```
import numpy as np
import cv2
import glob
import yaml
import pickle
import os

import pathlab
corner_x=8 # number of chessboard corner in x direction
corner_y=6 # number of chessboard corner in y direction

# termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.01)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((corner_y*corner_x,3), np.float32)
objp[:,:2] = np.mgrid[0:corner_x,0:corner_y].T.reshape(-1,2)

# Arrays to store object points and image points from a ll the images.
objpoints = [] # 3d point in real world space
pngpoints = [] # 2d points in image plane.

source_path = "E:\Plan B\Assignment3" #ini untuk source image kita simpan kat
mane??
#print(os.getcwd())
print('image found :',len(os.listdir(source_path)))

images = [source_path + '/' + f for f in glob.glob('*.png')]

#images = glob.glob('D:\Image Vision')


# path = 'results'
# pathlib.Path(path).mkdir(parents=True, exist_ok=True)

found = 0
for fname in images: # here, 10 can be changed to whatever number you like to
choose
    png = cv2.imread(fname) # capture frame by frame
    cv2.imshow('png', png)
    cv2.waitKey(500)
    print(fname)
    gray = cv2.cvtColor(png, cv2.COLOR_BGRA2GRAY)

    # find the chess noard corners
    ret, corners = cv2.findChessboardCorners(gray, (corner_x,corner_y), None)
    # if found, ass object points, image points (after refining them)
    if ret == True:

        objpoints.append(objp) #Certainly, every loop objp is the same in 3D
        corners2 = cv2.cornerSubPix(gray,corners,(20,5),(-1,-5),criteria)
        pngpoints.append(corners2)
        # Draw and display the corners
        png = cv2.drawChessboardCorners(png, (corner_x,corner_y), corners2, ret)
        found += 1
        cv2.imshow('chessboard', png)
        cv2.waitKey(0)
        # if you want to save images with detected corners
```

```
print("number of images used for calibration: ", found)

# when everything done, release the capture
# cap.release()
# cv2.destroyAllWindows()

#calibration of code
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, pngpoints,
gray.shape[::-1], None, None)

# transforms the matrix distortion coefficients to writeable lists
data= {'camera_matrix': np.asarray(mtx).tolist(), 'dist_coeff':
np.asarray(dist).tolist()}
print("camera_matrix",mtx)
print("..................................................................."
      ".....................................................................")
print("dist_coeff",dist)
# and save it to a file
with open("calibration_matrix.pickle", "w")as f:
    yaml.dump(data, f)

#undistort image

for fname in images: # here, 10 can be changed to whatever number you like to
choose
     #print(fname)
     png = cv2.imread(fname) # Capture frame-by-frame
     #cv2.imshow('png', png)
     #cv2.waitkey(500)
     h, w = png.shape[:2]

     newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx, dist , (w,h), 1, (w,h))
     #print(newcameramtx)
     #undistort
     dst = cv2.undistort(png, mtx, dist, None, newcameramtx)
     cv2.imshow('undistort', dst)
     cv2.waitKey(500)

     # crop the image
     x, y, w, h = roi
     dst = dst[y:y+h, x:x+w]
     #cv2.imshow('calibration.png',dst)
     cv2.imshow('undistort2', dst)
     cv2.waitKey(0)

     k = cv2.waitKey(1)
     if k % 256 == 27:
         # ESC pressed
         print("Escape hit, closing...")
         break

     '''elif k % 256 == 32:
         # SPACE pressed
         img_name = "opencv_frame_{}.png".format(img_counter)
         cv2.imwrite(img_name, frame)
         print("{} written!".format(img_name))
         img_counter += 1 '''
cv2.destroyAllWindows()
```