지 않는 경우
      - 이미지에 아무 것도 없는 경우
      - 객체와 다른 depth 생성되는 경우
      - 객체와 다른 3D visualization 되는 경우
  - 보고서 제출: 20
    - 실행 방법 및 결과 캡쳐하여 정리
    - 각 방식에 대한 설명 추가 시 최대 추가점수 10점

Homework2

- 제출기한
  - 2023년 5월 19일 11:59분
  - Delay penalty: -1점 / 1분

- 제출처
  - E-campus: 과제 및 평가 > HW2 - Stereo matching

- 제출물을 압축파일로 업로드

Thank you