PnP

1. ArUco마커의 정답값에 대한 Text 파일(result.txt), 임의의 사진을 읽는다
   a. 정답값 파일(result.txt) : 각 ArUco 마커의 ID와 World좌표계 기준 마커의 코너점(x,y,z) 4개
   b. 정답값 파일(result.txt)의 Format

      <ArUco 마커의 ID>

      <Corner 1 = x,y,z>

      <Corner 2 = x,y,z>

      <Corner 3 = x,y,z>

      <Corner 4 = x,y,z>

      … … …

   c. 예시

```
1
-1.7780474013548275 -0.5911724465900646 2.538114159560967
-1.6167995200194363 -0.5876025476704967 2.525722531509747
-1.6053059325272965 -0.5597621240987823 2.6867047324758473
-1.7675475951523465 -0.5622238510981535 2.6967577924941404
6
-0.7226252380666552 0.1373691807122849 3.0892130791345105
-0.5631206110358429 0.1405330320573944 3.086604420825473
-0.5659680213599725 0.300476203006917 3.0640311453438516
-0.7261388641456582 0.29693207389012743 3.0662599876236207
7
-0.7524670195106614 -0.6195602434516664 2.297376415423312
-0.5919010400274396 -0.6187448844051906 2.2918126337334934
-0.5843321542216886 -0.5886409587720803 2.4535007712913735
-0.745430237334934 -0.589250295386726 2.457878928920139
8
-0.4510691785365574 0.36744341926084717 3.0491309237523576
-0.29105369353627164 0.36708285496255644 3.0439622161950495
-0.29122848692562736 0.5265930726090436 3.013962988435374
-0.4515618018218716 0.5256628412010519 3.016055590584805
10
-0.3968960171405705 -0.5808066931458298 2.487796400680272
-0.2351234462287772 -0.5800962314640141 2.4813908431372544
-0.22817182822900586 -0.5504007906848454 2.6427155714800206
-0.38968479982507287 -0.5502610991533756 2.6485516747384668
```

   d. 관련 코드

```
def read_text_file(file_path):
    data = {}
    with open(file_path, 'r') as file:
            #파일의 각 줄을 읽음
        lines = file.readlines()
        current_id = None
        for line in lines:
            line=line.strip()
            if line:
                    #ArUco 마커의 ID를 읽고,
                if line.isdigit():
                    current_id = line.strip()
                    data[current_id] = []
                #해당 ID의 data에 points(x,y,z)를 입력함.
                else:
                    points = [float(val) for val in line.split()[:3]]
                    data[current_id].append(points)
    #결과값 data는 각 ArUco ID와 그에 해당되는 4개의 코너점들을 가지고 있음.
    return data

text_data = read_text_file(text_file_path)
```

2. 임의의 사진으로부터 ArUco 마커의 ID와 corner점들을 계산.

```
def detect_aruco(image_path):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    #'DICT_6X6_250' : ArUco 생성 시 설정하는 파라미터, 사용자의 환경에 맞게 수정가능함.
    aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
    parameters = cv2.aruco.DetectorParameters_create()
    #흑백영상으로부터 ArUco 마커 검출, 결과값은 ID와 각 corner 4개 점들
    corners, ids, _ = cv2.aruco.detectMarkers(gray, aruco_dict, parameters=parameters)
    return ids, corners

detected_ids, detected_corners = detect_aruco(image_path)
```

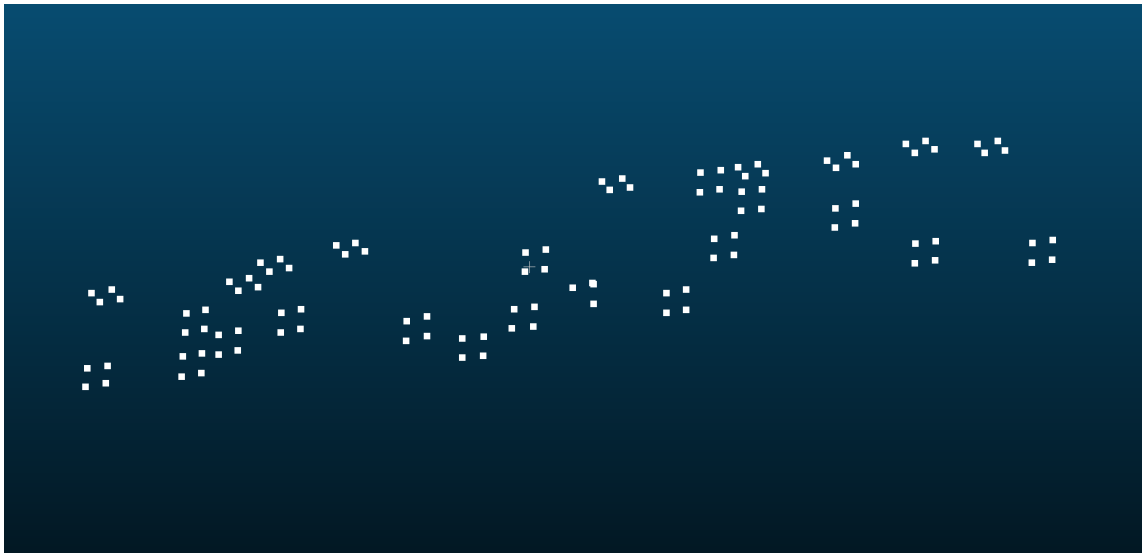
3. 1번의 결과와 2번의 결과 간의 ID 비교하여, 동일 ID 여부 확인 및 corner점들 저장

```
def compare_ids(text_data, detected_ids):
    matched_ids = []
    matched_points = []
    detected_ids_list = detected_ids.flatten().tolist()
    for id, points in text_data.items():
        if str(id) in map(str, detected_ids_list):
            matched_ids.append(id)
            matched_points.append(points)
    return matched_ids, matched_points

matched_ids, matched_points = compare_ids(text_data, detected_ids)
```

4. PnP 알고리즘

```
#PnP알고리즘을 돌리기 위해, 데이터 사전정렬
target_updated_corners = []
updated_3dpoint = []

for id_str,corner_points in zip(matched_ids,matched_points):
      # Find the index of the ID in detected_ids
    id_index = np.where(detected_ids == int(id_str))[1][0]
    target_updated_corners.append(detected_corners[id_index][0])
    updated_3dpoint.append(corner_points)

selected_target_updated_corners = [point for sublist in target_updated_corners
                                                                    for point in sublist]
selected_updated_3dpoint = [point for sublist in updated_3dpoint
                                                        for point in sublist]

#OpenCV의 PnP 알고리즘
#입력데이터
#    - 2차원 포인트들     : selected_target_updated_corners
#    - 3차원 포인트들     : selected_updated_3dpoint
#    - Camera Parameters : K(카메라 파라미터), D(왜곡함수)
#출력데이터
#    - rvec : 회전벡터
#    - tvec : 이동벡터
rvec, tvec = solve_pnp(np.array(selected_target_updated_corners),
                                            np.array(selected_updated_3dpoint), K, D)

#회전벡터를 회전행렬로 변환
R, _ = cv2.Rodrigues(rvec)

#최종 결과(Transformation Matrix Visualization)
T = tvec
print(np.hstack((R,T)))
```

5. 실행 결과
    a. 정답값 Visualization 예시



b. PnP 알고리즘 수행 결과

i. 입력영상

ii. 결과화면(Transformation Matrix)

```
Transformation Matrix
[[ 8.75443776e-01  3.08806633e-01 -3.71801907e-01  5.50578160e+00]
 [ 3.15071655e-01  2.18713041e-01  9.23522852e-01  7.19614119e+01]
 [ 3.66507908e-01 -9.25636575e-01  9.41747537e-02  2.09540841e+02]]
```