// https://docs.opencv.org/4.x/dc/dbb/tutorial\_py\_calibration.html

import numpy as np import cv2 as cv import glob

##### FIND CHESSBOARD CORNERS - OBJECT POINTS AND IMAGE POINTS ########

# termination criteria

criteria = (cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ,(6,5,0)

objp = np.zeros((17\*24,3), np.float32)

objp[:,:2] = np.mgrid[0:24,0:17].T.reshape(-1,2)

# Arrays to store object points and image points from all the images. objpoints = [] # 3d point in real world space

imgpoints = [] # 2d points in image plane.

images = glob.glob('X:\Sem 6\AI & ML\Exp 6\Calib - Copy\cali\*.png')

for image in images:

img = cv.imread(image)

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

# Find the chess board corners

ret, corners = cv.findChessboardCorners(gray, (24,17), None)

if ret == True:

objpoints.append(objp)

corners2 = cv.cornerSubPix(gray, corners, (11,11), (-1,-1), criteria) imgpoints.append(corners)

# Draw and display the corners #v.drawChessboardCorners(img, chessboardSize, corners2, ret) cv.drawChessboardCorners(img, (24,17), corners2, ret) cv.imshow('img', img)

cv.waitKey(1000) cv.destroyAllWindows()

##############CALIBRATION#########################

ret, mtx, dist, rvecs, tvecs = cv.calibrateCamera(objpoints, imgpoints, frameSize, None, None)

np.savez('B.npz', mtx=mtx, dist=dist, rvecs=rvecs, tvecs=tvecs) print('Calibration Completed')

###### UNDISTORTION ######

img = cv.imread('X:\Sem 6\AI & ML\Exp 6\Calib\image\_29.jpg') cv.imshow('img', img)

h, w = img.shape[:2]

newCameraMatrix, roi = cv.getOptimalNewCameraMatrix(mtx, dist, (w,h), 1, (w,h))

# Undistort

dst = cv.undistort(img, mtx, dist, None, newCameraMatrix)

# crop the image x, y, w, h = roi

dst = dst[y:y+h, x:x+w] cv.imwrite('caliResult1.png', dst)

# Undistort with Remapping

mapx, mapy = cv.initUndistortRectifyMap(mtx, dist, None, newCameraMatrix, (w,h), 5) dst = cv.remap(img, mapx, mapy, cv.INTER\_LINEAR)

# crop the image x, y, w, h = roi

dst = dst[y:y+h, x:x+w] cv.imwrite('caliResult2.png', dst)

# Reprojection Error mean\_error = 0

for i in range(len(objpoints)):

imgpoints2, \_ = cv.projectPoints(objpoints[i], rvecs[i], tvecs[i], mtx, dist) error = cv.norm(imgpoints[i], imgpoints2, cv.NORM\_L2)/len(imgpoints2) mean\_error += error

print( "total error: {}".format(mean\_error/len(objpoints)) ) np.savez('B.npz', mtx=mtx, dist=dist, rvecs=rvecs, tvecs=tvecs)

# Load previously saved data with np.load('B.npz') as file:

mtx, dist, rvecs, tvecs = [file[i] for i in ('mtx','dist','rvecs','tvecs')]

def draw(img, corners, imgpts): corner = tuple(corners[0].ravel())

img = cv.line(img, corner, tuple(imgpts[0].ravel()), (255,0,0), 10) img = cv.line(img, corner, tuple(imgpts[1].ravel()), (0,255,0), 10)

img = cv.line(img, corner, tuple(imgpts[2].ravel()), (0,0,255), 10)

return img

def drawBoxes(img, corners, imgpts): imgpts = np.int32(imgpts).reshape(-1,2)

# draw ground floor in green

img = cv.drawContours(img, [imgpts[:4]],-1,(0,255,0),-3)

# draw pillars in blue color

for i,j in zip(range(4),range(4,8)):

img = cv.line(img, tuple(imgpts[i]), tuple(imgpts[j]),(255),3)

# draw top layer in red color

img = cv.drawContours(img, [imgpts[4:]],-1,(0,0,255),3) return img

criteria = (cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

#objp = np.zeros((24\*17,3), np.float32) objp = np.zeros((9\*6,3), np.float32)

#objp[:,:2] = np.mgrid[0:24,0:17].T.reshape(-1,2)

objp[:,:2] = np.mgrid[0:9,0:6].T.reshape(-1,2)

axis = np.float32([[3,0,0], [0,3,0], [0,0,-3]]).reshape(-1,3)

axisBoxes = np.float32([[0,0,0], [0,3,0], [3,3,0], [3,0,0],[0,0,-3],[0,3,-3],[3,3,-3],[3,0,-3] ])

for image in glob.glob('X:\Sem 6\AI & ML\Exp 6\Calib\image\_29.png'): img = cv.imread(image)

gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)

#ret, corners = cv.findChessboardCorners(gray, (24,17),None) ret, corners = cv.findChessboardCorners(gray, (9,6), None)

if ret == True:

print('Pose Estimation Started')

corners2 = cv.cornerSubPix(gray,corners,(11,11),(-1,-1), criteria) #print(corners2)

# Find the rotation and translation vectors.

ret, rvecs, tvecs = cv.solvePnP(objp, corners2, mtx, dist)

# Project 3D points to image plane

imgpts, jac = cv.projectPoints(axisBoxes, rvecs, tvecs, mtx, dist)

img = drawBoxes(img,corners2,imgpts) cv.imshow('img',img)

# k = cv.waitKey(0) # if k == ord('s'):

cv.imwrite('pose.png', img) cv.waitKey(0)

#cv.destroyAllWindows()