Planning Framework Seminar

2021. 01. 25

Part 2-1. Sensor/Detector 프레임워크

김종혁

설명 내용

Sensor/Detector 프레임워크 소개

- Sensor/Detector 프레임워크 구조
  - Camera Interface 구조 및 카메라 구동
- DetectorInterface
  - 물체 인식을 위한 detector 구조
- ArucoStereo Detector
  - Aruco marker 및 stereo camera 기반 물체 인식
- MultiICP Detector
  - Mask RCNN + ICP w/ model matching 기반 3차원 물체 인식
- SceneBuilder & GeometryScene
  - SceneBuilder & GeometryScene 소개 및 Rviz에 Geometry 추가
- Combined Detector (ArucoStereo, MultiICP)
  - Combined Detector 구동
- 실행 예제 스크립트 영상

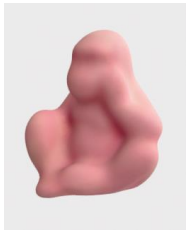
Sensor/Detector 프레임워크 소개

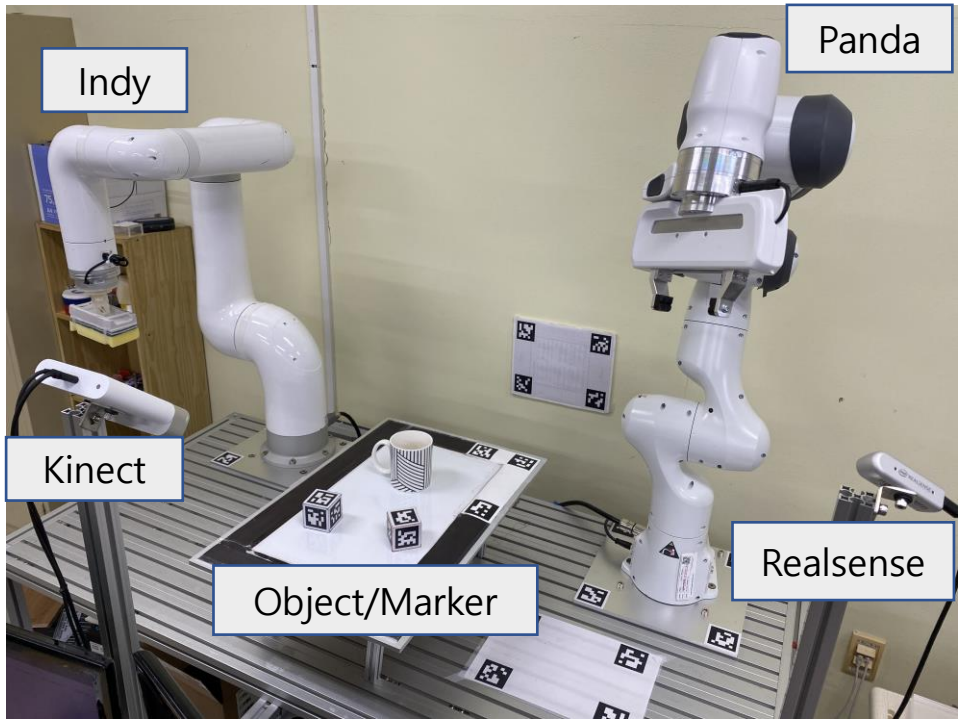
Sensor/Detector 프레임워크 구조

로봇 시스템

카메라 센서

물체/마커





※ 반드시 이 셋업에서 진행해야 하는 것이 아닌, 다른 카메라, 로봇 및 알고리즘을 사용하더라도 interface를 통일시킴으로써 framework와 연동이 가능. 따라서 이 프레임워크를 기반으로 확장성 있게 연구/작업을 진행할 수 있도록 하기 위함

Indy-Panda/Kinect-RealSense 셋업 예시

카메라 구동

kinect.py, realsense.py 참고

- Camera Interface 구조
  1) initialize/disconnect
  2) get_config
  3) get_image/get_depthmap/get_image_depthmap

1) 카메라 객체 생성 및 초기화
   - kinect = Kinect() / kinect.initialize()
   - realsense = RealSense() / realsense.initialize()

2) 카메라 내부변수, 왜곡계수, 뎁스 스케일
   - kinect.get_config()
   - realsense.get_config()

Depthmap에서의 pixel value scale[m]

get_config()

- This should return camera matrix (3x3) and distortion coefficients (4, 5, or 8 element vector)

```
1 camera_matrix, dist_coeffs, depth_scale = kinect.get_config()
2 print("camera_matrix: {} \n {}".format(camera_matrix.shape, camera_matrix))
3 print("dist_coeffs: {} \n {}".format(dist_coeffs.shape, dist_coeffs))
4 print("depth_scale: {}".format(depth_scale))
```

```
camera_matrix: (3, 3)
 [[1.82983423e+03 0.00000000e+00 1.91572046e+03]
 [0.00000000e+00 1.82983423e+03 1.09876086e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
dist_coeffs: (8,)
 [ 7.09966481e-01 -2.73409390e+00  1.45804870e-03 -3.24774766e-04
  1.44911301e+00  5.84310412e-01 -2.56374550e+00  1.38472950e+00]
depth_scale: 0.001
```

kinect config 정보

get_config()

- This should return camera matrix (3x3) and distortion coefficients (4, 5, or 8 element vector).
- Distortion coefficients for RealSense is [0,0,0,0,0]

```
1 camera_matrix, dist_coeffs, depth_scale = realsense.get_config()
2 print("camera_matrix: {} \n {}".format(camera_matrix.shape, camera_matrix))
3 print("dist_coeffs: {} \n {}".format(dist_coeffs.shape, dist_coeffs))
4 print("depth_scale: {}".format(depth_scale))
```

```
camera_matrix: (3, 3)
 [[1.39560388e+03 0.00000000e+00 9.62751587e+02]
 [0.00000000e+00 1.39531934e+03 5.47687012e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
dist_coeffs: (5,)
 [0. 0. 0. 0. 0.]
depth_scale: 0.0010000000475
```

realsense config 정보

카메라 구동

3) 이미지 촬영 (get_image(), get_depthmap(), get_image_depthmap())

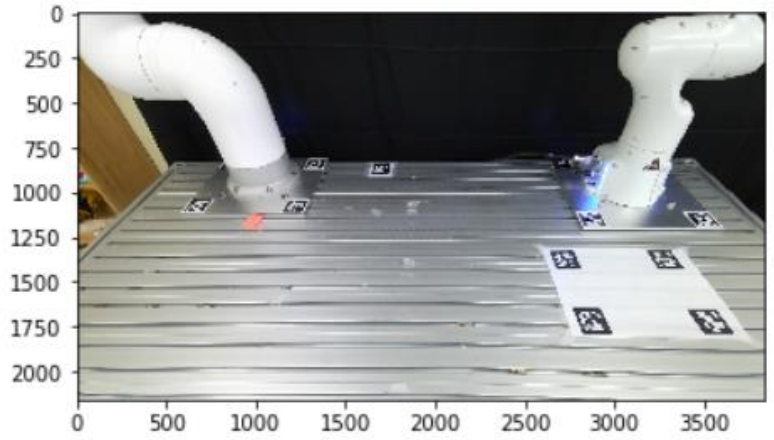
- kinect.get_image()
- realsense.get_image()

get_image()

- This will return camera image (RGB order)

```
1 plt.imshow(kinect.get_image()[:,:,[2,1,0]])
```

```
<matplotlib.image.AxesImage at 0x7f93398f7310>
```



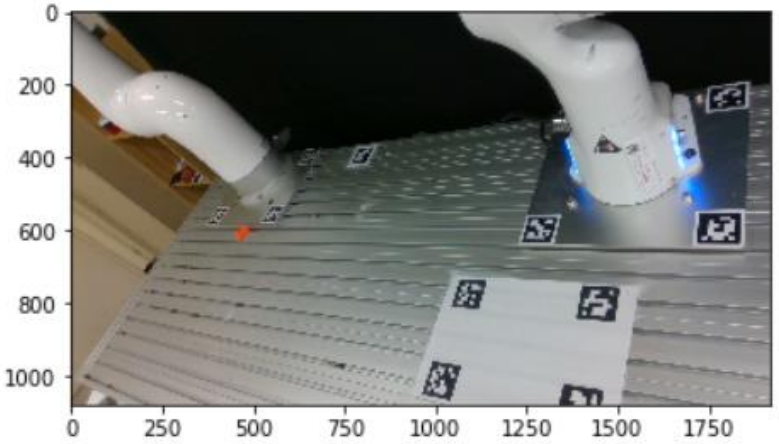
kinect로 이미지 촬영

get_image()

- This will return camera image (RGB order)

```
1 plt.imshow(realsense.get_image()[:,:,[2,1,0]])
```

```
<matplotlib.image.AxesImage at 0x7f9340bf51d0>
```



realsense로 이미지 촬영

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/2.Sensors.ipynb

Detector Interface

- Detector의 기본 구조(새로운 Detector 개발 시, 아래의 class method를 구현해야 함)
  1) initialize/disconnect
  2) calibrate (생략 가능)
  3) detect
  4) get_targets_of_levels/get_geometry_kwargs/add_item_axis

1) initialize, disconnect
   - Detector 구동 및 종료

2) calibrate
   - Detector 캘리브레이션(현재는 Stereo 카메라에서만 사용중)

3) detect(level_mask, name_mask, visualize)
   - level_mask – 명시한 DetectionLevel에 해당하는 물체만을 detect
   - name_mask – 명시한 이름에 해당하는 물체만을 detect
   - 물체의 pose가 저장된 dictionary를 리턴

DetectionLevel: 인식 시점에 따라 물체 분류

- ENVIRONMENT
- ROBOT
- MOVABLE
- ONLINE

Detector Interface

4-1) get_targets_of_levels(detection_level)

- 명시한 DetectionLevel에 해당하는 물체들의 이름을 리턴

4-2) get_geometry_kwargs(name)

- 명시한 geometry item의 argument들을 리턴 (geometry type, name, dims, color 등)

4-3) add_item_axis

- 명시한 geometry item의 detection 결과(coordinate)를 geometry scene에 표시

※ Geometry - Rviz에서 visualization되는 물체의 형상 (추후 설명)

ArucoStereo Detector

- 두 대의 카메라를 같이 사용하는 Stereo Camera Detector, 물체 인식을 위해 Aruco Marker 사용
- ArucoStereo 구조
  1) initialize/disconnect
  2) calibrate
  3) detect(level_mask, name_mask, visualize)

1) 스테레오 객체 생성 및 초기화
   - stereo = ArucoStereo(**arucomap**, camera_list)
   - stereo.initialize()

2) 스테레오 카메라 캘리브레이션
   - config1, config2, T_c12 = stereo.calibrate()

두 카메라 좌표계 간의 transformation

create ArucoStereo instance

```
1 stereo = ArucoStereo(aruco_map=get_aruco_map(),
2                      camera_list=[Kinect(), RealSense()])
```

initialize()

```
1 stereo.initialize()
```

ArucoStereo 객체 생성 및 초기화

calibrate()

- This calibrates relative displacement between two cameras. You need some ArUco markers in the field of view.
- The distance between two cameras tend to be calculated a few centimeters shorter than the actual distance.

cv2.stereoCalibrate

```
1 config1, config2, T_c12 = stereo.calibrate()
2 print("T_c12: \n {}".format(np.round(T_c12, 2)))
3 print("\n")
4 print("Check Distance! - %.2f cm"%(np.round(np.linalg.norm(T_c12[:3,3])*100, 2)))
```

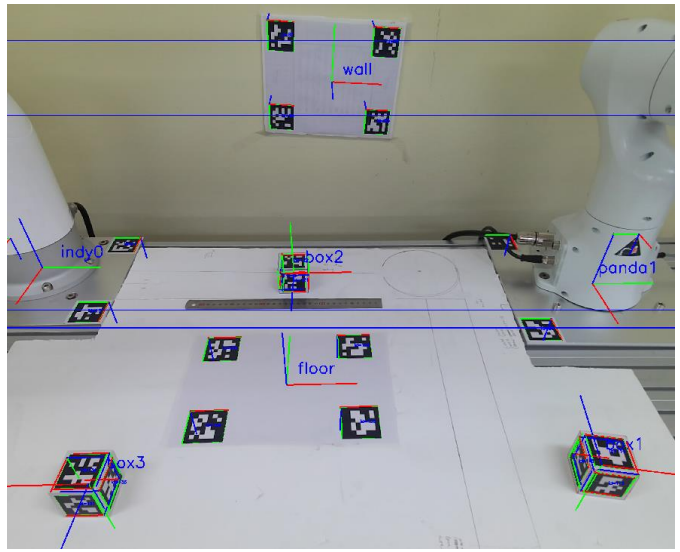
```
T_c12:
 [[ 0.81 -0.03 -0.59  0.72]
 [ 0.14  0.98  0.15 -0.15]
 [ 0.57 -0.2   0.8   0.06]
 [ 0.    0.    0.    1.  ]]


Check Distance! - 73.43 cm
```

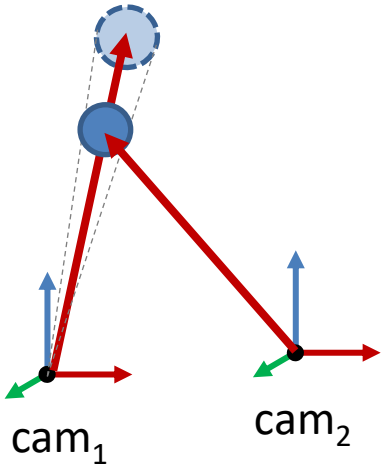
ArucoStereo Detector 캘리브레이션

ArucoStereo Detector

3) Aruco marker 기반 물체 인식

- 1개의 marker가 아닌 여러 개의 marker를 이용
- 1대의 카메라가 아닌 2대의 카메라를 이용





cv2.triangulatePoints

✓ **Multiple marker PnP**
✓ **Triangulate stereo camera**

- MarkerSet 설정 (arucomap) – name, marker index, size, marker coord w.r.t object(xyzrpy)

```
'box1':MarkerSet('box1',
                 dlevel=DetectionLevel.MOVABLE, gtype=GEOTYPE.BOX, dims=(0.05, 0.05,0.05), color=(0.8,0.3,0.3,1),
                 _list=[
                     ObjectMarker('box1', 111, 0.04, [0,0,0.025], (np.pi,0,0)),
                     ObjectMarker('box1', 112, 0.04, [0,0.025,0], (np.pi/2,0,0)),
                     ObjectMarker('box1', 113, 0.04, [0,0,-0.025], (0,0,0)),
                     ObjectMarker('box1', 114, 0.04, [0,-0.025,0], (-np.pi/2,0,0)),
                     ObjectMarker('box1', 115, 0.04, [0.025,0,0], (0,-np.pi/2,0)),
                     ObjectMarker('box1', 116, 0.04, [-0.025,0,0], (0,np.pi/2,0))
                 ]),
```

MarkerSet 설정 예시(marker_config.py 참고)

ArucoStereo Detector

3) Aruco marker 기반 물체 인식

- stereo.detect(level_mask, name_mask, visualize)

detect()

- level_mask - detects objects depending on levels
- name_mask - detects object with specified names
- visualize - visualize the detection result

level mask 지정 예시

```
print("Detect by Levels")
print("Robots on scene: {}".format(stereo.detect(level_mask=[DetectionLevel.ROBOT]).keys()))
print("Environments: {}".format(stereo.detect(level_mask=[DetectionLevel.ENVIRONMENT]).keys()))
print("Movable Objects: {}".format(stereo.detect(level_mask=[DetectionLevel.MOVABLE]).keys()))

print("Detect by Names")
detection_targets = ["floor"]
print("Try detect: {} - detected {}".format(detection_targets, stereo.detect(name_mask=detection_targets).keys()))

print("Detect and visualize - the detection result images from the two cameras will pop up")
stereo.detect(visualize=True)
```

name mask 지정 예시

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/src/pkg/detector/aruco

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/2.Sensors.ipynb

MultiICP Detector

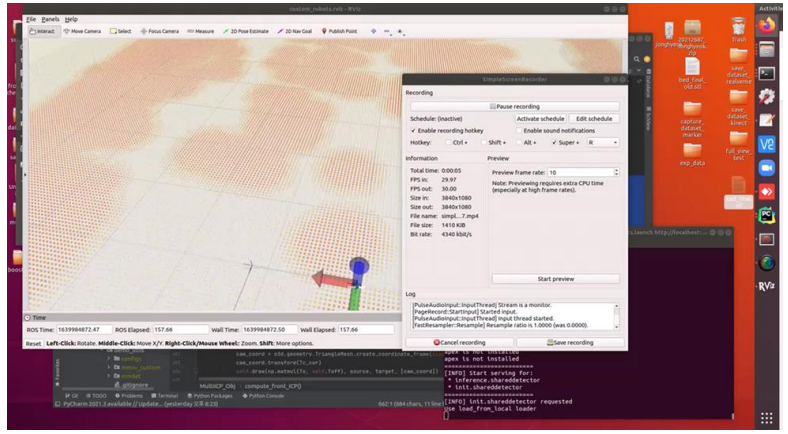
- Mask RCNN + ICP 기반의 모델 매칭을 통한 물체 인식 Detector
  1) 대상 영역 추출
  2) 초기값 추정
  3) 자세 결정 및 보정

RGBD image

Extract object region

Point cloud

Model



ICP(iterative Closest Point)

MultiICP 기반 인식 Flow

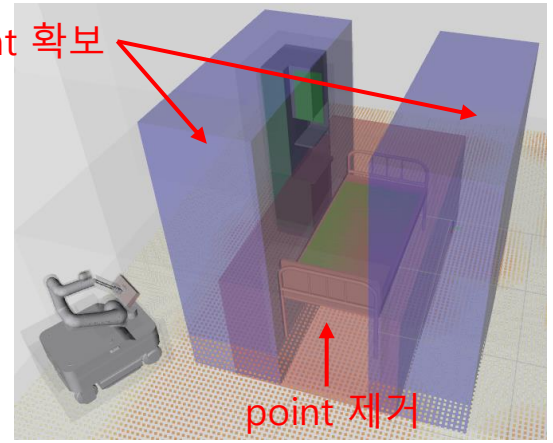
MultiICP Detector

1) 대상 영역 추출

- 방법 1: Mask RCNN으로 **인식이 가능**한 물체 (COCO Dataset에 들어있는 Class Object)에 해당
  - **Mask RCNN** 알고리즘으로 바로 **point cloud를 segmentation** (ex. 침대, 컵 등)
  - **Shared Detector**를 통해 촬영한 이미지로부터 인식한 물체의 영역을 획득(mask)
  - Color, depth 이미지에서 **mask**에 해당하는 부분들로부터 **물체의 point cloud** 생성
  - Shared Detector
    - Mask RCNN 기반 물체 인식

- 방법 2: Mask RCNN으로 **인식이 불가능**한 물체 (COCO Dataset에 없는 Class Object)에 해당
  - **Heuristic Rule**을 기반으로 **point cloud를 segmentation** (ex. 침대-상두대 관계)
  - 방법 1을 통해 인식한 물체에 대한 상대적인 관계를 설정. 상대적인 관계는 박스 형태로 생성하여 point cloud에서 제거 할 부분과 포함 할 부분을 구분
  - MaskRule (추상클래스)
    - MaskBoxRule: 박스 형태의 위치 관계로 물체 영역 추출
    - MaskBox: 박스를 생성하는 클래스

MaskBox 예시



MultiICP Detector

2) 초기값 추정 (for ICP)

- 자동 설정 → Point feature (FPFH) 기반 RANSAC
  - ICP의 **initial guess**를 제공하지 않았을 경우 작동
  - RANSAC을 통해 point feature matching을 해서 transformation 유추. 이를 초기값으로 사용
  - FPFH(**F**ast **P**oint **F**eature **H**istogram)
    - PFH: 인접한 포인트들의 normal vector와 거리를 이용하여 histogram을 만들고 이를 feature로 활용
    - FPFH → PFH의 Computation load를 줄이기 위해 인접한 포인트들 전부를 고려하지 않고 direct connection만 활용

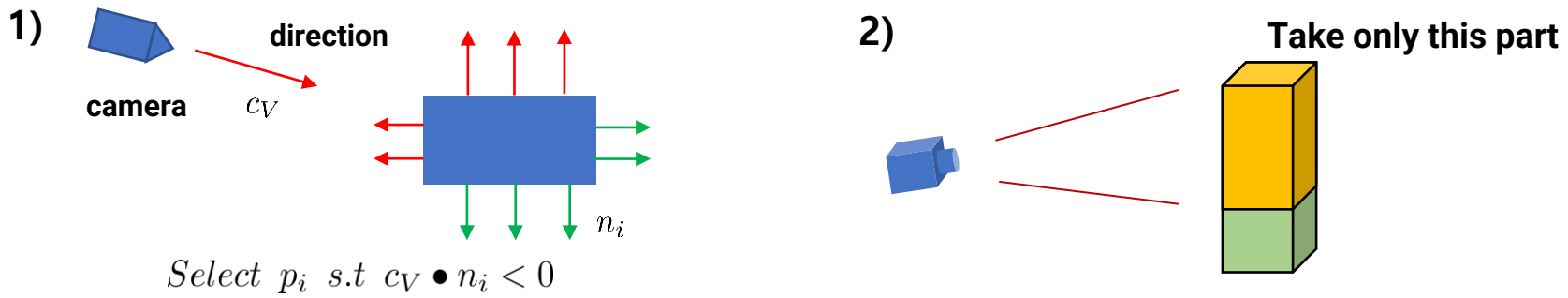
- 수동 지정
  - ICP의 **initial guess**를 사용자가 수동으로 설정
  - **Transformation matrix (Rot, Trans)**를 설정해주는 것
  - InitializeRule(추상클래스)
    - OffsetOnModelCoord: 포인트 클라우드 영역 중심 기준으로 오프셋 지정

MultiICP Detector

3) 자세 결정 / 보정

- ICP
  - 추출한 **물체의 point cloud**에 모델에서 샘플링한 **model point cloud**를 **정합**
  - Initial guess는 설정한 초기값으로 제공됨

- Front ICP
  - 실제로는 **물체의 일부분 데이터만**을 획득하기 때문에 획득한 데이터와 **가장 유사한 모델 부분만을 잘라서 매칭하는 전략**
  - Initial guess는 ICP를 통해 인식한 결과로 제공됨
  - 모델을 자르는 방법 (카메라 화면에서 최대한 가려지는 부분을 제거)
    - **1) Point normal processing**: 카메라 시점 벡터에 반대 방향에 해당하는 point normal을 가진 point만을 사용
    - **2) FOV filtering**: 카메라의 field-of-view에 들어오는 부분만을 사용



MultiICP Detector

※ 인식하고자 하는 물체의 정보 추가(config.py)

- ObjectInfo Class
  - name – 물체 이름
  - DetectionLevel - 물체의 detection level
  - Toff – Open3D에서의 STL 파일의 coordinate과 geometry scene에서의 coordinate 차이
  - scale – STL 파일에서의 치수와 실제 물체의 치수 간의 스케일
  - url – STL 파일의 경로

```
def get_obj_info():
    obj_info = {
        'cup': ObjectInfo('cup', dlevel=DetectionLevel.MOVABLE, gtype=GEOTYPE.CYLINDER,
                          dims=(0.4, 0.3, 0.01), color=(0.9, 0.9, 0.9, 0.2),
                          Toff=SE3(np.identity(3), (0, 0, 0)), scale=(1e-3,1e-3,1e-3),
                          url=RNB_PLANNING_DIR+'release/multiICP_data/cup.STL'),
```

```
'bed': ObjectInfo('bed', dlevel=DetectionLevel.ENVIRONMENT, gtype=GEOTYPE.BOX,
                  dims=(1.70,0.91,0.66), color=(0.9,0.9,0.9,0.2),
                  Toff=SE3([[0,1,0],[0,0,1],[1,0,0]], (0.455,0,1.02)), scale=(1e-3,1e-3,1e-3),
                  url=RNB_PLANNING_DIR+'release/multiICP_data/bed.STL'),
```

모델 config 추가 예시

MultiICP Detector

※ Heuristic Rule 추가(config.py)

- MaskBoxRule Class (Heuristic Rule)
  - 인식한 물체 주위에 Box를 만들어서 Box 별로 포함시키거나 포함시키지 않을 point 구분
  - config.py에서 box를 만들어내는 rule을 작성

```
##
# @param micp_closet on camera coordinates
# @param micp_bed on camera coordinates
def hrule_closet(micp_closet, micp_bed, mrule_closet):
    obj_info = get_obj_info()
    bed_dims = obj_info["bed"].dims
    CLOSET_LOCATION = check_closet_location(micp_closet.pcd, micp_bed.pcd, micp_bed.pose, bed_dims)

    mrule_closet.box_clear()
    # bed_box
    mrule_closet.add_box(MaskBox(Toff=SE3(np.identity(3), (0.02, 0, 0.5)),
                                 dims=(3, 1.6, 1.3), include=False))
    # bed_wall
    mrule_closet.add_box(MaskBox(Toff=SE3(np.identity(3), (-1.27, 0, 1.5)),
                                 dims=(0.5, 0.7, 0.3), include=False))
    # floor_box
    mrule_closet.add_box(MaskBox(Toff=SE3(np.identity(3), (0, 0, 0)), dims=(15, 15, 0.4), include=False))

    if CLOSET_LOCATION == "LEFT":
        # bed_left_space
        mrule_closet.add_box(MaskBox(Toff=SE3(np.identity(3), (0.02, -0.9, 1)),
                                     dims=(2.5, 1, 3), include=True))
    elif CLOSET_LOCATION == "RIGHT":
        # bed_right_space
        mrule_closet.add_box(MaskBox(Toff=SE3(np.identity(3), (0.02, 0.9, 1)),
                                     dims=(2.5, 1, 3), include=True))

    return mrule_closet
```

MaskBoxRule 생성 예시

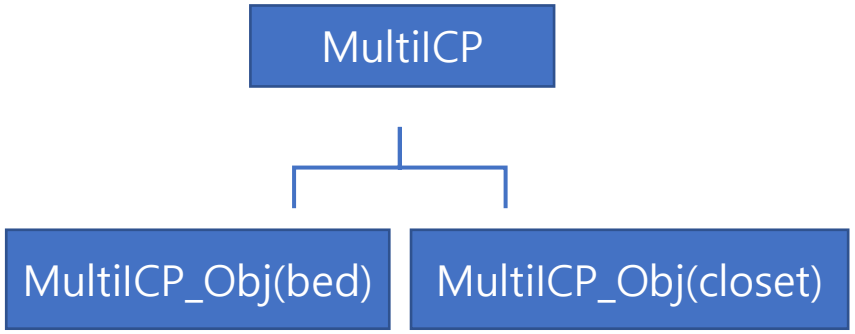
MaskBox Class

- ❖ Toff – 인식한 parent object로부터의 offset
- ❖ dims – MaskBox의 dimension
- ❖ include – MaskBox 내에 있는 point를 남길 것인지/버릴 것인지

MultiICP Detector

※ MultiICP 인식 개요: MultiICP Detector 생성 → MultiICP_Obj Class 생성 및 Detector에 추가 → 인식

- **MultiICP Detector** - 물체 인식을 전체적으로 총괄하는 클래스



- MultiICP 객체 생성 및 초기화
  - micp = MultiICP(camera) / MultiICP(None)
  - micp.initialize() / micp.initialize(config_list, img_dim)
  - 카메라를 연결해서 사용하지 않을 경우, 저장해놓은 이미지에 대해 테스트만 가능. 이때, config_list, img_dim을 수동으로 세팅해줘야함

create MultiICP instance

```
1  micp = MultiICP(None)
```

initialize()

```
1  # use manually given camera configs
2  config_list, img_dim = load_pickle(RNB_PLANNING_DIR+"release/multiICP_data/cam_configs.pkl")
3  micp.initialize(config_list, img_dim)
```

```
Camera is not set - skip initialization, use manually given camera configs
```

카메라가 연결되어 있지 않아 수동으로 파일 로드

```
micp = MultiICP(None)
cameraMatrix = np.array([[909.957763671875, 0., 638.3824462890625],
                         [0., 909.90283203125, 380.0085144042969],
                         [0., 0., 1]])
distCoeffs = np.array([0.]*5)
depth_scale=1 / 3999.999810010204
config_list = [cameraMatrix, distCoeffs, depth_scale]
img_dim = (720, 1280)
micp.initialize(config_list, img_dim)
```

카메라가 연결되어 있지 않아 수동으로 설정

MultiICP Detector

MultiICP_Obj

- ObjectInfo
- MaskBoxRule
- OffsetOnModelCoord

- **MultiICP_Obj** - 각각의 물체의 정보를 저장하는 클래스

- MultiICP_Obj 객체 생성 및 초기화
  - micp_bed = MultiICP_Obj(ObjectInfo, None, OffsetOnModelCoord)
  - micp_closet = MultiICP_Obj(ObjectInfo, MaskBoxRule, OffsetOnModelCoord)

```
# Load config file of object information
obj_info_dict = get_obj_info()
```

```
# object items you want to detect
micp_bed = MultiICP_Obj(obj_info_dict["bed"], None,
                        OffsetOnModelCoord("bed", R=np.matmul(T_cb[:3, :3], Rot_axis(3, np.pi)),
                                           offset=np.matmul(T_cb[:3, :3], (1.1 * 0.5, 0, -0.5))))
```

Bed의 MultiICP_Obj 객체 생성 – MaskBoxRule 없음

```
mrule_closet = MaskBoxRule("closet", "bed", merge_rule=np.all)
mrule_closet.update_rule = ClosetRuleFun(mrule_closet)
```

← Closet의 MaskBoxRule 생성

```
micp_closet = MultiICP_Obj(obj_info_dict["closet"],
                           mrule_closet,
                           OffsetOnModelCoord("closet",
                                              offset=(0, 1, 0.3),
                                              use_median=True
                                              ))
```

← Closet의 MultiICP_Obj 객체 생성 – MaskBoxRule 있음

MultiICP Detector

- MultiICP 객체의 config 파일 설정 (생성한 MultiICP_Obj Class를 Detector에 추가하는 과정)
  - micp.set_config(micp_dict, shared detector, combined robot, viewpoint)
    1) micp_dict - 인식하려는 물체들의 MultiICP_Obj class dictionary
    2) Shared detector - Mask RCNN Detector
    3) Combined Robot - Combined Robot Class
    4) viewpoint - camera 시점 geometry

```
1 # set config information for micp
2 micp.set_config(micp_dict, sd, crob, viewpoint)
```

1) micp_dict – 인식하려는 물체들의 MultiICP_Obj class dictionary

```
1 micp_dict = {"bed": micp_bed, "closet": micp_closet}
```

micp_dict 예시 (bed와 closet MultiICP_Obj class를 갖고 있음)

MultiICP Detector

- MultiICP 객체의 config 파일 설정
  - micp.set_config(micp_dict, shared detector, combined robot, viewpoint)
    1) micp_dict - 인식하려는 물체들의 MultiICP_Obj class dictionary
    2) Shared detector - Mask RCNN Detector
    3) Combined Robot - Combined Robot Class
    4) viewpoint - camera 시점 geometry

2) Shared detector – Mask RCNN Detector

Run shared detector for object detection on bash

- To skip detector initiailzation, run shared detector on separate terminal as follows
- set proper values to dims: desired size of MultiICP = reversed(micp.dsize)+(3,)

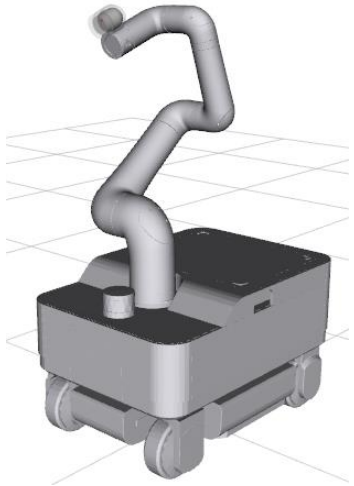
```
python3 $RNG_PLANNING_DIR/src/pkg/detector/multiICP/shared_detector.py --dims='(720,1280,3)'
```

```
from pkg.detector.multiICP.shared_detector import SharedDetectorGen
sd = SharedDetectorGen(tuple(reversed(micp.dsize))+(3,))()
sd.init()
```

Shared Detector 구동

MultiICP Detector

- MultiICP 객체의 config 파일 설정
  - micp.set_config(micp_dict, shared detector, combined robot, viewpoint)
    1) micp_dict - 인식하려는 물체들의 MultiICP_Obj class dictionary
    2) Shared detector - Mask RCNN Detector
    3) Combined Robot - Combined Robot Class
    4) viewpoint - camera 시점 geometry



Combined Robot 생성시
Rviz 화면(모바일로봇-인디)

3) Combined Robot – CombinedRobot Class
  - 두 대의 Robot을 함께 구동하는 클래스(ex. 인디+판다/모바일로봇+인디)

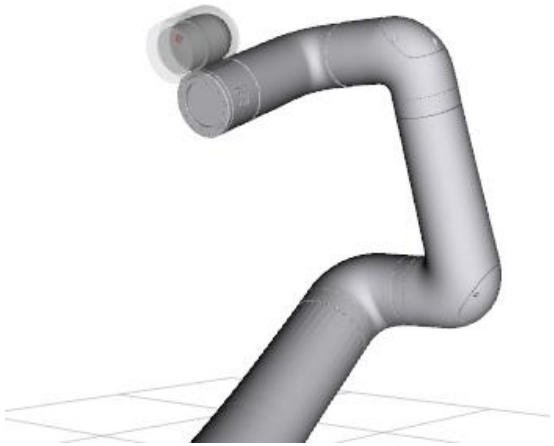
```
mobile_config = RobotConfig(0, RobotType.kmb, ((0,0,0), (0,0,0)),
                "{}/{}".format(None, None))
robot_config = RobotConfig(1, RobotType.indy7,
                           (INDY_BASE_OFFSET, INDY_BASE_RPY),
                           None, root_on="kmb0_platform",
                           specs={"no_sdk":True})
MOBILE_NAME = mobile_config.get_indexed_name()
ROBOT_NAME = robot_config.get_indexed_name()
crob = CombinedRobot(robots_on_scene=[mobile_config, robot_config]
              , connection_list=[False, False])
```

Combined Robot 생성예시
(모바일로봇+인디)

MultiICP Detector

- MultiICP 객체의 config 파일 설정
  - micp.set_config(micp_dict, shared detector, combined robot, viewpoint)
    1) micp_dict - 인식하려는 물체들의 MultiICP_Obj class dictionary
    2) Shared detector - Mask RCNN Detector
    3) Combined Robot - Combined Robot Class
    4) viewpoint - camera 시점 geometry

4) viewpoint – camera 시점 geometry



camera geometry 예시

```
gscene.create_safe(gtype=GEOTYPE.CYLINDER, name="cam", link_name=tool_link,
                   dims=(0.061, 0.061, 0.026), center=(-0.0785, 0, 0.013), rpy=(0, 0, 0),
                   color=(0.8, 0.8, 0.8, 0.5), display=True, fixed=True, collision=False)
```

```
viewpoint = gscene.create_safe(gtype=GEOTYPE.SPHERE, name="viewpoint", link_name=tool_link,
                               dims=(0.01, 0.01, 0.01), center=(-0.013, 0, 0), rpy=(0, 0, -np.pi / 2),
                               color=(1, 0, 0, 0.3), display=True, fixed=True, collision=False, parent="cam")
```

viewpoint geometry 생성

MultiICP Detector

- MultiICP 기반 물체 인식
  - 카메라 없이 사용했을 경우, 저장해놓은 이미지에 대해 테스트만 가능하며 저장한 이미지를 로드하고 MultiICP Class에 저장한 뒤 인식 → micp.cache_sensor(color_image, depth_image, Q)

Load example data (for bed, closet)

- You need to prepare example data and stl model to test MultiICP detector
- Use color image, depth image and csv file which has joint values Q with cam_intrins, depth_scale
- file path: release/multiICP_data/

```
1  def load_rdict(file_name):
2      rdict = {}
3
4      rdict['color'] = cv2.imread(
5          os.path.join('../release/multiICP_data/', file_name + '.jpg'), flags=cv2.IMREAD_UNCHANGED)
6      rdict['depth'] = cv2.imread(
7          os.path.join('../release/multiICP_data/', file_name + '.png'), flags=cv2.IMREAD_UNCHANGED)
8
9      Q = np.loadtxt(os.path.join('../release/multiICP_data/', file_name + '.csvd'), delimiter=",")
10     return rdict, np.array(Q)
11
12
13 rdict, Qtest = load_rdict("test_1")
14 color_img = rdict['color']
15 depth_img = rdict['depth']
```

저장해놓은 이미지 로드 예시

```
# save test data at micp class by using cache_sensor()
micp.cache_sensor(color_img, depth_img, Qtest)
```

- micp.detect(name_mask, level_mask, visualize)

```
pose_dict = micp.detect(visualize=True)
```

인식결과 물체의 자세를 dictionary에 저장해서 반환

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/3.1.MultiICP.ipynb

SceneBuilder & GeometryScene

- SceneBuilder: Detector로 인식한 결과를 환경 모델에 추가
- GeometryScene: 물체의 형상, 환경 모델링 및 Rviz를 통한 시각화

- SceneBuilder 객체 생성
  - Scene_builder = SceneBuilder.instance(detector)

Create SceneBuilder instance

```
1 from pkg.geometry.builder.scene_builder import SceneBuilder
2 scene_builder = SceneBuilder.instance(detector=stereo)
```

Detector를 SceneBuilder에 연동

- GeometryScene 생성 및 Robot Geometry 추가
  - gscene = scene_builder.create_gscene(combined robot)
  - gtems = scene_builder.add_robot_geometries(color, display, collision)

create_gscene()

- Create a GeometryScene instance

```
1 gscene = scene_builder.create_gscene(combined_robot)
```

Geometry scene 생성

add_robot_geometries()

- Add collision boundaries defined in the URDF/Xacro file of each robots

```
1 gtems = scene_builder.add_robot_geometries(color=(0,1,0,0.5), display=True, collision=True)
```

Robot geometry 추가

SceneBuilder & GeometryScene

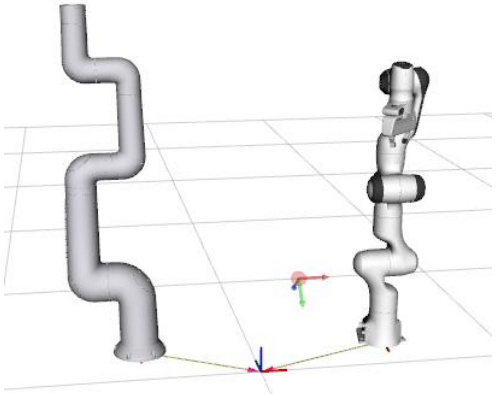
- SceneBuilder 물체 인식 및 scene에 추가
  - gtem_dict = scene_builder.detect_and_register(item_names, level_mask)

detect_and_register()

- Detect items in the field of view and register them to the GeometryScene
- They will appear in the RVIZ

```
1 gtem_dict = scene_builder.detect_and_register(level_mask=[DetectionLevel.ENVIRONMENT])
2 gtem_dict = scene_builder.detect_and_register(level_mask=[DetectionLevel.MOVABLE])
```

Detector로 인식한 결과를 scene에 업데이트



Combined Robot 생성시
Rviz 화면(Indy-Panda)

- GeometryScene에 Geometry 추가
  - gscene.create_safe(name, link name, geometry type, center, rpy, dims,
    color, display, collision, fixed, parent)

```
from pkg.geometry.geotype import GEOTYPE

# generate box geometry
gscene.create_safe(name="test_box", link_name="base_link", gtype=GEOTYPE.BOX,
                                center=(-0.5,0.0,0.05), rpy=(0,0,0), dims=(0.1,0.1,0.1),
                                color=(1,0,0,1), display=True, collision=True, fixed=True)
```

Box geometry 추가 예시

SceneBuilder & GeometryScene

- GeometryScene에서 로봇의 자세와 움직임 시각화
  - gscene.show_pose(pose)
  - gscene.show_motion(pose list, period)

show_pose()

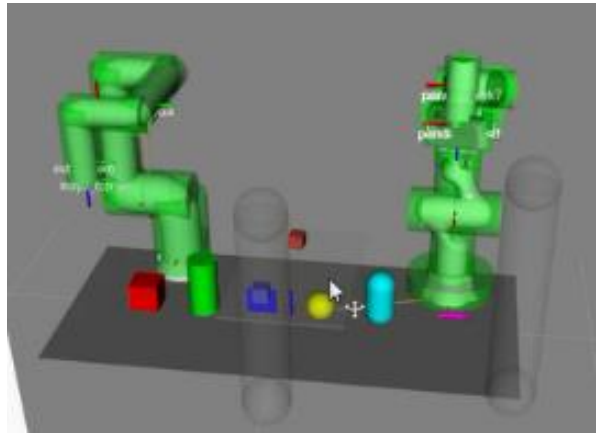
- show pose in RVIZ

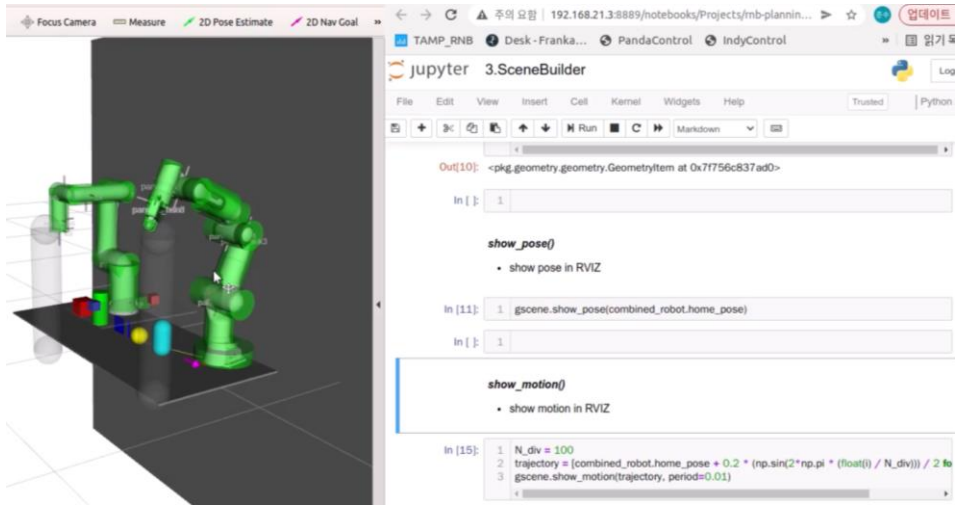
```
1 gscene.show_pose(combined_robot.home_pose)
```

show_motion()

- show motion in RVIZ

```
1 N_div = 100
2 trajectory = [combined_robot.home_pose + 0.2 * (np.sin(2*np.pi * (float(i) / N_div))) / 2 for i in range(N_div)]
3 gscene.show_motion(trajectory, period=0.01)
```





show_pose, show_motion 예시

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/3.SceneBuilder.ipynb

ArucoStereo - MultiICP Combined Detector

- Aruco와 MultiICP Combined Detection
- CombinedDetector 객체 생성 및 SceneBuilder 연동
  - detector = CombinedDetector(detector list)

```
detector = CombinedDetector([stereo, micp])
```

Combined Detector 생성

create SceneBuilder instance

```
1 scene_builder = SceneBuilder.instance(detector=detector)
```

- 스테레오 객체 초기화 및 캘리브레이션
  - stereo = ArucoStereo(arucomap, camera_list)
  - stereo.initialize(), stereo.calibrate()

create ArucoStereo instance

```
1 stereo = ArucoStereo(aruco_map=get_aruco_map(),
2                       camera_list=[kn, rs])
3 stereo.initialize()
4
5 time.sleep(1) # Let the camera have some time to get stable
6 stereo.calibrate()
```

- MultiICP 객체 생성 및 초기화
  - micp = MultiICP(camera)
  - micp.initialize()

create multiICP instance

```
1 micp = MultiICP(kn)
2 micp.initialize()
```

ArucoStereo, MultiICP 두 개의 Detector 생성

ArucoStereo - MultiICP Combined Detector

- ArucoStereo Detector를 통해 Robot 위치 업데이트 및 scene 생성

set reference coordinate and viewpoint (by StereoAruco)

```
T_kn = stereo.ref_coord_inv
```

Detector의 reference coordinate 설정

create geometry scene

```
crob = CombinedRobot(
    robots_on_scene=[
        RobotConfig(0, RobotType.indy7, ((0,-0.3,0), (0,0,0)), None),
        RobotConfig(1, RobotType.panda, ((0,0.3,0), (0,0,0)), None)],
    connection_list=[False, False])

xyz_rpy_robots = scene_builder.detect_items(level_mask=[DetectionLevel.ROBOT])
crob.update_robot_pos_dict(xyz_rpy_robots=xyz_rpy_robots)
gscene = scene_builder.create_gscene(crob)
gscene.show_pose(crob.home_pose)
viewpoint = gscene.create_safe(gtype=GEOTYPE.SPHERE, name="viewpoint", link_name="base_link",
                               dims=(0.05, 0.05, 0.02), center=T_kn[:3,3], rpy=Rot2rpy(T_kn[:3,:3]),
                               color=(1, 0, 0, 0.3), display=True, fixed=True, collision=False)
```

Combined Robot 및 Geometry scene 생성

- MultiICP Detector를 통해 마커없이 물체 인식

detect_and_register()

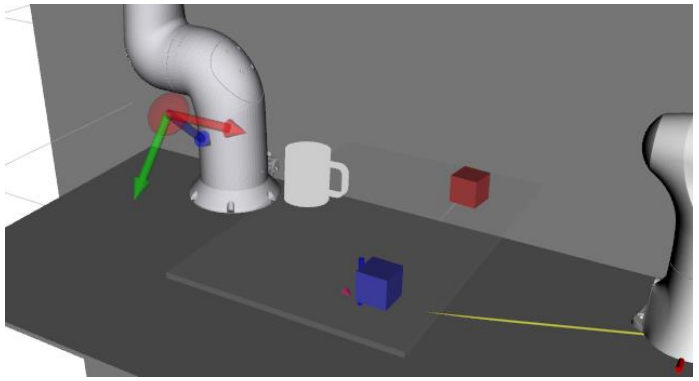
- Detect items in the field of view and register them to the GeometryScene
- They will appear in the RVIZ

```
gtem_dict = scene_builder.detect_and_register(level_mask=[DetectionLevel.ENVIRONMENT])
```

```
name_mask is []
```

```
gtem_dict = scene_builder.detect_and_register(level_mask=[DetectionLevel.MOVABLE],
                                              visualize=True)
```

```
name_mask is ['cup']
===== Detected : cup, 1 object(s) =====
```

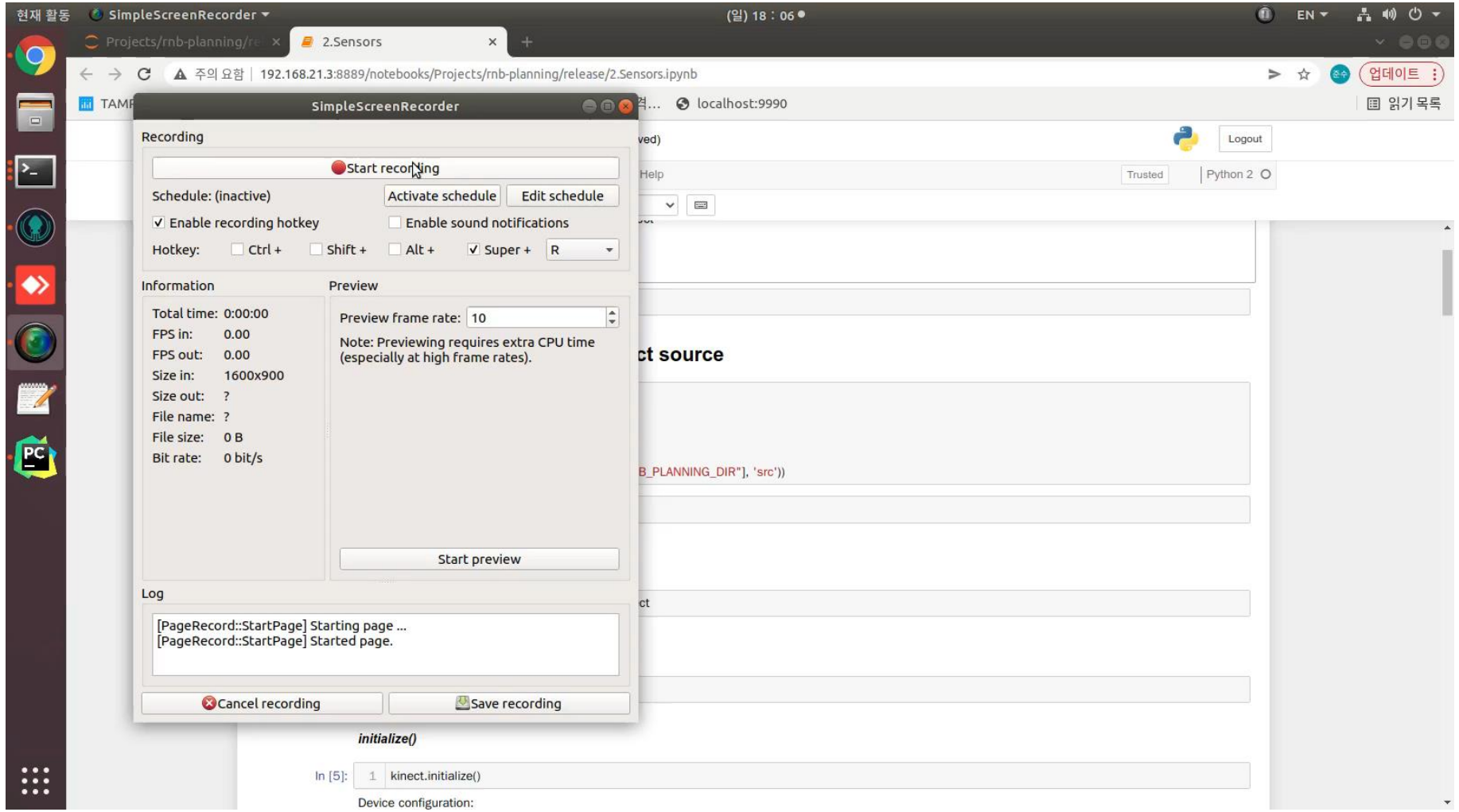


인식 결과 scene에 cup 추가

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/3.2.MultiICP-Combined.ipynb

실행 예제

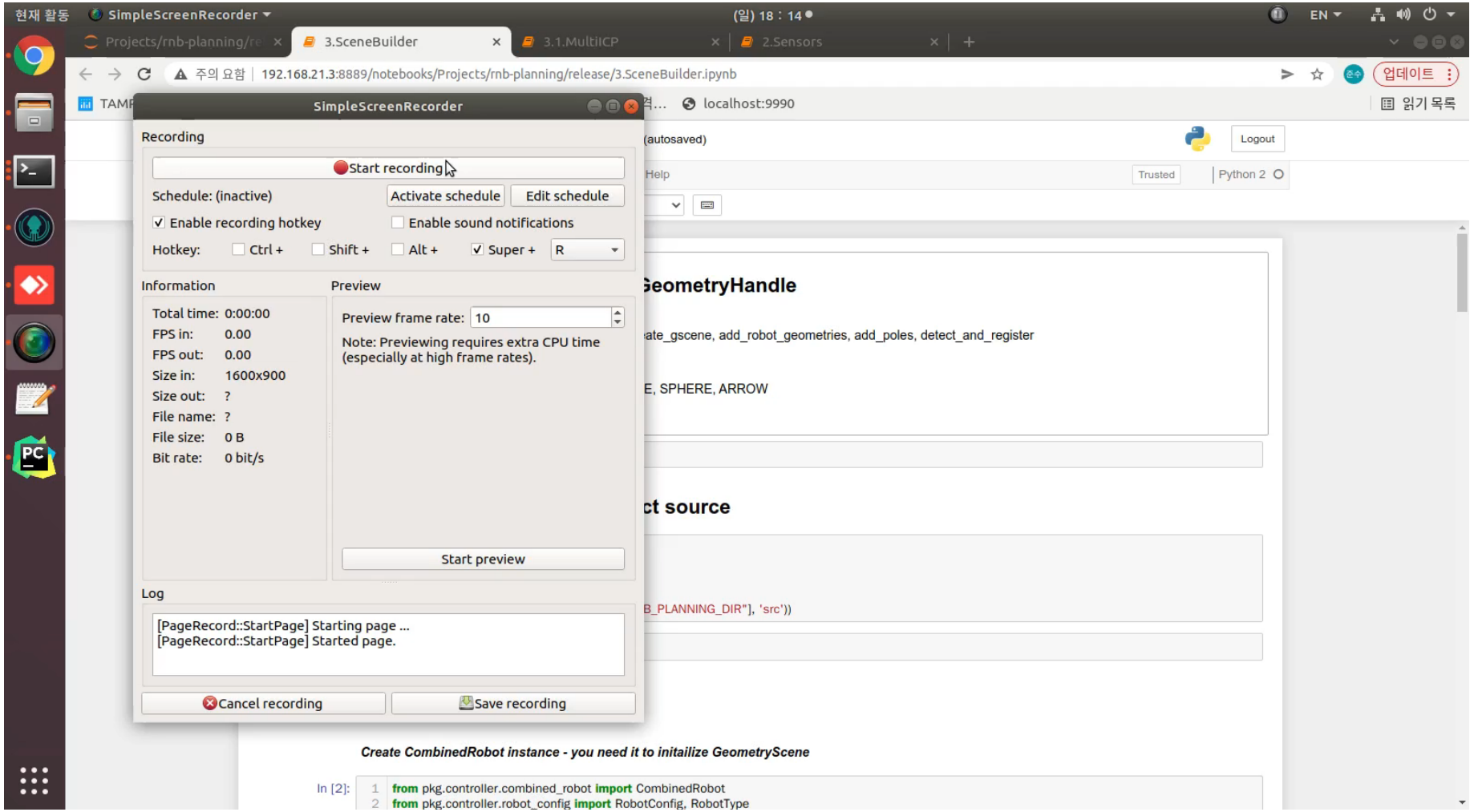
- Sensor 예제 script (Kinect, RealSense, ArucoStereo)



참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/2.Sensors.ipynb

실행 예제

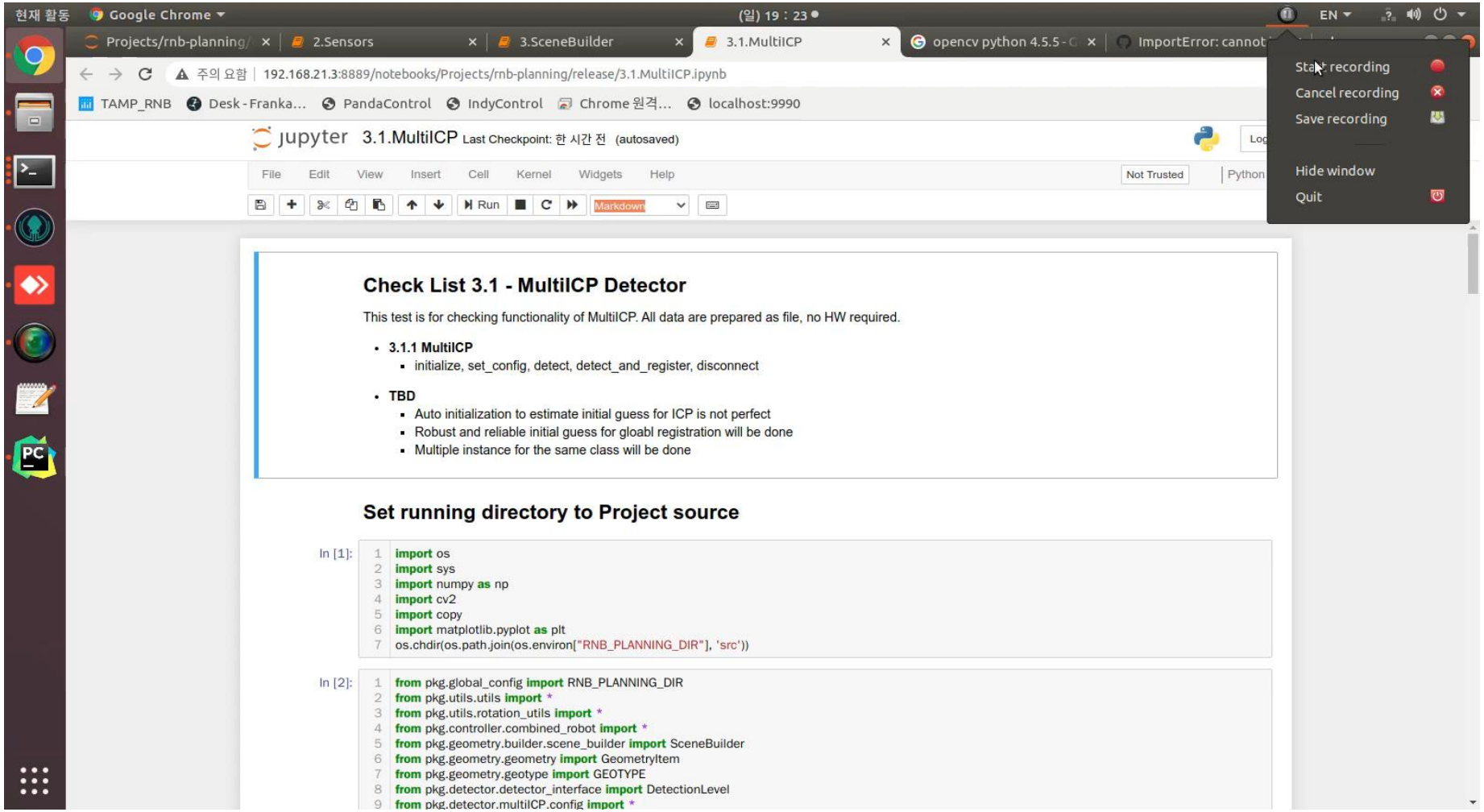
- SceneBuilder & GeometryScene 예제 script



참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/3.SceneBuilder.ipynb

실행 예제

- MultiICP 예제 script (bed, closet 인식)



참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/3.1.MultiICP.ipynb

실행 예제

- Combined Detector (ArucoStereo, MultiICP) 예제 script

실제 위치

참고 자료: https://github.com/rnb-disinfection/rnb-planning/tree/develop/release/3.2.MultiICP-Combined.ipynb

APPENDIX

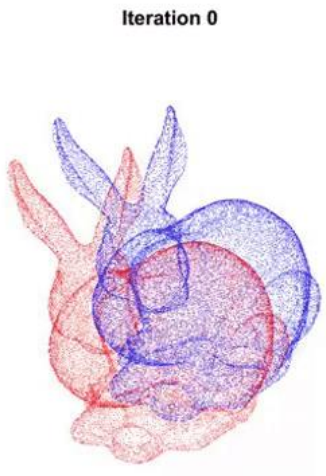
ICP(Iterative Closest Point)[1]

6DoF Pose ↑ CAD model ↓ Observed Data ↓

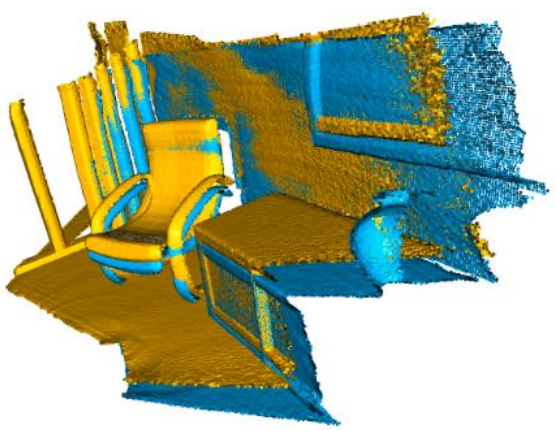
- **Optimize** below objective so that we obtain the **transform T** which align **source** and **target point cloud**

$$E(T) = \sum_{(p,q)\in K} \|p - Tq\|^2$$

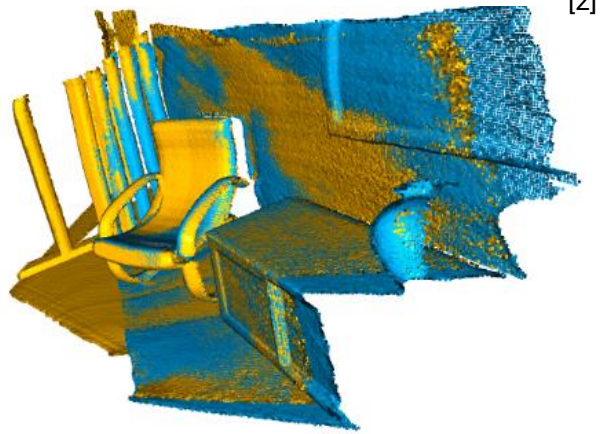
correspondence set K={(p,q)}
from target point cloud P, source point cloud Q



[2]



Transform T →



Before align

After align

[1] Paul, J. "BESL and Neil MCKAY, A methode for registration of 3d shapes." IEEE Transactions on pattern analysis and machine intelligence 14.2 (1992): 239-256.

[2] Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. "Open3D: A modern library for 3D data processing." arXiv preprint arXiv:1801.09847 (2018).