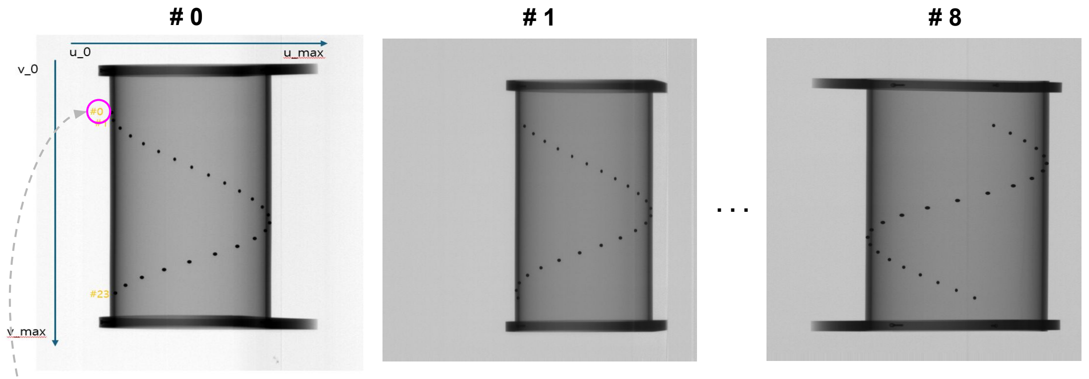
X-Ray 3d Bbox 복원 데모

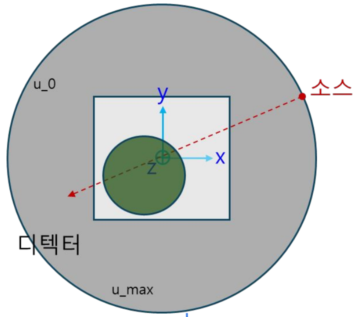
3d object detection

25.04.28

https://docs.google.com/presentation/d/1K-YlyjJVZBeRO8yU0_2Thm0z8kVtE5_Q4oGlCHkZG5k/edit?usp=sharing

Calibration (calibrate.py)





data\calibration

```
├─ 6679
│   ├─ botleft_00.png
│   ├─ botleft_01.png
│   ├─ ...
│   ├─ botleft 08.png
│   ├─ 6679 2d.npy
│   ├─ 6679_3d.npy
│   └─ botleft(6679).txt
```

beads 촬영 geometry-2.pptx

```
6680
6681
6682
총 4Set
```

🟦	INPUT
🟥	OUTPUT

퀀텀 좌표

```
View# Bead# U  V
0  0    213.207   198.891
0  1    213.675   216.175
0  2    222.618   233.605
0  3    238.172   250.856
...                ...
0 23    222.061   589.936
1  0    288.475   199.204
1  1    306.668   216.406
1  2    329.719   233.795
1  3    356.397   250.913
...                ...
8 21    347.864   556.783
8 22    385.248   574.365
8 23    422.463   591.869
```

6679_2d.npy
비드 중심점의 이미지 좌표

2

이미지의 V 값(y)이
3d z축에 해당함!!!

```
-108.2, 89.5,   198.891
-146.0, 84.8,   216.175
           ...
 -67.7,  84.7,  591.869
```

6679_3d.npy
전달받은 "beads 촬영 geometry-2.pptx" 에서
X, Y 값을 가져옴

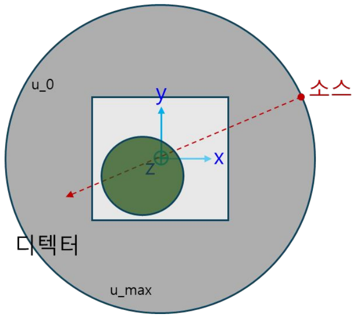
← data/calibration/make_2d_npy.py
botleft(6679).txt 를 읽어서 생성됨

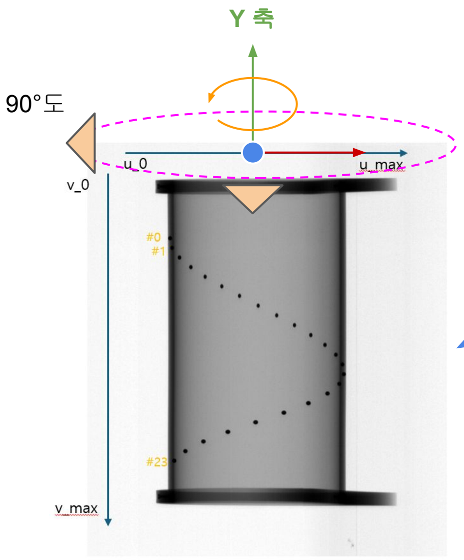
1

비드번호	X(mm)	Y(mm)	비드번호	X(mm)	Y(mm)
#0	-108.2	89.5	#12	-106.9	-203.7
#1	-146.0	84.8	#13	-68.7	-198.7
#2	-181.3	70.0	#14	-33.1	-183.9
#3	-211.4	46.8	#15	-2.7	-160.6
#4	-234.6	16.6	#16	20.4	-130.4
#5	-248.9	-18.9	#17	35.1	-95.0
#6	-253.6	-57.3	#18	40.2	-57.2
#7	-248.6	-94.9	#19	35.4	-19.2
#8	-233.9	-130.4	#20	20.7	16.2
#9	-210.7	-160.3	#21	-2.5	46.6
#10	-180.1	-184.0	#22	-32.7	69.8
#11	-144.5	-198.9	#23	-67.7	84.7

spiralbeads 촬영 geometry-2.pptx

Calibration (calibrate.py)





- 이미지와의 정렬을 위해서는 beads_3d 의 Y, Z 값 flip, X 값 반전이 필요함(원통이 세워짐)
- 카메라는 Y 축 기준으로 회전시킨 후,
- 다시 (X, Z) 평면상에 이동시킴
- Y 축 독립이므로 비드의 X 좌표값으로만 loss 계산

beads_3d = 6679_3d.npy 에서 읽어온 값

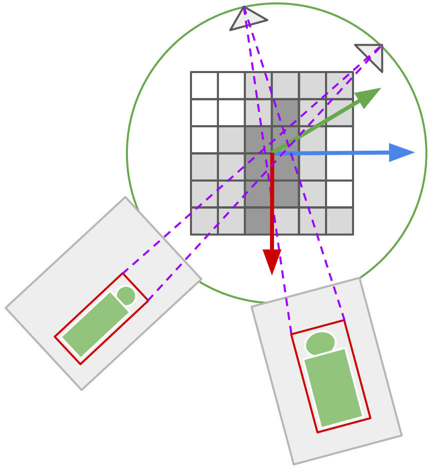
```
// X 축 flip
beads_3d_flipped = beads_3d.clone()
beads_3d_flipped[:, 0] = -beads_3d_flipped[:, 0]

// Y, Z 축 flip
X = beads_3d_flipped[:, 0]
Y = beads_3d_flipped[:, 2]
Z = beads_3d_flipped[:, 1]
```

```
// Y 축 회전(theta) 후, (X, Z) 평면 이동(Tx, Tz)
X_ = X * torch.cos(theta) + Z * torch.sin(theta) + Tx
Z_ = Z * torch.cos(theta) - X * torch.sin(theta) + Tz

// Projection 된 x 좌표와의 loss 계산
u = Cx + (DSD * X_) / Z_
error = torch.nn.functional.mse_loss(u, beads_2d[:,0])
```

```
// Tx, Tz, theta, DSD_np 순서 --> shape = (9, 4)
[[-6.33441865e-01,  1.05138281e+03, -1.82991852e+02, 1.10000000e+03],
 [-1.87607169e+00,  1.05566675e+03, -1.13830719e+02, 1.10000000e+03],
 [-2.32999110e+00,  1.07070251e+03, -4.32817192e+01, 1.10000000e+03],
 [-7.74644375e-01,  1.06971704e+03,  2.55007267e+01, 1.10000000e+03],
 [ 4.20448750e-01,  1.06027283e+03,  9.58136826e+01, 1.10000000e+03],
 [-7.25076199e-01,  1.04909424e+03, -1.48666901e+02, 1.10000000e+03],
 [-2.09228086e+00,  1.06667310e+03, -7.83258438e+01, 1.10000000e+03],
 [ 3.87181304e-02,  1.06869507e+03, -8.68330574e+00, 1.10000000e+03],
 [ 1.50599396e+00,  1.06495752e+03,  6.03852539e+01, 1.10000000e+03]]
```

Calibration 최종 결과(50,000 iter) 파일 → output/calibration_results.npy

3d Bbox 복원 (visual_hull.py)



visual hull 수행

객체별 9-view 별 bbox 좌표
shape = (9, 4)

Candidates count 5

```
[[[462. 311. 630. 400.]
  [424. 305. 450. 395.]
  [ 95. 299. 335. 391.]
  [  0. 295. 212. 385.]
  [ 61. 292. 170. 381.]
  [457. 286. 575. 377.]
  [248. 281. 400. 372.]
  [ 16. 276. 269. 367.]
  [ 16. 272. 183. 362.]]

 [[  0.   0.   0.   0.]
  [382. 141. 547. 270.]
  [239. 137. 520. 269.]
  [141. 128. 354. 261.]
  [  0.   0.   0.   0.]
  [436. 118. 522. 252.]
  [315. 119. 551. 248.]
  [178. 111. 450. 245.]
  [133. 105. 254. 238.]]

...

 [[  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [437. 115. 522. 252.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]]]
```

[bboxes_candidates]
shape = (5, 9, 4)

data\

```
├─ raw_voxel
│  ├─ 35778
│  │  ├─ 35778_512x512x645.npy
│  │  └─ 35778_512x512x645.obj
├─ inference_results
│  ├─ 35778
│  │  ├─ 00035778_0_he.json
│  │  ├─ 00035778_0_he.png
│  │  ├─ ...
│  │  ├─ 00035778_8_he.json
│  │  └─ 00035778_8_he.png
```

🟦 INPUT

🟥 OUTPUT

전달받은 .raw 파일로
부터 생성된 npy 파일
(1_raw_to_npy.py)

3d Bbox 복원 (visual_hull.py)

[3d Bbox 복원 결과 확인 방법]

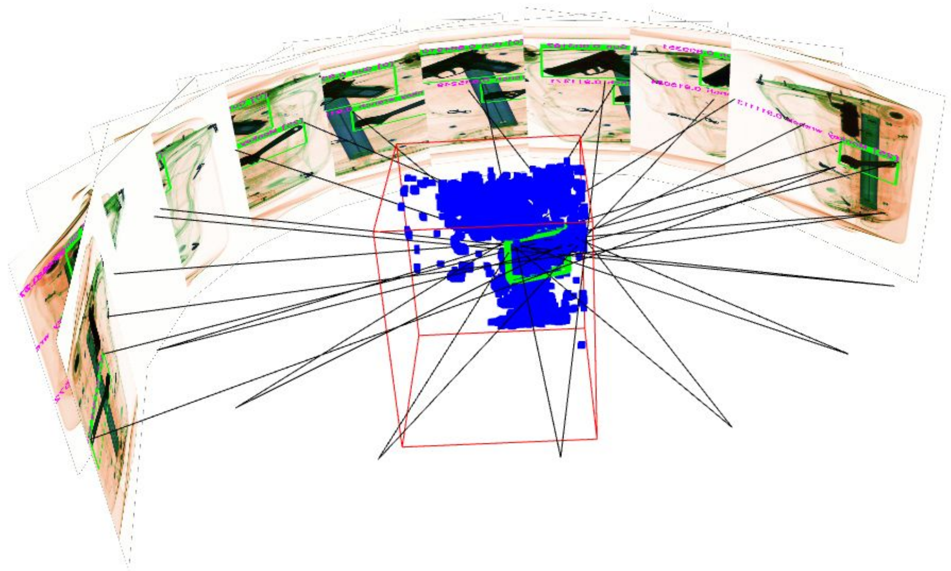
README.md 참고하여 테스트

// 이하 콘솔 출력 메시지

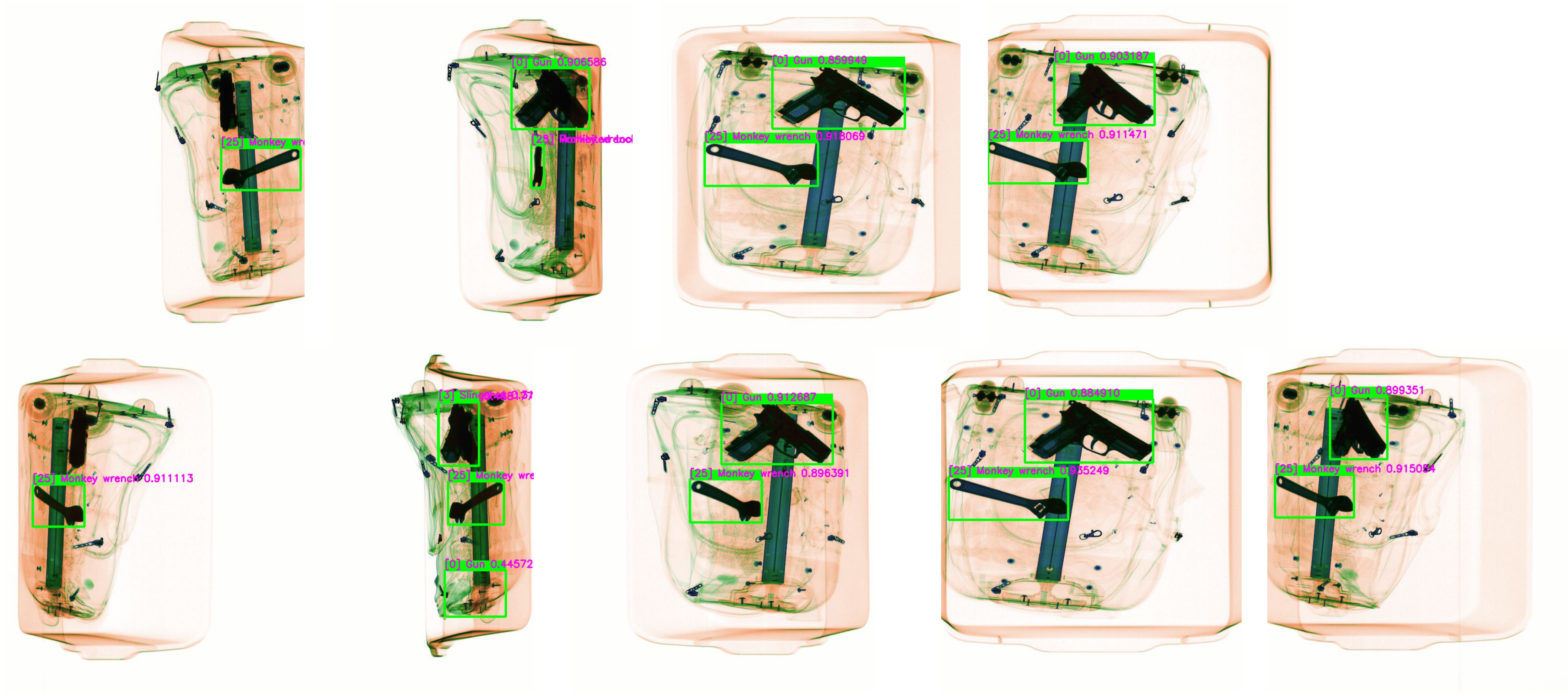
```
Select 3D object index you want to visualize, you have 0 ~ 1 objects
If you want to quit, press 'q'
Index : 0

Object 0 Information:

Class name: Monkey wrench

3D Cuboid Coordinates:

3D Points:
Point 0: (212, 251.8, 170)
Point 1: (203, 251.8, 197)
Point 2: (35, 251.8, 145)
Point 3: (43, 251.8, 118)
Point 4: (212, 386.2, 170)
Point 5: (203, 386.2, 197)
Point 6: (35, 386.2, 145)
Point 7: (43, 386.2, 118)
```

→ e.g, data\bbox3d\35671\Scissors-A.json 에 저장됨

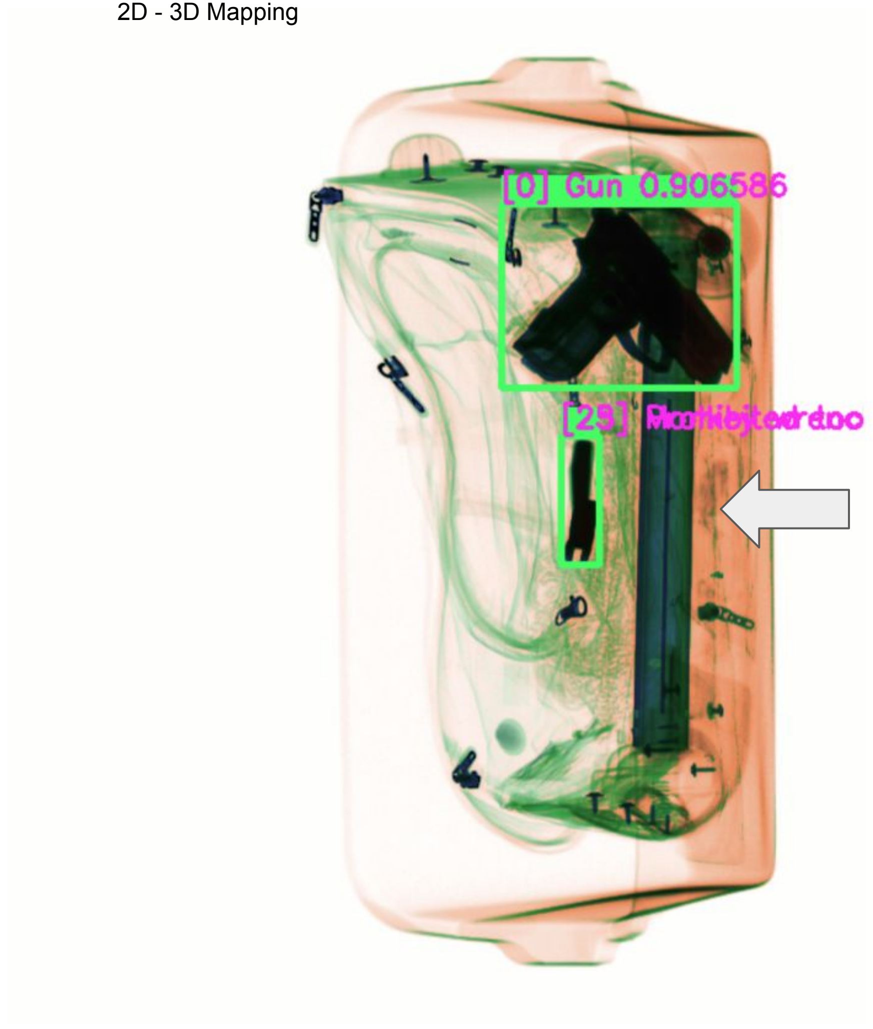
```
You can turn on and off image with key number 0 ~ 8
```

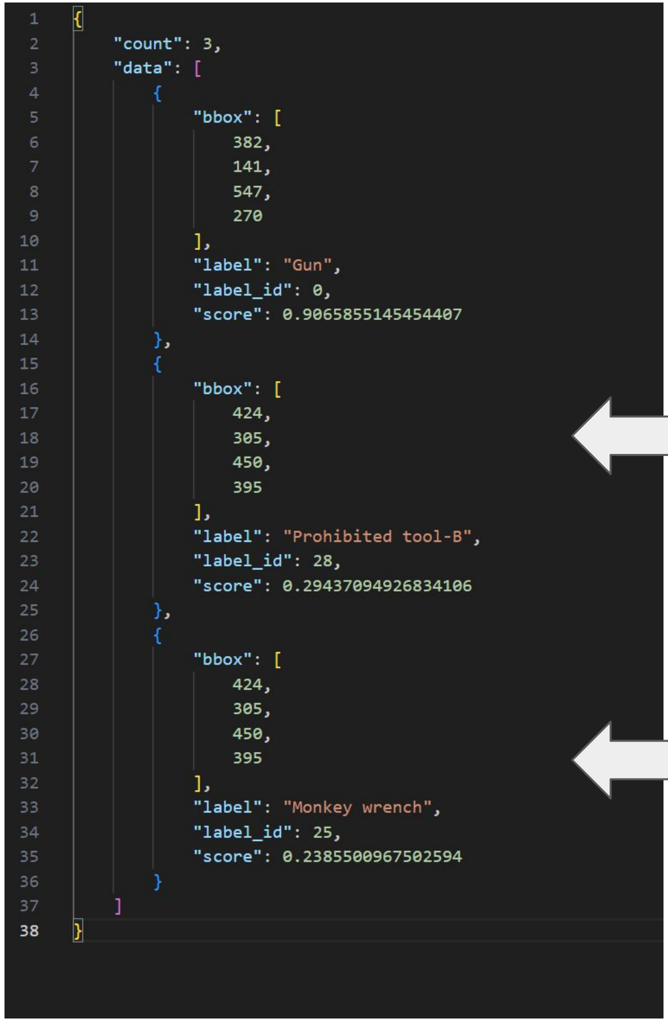


2D - 3D Mapping


[25] Monkey wr
[0] Gun 0.906586
[0] Gun 0.859949
[25] Monkey wrench 0.918069
[0] Gun 0.903187
[25] Monkey wrench 0.911471
[25] Monkey wrench 0.911113
[25] Monkey wre
[0] Gun 0.44572
[0] Gun 0.912687
[25] Monkey wrench 0.896391
[0] Gun 0.884910
[25] Monkey wrench 0.935249
[0] Gun 0.899351
[25] Monkey wrench 0.915054

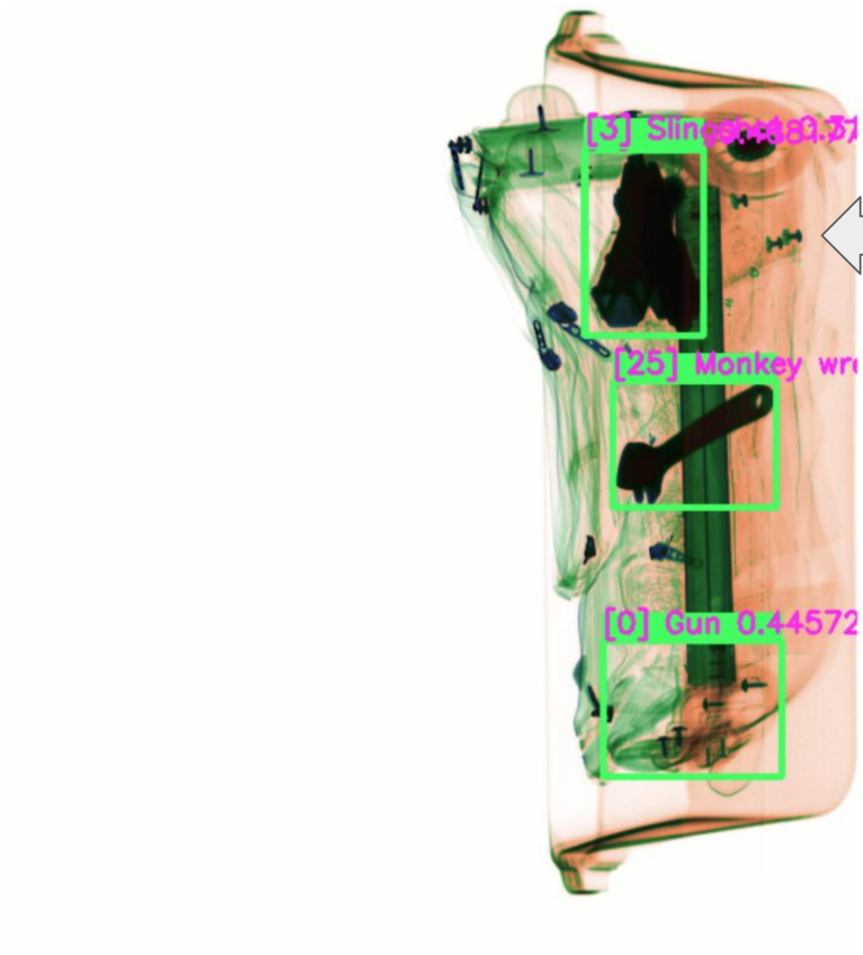
2D - 3D Mapping

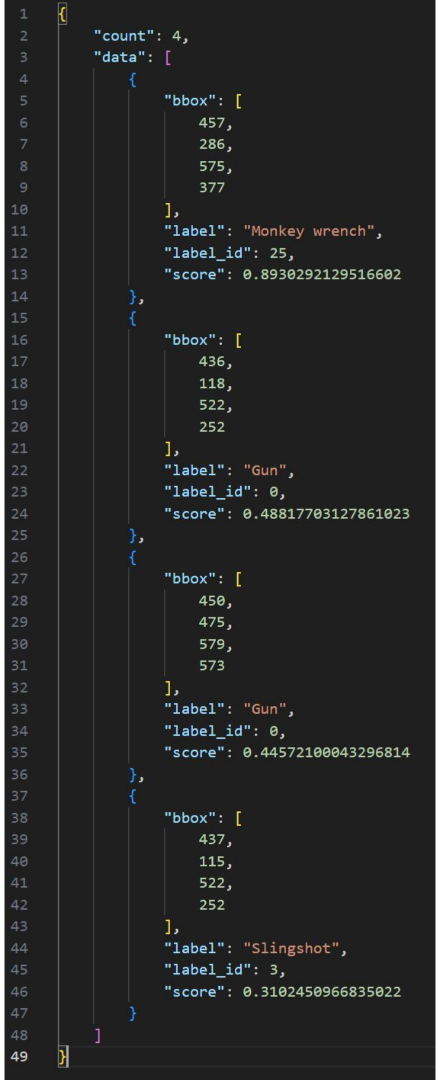




```
{
    "count": 3,
    "data": [
        {
            "bbox": [
                382,
                141,
                547,
                270
            ],
            "label": "Gun",
            "label_id": 0,
            "score": 0.9065855145454407
        },
        {
            "bbox": [
                424,
                305,
                450,
                395
            ],
            "label": "Prohibited tool-B",
            "label_id": 28,
            "score": 0.29437094926834106
        },
        {
            "bbox": [
                424,
                305,
                450,
                395
            ],
            "label": "Monkey wrench",
            "label_id": 25,
            "score": 0.2385500967502594
        }
    ]
}
```

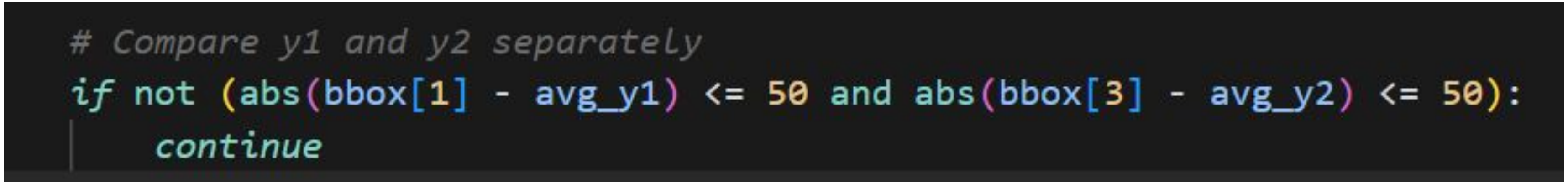
2D - 3D Mapping





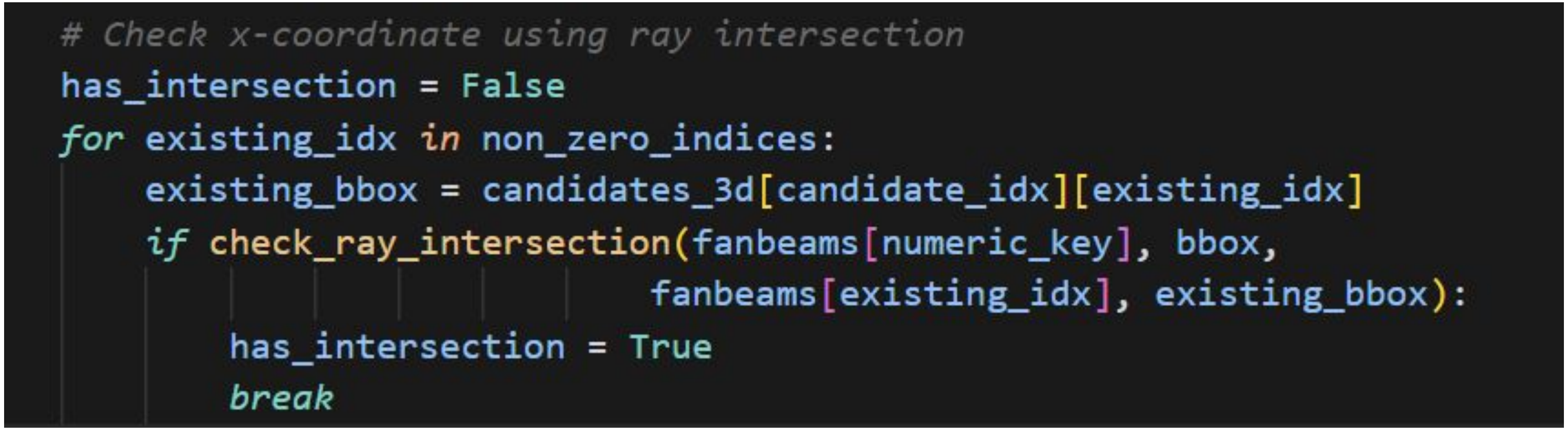
```json
{
    "count": 4,
    "data": [
        {
            "bbox": [
                457,
                286,
                575,
                377
            ],
            "label": "Monkey wrench",
            "label_id": 25,
            "score": 0.8930292129516602
        },
        {
            "bbox": [
                436,
                118,
                522,
                252
            ],
            "label": "Gun",
            "label_id": 0,
            "score": 0.48817703127861023
        },
        {
            "bbox": [
                450,
                475,
                579,
                573
            ],
            "label": "Gun",
            "label_id": 0,
            "score": 0.44572100043296814
        },
        {
            "bbox": [
                437,
                115,
                522,
                252
            ],
            "label": "Slingshot",
            "label_id": 3,
            "score": 0.3102450966835022
        }
    ]
}
```

1. 이미지 y 축 기준 필터링

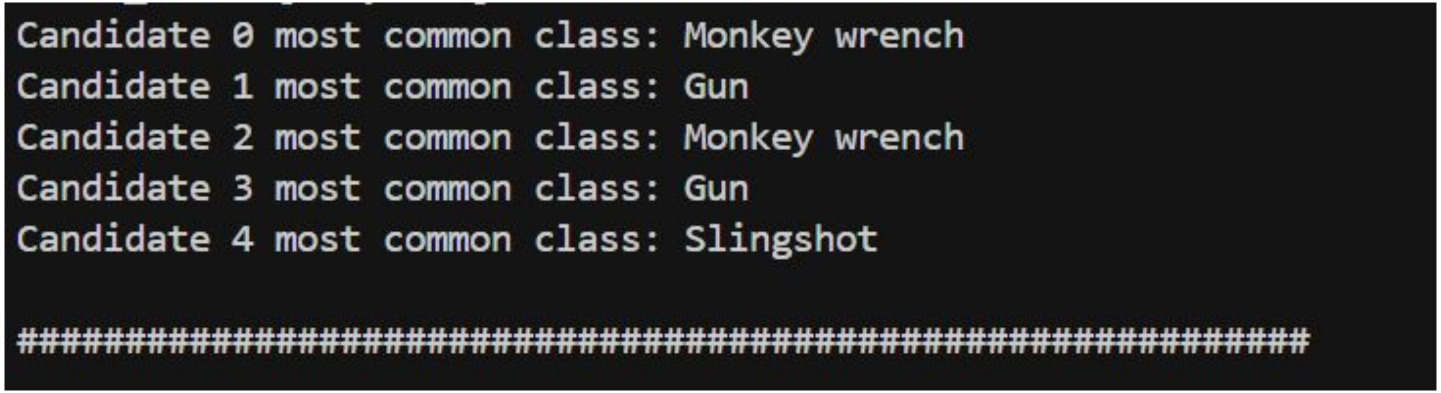


```python
# Compare y1 and y2 separately
if not (abs(bbox[1] - avg_y1) <= 50 and abs(bbox[3] - avg_y2) <= 50):
    continue
```

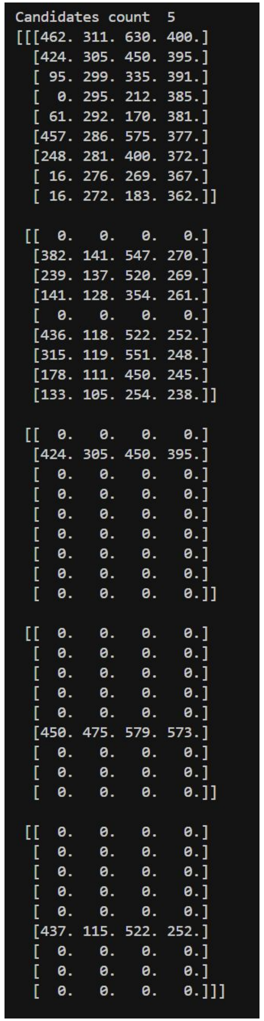
2. 이미지 x 축 기준 필터링



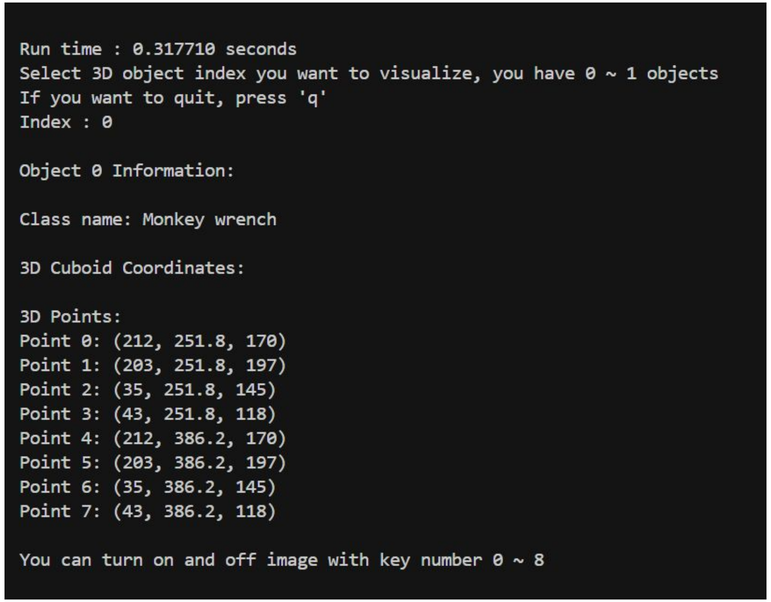
```python
# Check x-coordinate using ray intersection
has_intersection = False
for existing_idx in non_zero_indices:
    existing_bbox = candidates_3d[candidate_idx][existing_idx]
    if check_ray_intersection(fanbeams[numeric_key], bbox,
                              fanbeams[existing_idx], existing_bbox):
        has_intersection = True
        break
```



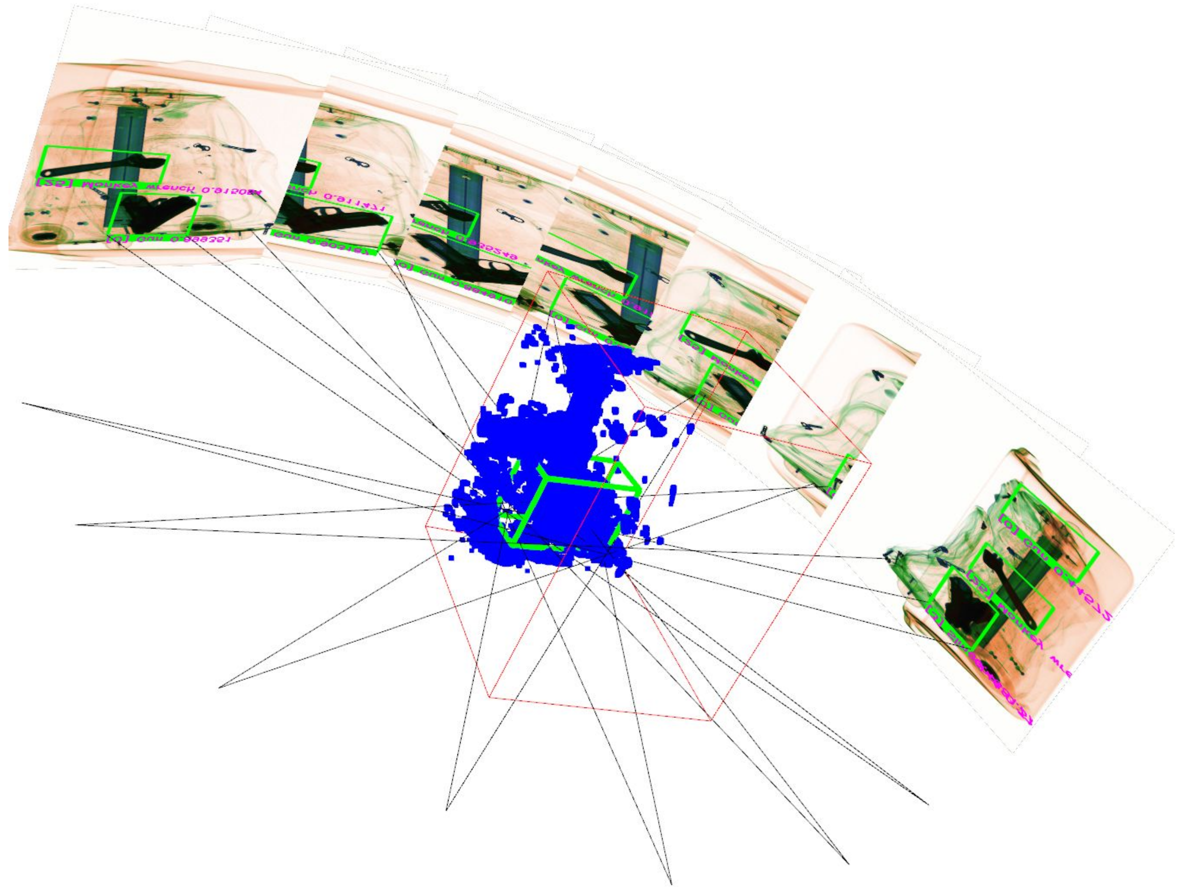
```
Candidate 0 most common class: Monkey wrench
Candidate 1 most common class: Gun
Candidate 2 most common class: Monkey wrench
Candidate 3 most common class: Gun
Candidate 4 most common class: Slingshot

##########################################################################
```



```
Candidates count  5
[[[462. 311. 630. 400.]
  [424. 305. 450. 395.]
  [ 95. 299. 335. 391.]
  [  0. 295. 212. 385.]
  [ 61. 292. 170. 381.]
  [457. 286. 575. 377.]
  [248. 281. 400. 372.]
  [ 16. 276. 269. 367.]
  [ 16. 272. 183. 362.]]

 [[  0.   0.   0.   0.]
  [382. 141. 547. 270.]
  [239. 137. 520. 269.]
  [141. 128. 354. 261.]
  [  0.   0.   0.   0.]
  [436. 118. 522. 252.]
  [315. 119. 551. 248.]
  [178. 111. 450. 245.]
  [133. 105. 254. 238.]]

 [[  0.   0.   0.   0.]
  [424. 305. 450. 395.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]]

 [[  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [450. 475. 579. 573.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]]

 [[  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [437. 115. 522. 252.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]
  [  0.   0.   0.   0.]]]
```



```
Run time : 0.317710 seconds
Select 3D object index you want to visualize, you have 0 ~ 1 objects
If you want to quit, press 'q'
Index : 0

Object 0 Information:

Class name: Monkey wrench

3D Cuboid Coordinates:

3D Points:
Point 0: (212, 251.8, 170)
Point 1: (203, 251.8, 197)
Point 2: (35, 251.8, 145)
Point 3: (43, 251.8, 118)
Point 4: (212, 386.2, 170)
Point 5: (203, 386.2, 197)
Point 6: (35, 386.2, 145)
Point 7: (43, 386.2, 118)

You can turn on and off image with key number 0 ~ 8
```



EOD