In [ ]:
```
import numpy as np
import cv2 as cv
import glob
import pickle
```

In [ ]:
```
################ FIND CHESSBOARD CORNERS - OBJECT POINTS AND IMAGE POINTS ######
chessboardSize = (10,7)
frameSize = (1280,720) # realsense d435i
# frameSize = (3024,4032) # iphone 13

# termination criteria
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)

objp = np.zeros((chessboardSize[0] * chessboardSize[1], 3), np.float32)
objp[:,:2] = np.mgrid[0:chessboardSize[0],0:chessboardSize[1]].T.reshape(-1,2)

size_of_chessboard_squares_mm = 20
objp = objp * size_of_chessboard_squares_mm

# 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.

# for RSd435i camera
images = glob.glob('cameraCalibration_srcdata/RSd435i/*.png')
# for iphone 13 camera
# images = glob.glob('cameraCalibration_srcdata/iphone13/*.png')
print(images)

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, chessboardSize, None)
    # If found, add object points, image points (after refining them)
    if ret == True:

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

        # Draw and display the corners
        cv.drawChessboardCorners(img, chessboardSize, corners2, ret)
        img = cv.resize(img, (720, 1080))
        cv.imshow('img', img)
        cv.waitKey(1000)

cv.destroyAllWindows()

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

ret, cameraMatrix, dist, rvecs, tvecs = cv.calibrateCamera(objpoints, imgpoints,

# Save the camera calibration result
calibration_data = {
    "cameraMatrix": cameraMatrix,
    "dist": dist,
    "rvecs": rvecs,
```

```
    "tvecs": tvecs
}

with open('cameraCalibration_srcdata/calibration_data_RSd435i.pkl', 'wb') as f:
    pickle.dump(calibration_data, f)

print("Calibration parameters saved successfully.")
```

```
['cameraCalibration_srcdata/RSd435i\\10_Color.png', 'cameraCalibration_srcdata/RS
d435i\\11_Color.png', 'cameraCalibration_srcdata/RSd435i\\12_Color.png', 'cameraC
alibration_srcdata/RSd435i\\13_Color.png', 'cameraCalibration_srcdata/RSd435i\\14
_Color.png', 'cameraCalibration_srcdata/RSd435i\\15_Color.png', 'cameraCalibratio
n_srcdata/RSd435i\\16_Color.png', 'cameraCalibration_srcdata/RSd435i\\17_Color.pn
g', 'cameraCalibration_srcdata/RSd435i\\1_Color.png', 'cameraCalibration_srcdata/
RSd435i\\2_Color.png', 'cameraCalibration_srcdata/RSd435i\\3_Color.png', 'cameraC
alibration_srcdata/RSd435i\\4_Color.png', 'cameraCalibration_srcdata/RSd435i\\5_C
olor.png', 'cameraCalibration_srcdata/RSd435i\\6_Color.png', 'cameraCalibration_s
rcdata/RSd435i\\7_Color.png', 'cameraCalibration_srcdata/RSd435i\\8_Color.png',
'cameraCalibration_srcdata/RSd435i\\9_Color.png']
Calibration parameters saved successfully.
```

In [ ]:
```
# Reprojection Error
mean_error = 0

for i in range(len(objpoints)):
    imgpoints2, _ = cv.projectPoints(objpoints[i], rvecs[i], tvecs[i], cameraMat
    error = cv.norm(imgpoints[i], imgpoints2, cv.NORM_L2)/len(imgpoints2)
    mean_error += error

print( "total error: {}".format(mean_error/len(objpoints)) )
```

```
total error: 0.03223261797395963
```

In [ ]:
```
# read the calibration data
with open('cameraCalibration_srcdata/calibration_data_RSd435i.pkl', 'rb') as f:
    calibration_data = pickle.load(f)
    cameraMatrix = calibration_data["cameraMatrix"]
    dist = calibration_data["dist"]
    rvecs = calibration_data["rvecs"]
    tvecs = calibration_data["tvecs"]

# read the image
img = cv.imread('cameraCalibration_srcdata/distorted/RSd435i/bg_removed.png')
h, w = img.shape[:2]

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

# Undistort
dst = cv.undistort(img, cameraMatrix, dist, None, newCameraMatrix)

# crop the image
x, y, w, h = roi
dst_crop = dst[y:y+h, x:x+w]

cv.imwrite('cameraCalibration_srcdata/undistorted/RSd435i/bg_removed.jpg', dst)
cv.imwrite('cameraCalibration_srcdata/undistorted/RSd435i/bg_removed_crop.jpg',
print("Undistorted image saved successfully.")
```

```
Undistorted image saved successfully.
```