ELEC 6910A Project 3

Camera Calibration

Professor: TAN, Ping

Due Nov. 19, 2023

1 Camera Calibration

In this project, you will calibrate the intrinsic and extrinsic matrix of a **SINGLE** camera. Before we step into the calibration, we will walk you through the basic background of the camera calibration. In the programming section, you will first learn to find the functions provided by opencv or matlab library to detect the feature point on the checkboard or the apriltag. Then, you can use the location of the feature points on different images to compute the camera intrinsic matrix. Given the known intrinsic matrix, you can further compute the camera extrinsic matrix for each image. Finally, you will also need to compute the distortion parameters of the camera.

In this project, you should implement the code by yourself. Every script and function you write in this section should be included in the *matlab/* or *python/* directory. The file called "main.py" or "main.m" will be run as the program entry to evaluate the result.

The data is exclusively used in this course only. You can not release the dataset to any other entity. You should sign the data usage agreement file, and submit it with the code and result. The submission without the signed data usage agreement file will be ignored, and you will get 0 points in this assignment.

Post questions to Canvas so everybody can share unless the questions are private. Please look at Canvas first if similar questions have been posted.

2 Method

2.1 Dataset

This project utilizes a specialized dataset provided by an anonymous company designed for camera calibration, both intrinsic and extrinsic. The dataset comprises 36 images, each featuring an individual, Baowen ZHANG, holding an AprilTag calibration board. This uniquely designed board aids in accurate and efficient calibration processes. It features tags from the "Tag36h11" family, well-known for its robustness in various scenarios.

Each tag on the calibration board measures 8.8 cm in size, and the spacing between the tags is precisely 2.64 cm. This specific setup ensures a high degree of accuracy in the calibration process, making it suitable for various computer vision applications that require exact measurements and pose estimation.

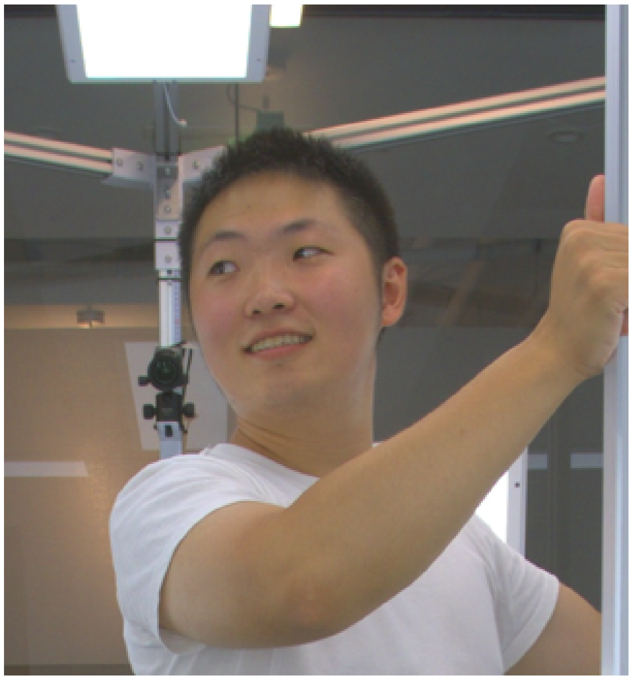

Figure 1: This is Baowen Zhang, please say Hi to him.

This dataset is confidential and cannot be disclosed or shared publicly. All individuals accessing or using this dataset must sign a Data Usage Agreement to ensure the privacy and security of the data.

2.2 Marker Dectection

In recent years, AprilTag has gained significant attention as a robust visual fiducial marker system designed to provide accurate and computationally-efficient pose estimation. These markers are widely used for various applications such as robotics, augmented reality, and camera calibration. Unlike traditional QR codes, AprilTags are engineered to be quickly and accurately identified, even under partial occlusion or extreme viewing angles.

In Matlab, the AprilTag detection can be accomplished using the detectAprilTag function from the Computer Vision Toolbox. This function takes an image as input and returns detected tag corners and IDs. In Python, libraries such as apriltag can be used, where the detect function is commonly employed. This function also takes an image and returns a list of dictionaries, each containing the tag corners, ID, and other geometric information.

Both Matlab and Python methods return the tag's ID and corner coordinates in the image frame, which can be further utilized for calculating the tag's pose relative to the camera. These returned values serve as essential data points for various downstream applications, making AprilTag an indispensable tool in computer vision projects.

2.3 Intrinsic Matrix esstimation

To calibrate the intrinsic parameters of a camera, AprilTags provides a convenient and efficient method. The process begins by capturing images of an AprilTag calibration board from different angles and orientations. For each image, the corners of the AprilTags are detected, and these corner points serve as correspondences for homography estimation.

Rather than utilizing the built-in functions for homography computation available in OpenCV or MATLAB, this project requires using a custom function to estimate the homography, which is implemented in the previous project 2. The built-in functions, like cv2.findHomography(), are still not allowed. Once the homography is estimated for multiple views, these matrices can be utilized to extract the intrinsic parameters of the camera. The output should be a 3×3 intrinsic matrix only.

For more details on the methodologies and concepts involved in this process, please refer to the course slides and materials.

2.4 Extrinsic Matrix esstimation

Calibrating the extrinsic parameters of a camera involves determining its position and orientation relative to a world coordinate system. AprilTags serve as an effective tool for this purpose. By capturing images of an AprilTag calibration board from various angles, you can establish point correspondences needed for extrinsic parameter estimation.

It's essential to note that each camera's extrinsic parameters are calibrated independently. This means that the position and orientation are unique since the extrinsic matrices are based on the world coordinates under the Apriltag plane. The output should be a 4×4 matrix for each image.

For a comprehensive understanding of the methods and algorithms involved in extrinsic calibration using AprilTags, please consult the relevant course materials and slides.

2.5 Camera Distortion Parameters

Once you've successfully calibrated your camera's intrinsic and extrinsic parameters using AprilTags, the next step is to correct for lens distortions. Typically, there are two main types of distortions: radial and tangential. Radial distortion manifests as either "barrel" or "pincushion" distortions and is often described by parameters k_1, k_2 , and k_3. Tangential distortion occurs when the lens and the image plane are not parallel and is usually represented by parameters p_1 and p_2.

To estimate these distortion parameters, you can make use of the same sets of point correspondences obtained during intrinsic and extrinsic calibration. A common approach is to optimize a cost function that measures the difference between the observed image points and the ones projected from 3D space, taking into account the distortion parameters.

You should output k_1, k_2, k_3, p_1, p_2 in a row. You should output the evaluated result under this camera distortion parameter in any format.

Note: you can optimize k_1 and k_2 only and leave k_3, p_1, p_2 as 0. To optimize all parameters, you can use the non-linear optimization function in matlab and python.

3 Submission

You only need to upload **ONE** zip file containing the code, the README file, and the results. The code should be in Matlab format. Implementing another language will be skipped, and get 0 pts in this project.

Many of the algorithms you will be implementing as part of this project are functions in the Matlab image processing toolbox. You are not allowed to use these functions in this project without permission. However, You may compare your output to the output generated by the image processing toolboxes to ensure you are on the right track. You should write the necessary information in the README file, including the environment, the cmd, the file format, etc. You should make the code in the submission self-contained. The script output should be matched with the file in the results folder.

Cite the paper, GitHub repo, or code url if you use or reference the code online. Please keep academic integrity; plagiarism is not tolerated in this course.

4 Tips

You can access the Matlab with a proper subscription via virtual-barn.

You can use python packages like, cv2, numpy, to finish this project.