ArUCo Marker 실습

ai-contents

Exported on 06/18/2023

Table of Contents

1 1. 코드 및 라이브러리 설치

- 해당 내용은 Window 10 기준입니다.

1-1. 코드 설치

- 공유된 ArUCo 마커 실습 데이터 폴더에서 ‘pose_QrUCo.zip’을 다운받고, 압축 해제를 합니다.
- 압축이 해제된 코드 폴더를 'C:/Users/사용자이름/' 위치로 이동합니다.

1-2. 가상환경 생성 및 numpy, opencv 등 필요 라이브러리 설치

- python version : 3.9
- 환경이름 : mobility(원하는 이름으로 생성해도 됩니다.)

```
conda create -n mobility python=3.9
conda activate mobility
pip install numpy opencv-python
pip install opencv-contrib-python==4.6.0.66
```

---

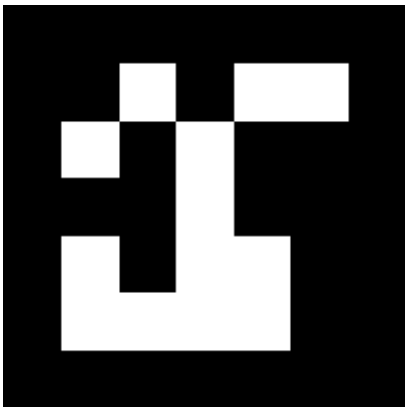
2 2. 코드 사용 방법

2-1. ArUCo 마커 생성

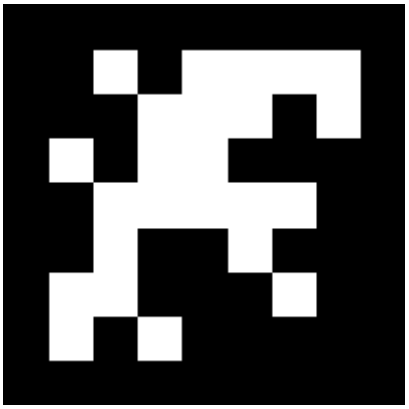
- 명령어

```
#
cd pose_ArUCo

# ArUCo 1
python 1_generate_aruco_tags.py --id 23 --type DICT_5X5_100 --output aruco_marker/

# ArUCo 2
python 1_generate_aruco_tags.py --id 23 --type DICT_7X7_100 --output aruco_marker/
'''
입력
  id는 생성할 ArUCo 마커의 ID(ex. 23)
  type은 생성할 ArUCo 마커의 타입(ex. DICT_5X5_100)
  output은 ArUCo 마커의 이미지를 저장할 폴더의 경로(ex. aruco_marker/)

출력
  ArUCo 이미지
'''
```

출력 결과



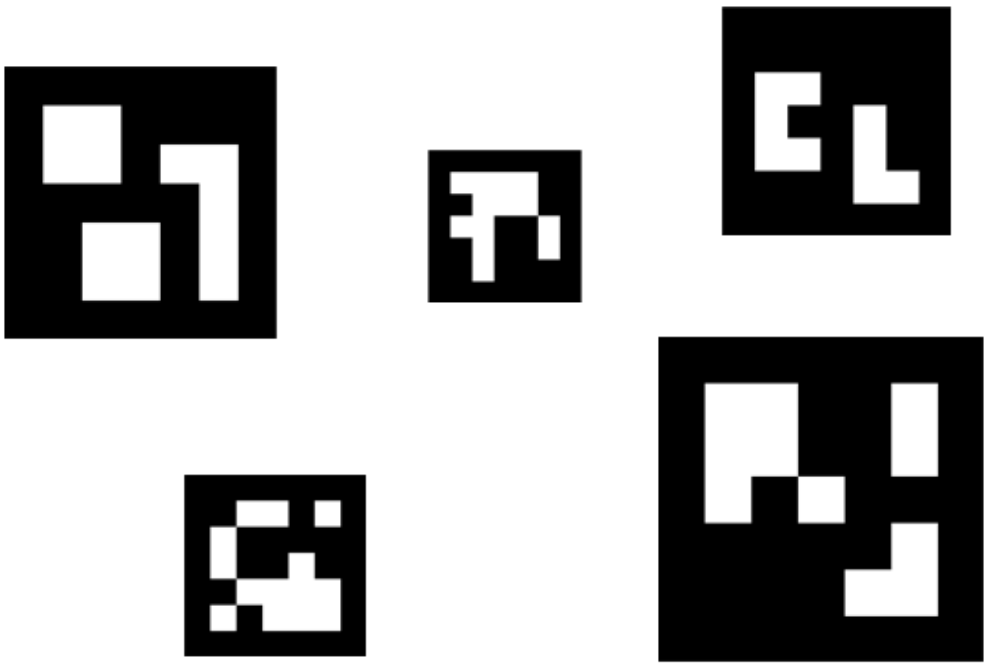
1 5X5 ArUCo 마커



2 7X7 ArUCo 마커

2-2. ArUCo 마커 검출

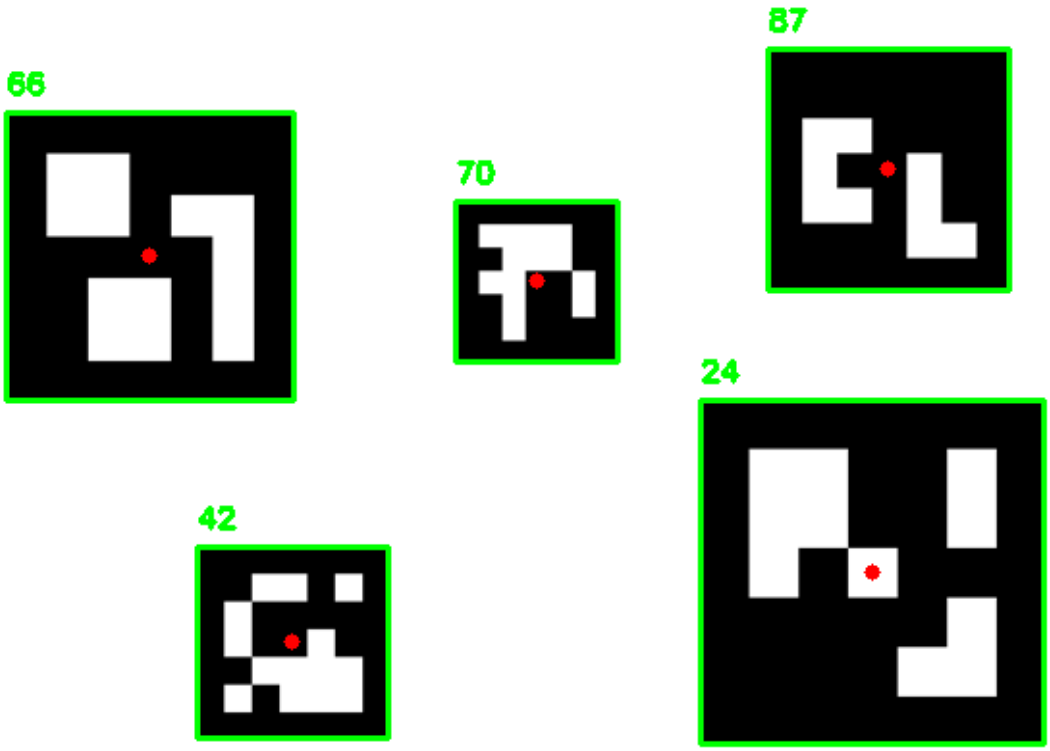
- 검출하고자 하는 마커들의 이미지가 필요합니다.
- 위에서 생성한 마커를 쓰거나 첨부되어 있는 예시(aruco_marker폴더 내 test_image_1.png)를 사용해도 좋습니다.



- 명령어

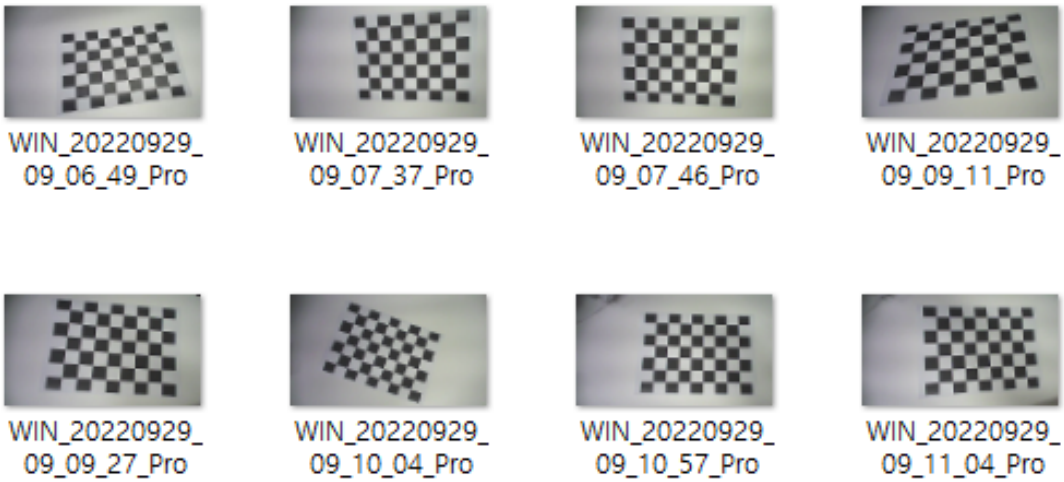
```
python 2_detect_aruco_images.py --image aruco_marker/test_image_1.png \
--type DICT_5X5_100 --output aruco_marker/
'''
입력
  image는 검출하고자 하는 마커 이미지의 경로(ex. aruco_marker/test_image_1.png)
  type은 검출하고자 하는 마커의 타입(ex. DICT_5X5_100)
  output은 검출된 마커에 대한 이미지를 저장할 경로(ex. aruco_marker/)

출력
  검출된 ArUCo marker ID 및 중앙점
'''
```

- 출력 결과



2-3. 카메라 캘리브레이션

- 준비
  - 체스보드 파일을 인쇄합니다.(A4용지)
  - 캘리브레이션을 하고자 하는 카메라로 해당 채스보드 사진을 7~10장 가량 촬영합니다.
    - 위에 첨부되어 있는 체스보드는 예시이므로 본인이 실제 사용할 카메라로 촬영해야 합니다.
  - 촬영한 사진을 'calibration_checkerboard' 폴더 안으로 옮깁니다.



- 명령어

```
python 3_calibration.py --dir calibration_checkerboard/ --width 8 --height 6 \
--square_size 0.03
'''
입력
  dir은 촬영한 체커보드 이미지가 들어있는 폴더(ex. calibration_checkerboard/)
  width는 체커보드의 너비 픽셀 수(ex. 8)
  height는 체커보드의 높이 픽셀 수(ex. 6)
  square_size는 체커보드 한 셀의 실제 길이(m 단위)(ex. 0.03)

출력
  calibration_matrix.npy는 카메라 내부 행렬이 저장된 파일
    ex. [[1.33997373e+03 0.00000000e+00 7.67833751e+02]
         [0.00000000e+00 1.33849566e+03 3.93775659e+02]
         [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
  distortion_coefficients.npy는 렌즈 왜곡 계수 값이 저장된 파일
    ex. [[0.23732017 -1.60298151 -0.01527387 -0.00997957
        6.76024172]]
'''
```

2-4. 3차원 자세 추정(카메라 좌표계 기준)

- 명령어

```
python 4_pose_estimation.py --K_Matrix calibration_matrix.npy \
--D_Coeff distortion_coefficients.npy --type DICT_5X5_100
```

```
'''
입력
  K_Matrix는 내부 카메라 행렬에 대한 파일 경로(ex. calibration_matrix.npy)
  D_Coeff는 렌즈 왜곡 계수에 대한 파일 경로(ex. distortion_coefficients.npy)
  type은 검출하고자 하는 마커의 타입(ex. DICT_5X5_100)

출력
  ArUCo marker의  3차원 자세
  4_pose_estimation.py에서 아래의 변수에  6차원 자세 값이 저장됨
    - rvec:회전 벡터(3X1)
    - tvec:이동 벡터(3X1)
'''
```