로봇센서 및 데이터 중간고사 Report

담당교수 : 황효석 교수님

과목 : 로봇센서데이터처리

2018102094_소프트웨어융합학과_김대호

목차

1. 머리말

2. 사전 지식

2-1) 카메라 지식

2-2) Coordinate

3. Camera Calibration

3-1) Camera Calibrate

3-2) Intrinsic Parameter

3-3) Extrinsic Parameter

4. Disparity Map

4-1) Disparity

4-2) Stereo matching

5. 3D Reconstruction

5-1) Disparity Map을 이용한 Depth 연산

5-2) Intrinsic 변수를 이용한 X, Y, Z 복원

5-3) Point cloud 표현

6. 꼬리말

6-1) 과제를 마치며

6-2) 참고문헌

1. 머리말

본 보고서는 로봇센서데이터처리 과목 중간고사 대체과제에 사용한 코드의 이론적 배경을 바탕으로 작성한 코드의 개요를 설명하는 글이다. 제시된 문제 (Camera Calibration, Stereo matching, 3D Reconstruction, Visualization) 순서를 따라 본 보고서의 내용을 구성하였으며 이에 필요한 관련 지식을 먼저 설명하고 본 내용으로 들어가고자 한다. 본 보고서의 내용은 로봇센서데이터처리의 수업을 기반으로 한다.

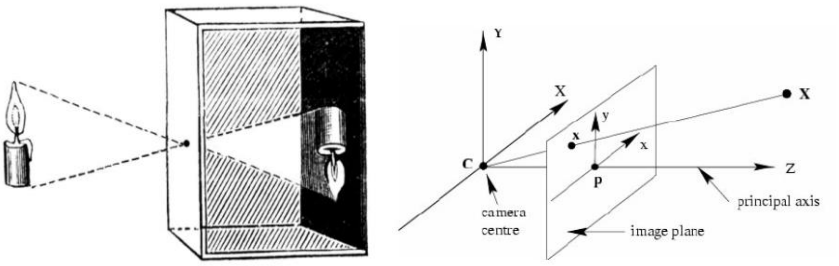
2. 사전 지식

본 시험문제를 풀기위해 문제에서 직접 등장하지는 않았지만 알고 있어야 하는 용어들을 미리 정리하고자 한다.

2-1) 카메라 지식

공간상에 존재하는 물체를 카메라로 촬영한다는 것은 3D 공간상의 물체에 반사된 빛이 카메라 렌즈를 통과하여 카메라 센서에 빛을 주는 과정을 수행하는 것이다.

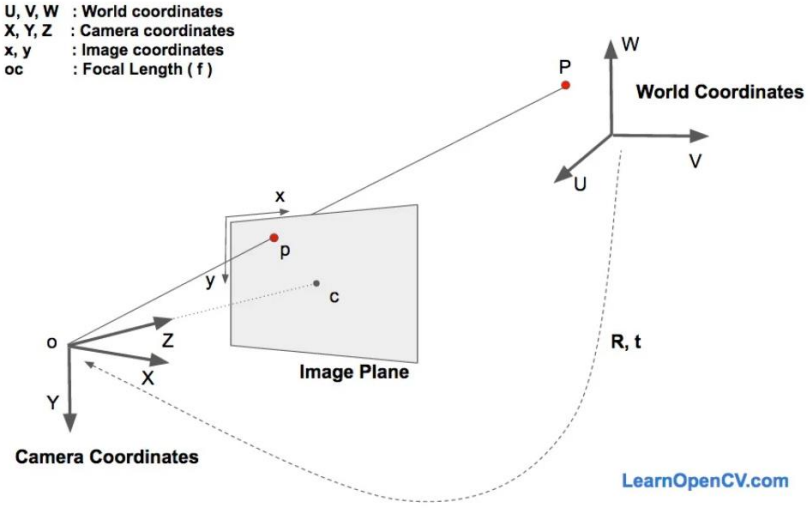
이때 카메라 렌즈의 중심을 통과하는 빛이 핀홀 구멍을 중심으로 점대칭 되어 상이 뒤집혀 나타나는 카메라 모델을 핀홀 카메라 모델이라 한다. 핀홀 카메라 모델을 사용할 경우 뒤집혀진 상을 다시 뒤집기 위해 핀홀을 중심으로 다시 상을 뒤집어 나타낸다. 이때 상이 맺히는 공간을 Image Plane이라 한다.



<좌. Pinhole camera 우. Image plane>

2-2) Coordinate

3D공간에 존재하는 물체를 2D인 Image plane에 나타내기 위해서는 좌표 공간을 동일하게 대응시켜야 한다. 이때 World Coordinate를 수행하여 3D 공간의 좌표계를 Camera Coordinate 좌표계와 동일선상으로 대응시켜준다. Camera Coordinate 좌표계는 Image plane의 횡으로 이어지는 면을 X 축, 종으로 떨어지는 면을 Y 축, Pinhole 과 Image Plane 에서 수선을 그은 선을 Z 축이다. 이후 Camera Coordinate를 하여 3D 공간의 물체를 Image plane 으로 투영시킨다. Image plane의 원점은 Z축과 Image plane 이 교차하는 점이 되는데 Image plane의 좌측 상단으로 원점을 이동시키는 과정을 Image Coordinate라 한다.



<World Coodinate와 Camera Coordinate>

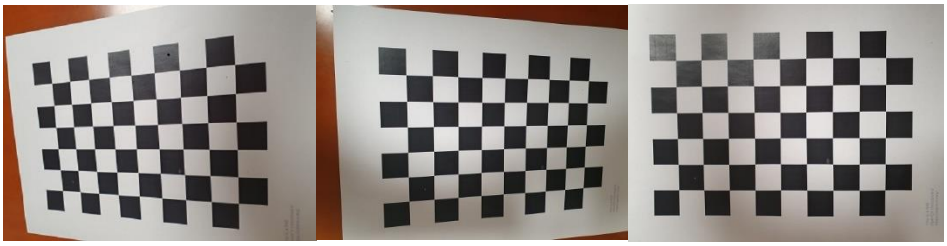
3. Camera calibration

1, 2번 문제는 Camera calibration에 관한 문제이다. 이에 Camera calibration이 무엇인지. 어떤 과정을 통해 구하는 것인지 그렇게 해서 구한 값들이 무슨 원리와 의미를 지니는지 알아보고자 한다.

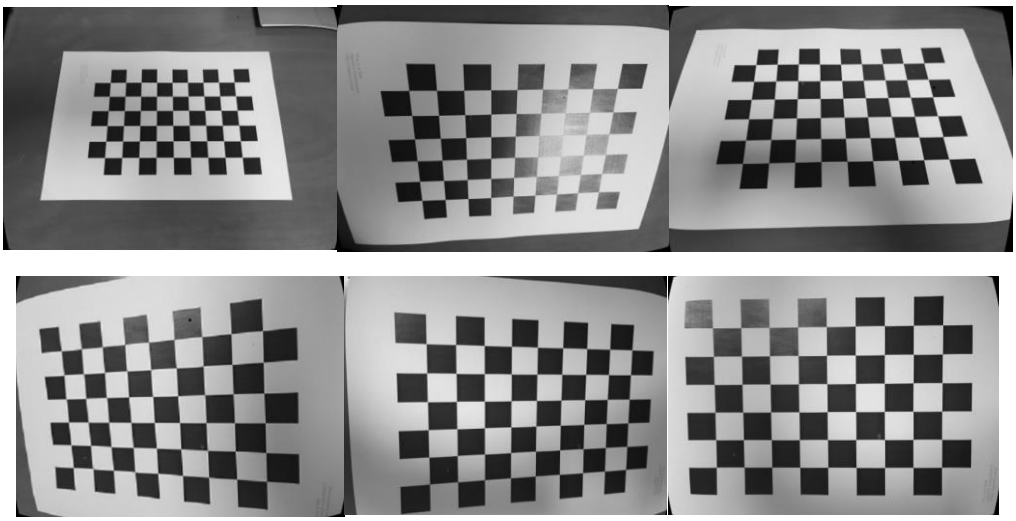
3-1) Camera calibrate

우리가 사용하는 카메라는 렌즈를 이용하여 빛을 수집한다. 이때 렌즈의 가장자리를 지나는 빛과 렌즈의 중심부를 지나는 빛의 굴절율이 다르기 때문에 사진을 찍을 경우 왜곡된 사진이 나타난다. 따라서 왜곡된 사진을 왜곡되지 않은 사진으로 만들기 위해 필요한 수를 구하는 과정이 필요하다. 보정을 위해 사용하는 Matrix를 Project matrix라 하며 Intrinsic parameter와 Extrinsic parameter로 구성되어 있다. 따라서 Camera calibration 을 수행한다는 것은 Intrinsic parameter와 Extrinsic parameter을 찾는 과정이다.

이를 구하기 위해선 먼저 문제에서 제공된 Calibration_pattern 사진을 다각도로 촬영하여. 본 카메라 사진의 왜곡 계수를 찾아야 한다. openCV에서는 체스 무늬 패턴을 인식하여 이를 찾아주는 메서드 findChessboardCorners를 제공하고 있다. 이후 찾은 값을 바탕으로 메서드 opencv 의 calibrateCamera 메서드를 수행하면 왜곡계수와, Intrinsic parameter, Extrinsic parameter를 알아낼 수 있다.



<Calibration_pattern의 다각도 촬영>



<Calibration_patternd의 보정 결과>

<Stereo image L R 보정결과>

3-2) Intrinsic parameter

Project matrix는 다음의 matrix 행렬식으로 표현된다.

$$\begin{bmatrix} s_x f & \beta & c_x \\ 0 & s_y f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

<Project matrix>

이때 앞의 (3 x 3) 매트릭스를 Intrinsic parameter라 한다. Intrinsic parameter은 카메라 외부 공간속의 world coordinate의 (X, Y ,Z) 값을 Image plane으로 투영시켜주는 매트릭스이다.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

<Intrinsic Parameter>

3-3) Extrinsic parameter

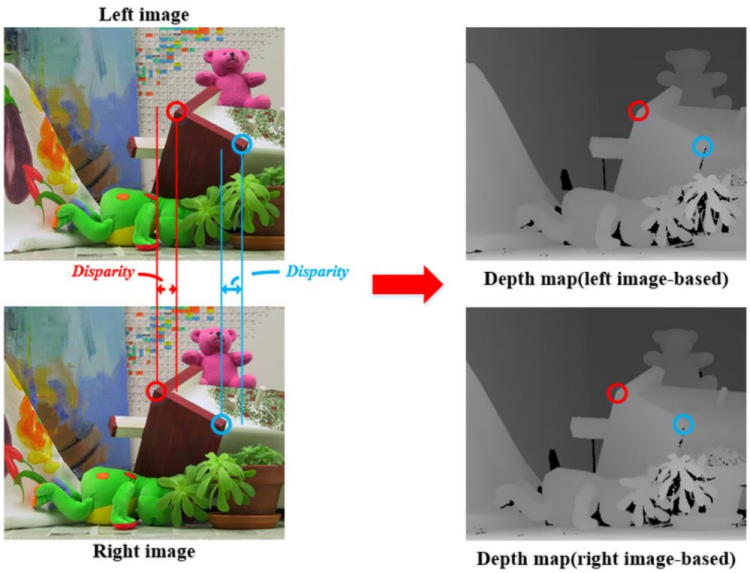
Rotation matrix 와 Translation vector로 구성된 행렬이다. Extrinsic parameter는 World coordinate의 좌표가 Image coordinate와 대응이 불가능한 경우 이를 보정해주는 행렬이다.

4. Disparity Map

3번 문제는 Disparity map을 구하는 문제이다. 이에 Disparity에 대한 이해와 그 과정에서 사용되는 Stereo matching에 대한 이해가 필요하다.

4-1) Disparity

같은 물체를 수직축은 같고 수평선으로만 떨어트린 두 카메라로 촬영하여 두 사진을 비교해보면, 카메라와 가까이 있는 물체가 더 큰 좌우폭을 가지게 된다. Disparity란 스테레오 사진에서 수평선상으로 같은 점을 대응시켜서 떨어진 정도를 나타낸 값이다.
Disparity map 이란 모든 값을 비교하여 좌우 사진의 Disparity 값을 Gray Scale로 나타낸 Image를 말한다.

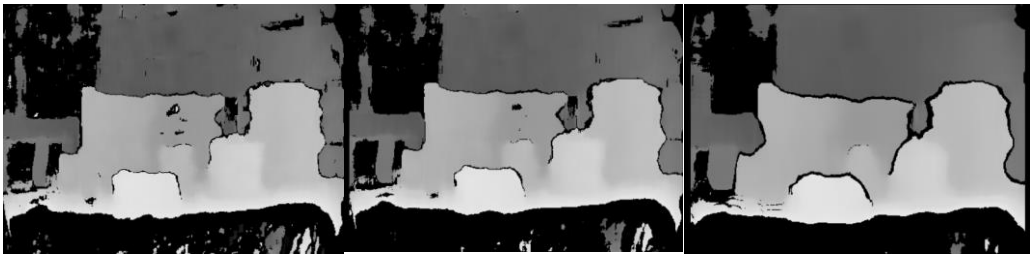


<Stereo matching으로 Depth map 생성 >

4-2) Stereo matching

Disparity를 구하기 위해 Stereo 사진을 완전히 같은 위치로 만들어 주는 Rectification 과정을 수행한다. 이후 Dense matching을 해주기 위해 patch를 만들어 matching 한다.

OpenCv에서는 StereoBM_create 라는 메서드를 제공하여 BlockSize 라는 입력변수에 크기를 주어 patch 사이즈를 조정할 수 있게 해준다. Block size가 크다면 smooth한 Disparity map을 얻을 수 있다. 이후 compute 메서드를 수행해주어서 좌우 값을 matching 시켜 disparity map을 얻을 수 있다.



<좌. Blocksize 15, 중. Blocksize 19, 우. Block size 33>

이렇게 얻은 Disparity map엔 노이즈와 3D Reconstruction을 수행할 때 불필요 한 값들이 생겨나기 때문에 Disparity map 의 min값과 max값을 조정해주어야 한다.

```
for a in np.arange(h):
    for b in np.arange(w):
        if (disp8[a][b] <= disp_min or disp8[a][b] >= disp_max ) :
            disp8[a][b] = 0
```

<Disparity map 최대 최소값 지정>

5. 3D Reconstruction

4번 문제를 풀기 위해서 Image plane의 값들을 Re-Project 해주어야 한다. 즉 Camera Coordinate 해주어 3D 물체를 Project matrix를 이용하여 Image plane의 2D 값으로 변환시킨 과정을 역으로 진행해야 한다.

5-1) Disparity Map을 이용한 Depth 연산

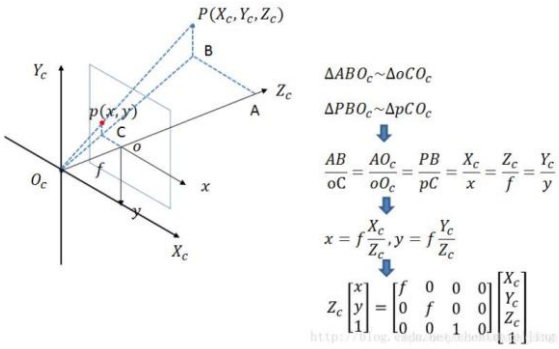
위의 과정의 시행하기에 필요한 값으로 Depth 값이 필요하다. 이는 Disparity map을 이용해 구할 수 있다. Disparity Map에서는 물체가 가까이 있을수록 높은 값을 가지는데 Depth Map 은 정반대로 물체가 가까이 있을수록 낮은 값을 가진다. 즉 Gray Scale로 표현된 두 Map의 같은 픽셀의 값을 더하면 255이 되도록 만들 수 있다. 이 점을 활용하면 Disparity map을 이용하여 Depth map을 구해줄 수 있다.

```
h, w = disp8.shape
depth = disp8
for i in np.arange(h):
    for j in np.arange(w):
        depth[i][j] = 255 - disp8[i][j]
```

<depth 생성 코드>

5-2) Intrinsic parameter를 이용한 X, Y, Z 복원

2D로 투영하는 과정을 역으로 하기 위해서 가장 마지막 과정인 Image coordinate과정부터 역으로 시행해야 한다. 즉 Image plane의 X, Y 값을 World coordinate 좌표계의 X, Y, Z 값으로 복원시켜줘야 한다. 이를 위해 Image coordinate를 시행하면서 움직인 원점을 이동시켜야 한다. 움직인 값 Cx, Cy은 Intrinsic parameter를 인덱싱하여 얻을 수 있다. 이후 구하고자 하는 World coordinate의 X 값을 a, Image plane 의 한 점의 인덱스를 (u, v), Intrinsic parameter를 인덱싱하여 구한 fx 값을 World coordinate와 image coordinate에 대응시키면 fx : (u - Cx) = depth : a 의 비율을 가지게 된다. 마찬가지의 방법으로 World coordinate Y 값을 구하면 World Coordinate (X, Y, Z)의 값을 구할 수 있다.




```
fx = K[0][0]
fy = K[1][1]
cx = K[0][2]
cy = K[1][2]

pc_points = np.zeros((h * w, 3))
i = 0
for v in range(h):
    for u in range(w):
        x = (u - cx) * depth[v, u] / fx
        y = (v - cy) * depth[v, u] / fy
        z = depth[v, u]
        pc_points[i] = (x, y, z)
        i = i + 1
```

<좌. World, Camera coordinate 닮음비, 우. 닮음비 적용 코드>

5-3) Point cloud 표현

완전한 3D Reconstruction을 위해 2D Image의 RGB 값도 수정해야 하는데 3D Reconstruction을 위해 사용하는 Open3d는 0 ~1 사이의 값을 입력변수로 사용하기에 RGB 값에 255를 나눠줘야 한다.

최종적으로 Disparity map 에서 0으로 맞춰준 불필요하는 값을 Point cloud 에서 제거하는 과정을 수행하여 다음과 같은 결과물을 얻을 수 있다.

```
rgb = cv2.cvtColor(imgLU, cv2.COLOR_BGR2RGB)
rgb = rgb.astype(np.float32)/255.0
rgb = rgb.reshape(h*w, 3)
pc_color = rgb
```

```
mask = pc_points[:,2] < 255
pc_points = pc_points[mask]
pc_color = pc_color[mask]
```

<좌. RGB 컬러 복원 코드, 우. Filter 코드>

<3D Reconstruction 결과>

6. 꼬리말

6-1) 과제를 마치며

이번 과제는 결과적으로 기입한 코드의 줄은 길지 않다. 하지만 사용하려는 메서드의 사용이유와 그에 맞는 입력 변수를 명확하게 알아야 원하는 결과값을 출력할 수 있었다. 3번 Disparity 문제에서 4번 3D Reconstruction 하는 문제가 특히 그러하였다.

Open3d의 Vector3dvecto의 입력 변수의 타입은 Matrix 이지만 OpenCV의 복원값은 Tensor 였기 때문에 차원을 변환시켜야 했다. 또한 Disparity map의 최대 최소 범위를 지정하고 3D Reconstruction 해줬지만 노이즈 값이 생겼는데 이러한 이유는 pc_points를 최종적으로 Filter 해주지 않기 때문이었다. Point cloud의 RGB 값을 입력해주지 않았기 때문에 Visual 해주지 않았기 때문에 출력이 되지 않을 것이라 생각했다. 하지만 Open3d 의 Point cloud는 RGB 값을 지정해주지 않아도 Point 만 존재한다면 3D 로 Visual 해주기 때문에 Filter값을 입력해주어야 했다.

이번 과제에서 한계점으로 내가 직접 찍은 사진으로 Disparity Map 이 나오지 않았다. 이는 내가 카메라를 이동해주면서 2장의 사진을 찍은 결과물이 Rectification이 명확하게 되지 않았기 때문이라 생각한다.

이번 과제에선 2D의 사진을 3D로 변환해줬지만 Real Time으로 영상을 찍으면서 3D로 복원하는 과정도 시도할 수 있겠다.

6-2) 참고문헌

그림 1. Pinhole camera ("https://markellisimagery.com/pinhole-camera")

그림 2. Camera principle ("https://www.researchgate.net/figure/Schematic-view-of-a-pinhole-camera-The-image-plane-is-shown-in-front-of-the-camera_fig2_265190953")

그림 3. Disparity map ("https://vision.middlebury.edu/stereo/")

논문 1. Daniel Scharstein. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms . 1-4. 2002

서적 1. DiegoJohnson https://www.kaggle.com/c/pku-autonomous-driving/discussion/120083 .2020