Lab #6: Camera Calibration Using OpenCV

Fisea B., School of Computer Science, UNNC

March 26, 2025

Objective

To understand and implement camera calibration using a chessboard pattern and OpenCV. The lab will cover:

1. Detecting chessboard corners in images.
2. Establishing correspondences between 3D real-world points and 2D image points.
3. Computing the camera's intrinsic parameters and distortion coefficients.

Lab Outline

1. Introduction

What is Camera Calibration?

- Camera calibration is the process of estimating the intrinsic and extrinsic parameters of a camera.
- **Intrinsic parameters**: Focal length, optical center, skew, etc.
- **Extrinsic parameters**: Rotation and translation of the camera relative to the world.

Why is Camera Calibration Important?

- Corrects lens distortion:
  - **Radial distortion**: Causes straight lines to appear curved.
  - **Tangential distortion**: Occurs due to lens misalignment.
- Enables accurate 3D reconstruction.
- Essential for computer vision tasks like augmented reality, robotics, and 3D mapping.

Example: Stereo Applications

For stereo applications(we will discuss in the next lecture), distortions need to be corrected first. To find these parameters, we must provide some sample images of a well-defined pattern (e.g., a chessboard). We find some specific points of which we already know the relative positions (e.g., square corners in the chessboard). We know the coordinates of these points in real-world space and their corresponding coordinates in the image, so we can solve for the distortion coefficients. For better results, we need at least 10 test patterns.

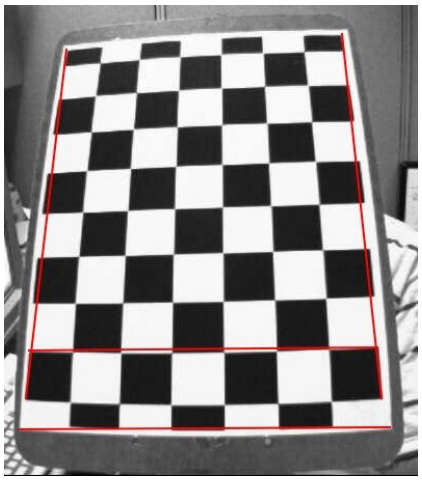

Figure 1: (a)

Figure 2: (b)

Figure 3: (a) radial distorted image (b) Corrected using camera caliberation parameters

2. Tools and Libraries

- **OpenCV**: A library for computer vision tasks.
- **NumPy**: A library for numerical computations in Python.
- **Chessboard Pattern**: A known pattern used for calibration.

3. Lab Setup

Requirements:

- Python installed with OpenCV and NumPy.
- A set of images of a chessboard pattern taken from different angles. (download it from moodle)
- A chessboard with a known number of inner corners (e.g., 7x6).

Environment Setup:

To set up the environment for this lab, follow these steps:

1. Create a new conda environment:

```
conda create --name CV python=3.8
```

2. Activate the environment:

```
conda activate CV
```

3. Install the required libraries:

```
pip install opencv-python opencv-contrib-python numpy
```

4. Install Jupyter Notebook (optional, for interactive coding):

```
pip install notebook
```

4. Steps in Camera Calibration

1. **Prepare Object Points:**
   - Define the 3D coordinates of the chessboard corners in the real world.
   - Example: `objp = np.zeros((6*7, 3), np.float32)`.
   - For simplicity, assume the chessboard is stationary on the XY plane (so $Z = 0$ always), and the camera is moved accordingly. This reduces the problem to finding only X and Y values.
   - The 3D points are called **object points**, and the 2D points in the image are called **image points**.
2. **Detect Chessboard Corners:**
   - Use `cv.findChessboardCorners` to detect corners in the images.
   - Pass the pattern type (e.g., 7x6 grid for a chessboard with 7x7 internal corners).
   - The function returns the corner points and a boolean (`retval`) indicating whether the pattern was detected.
   - Refine the corners using `cv.cornerSubPix` for better accuracy.
3. **Store Correspondences:**
   - Store the 3D object points (`objpoints`) and 2D image points (`imgpoints`) for each image.

- These correspondences are used to compute the camera parameters.

4. **Calibrate the Camera:**
   - Use `cv.calibrateCamera` to compute the camera matrix and distortion coefficients.
   - The function returns the following:
     - **retval**: The root mean square (RMS) reprojection error, indicating the accuracy of the calibration.
     - **cameraMatrix**: The 3x3 intrinsic matrix containing focal lengths (`fx`, `fy`) and optical center (`cx`, `cy`).
     - **distCoeffs**: The distortion coefficients (e.g., radial and tangential distortion).
     - **rvecs**: A list of rotation vectors (one for each image) describing the rotation of the world frame relative to the camera frame.
     - **tvecs**: A list of translation vectors (one for each image) describing the translation of the world frame relative to the camera frame.
5. **Undistort and Crop the Image:**
   - Use the computed camera matrix and distortion coefficients to undistort an image.
   - The `cv.undistort` function corrects lens distortion in the image.
   - Optionally, use `cv.getOptimalNewCameraMatrix` to compute a new camera matrix for better results.
   - Crop the undistorted image to remove black borders caused by the undistortion process.
   - Example:
     - Load a test image: `img = cv.imread('./chessboard/left12.jpg')`.
     - Compute the optimal new camera matrix: `newcameramtx, roi = cv.getOptimalNewCameraMatrix(mtx, dist, (w, h), 1, (w, h))`.
     - Undistort the image: `dst = cv.undistort(img, mtx, dist, None, newcameramtx)`.
     - Crop the image using the region of interest (ROI): `dst = dst[y:y+h, x:x+w]`.
     - Save or display the undistorted and cropped image.
6. **Evaluate Results:**
   - Display the undistorted image to verify the calibration.
   - Compare the original distorted image with the undistorted image to observe the correction of lens distortion.
   - Check the printed camera matrix and distortion coefficients for accuracy.

5. Code Implementation

```
import numpy as np
import cv2 as cv
import os

# 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((6*7,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:6].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.

# Specify the directory path
directory_path = 'chessboard'

# List all files in the directory
image_files = [f for f in os.listdir(directory_path) if f.endswith(
    '.jpg')]

for fname in image_files:
    # Construct the full file path
    file_path = os.path.join(directory_path, fname)

    # Read the image
    img = cv.imread(file_path)
    if img is None:
        print(f"Error: Unable to read image {file_path}")
        continue

    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    # Find the chess board corners
    ret, corners = cv.findChessboardCorners(gray, (7,6), 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(corners2)

        # Draw and display the corners
        cv.drawChessboardCorners(img, (7,6), corners2, ret)
        cv.imshow('img', img)
        cv.waitKey(500)
```

```
cv.destroyAllWindows()

# Calibrate camera
ret, mtx, dist, rvecs, tvecs = cv.calibrateCamera(objpoints,
    imgpoints, gray.shape[::-1], None, None)

# Print results
print("Camera matrix:\n", mtx)
print("Distortion coefficients:\n", dist)

# Undistort an image
img = cv.imread('./chessboard/left12.jpg') #test image
h,  w = img.shape[:2]
newcameramtx, roi = cv.getOptimalNewCameraMatrix(mtx, dist, (w,h),
    1, (w,h))

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

# crop the image
x, y, w, h = roi
dst = dst[y:y+h, x:x+w]
cv.imwrite('calibresult.png', dst)

# Display undistorted image
cv.imshow('Undistorted Image', dst)
cv.waitKey(0)
cv.destroyAllWindows()
```

6. Expected Output

- Display the detected corners on the chessboard images.
- Print the camera matrix and distortion coefficients.
- Show undistorted images to demonstrate the effect of calibration.

7. Discussion

Challenges:

- Ensuring the chessboard is fully visible in the images.
- Handling poor lighting or blurry images.

Applications:

- Augmented reality.
- 3D reconstruction.
- Robotics and autonomous navigation.

8. Exercises (Lab or Homework Assignment)

1. Capture your own set of chessboard images using your phone camera and calibrate the camera. This exercise will help you understand the intrinsic parameters of your phone camera, such as focal length, optical center, and distortion coefficients.
2. Modify the code to use a different calibration pattern (e.g., a circle grid), see Figure 4
3. Experiment with different termination criteria for corner refinement.

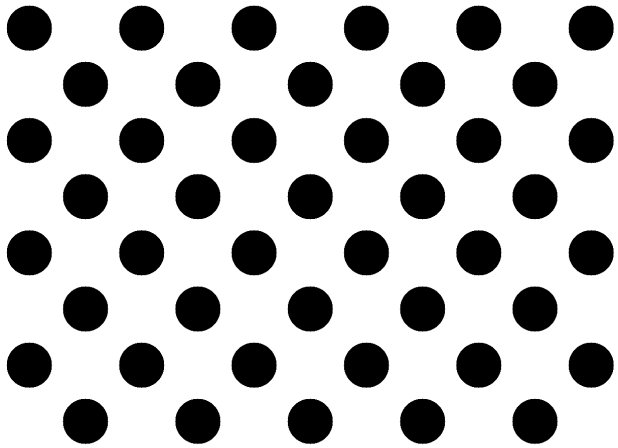

Figure 4: Circle grid

9. References

- OpenCV documentation: `https://docs.opencv.org/`.
- Zhang, Z. (2000). A flexible new technique for camera calibration.