CS 763/CS 764: Lab Six

Extrinsically, focusing on the focal length

- Announced: 02/21 14:00, Posted: 02/21 14:00 , Inlab Due: 02/21 22:30, Outlab Due: March 04, 22:30

1 Focus!

1.1 Overview

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 phone is different(!), we will get different correct answers.

1.2 Tasks

1. Take multiple photos. Recall the kind of pictures you took for the `Camera calibration` task in **Lab05**. Using the same or similar photographs, save all images in `./data/calib_images` directory. You are free to take different pictures from the previous lab, and perhaps high quality photos. It's important for you and the TA to retrieve the focal length of the camera from the images (we use tools like `exif`).

2. As before calibrate the camera using the same method as earlier. You are free to use any number of images for this task.

3. Determine the type of camera that your camera uses from the manufacturer site. Make sure you check out multiple sites to confirm consistent information, in particular the units. Note down the focal length and make sure you verify this value from the images stored. (See Item 1).

4. Using the camera calibration matrix determined in Step 2 and the sensor specifications in Step 3, find the focal length of your camera in `mm`. Print the value of the computed focal length on the screen.

Run: `python3 find_focal.py`

Oh, we are silent on how many different focal lengths you will report. YMMV. We will wait for your reflection essay.

2 Outlab

2.1 Overview

Augmented Reality (AR) is the overlaying of digital (synthetic) data onto the real world. In this problem, a *synthetic* book will be overlayed on a plane in an image, such that it is perceived as if the book actually lies in the scene. This requires the knowledge of the camera for projecting the book from a particular viewpoint.

2.2 Tasks

1. In this lab, a "book" is specified by three dimensions. We refer to the longest dimension as "length", the shortest as "width", and the remaining as "breadth". This information is to be specified in the command line argument.

2. All necessary data (input and output) for this problem is to be stored in `data/augment_book/`. The input images of the plane (a "wall") will be provided. The intrinsic camera matrix has been provided in `intrinsic.txt`.

   We will judge your book by the cover, so we provide two textures in `front.png` and `side.png`.

   Read the data in your program. We would like to see a `python` function `get_data` for this.

3. We want to "place" the virtual book sticking (and floating) on the wall for an upcoming Harry Potter movie. A picture is provided for reference. Notice that the approximate layout of how the book is sticking to the wall. The "back cover" (last page) is stuck to the "wall" (i.e. the sound proofing fixture). The front cover is hanging like a cantilever. The book is aligned parallel to the holes of the wall, so that one can read the text if one were to fly and rotate ourselves. One example as shown in Fig. 1.

   Draw the boundary edges of the implicit cuboid corresponding to the book on each of the provided images. The key to doing this is to use `cv.solvePnP`. The boundary must be in red, and make sure to use a thick pencil.

   We would like to see a `python` function `draw_book_on_img` for this.

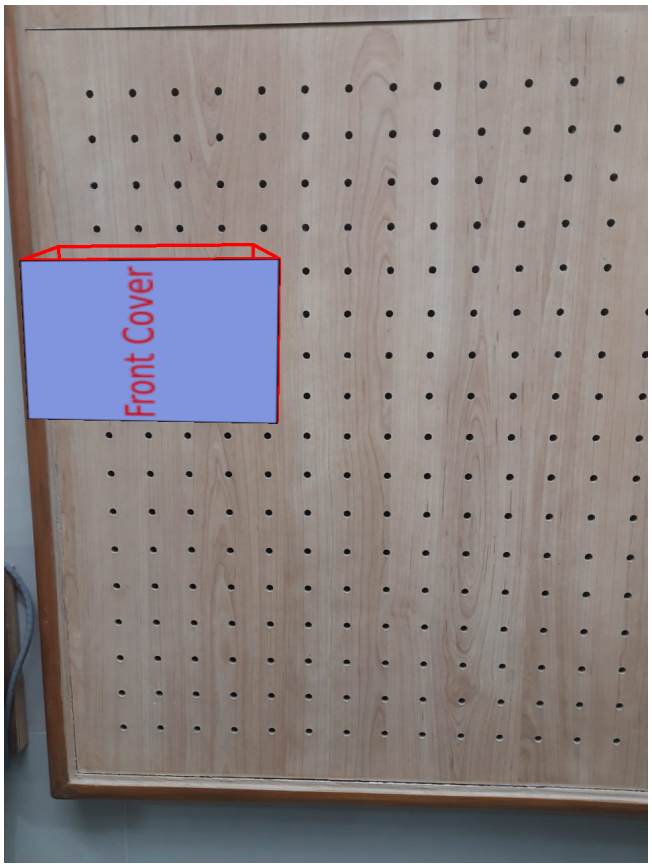

Figure 1: A book anchored to the wall like a cantilever.

4. We are almost there. Use the provided images to "stick" the texture of the front and side to make the scene more realistic. Save the images on disk.

   We would like to see a `python` function `texture_book_on_img` for this.

5. Currently there is a single view of the book. In fact, the side plane of the book is occluded, and we can't see it! Fixing the book at exactly the same location, provide 3 other views of the book: the constraint here is that the camera is placed at sufficiently different locations. (If, in the example provided, the camera was at a different location (for instance if the camera man was squatting and looking up) the side would have been visible.)

   Feel free to invent your textures for the 4 remaining planes of the book.

2.3 Run

The code should display all the `textured` images. The program will be executed as

```
python code/augment_book.py data/augment_book -w 1.5 -l 5 -b 3
```

3 Submission Guidelines

We are not repeating this. You should know the drill by now.