Robotics
Practical 5: Planar Camera Calibration

Andrew Davison
a.davison@imperial.ac.uk

1 Introduction

In this practical we will be introducing the use of a Raspberry Pi camera to the robotics kits, and showing how it can be calibrated to make accurate planar measurements.

This practical will be UNASSESSED. It is important preparation for taking part in the final challenge competition, which will be the final practical of term set next week, where we will use the camera in a path planning/obstacle avoidance competition.

2 Camera Connection

Come and see us in the lab and we will give you a Raspberry Pi wide angle camera. This is a tiny colour camera which connects directly to your Raspberry Pi via a ribbon cable. We will help you to unscrew your BrickPi case in order to access the camera connector, which is in the middle of the Pi. There is a plastic bar which protects the camera connector, which you need to take out, then place back in to pinch the ribbon cable in place. See here for more details: `https://projects.raspberrypi.org/en/projects/getting-started-with-picamera`. Please be careful with the ribbon cable and connector, and avoid pulling or twisting it. Once you have connected the camera and re-assembled your case, you may need to reconfigure your robot a bit to allow the camera to be mounted in the right place on your robot. Please keep the camera safe with the rest of your kit, and return it to us at the end of term.

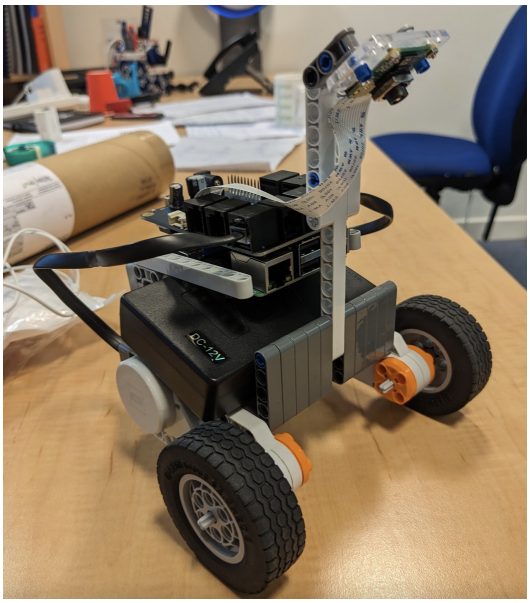
3 Camera Mounting

We have attached a custom LEGO-compatible plastic plate to each camera, which you can use to attach it to your robot.

You should design a camera mount for the front of your robot, such that the camera is ideally lifted up at the front of the robot and pointing at a downward angle of maybe 30–40 degrees so that most of the field of view covers the floor in front of the robot. In the challenge next week you should experiment with the ideal mounting position of your camera to see what works best. If it is pointing downwards at a high angle, it will capture high resolution information about the floor just in front of the robot, but will not be able to look far ahead; while a flatter angle will have a greater range, but less accuracy, and may have a blind region close to the robot.

It is important that your camera mount is as rigid as possible, so that the position of the camera relative to the robot stays fixed. Any time that the position of the camera might have moved, you will need to recalibrate it. Here is an example with quite a high downward-looking angle — probably a smaller

downward-angle than this will be better for next week because this robot currently cannot see very far ahead.



4 Capturing, Displaying and Processing Images

Images are captured using functions from the `picam2` Python module, and we will be using OpenCV from the `cv2` module for image processing. Both of these are already installed on your existing SD cards. We have also included functionality in the web interface you are already using to display images from the camera.

First download and run this example program from the web interface, which will capture and display images from the camera so that you can test that it is working:

`http://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/picamera.py`

5 Calibrating the Camera as a Planar Sensor

In preparation for the final challenge, the main task this week is to calibrate your camera as a planar sensor. As explained in lectures, when a perspective camera observes the ground plane, the relationship between pixel coordinates (u, v) and planar coordinates (x, y) in the robot frame of reference is described by a fixed *ground plane homography*. This is a 3×3 matrix which depends on $\mathtt{R}^{CR}$, the rotation matrix relating robot frame R and camera frame C, $\mathtt{t}^R$, the vector from the robot centre to the camera centre, and the *camera calibration matrix* $\mathtt{K}$:

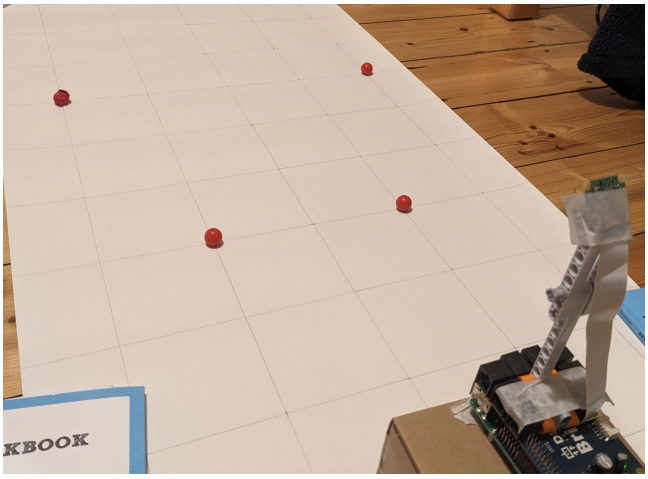
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \propto \mathtt{H} \begin{pmatrix} x^R \\ y^R \\ 1 \end{pmatrix} ,$$

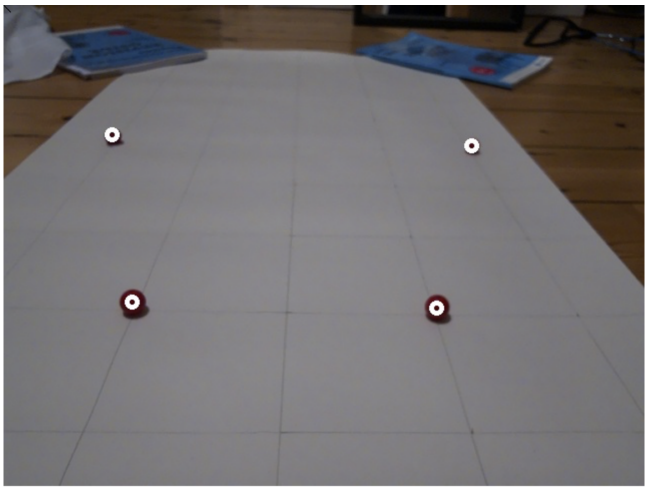
where $\mathtt{H} = \mathtt{K}\mathtt{R}^{CR}\mathtt{T}$.

Rather than needing the calibrate each of the elements of the transformation and projection separately, we will directly estimate the elements of $\mathtt{H}$ using measurements of image to ground place correspondences.

We have pieces of A1 paper with 2cm printed grids, and red markers which you can use for calibration.

Define a position at one end of the paper as the origin of your robot coordinate system, and place the robot centre accurately there and aligned with the x axis of the paper. Then place four red markers on the paper at obvious grid intersections which are visible in the camera's field of view, and record their positions. It is best to place the markers as far away from each other as possible so that they span the whole image.





We have provided a program which will detect the positions of markers in the image from the camera, and display and print out the image coordinates.

`http://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/picameraredblobs.py`

This program works by applying OpenCV image processing to the image in several steps:

1. The image is converted from RGB (Red, Green, Blue) to the HSV (Hue, Saturation, Value) image space, where information about the type of colour is contained in the Hue channel.
2. A mask is generated for pixels where the Hue values are in a range indicating red colour. (Note that in the program this is done in two steps, because in the Hue space used in OpenCV, red lies either side of zero. For other colours, only a single range would be needed.)
3. We find "connected components", regions of connected pixels within the mask, which have a certain minimum size. (This rules out any small red regions which may be present just due to image noise.)
4. The image coordinates of the centres of these regions are reported.

Using the program, find the image coordinates of your four markers. Check which image coordinates correspond to which scene coordinates, and note them all down. You will get four sets of (x_i, y_i, u_i, v_i) corresponding coordinates.

6 Calculating the Homography and Visualising the Calibration with a Grid

We have provided a final program into which you can enter your four sets of corresponding coordinates. The program will then calculate the homography and inverse homography, and then use these to display a calibrated grid on top of your image:

`http://www.doc.ic.ac.uk/~ajd/Robotics/RoboticsResources/picamerahomographygr`
`py`

If you have done the calibration correctly, you should see the grid line up well with your paper grid when you place the robot on the origin. The robot now has the ability to accurately estimate the (x, y) planar coordinates in the robot frame of any (u, v) image point it observes which lies on the ground plane.

Acknowledgements

Thank you to Riku Murai, Marwan Taher and the other TAs on the robotics course for massive help with preparing this practical.