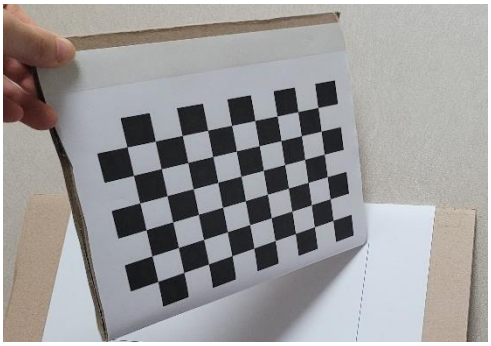
CSE404 Project 3

201811118 이 구

1. Result & Discussion

이번 프로젝트에서는 스마트폰 카메라를 이용한 camera calibration과 3d coordinate에서 다른 3d coordinate로의 변환을 수행한다. 먼저, camera calibration 과정에 대해 다룬 후, 3d coordinate 변환을 기반으로 한 직사각형의 길이 측정에 대해 다룬다.

먼저, camera calibration을 위해서는 일반적으로 checker board를 활용한다. 사용한 checker board의 모습은 아래(왼쪽)와 같다. 한 칸의 길이는 18mm이며, row, column의 교차점 수는 각각 9개, 6개이다. Camera Calibration을 위해서는 최소 3장의 이미지가 필요하며, 이번 프로젝트에서는 checker board를 촬영한 18장의 사진을 활용하여 수행하였다. 이를 통해 얻은 camera의 intrinsic matrix는 아래(오른쪽)와 같다. 촬영한 이미지의 해상도는 4032 x 2268 이다.



[[3105.37, 0.00, 2113.02],

[0.00, 3106.35, 1172.21],

[0.00, 0.00, 1.00]]

이후, 3d coordinate 변환을 기반으로 한 직사각형(가로 20cm, 세로 10cm)의 길이 측정을 진행하였다. 총 5가지의 상황에서 왼쪽 위, 오른쪽 위, 왼쪽 아래, 오른쪽 아래 꼭짓점에 checker board를 붙인 상태에서 촬영하였다. 이때, 실제로 직사각형의 꼭짓점과 맞닿는 부분은 checker board의 교차점이 아니므로, offset을 고려해야 한다. 이때, 직접 측정한 offset이 가로, 세로 30mm 이므로, 이를 고려해서 transform을 수행하였다.

이를 통해 각 꼭짓점 별로 transform을 수행한 후, 해당 포인트들 사이의 Euclidean distance를 계산한 결과는 아래와 같다. 모든 결과는 소수점 아래 둘째 자리에서 반올림하였다.

	왼쪽 변 (mm)	*오른쪽 변 (mm)*	*윗쪽 변 (mm)*	*아랫쪽 변 (mm)*
Case 1	*105.8*	*96.6*	*201.9*	*199.6*
Case 2	*96.7*	*95.9*	*200.5*	*199.6*
Case 3	*99.1*	*95.8*	*200.5*	*197.9*
Case 4	*100.0*	*97.9*	*195.0*	*192.3*
Case 5	*97.5*	*100.4*	*191.2*	*196.8*
Mean	*99.8*	*97.3*	*197.8*	*197.2*
Variance	*10.3*	*2.9*	*16.6*	*7.2*

약간의 오차는 존재하지만, 모든 케이스에서 길이를 잘 추정한 것을 확인할 수 있다. 사진을 촬영하는 과정에서 checker board의 끝점과 직사각형의 꼭짓점 사이가 떨어지는 경우가 종종 발생했으며, 화면을 확인하며 사진을 촬영하다 보니 발생한 motion blur 등이 오차의 원인이 될 수 있다.

2. Code

2.1. Calibration

Intrinsic parameter와 distortion 정보를 저장한다.

```python
import cv2
import numpy as np

# np.set__printoptions(precision=2, suppress=True)

wp = 9
hp = 6
length = 18 # mm
directory = "/home/gulee/Desktop/ComputerVision/Project3/intrinsic/"
imageExtension = '.jpg' # ext
startImageNum = 0
endImageNum = 17

objp = np.zeros((wp*hp, 3), np.float32)
objp[:, :2] = np.mgrid[0:wp, 0:hp].T.reshape(-1,2)
objp[:, :2] *= length # checkboard 길이

objpoints = []
imgpoints = []
findImages = []

for i in range(startImageNum, endImageNum):
    img = cv2.imread(directory + str(i) + imageExtension)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(gray, (wp, hp), None)
    img_shape = gray.shape[::-1] # 사진 가로, 세로

    if ret==True:
        print(f'{i}th image calib pattern finding success')
        objpoints.append(objp)
        imgpoints.append(corners)
        findImages.append(i)

    else:
        print(f'{i}th image calib pattern finding failed')

rt, M, D, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, img_shape, None, None)
print(f'Intrinsic matrix M: \n {M}')
np.savetxt(directory + "Cal_intrinsic.txt", M, fmt="%.2f")
np.savetxt(directory + "Cal_distortion.txt", D, fmt="%.2f")


W = np.full((3, 4), 0.0)
R = np.full((3, 3), 0.0)

for i, no in enumerate(findImages):
    rvec = rvecs[i]
    tvec = tvecs[i]

    cv2.Rodrigues(rvec, R)
    W[0:3, 0:3] = R
    W[0:3, 3:4] = tvec
    print(f'{no}th Extrinsic matrix W: \n{W}')
    np.savetxt(directory + "Cal_extrinsic" + str(no) + ".txt", W, fmt='%.2f')

print(f"Distorsion coefficient: \n{D}")
print(f"Reprojection error: {rt}")
```

2.2. Camera Tracking

2.1에서 얻은 정보를 이용하여, 3D coordinate에서의 변환을 수행하고 그 결과를 출력한다.

```
import cv2
import numpy as np

np.set_printoptions(precision=2, suppress=True)

wp = 9
hp = 6
length = 18 # mm
directory = "/home/gulee/Desktop/ComputerVision/Project3/intrinsic/"
target_directory = "/home/gulee/Desktop/ComputerVision/Project3/try1/"
imageExtension = '.jpg' # ext
startImageNum = 0
endImageNum = 3

objp = np.zeros((wp*hp, 3), np.float32)
objp[:, :2] = np.mgrid[0:wp, 0:hp].T.reshape(-1,2)
objp[:, :2] *= length # checkboard 길이

intrinsic = np.loadtxt("/home/gulee/Desktop/ComputerVision/Project3/intrinsic/Cal_intrinsic.txt")
distorsion = np.loadtxt("/home/gulee/Desktop/ComputerVision/Project3/intrinsic/Cal_distortion.txt")

W = np.full((3,4), 0.0)
R = np.full((3,3), 0.0)

# try1
pts = objp[0]-30
pts[-1] = 0
pts = np.append(pts, 1)

for i in range(startImageNum, endImageNum+1):
    img = cv2.imread(target_directory + str(i) + imageExtension)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(gray, (wp, hp), None)
    img_shape = gray.shape[::-1]

    if ret==True:
        # print(f'{i}th img pattern recog success')
        ret, rvec, tvec = cv2.solvePnP(objp, corners, intrinsic, distorsion)

        cv2.Rodrigues(rvec, R)
        W[0:3, 0:3] = R
        W[0:3, 3:4] = tvec
        # print(f'{i}th HTM matrix W:\n{W}')
        np.savetxt(target_directory + "\HTM_"+str(i)+".txt", W, fmt="%.2f")

        # print(f'{i}th Projection matrix P:\n{P}')

        # print(f'{i}th inverse Projection matrix P:\n{inverse_P}')

        pts_3d = np.dot(W, pts)
        print(f'{i}th 3d points:\n{pts_3d}')

    else:
        print(f'{i}th image pattern point recog failed')
```

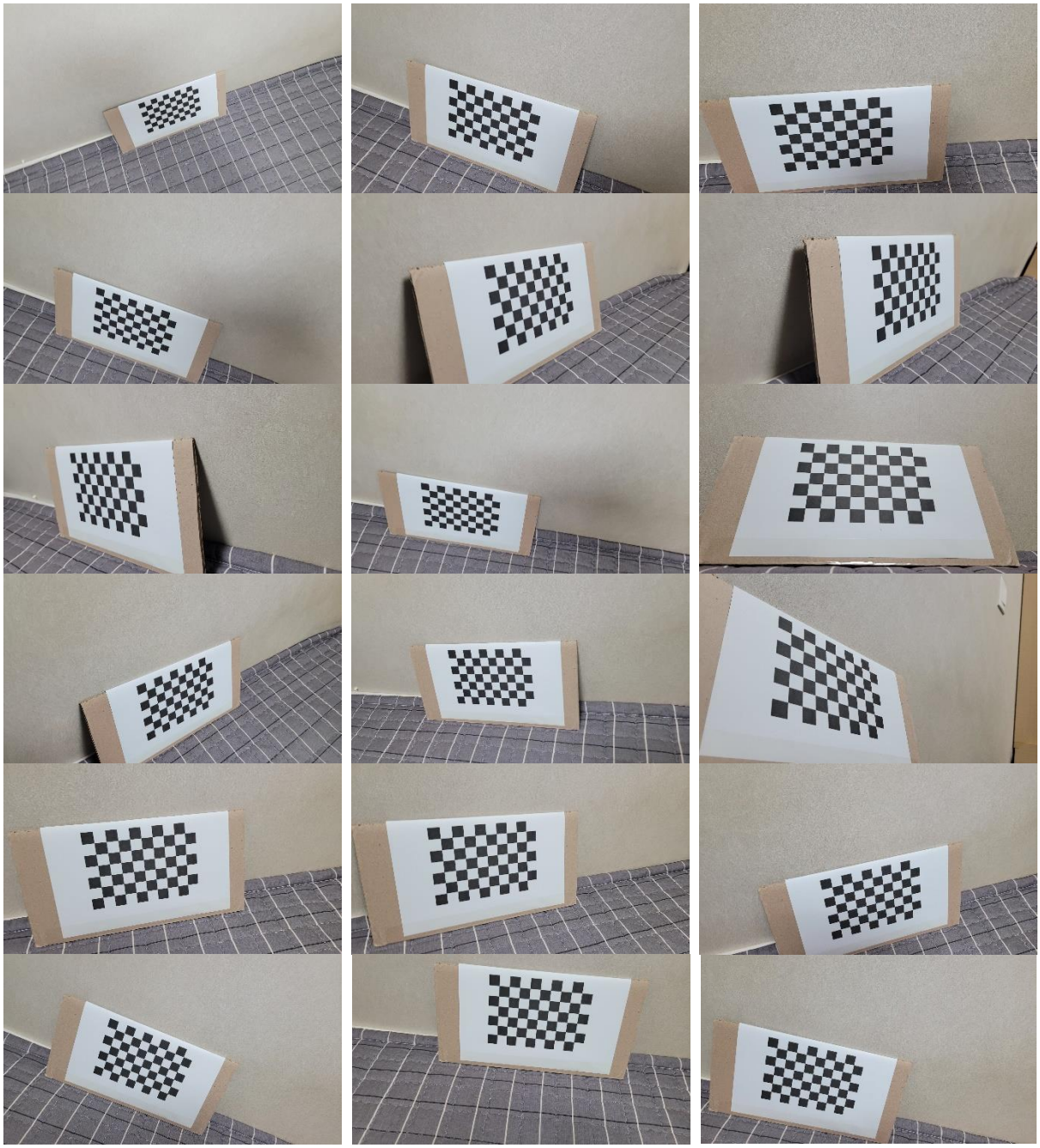
2.3. Calculate Distance

2.2에서 출력한 정보를 이용하여, 직사각형의 각 변의 길이를 계산하고 출력한다.

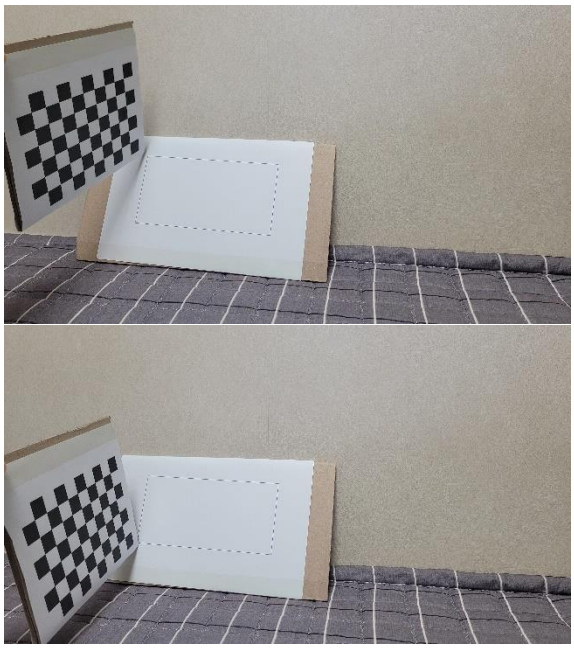
```python
import numpy as np
import math

def calc_side_length(pts):
    def dist(x, y):
        return math.sqrt((x[0]-y[0])**2 + (x[1]-y[1])**2 + (x[2]-y[2])**2)

    left_length = dist(pts[2], pts[0])
    right_length = dist(pts[3], pts[1])
    top_length = dist(pts[2], pts[3])
    bottom_length = dist(pts[1], pts[0])

    print(f"left: {left_length}, right: {right_length}, top: {top_length}, bottom: {bottom_length}")
    return


if __name__ == "__main__":

    case_1 = [
        [-231.19, 81.99, 620.25],
        [-34.69, 88.16, 585.5 ],
        [-230.12, -22.32, 637.8 ],
        [-31.16, -5.81, 607.46]    ,
    ]


    case_2 = [
        [-216.39, 58.85, 620.2 ],
        [-20.73, 71.12, 582.68],
        [-211.43, -35.93, 638.9 ],
        [-13.16, -22.29, 602.8 ],
    ]
    case_3 = [
        [-94.3, 75.47, 702.37],
        [ 94.68, 88.86, 645.32],
        [-82.49, -15.37, 740.09],
        [108.88, 1.86, 682.78],
    ]

    case_4 = [
        [-81.95, 77.35, 559.8 ],
        [109.97, 86.5, 568.01],
        [-80.26, -17.57, 591.4 ],
        [113.78, -3.92, 605.45],
    ]

    case_5 = [
        [-128.78, 59.19, 703.01],
        [ 67.27, 66.58, 718.43],
        [-127.66, -32.58, 735.89],
        [ 62.35, -26.28, 756.26],
    ]

    calc_side_length(case_1)
    calc_side_length(case_2)
    calc_side_length(case_3)
    calc_side_length(case_4)
    calc_side_length(case_5)
```

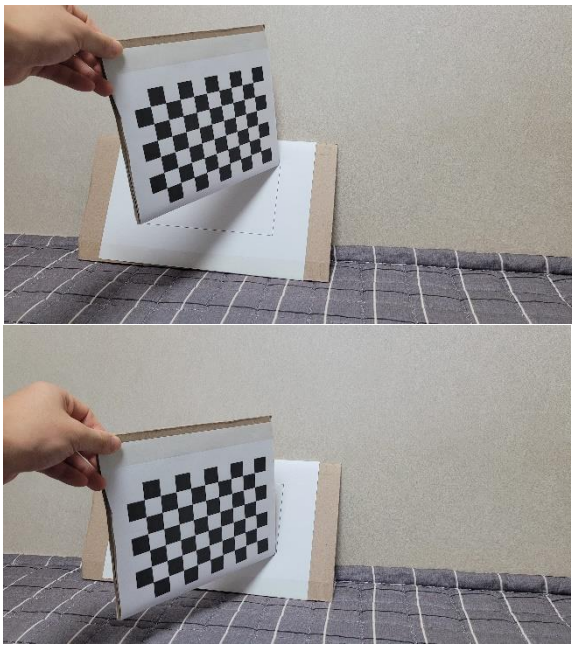
3. Image

3.1. Calibration

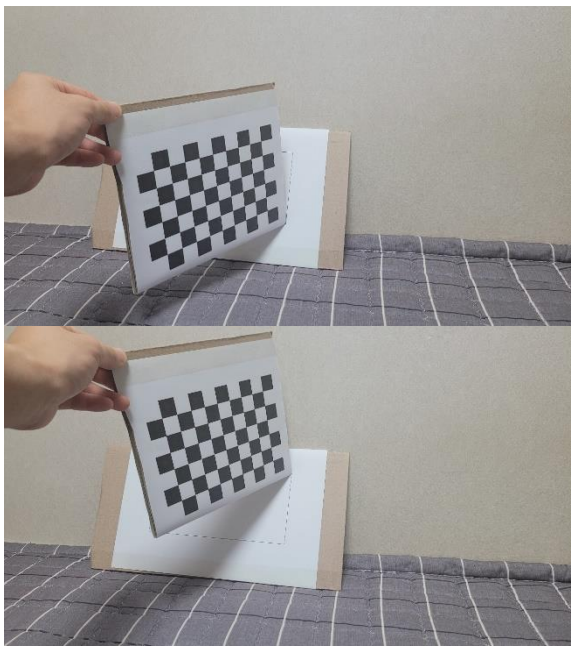


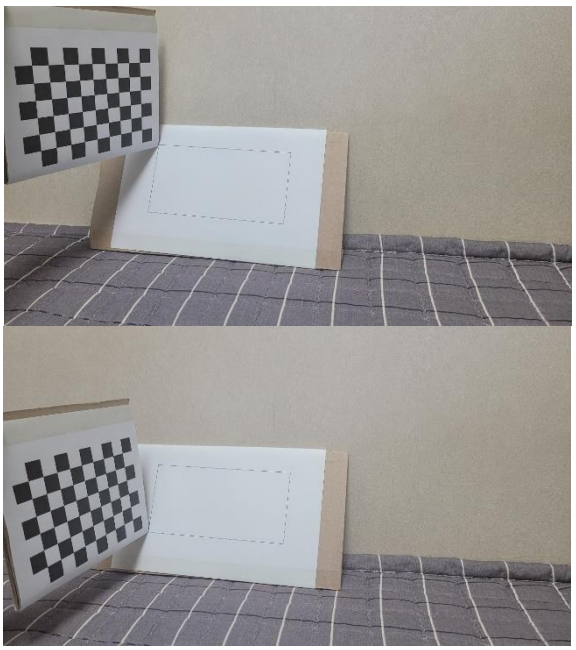
3.2. Case 1



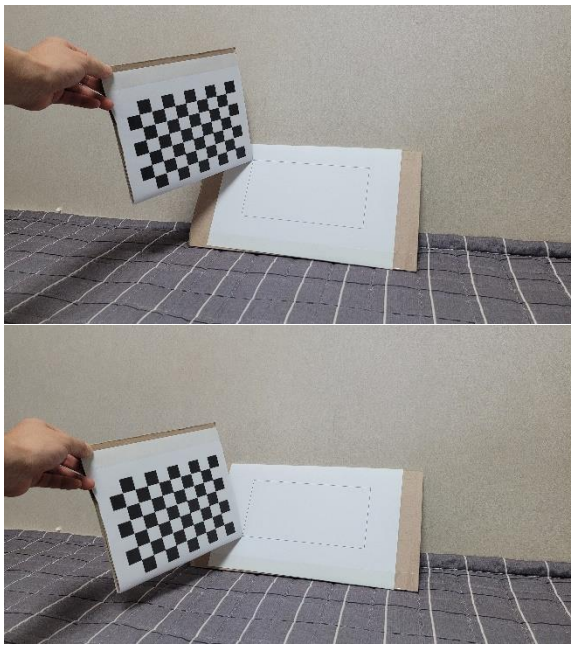


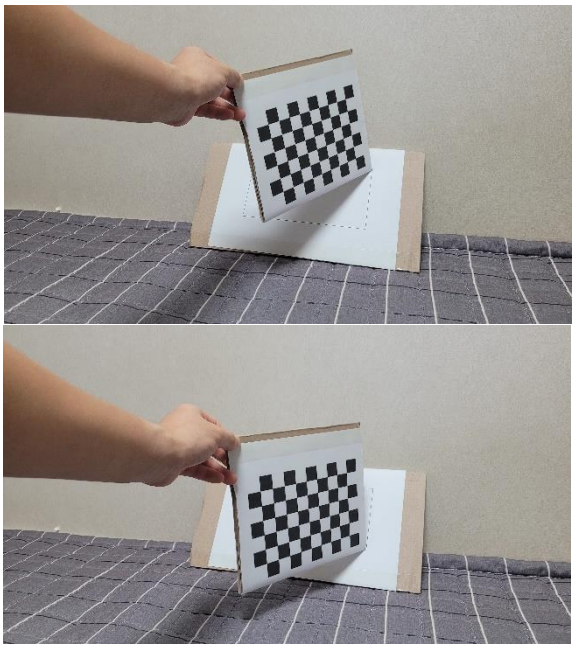
3.3. Case 2



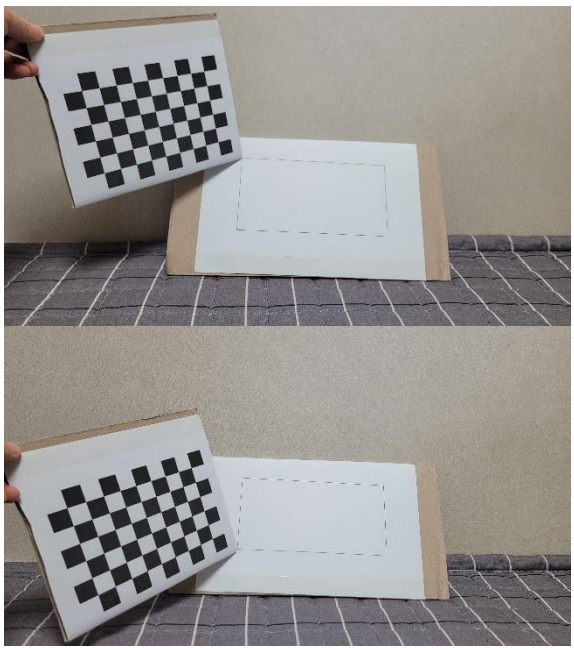


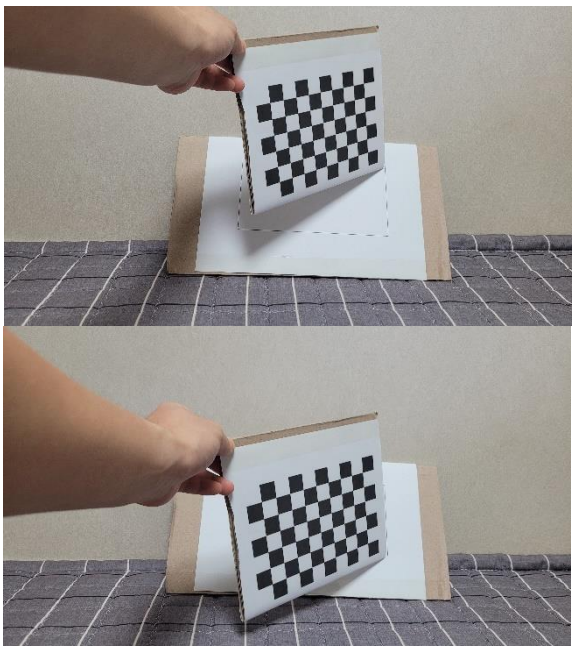
3.4. Case 3





3.5. Case 4





3.6. Case 5