Lab 6: Basic Computer Vision

1.104 Team
Department of Civil and Environmental Engineering
Massachusetts Institute of Technology

Introduction

In this lab, you will learn about camera calibration, a fundamental process in computer vision. Camera calibration is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) and the 3D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters). By the end of this lab, you'll have calibrated your own smartphone camera and understand how these parameters can be used in various computer vision tasks.

1 The Pinhole Camera Model

1.1 Theoretical Background

The pinhole camera model is the simplest mathematical model of a camera. In this model, light passes through a single point (the pinhole) and projects an inverted image on the opposite wall of the camera. The pinhole camera model is shown in Figure 1

The pinhole camera model describes how 3D world points are projected onto a 2D image plane. This projection can be mathematically represented as:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} p_x^c \\ p_y^c \\ p_z^c \\ 1 \end{bmatrix} \tag{1}$$

This equation can be simplified as:

$$s\mathbf{m} = K[R|t]\mathbf{M} \tag{2}$$

Where:

- (p_x^c, p_y^c, p_z^c) are the 3D coordinates of a point in the world coordinate system
- (u, v) are the 2D coordinates of the projection point on the image plane
- s is a scale factor
- K is the camera intrinsic matrix (also called the calibration matrix)
- $[R|t]$ is the extrinsic matrix, describing the camera's position in the world

1.2 Camera Parameters

There are two sets of parameters we need to determine during camera calibration:

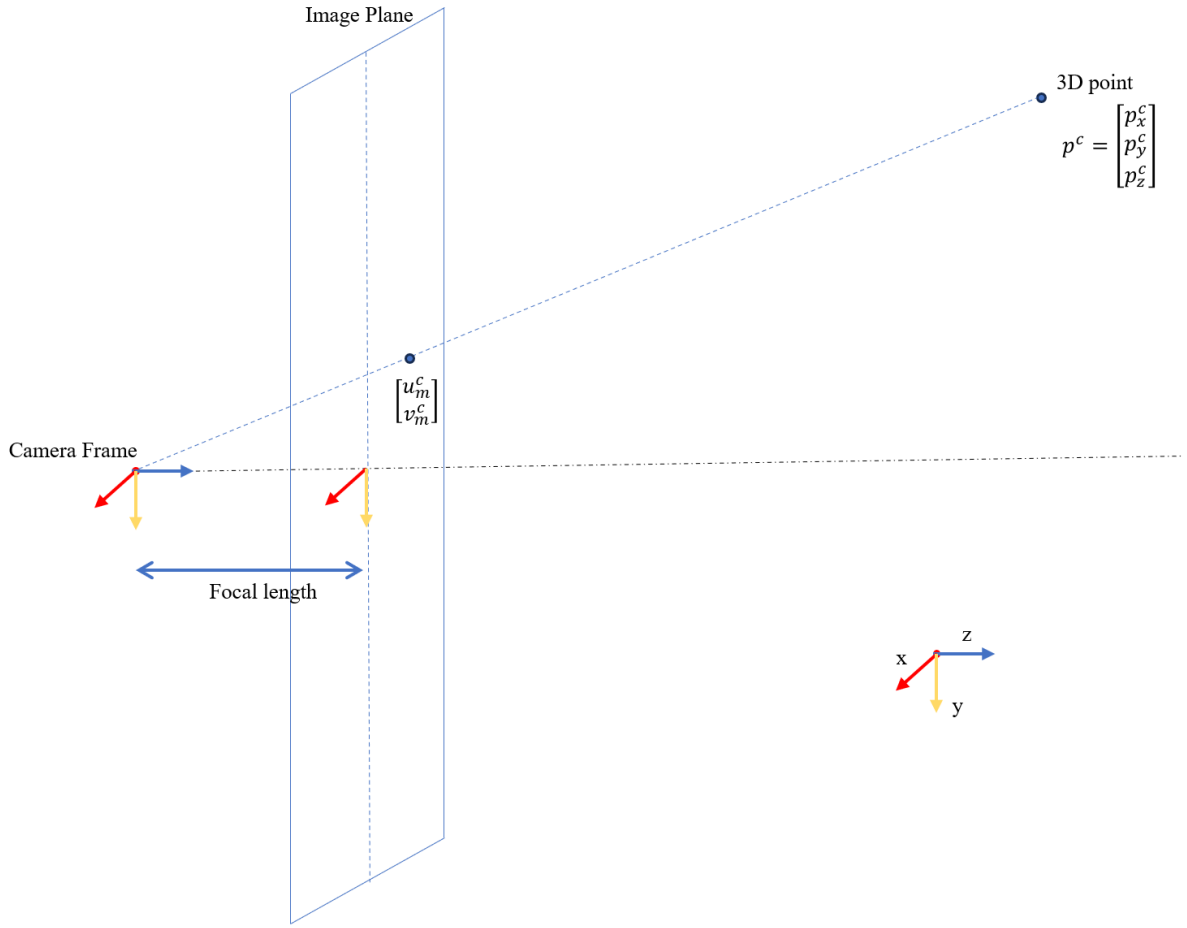

Figure 1: Illustration of the pinhole camera model

1.2.1 Intrinsic Parameters

These parameters define the internal optical and geometric characteristics of the camera:

- **Focal Length** (f_x, f_y): The distance from the camera center to the image plane, measured in pixels for the x and y directions. If the pixel is square, $f_x = f_y$.
- **Principal Point** (c_x, c_y): The coordinates of the principal point (where the optical axis intersects the image plane) in the image coordinate system.
- **Skew Coefficient**: A parameter describing the angle between the x and y axes. For most cameras, it is 0, meaning the x and y axes are perpendicular.

The intrinsic matrix K is represented as:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Where s is the skew coefficient, which is typically zero for modern cameras.

1.2.2 Extrinsic Parameters

These parameters define the position and orientation of the camera in the world coordinate system:

- **Rotation Matrix** (R): A 3×3 matrix that describes the orientation of the camera in the world coordinate system.
- **Translation Vector** (t): A 3×1 vector that describes the position of the camera center in the world coordinate system.

1.2.3 Lens Distortion Parameters

Real cameras have lenses that introduce distortions. These distortions can be modeled with additional parameters:

- **Radial Distortion Coefficients** (k_1, k_2, k_3): Model the radial distortion of the lens.
- **Tangential Distortion Coefficients** (p_1, p_2): Model the tangential distortion caused by the lens not being perfectly parallel to the image plane.

2 Calibration Method: Zhang's Method

In this lab, we will use Zhang's method for camera calibration. This method requires the camera to observe a planar pattern (a checkerboard in our case) from multiple viewpoints. The pattern can be freely moved (or the camera can be moved while the pattern stays fixed).

The basic steps of Zhang's method are:

1. Detect the corners of the checkerboard in multiple images taken from different viewpoints.
2. Establish the correspondence between the 3D coordinates of the corners on the checkerboard and their 2D projections in the images.
3. Solve for the camera intrinsic and extrinsic parameters using these correspondences.
4. Refine the parameters using nonlinear optimization to minimize the reprojection error.

Problem 1: Why we use the corners as the features for detection?

3 Experimental Procedure

3.1 Equipment Required

- A smartphone with a camera
- A printed checkerboard pattern (provided)
- A computer with Python and the necessary libraries installed

3.2 Part 1: Preparation and Data Collection

1. Check the checkerboard pattern provided with this lab. Make sure it's printed on a flat surface and is not bent or damaged.
2. On the lab laptop, open the `VS Code`, and press `Ctrl+‘` to open the terminal.
3. Run the following command to move to the `Desktop` folder (use `tab` to auto-complete the folder name):

   ```
   cd .\Desktop\
   ```

4. Run the following command to clone the lab repository to your desktop:

   ```
   git clone https://github.com/LiJiarui111/MIT-1104-Basic-CV-Lab
   ```

5. Select "Open Folder" in VS Code, and open the folder `MIT-1104-Lab5-Basic-CV-2025` on the desktop (select "I trust the authors" if prompted).
6. Create two folders on your computer named `calibration_images` and `object_images` under your project directory where you will store your images.
7. Take approximately 15 photos of the checkerboard pattern using your smartphone. Follow these guidelines:
   - Make sure the entire checkerboard is visible in each image.
   - Take photos from various angles and distances.
   - Include some photos where the checkerboard is near the edge of the frame.
   - Include some photos where the checkerboard is rotated.
   - Make sure the checkerboard is well-lit and in focus.
8. Next, take 5 to 10 photos of your personal belongings (such as your laptop and smartphone) along with the fruit models provided in the lab. Ensure you capture these items from different angles, distances, and varying lighting conditions.
9. Transfer the images to your computer. Place the checkerboard photos into the `calibration_images` folder, and the photos of your belongings/fruits into the `object_images` folder. For the transfer, I recommend using email to send the images to yourself, and then downloading them to your computer.

10. Rename the images sequentially within their folders, such as `img1.jpg`, `img2.jpg`, etc.

Problem 2: Why do we need to take multiple images of the checkerboard? Wouldn't a single image be sufficient for calibration?

3.3 Part 2: Camera Calibration

1. Open your terminal in VS Code (use `Ctrl+‘` to open the terminal).
2. Navigate to the project folder containing the Python scripts (you should have already been in this folder, but if not, navigate to the folder using `cd` command).
3. Before running any code, **read the README.md file to install necessary packages**, understand the requirements and potential issues (you can use the preview button on the up right corner of the file to view it).
4. Run the following command to perform the camera calibration:

```
python camera_calibration.py --dir .\calibration_images\
--rows 7 --cols 5 --square_size 0.030
```

   Note: The above command assumes a 7×5 internal corner checkerboard pattern with squares of 30mm (0.030m) size. Adjust the parameters if your checkerboard is different.

5. The script will process all the images in the `calibration_images` folder, detect the checkerboard corners, and calculate the camera parameters.
6. After the calibration is complete, the script will display the reprojection error and save the calibration parameters to a file named `camera_calibration.yaml`.
7. Check the `camera_calibration.yaml` file. Taking the intrinsic parameter matrix K as an reference, what are the focal length, principal point, and distortion coefficients? Are f_x and f_y the same?
8. The script will also generate 5 undistorted versions of your images and display them alongside the original images. Is the distortion significant?

Problem 3: What does the reprojection error measure, and why is it important in camera calibration?

3.4 Part 3: Visualize the Results

1. Run the visualization script to see the effects of calibration:

```
python visualize_calibration.py --calib_file camera_calibration.yaml
--image_path .\calibration_images\img1.jpg
```

2. The script will display:
   - The original image
   - The undistorted image
   - A visualization of the camera's position and orientation relative to the checkerboard

   Tune the visualization parameters to get a better view. For example, you can change the `pad` at line 151 of `visualize_calibration.py`.

3. Try running the script with different images from your calibration set. Is the estimated camera position and orientation consistent with the actual position and orientation of the camera?

Problem 4: Now you have the camera's relative position and orientation to the checkerboard coordinates (world coordinates). What do you think this information can be used for?

Optional Problem: Suppose you have a calibrated camera (which means the intrinsic parameters are known), is it possible to get the position and orientation information of the camera without using the checkerboard? Moreover, given two images of one object taken by the same camera, is it possible to get the translation and rotation between the camera frames of the two images?

4 Object Recognition with YOLO

4.1 Introduction to Object Recognition

In the previous sections, you learned how a camera maps the 3D world to a 2D image using calibration. Now, we will explore how computers can actually "understand" what is inside those 2D images.

We will use a popular AI model called **YOLO** (You Only Look Once). YOLO is a state-of-the-art, real-time object detection system. It is designed to look at an image and immediately recognize and locate multiple objects within it. Because the model provided on your laptops has been pre-trained on hundreds of thousands of images, it already knows what common objects—like laptops, cell phones, and apples—look like. You don't need to write any complex AI code to use it!

4.2 Running the Object Detector

Let's test the YOLO model on the photos you took of your belongings and the fruit models.

1. Open your terminal in VS Code (use `Ctrl+‘` to open the terminal).
2. Navigate to the project folder containing the Python scripts (you should have already been in this folder, but if not, navigate to the folder using the `cd` command).
3. Run the following command to apply the YOLO model to your images:

```
python run_yolo.py --dir .\object_images\
```

4. The script will process all the images in the `object_images` folder and save the annotated results in a new folder named `yolo_results`.

5. Navigate to the `yolo_results` folder and open the images to view the output.

4.3 Expected Output and Analysis

When you open the processed images, you should see "bounding boxes" (colored rectangles) drawn around the objects the model successfully recognized.

Above each box, there will be two pieces of information:

- **The Label**: What the AI thinks the object is (e.g., "laptop", "cell phone", "apple").
- **The Confidence Score**: A decimal number between 0.0 and 1.0. This represents how certain the AI is about its prediction (where 0.85 means it is 85% confident).

Problem 5: Look closely at the images in your `yolo_results` folder. Did the model make any mistakes? For example, did it fail to detect an object entirely, or did it label an object incorrectly? Based on how the photo was taken, what factors (such as lighting, angle, distance, or the object being partially hidden) do you think might have confused the model?

Problem 6: Why is it useful for an AI model to output a "confidence score" rather than just giving a definitive label? Give a brief example of how a confidence score might be used to make safety decisions in a real-world engineering application, like a self-driving car.

5 Report

After completing the lab, prepare a brief report that includes:

1. The intrinsic parameters of your camera
2. The distortion coefficients and the reprojection error
3. The visualization of the camera's position and orientation relative to the checkerboard.
4. Your answers to the problems

6 Conclusion

In this lab, you have learned about the pinhole camera model and how to calibrate a camera to determine its intrinsic and extrinsic parameters. These parameters are fundamental in many computer vision applications. Understanding camera calibration provides a solid foundation for more advanced topics in computer vision.

7 References

1. Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), 1330-1334.

2. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc.

3. OpenCV Documentation: Camera Calibration and 3D Reconstruction. https://docs.opencv.org/master/d9/d0c/group__calib3d.html

4. Yang, Heng. 2023. "Optimal Control and Estimation." November 14, 2023. https://hankyang.seas.harvard.edu/OptimalControlEstimation/.