Lab 5 – Camera Calibration

- Chessboard calibration
- Projection of 3D points in image with camera model
- Calibrating your camera
- External calibration

5.1 Chessboard calibration

Run and test the file `chessboard.py`. This code detects corners in a chessboard pattern using openCV functions and shows the results of the detection for a series of images.

Use the available code to calibrate the camera used in the provided images (`left01.jpg` to `left13.jpg`). You need to use the function `calibrateCamera`. Help on this function can be found in https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#calibratecamera. Last parameter should be 0 (not using previous information for calibration), and do not specify a termination criteria (not using last parameter).

You need to create the following variables to receive the results of the camera calibration process.

You might show in the standard output the calibration results using:

```
print("Intrinsics: ")
print (intrinsics)
print("Distortion : ")
print(distortion)
for i in range(len(tvecs)):
    print ("Translations(%d) : " % i )
    print(tvecs[0])
    print ("Rotation(%d) : " % i )
    print(rvecs[0])
```

After a successful calibration, save the intrinsic and distortion matrices in a npz file using `savez` as in the following code.

```
np.savez('camera.npz', intrinsics=intrinsics, distortion=distortion)
```

Analyze the code for filling the 3D coordinates of the pattern. Imagine that you use another pattern with a different size what should you do to get the distance correctly evaluated?

Optional

You might improve the precision of the corner detection by using the `cornerSubPix` function after the call to `FindChessboardCorners`.

5.2 Projection of 3D points in the image

Use the function `cvProjectPoints()` to project an orthogonal line (normal) or a wireframe cube in one of the calibration image (the first for example) after the calibration code. Use the rotation and translation vectors from the calibration to project the 3D point in the correct positions in the image.

5.3 Using Camera on your computer

Modify the code to use the camera from your computer to process the chessboard (comment the code for reading the provided images to allow switching between camera and provided images). Calibrate you camera with several chessboard images (use `cvWaitKey()`to move the chessboard position and a pre-defined number of images, for example 10). Be careful to check if the available chessboard is like the one in the provided images. If not, modify the code accordingly. If you want real metric distances, you need to update the code with the real distances of the used chessboard. Save the calibration parameters to a file.

Sample code for accessing an image is `openCV` is as follow:

```
import cv2
capture = cv2.VideoCapture(0)
while (True):
    ret, frame = capture.read()
    cv2.imshow('video', frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
            break

capture.release()
cv2.destroyAllWindows()
```

5.4 External calibration

Calibrate a camera (using the given images or using your computer camera) and save the camera parameter file with another name.

Modify the previous examples to read the intrinsic and distortion parameters from the file and perform external parameters calibration (using function `solvePnP`) for a single image with the calibration pattern.

Read the file using the following code:

```
with np.load('camera_params.npz') as data:
    intrinsics = data['intrinsics']
    distortion = data['distortion']
print(intrinsics)
print(distortion)
```

Remember that in this case, the external calibration should be performed for each single image returning its position (rotation and orientation).

Optional

Compute the external calibration parameters for the camera of your computer while seeing a live feed of a pattern and project a cube (or normal vector) on the processed images live on the pattern.