CS 763/CS 764: Lab Five

Whose Camera is that?

- Announced: 02/14 14:00, Posted: 02/14 14:00 , Inlab Due: 02/14 16:30 Outlab Due: 02/19 22:30

We can project a 3D point in the real world to a 2D point in the image using the projection matrix (or camera matrix). This matrix has 2 components, the extrinsic camera matrix that includes the parameters for translation and rotation and the intrinsic camera matrix that includes the parameters for focal length, principal point offset and axis skew. The extrinsic camera matrix will vary depending on, say, where the shooter is present, but the intrinsic camera matrix remains the same (in most cases, unless, e.g. the shooter alters the zoom setting).

The forward problem is the projection, given the camera matrix. The inverse problem is to find that matrix. Whose camera is that?

1 Inlab: Elements of Camera Calibration

To discover the camera, we need to know something about the relation of a few items in the world and the corresponding image. A flat pattern (such as a chessboard) is often used for camera calibration since we can take the plane of the object to be the X-Y plane (i.e., the Z-coordinate of each point is 0) and the repeated pattern allows one to extrapolate the 3D coordinates of the other points, given that we fix a point.

Fortunately a flat pattern is easily available. For our lab we will use the drilled hole pattern which you can find on the walls of the room (sound dampners).

1.1 Task: Take two photos

Look at Figure 1 to get an idea of the kind of photos that you have to take. Save the images that you took in `./data/calib_images` directory.

You are going to take two, yes, two photos.

Figure 1: We took these photos. You should take two pictures like this. Notice the points at the lower left corner which are marked as A, B, C, D, E, F

1.2 Calibrate!

Perform the following tasks.

1. Assume the picture of the pattern ("the plane") you have taken lies on the Z=0 plane. Furthermore assume that point A is on the Z axis. It is given to you that the distance between two adjacent (say in the X-direction) holes in the pattern is 3cm. Record the 3D coordinates of six points in each image in your notebook.

2. Figure out the sensor coordinates for these six points (any which way you like) (recall Lab00). You need to do these for the two photos (pictures, images) you have taken.

3. For any one picture, highlight the chosen points on the image and save the image as `./data/true.jpg`. For example, draw a circle on the location of the chosen points.

4. Next, using `cv2.calibrateCamera()`, based on the method described in the class, obtain the intrinsic parameters.

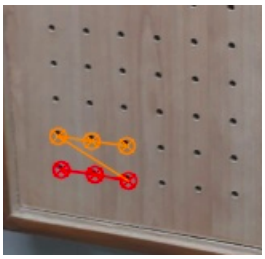
   Q: What are the returned values of this function? Can you explain what each of them means? It's OK if you don't know all but you should at least know those which have been discussed in the class.

5. (Did that work?) Now that you have calculated the camera parameters, we want to make sure that these are the correct parameters. One way to do this is to pretend we don't have our camera or image, and use the OpenCV function `projectPoints()` to act like a camera for us.

   The game plan is to assume that the sensor coordinates you have found earlier (in Step 2) is the ground truth, and figure out the the re-projection error. For the same image that you chose in Step 3,

   - For each of the world coordinate point as hypothetically seen from the first camera, calculate the projected points using `cv2.projectPoints()`.
   - Calculate the mean L2 norm between the actual and projected image points (the re-projection error). Lower values of re-projection error mean that our camera was calibrated successfully.
   - Draw the projected points on the image and save it as `./data/projected.jpg`.

(a) Output image



(b) Zoomed left image

Figure 2: Sample image after drawing the projected points using our synthetic camera. (We don't require that you draw the points in the exact same way.)

1.3 Run

We don't expect you to parse the command line arguments for this lab. Your code should save the images as mentioned in the above sections and should report the calculated re-projection error on the command line.

Run: `python3 camera_calibrate.py`

1.3.1 Action

Get your annotated pictures (`true` and `projected`) approved by your TA. You can take as many pictures as you like but you must submit (and work on) exactly two photos.

2 Outlab

2.1 What camera is that?

In this part we have some data that was collected from two pictures (of the same pattern used in the inlab) taken by your TA. We don't know where she took the picture from, other than the fact that it was as if she was with you during the lab session. Your goal is to discover **her** camera parameters.

You are given a text file named `points.txt` in the `./data` folder which contains a list of 12 points that represent 2D coordinates of some 6 points in the two images (the same set of points are taken for both the images). The first 6 points in the text file correspond to image 1 and the remaining correspond to image 2.

Your task is to find the re-projection error just like the inlab.

2.1.1 Run

Your code should take the path of the text file in the command line argument and return the re-projection error on the command line.

Run: `python3 reproject.py -f <path_to_text_file>`

2.2 Not released at this time

Submission Guidelines

1. The top assignment directory should contain a folder `inlab` which includes the inlab submission as detailed below.

   (a) Refer to post 46 for format of `readme.txt`. Do include a `readme.txt` (telling me whatever you want to tell me including any external help that you may have taken).

   (b) All other instructions mentioned earlier apply.

   (c) Your inlab submission folder should look something like:

```
150050001_130010009_140076001_lab05_camera/
├── inlab
│   ├── ReflectionEssay.pdf
│   ├── code
│   │   ├── camera_calibrate.py
│   │   └── reproject.py
│   ├── data
│   │   ├── calib_images
│   │   │   └── <Images taken for task 1.1>
│   │   ├── projected.jpg
│   │   └── true.jpg
│   ├── convincingDirectory
│   └── readme.txt
└── outlab
    └── data
        └── points.txt
```