CS 763/CS 764: Lab Five

Camera Calibration and Back Projection

- Announced: 03/11. Due: 03/17 5PM
- This is a group assignment

We project a 3D point in the real world to a 2D point in the image using the camera matrix. This matrix has 2 components: the extrinsic parameters that account for translation and rotation, and the intrinsic parameters that includes the parameters for focal length, principal point offset, and axis skew. The extrinsic camera matrix will vary depending on, say, where the picture taker was 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, among other things, to find the camera matrix and back projecting. You will work with both directions in this lab exercise.

1 Whose Camera is That?

First, we look at the inverse problem. In this part you shall use one or more images to determine camera parameters.

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.

An example of such a regular pattern is the drilled hole pattern shown in Figure 1. Your group should look for a similar pattern in your immediate surroundings. Try and find a pattern along a vertical surface like a wall, this will help you save effort in the next lab.

1.1 Focus!

Just like we know how much memory we have on our computer, and we can verify the same, we would like to verify that the focal length of the camera we have in our smartphone is as advertised. This should be doable right?

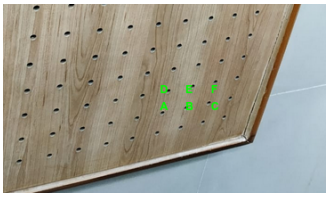
The cool thing about this is just like every one of your phones are different(!), we will get different correct answers for the focal length. The more phones you have, the more correct answers you will get, so this is a good time to use the diversity in the group. (As an aside, Apple lovers may find this part a damp squib in their learning endeavours; in contrast, the older your Android phone is, the more opportunities you have to learn.)

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

1.1.1 Calibrate Your Camera!

We will calibrate our camera using a program. you are going to with work with two, yes, two photos in this sub-section.

1. Save the two images in `./data/calib_images` directory. Note that, for you and the TA to be able to retrieve the focal length of the camera from the image meta-data (using tools like `exif`),



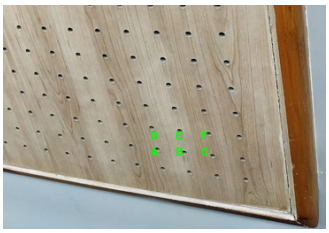


Figure 1: We took these photos. You should take two pictures like this. Notice the points at the lower right corner which are marked as A, B, C, D, E, F.

your photos will need to be in their original form. Transferring a picture from your phone to your PC using a messenger application may create problems. You want those exact bytes the camera manufacturer gives you in your submission.

2. 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. In the example, it is given to you that the distance between two adjacent (say in the X-direction) markers in the pattern is 3cm (This need not be accurate as long as the pattern is regular). If a regular pattern is not accessible, you can measure the relevant real world distances. In either case, be sure to mention the 3D coordinates of the six points in your reflection essay.

3. Figure out the sensor (image) coordinates for these six points (recall manual point correspondences in earlier labs). You need to do these for the two photos (pictures, images) you have taken. You don't necessarily need retain the original image resolution for this step. (What did we mean by this statement?)

4. Next, using `cv2.calibrateCamera()` obtain the intrinsic parameters.

   Q1: 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. (Also answer some of the implicit questions scattered around in the description of this problem statement.)

1.1.2 Truth in Advertisement

In this part, we want to be sure we understand the literature provided by the manufacturer.

1. Determine the type of camera that your phone uses from the manufacturer site. In particular, note the sensor specifications. Make sure you check multiple sites to confirm consistent information, in particular the units. Your Google-fu skills are likely to be important if you have an Android phone. Note down the focal length. Make sure you verify this value from the images stored (See Item 1 in Section 1.1.1 using, e.g., `exif` (CLI on Linux).

2. Using the camera calibration matrix determined earlier in Item 4 in Section 1.1.1. and the sensor specifications in the previous step, output the focal length of your camera in `mm`. Print the value of the computed focal length on the screen, following the run template:

   Run: `python3 find_focal.py`

Q2: Report your observations such as the number of focal lengths, discrepancies, and so on. This part is likely to be different for different groups, so please don't try to get the "correct answer" like you most likely did in your practical session in Class XII. Just tell us what you did.

1.2 Reproject!

In this part, we ask you to verify the accuracy of your estimate to the inverse problem using re-projection. Here we are going to relax the assumption on the number of pictures. Using the same or similar photographs, perform the following tasks.

1.2.1 Tasks

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

2. As before calibrate the camera, but this time use any number of images for this task.

   Q3: Comment on the role of the number of images

3. 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 your 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 image that you chose,

   - For each of the world coordinate point as hypothetically seen from the 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 using a different marker and save it as `./data/projected.jpg`.

Run: `python3 camera_calibrate.py`

Figure 2: Sample image after drawing projected points. Original locations maked with circles and reprojected points marked with crosses (You can be creative, we don't require that you draw the points in the exact same way)

Note

You can take as many pictures as you like but you must submit (and work on) the output of exactly two of those photos.

1.3 I wish I had that mobile phone!

In this part we have some data that was collected from two pictures 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 in the previous task.

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>`

Q4: Were you able to complete this task without the image size? Discuss the presence or absence of this information.

2 Submission Guidelines

1. The top assignment directory should include the lab submission as detailed below.

   (a) Do include a `readme.txt` (telling me whatever you want to tell me including any external help that you may have taken). Don't forget to include your honor code here (or your name and roll number). All members of a group are expected to (electronically) sign the honor code and the percentages (see below). This is a text file, not `pdf`, not `docx`.

   The `readme.txt` will contain individual contributions of each of the team members. If the assignment is finally worth 80 marks as graded by the TA, then a contribution of the form `80, 100, 60` (in sorted order of roll numbers) will result in (respectively) marks `64, 80, 48`. Do this for each question separately. A person claiming 100% is basically saying that (s)he can reproduce the entire assignment without the help of the other team members.

   (b) `ReflectionEssay.pdf`: Should contain the explanation for all the questions implicitly and explicitly raised. Provide an output of a sample run. Explain what you learnt in this assignment, and how this assignment improved your understanding. What will someone who reads this gain? Can this be a blog post, which if read end-to-end someone not in your class (but in your batch) will understand?

   (c) A directory called `code` which contains all source files, and only source files (no output junk files). The mapping of code file to questions should be obvious and canonical.

   (d) A directory called `results` to store output that needs to be saved (this will typically be explicitly stated in the question).

   (e) A directory called `data` on similar lines to code, whenever relevant. Note that your code should read your data in a relative manner.

   (f) Source files, and only source files (no output junk files). Mac users please don't include junk files such as `.DS_store`. We are not using MacOS.

   (g) Create a directory called `convincingDirectory` which contains anything else you want to share to convince the grader that you have solved the problem. We don't promise to look at this (especially if the code passes the tests) but who knows? This is your chance.

2. Once you have completed all the questions and are ready to make a submission, prepend the roll numbers of all members in your group to the top assignment directory name and create a submission folder that looks like (for group but you get the idea) this
   `130010009_140076001_150050001_lab0X_description.tgz`

   Please stick to `.tar.gz`. Do not use `.zip`. Do not use `.rar`

3. Your lab submission folder should look something like this:

```
130010009_140076001_150050001_lab03_calibrate/
├── ReflectionEssay.pdf
├── code
│   ├── camera_calibrate.py
│   ├── find_focal.py
│   └── reproject.py
├── data
│   ├── calib_images
│   │   └──
│   ├── true.jpg
│   ├── points.txt
│   └── projected.jpg
├── convincingDirectory
└── readme.txt
```

4. Submission. Very very important.

   (a) Submit on **Moodle** at the course **CS-763**.

   (b) The lexicographic smallest roll number in the group should submit the entire payload (with all the technical stuff).

   (c) All other roll numbers submit only `readme.txt` as discussed above.