TTK4255: Robotic Vision

Homework 5: Camera calibration

Originally created by Simen Haugo, and modified by Mau Hing Yip.

Instructions

First make sure to read `about.pdf` in the "Course work" page on Blackboard. To get your assignment approved, you need to complete any 60%. Upload the requested answers and figures as a single PDF. You may collaborate with other students and submit the same report, but you still need to upload individually on Blackboard. Please write your collaborators' names on your report's front page. If you want detailed feedback, please indicate so on the front page.

About the assignment

The aim of this assignment is to give you experience in calibrating a camera and interpreting the results. The camera model you are asked to use is the perspective camera model extended with a polynomial distortion model. We provide a brief review of this model, and the first two parts involve doing simplified analyses involving the model.

In the last part you are asked to calibrate a real camera using the images provided in the data archive. The images are of a planar checkerboard pattern, which provides a regular grid of equally-spaced and well-localized points — the intersections of the black and white squares (aka the internal "corners"). Their coordinates are of the form $(i\lambda, j\lambda)$ where i, j are integers ≥ 0, and λ is a scaling factor. The pattern was displayed on a flat computer monitor, so any deviation from this is likely negligible. The scaling factor has not been measured, but this is unimportant if only the intrinsics are of interest.

Polynomial distortion model

A significant source of error in the perspective camera model is caused by the camera's lens. Many lenses are designed with perspective projection (aka rectilinear projection) as a goal, but may deviate from it due to competing goals and manufacturing limitations. Deviation from a specific projection is often called (lens) distortion. There are many extensions of the perspective camera model intended to describe the distortion present in rectilinear-type lenses. A popular extension is a polynomial radial and tangential distortion model. Let $x = X/Z$ and $y = Y/Z$. Then, a point with camera coordinates (X, Y, Z) is projected to the pixel coordinates

$$u = c_x + (x + \delta_x) f_x, \tag{1}$$
$$v = c_y + (y + \delta_y) f_y, \tag{2}$$

where δ_x and δ_y each are defined by polynomials in x, y, and $r = \sqrt{x^2 + y^2}$:

$$\delta_x = (k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots)x + 2p_1 xy + (r^2 + 2x^2)p_2, \tag{3}$$
$$\delta_y = \underbrace{(k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots)y}_{\text{Radial}} + \underbrace{(r^2 + 2y^2)p_1 + 2p_2 xy.}_{\text{Tangential}} \tag{4}$$

The distortion coefficients $k_1, k_2, ..., p_1, p_2$ are estimated via calibration, along with c_x, c_y, f_x, f_y. More tangential distortion coefficients can also be used (in a more complex form of the model), but for modern cameras it is often limited to two. This model can be found in many vision software packages, and camera calibration tools, including OpenCV's calibration module (link), Kalibr (link) and Matlab's Computer Vision Toolbox (link).

"Distortion" suggests that the deviation is an imperfection that needs to be corrected before anything useful can be done. However, many computer vision algorithms are trivially applicable with (or have been generalized to) more general camera models, such as the above. For example, it is entirely possible to compute reprojection errors, thus enabling the use of iterative reprojection error minimization. Moreover, if the camera is calibrated, then any points in the image can be converted into back-projection rays. Several initialization-free methods have been formulated for rays rather than image points, see e.g. OpenGV (link).

Nonetheless, transforming the input image (or points within it) into some other projection is sometimes desirable, and allows one to use a simpler camera model afterward. Performing this transformation is often called "undistortion". Beware that some authors and software packages consider a perspective projection to be somehow universally ideal, and regard any deviation from perspective as distortion, even if such deviation is intentional (as in fisheye lenses).

Part 1 Polynomial distortion model (25%)

Task 1.1: (5%) If a point is represented with spherical coordinates around the camera origin, then its perspective projection can be written as

$$u = c_x + f \tan\theta \cos\phi, \tag{5}$$
$$v = c_y + f \tan\theta \sin\phi, \tag{6}$$

where θ and ϕ are related to the Cartesian camera-frame coordinates by

$$X = \lambda \sin\theta \cos\phi, \tag{7}$$
$$Y = \lambda \sin\theta \sin\phi, \tag{8}$$
$$Z = \lambda \cos\theta. \tag{9}$$

Use this to explain why it can be problematic to transform images (or points within images) with an angle of view approaching or exceeding $180°$ into a perspective projection.

Task 1.2: (10%) Consider a perspective camera model extended with two radial and two tangential distortion coefficients. The model has been calibrated on images of width W and height H. Consider an image taken by the camera and suggest how the parameters $(c_x, c_y, f_x, f_y, k_1, k_2, p_1, p_2)$ should be adjusted to remain valid, when...

(a) ...the image is downscaled by 1/2 to width $W' = W/2$ and height $H' = H/2$.

(b) ...the image is cropped to width $W' = W - (l + r)$ by subtracting $l \geq 0$ pixels from the left and $r \geq 0$ pixels from the right.

Task 1.3: (10%) The images in Fig. 1 have been modified to exhibit radial distortion. Suppose the resulting projection can be modeled acceptably using a perspective projection with the described polynomial distortion model, with just one radial distortion coefficient (k_1). For which image will k_1 be positive and for which will it be negative?

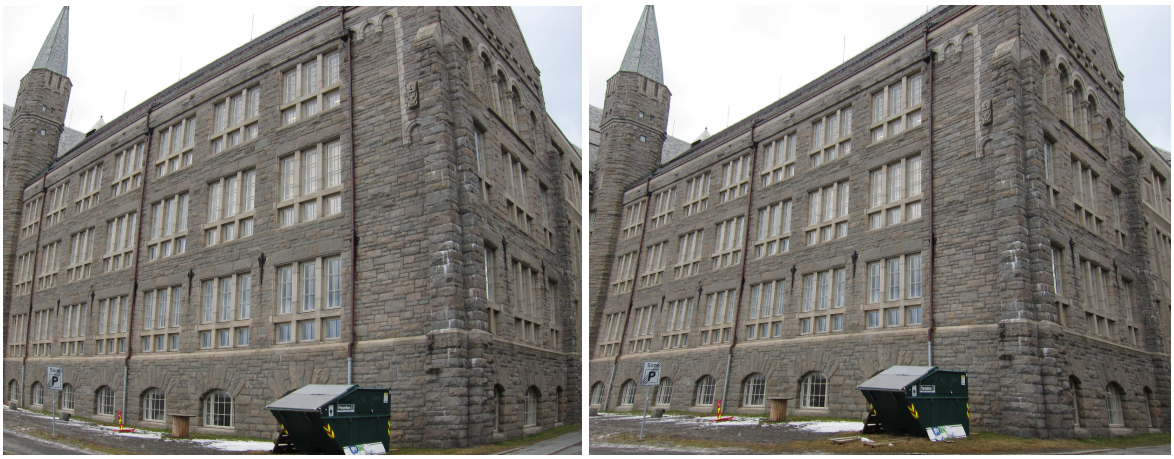

Figure 1: Images with simulated radial distortion.

Parameter	$\hat{p}$	$\hat{\sigma}_{\hat{p}}$	Parameter	$\hat{p}$	$\hat{\sigma}_{\hat{p}}$
f_x	2359.40946	0.84200	k_1	-0.06652	0.00109
f_y	2359.61091	0.76171	k_2	0.06534	0.00624
c_x	1370.05852	1.25225	k_3	-0.07555	0.01126
c_y	1059.63818	0.98041	p_1	0.00065	0.00011
			p_2	-0.00419	0.00014

Table 1: Results from a camera calibration.

Part 2 Interpreting uncertainty in the intrinsics (25%)

Table 1 shows results from a camera calibration. Listed is each parameter's maximum likelihood estimate ($\hat{p}$), and its estimated standard deviation ($\hat{\sigma}_{\hat{p}}$). The standard deviations are simple indicators of the uncertainty in the estimates, which should always be considered when assessing the quality of a calibration. If the estimates are independent[1], the standard deviations can be used to form confidence intervals. For example, if an estimate is normal-distributed, multiplying its standard deviation by 1.96 gives the half-width of a 95%-confidence interval (link).

It should be clear how error in the estimate of the principal point translates into error in the image; in the projection of a point, any error in (c_x, c_y) results in an equal error in (u, v). Less clear is the effect of error in f_x and f_y and the distortion coefficients. The resulting error will not have the same magnitude everywhere in the image. Instead, its magnitude reaches a maximum near the sides or corners of the image.

Task 2.1: (10%) Assuming the image has 2816 columns and 2112 rows, calculate the maximum error in u caused by using the maximum likelihood estimate of f_x, when the true value is 1.96 standard deviations off. Assume the other parameters are equal to their maximum likelihood estimates.

Tip: A point that is projected approximately to the lower-right corner using the maximum likelihood estimates has $X/Z = 0.64$ and $Y/Z = 0.46$.

Task 2.2: (15%) Calculate the maximum error in (u, v), measured with the L_2-norm, caused by using the maximum likelihood estimates of p_1 and p_2, when their true values can be anywhere in the respective 95%-confidence intervals. Assume the other parameters are equal to their maximum likelihood estimates.

Tip: The provided script `task22.py/m` implements the camera model with distortion. You may use the script as aid in your calculations.

[1]The estimates will be correlated because the parameters do not have mutually independent effects on the projection. Ideally, you consider the full covariance matrix (and the corresponding confidence ellipses), but some calibration tools only provide the "standard deviations", which are the square-rooted entries on the diagonal of the covariance matrix.

Part 3 Camera calibration (50%)

In this part you will calibrate a real camera. You will need either of the following:

- (Matlab) Single Camera Calibrator App (link) from the Computer Vision Toolbox (link). You can install new toolboxes to an existing Matlab installation by rerunning the installer (link).
- (Python) OpenCV Calibration Module. We provide a script `calibrate_camera.py` that brings together the required functionality, which is largely based on the official tutorial (link). To use this you must install OpenCV (link) with the option `pip install opencv-contrib-python`.

Task 3.1: (10%) Included in the zip is a set of images of a checkerboard, taken with the same camera. Calibrate the camera using a perspective camera model with three radial and two tangential distortion coefficients. Run the `show_calibration_results` script and include the generated figures.

Task 3.2: (10%) One figure shows the detected checkerboard points from all the images together. Does the plot suggest anything about this calibration, e.g. should more/other images be taken?

Task 3.3: (10%) One figure shows the mean reprojection error per image. Some images will have nearly twice the error as the image with the smallest error. What could be causing this? Describe experiments you could do to test your hypotheses (you don't need to actually do the experiments).

Task 3.4: (20%) A visual understanding of the uncertainty of the distortion coefficients can be gotten by sampling a few hypothetical outcomes from a normal distribution defined by the estimated covariance matrix, and undistorting an image with each set of hypothetical parameter values.

Write a script to assess the uncertainty via this approach. You may keep the intrinsic matrix fixed, and only sample from the distribution of the distortion coefficients. You may assume that these are independently normal-distributed, with mean and standard deviation equal to the corresponding values printed by the above script. Matlab users may use `undistortImage` (link), and should also read how to create a `cameraParameters` object (link). OpenCV users may use `cv.undistort` (link), and should consult the usage example in the calibration tutorial. Save the images to disk and open one in your image viewer. Then you can quickly flip between the images to more clearly see the differences.

Reflect on the following:

(a) Should the image you choose to undistort for this analysis be one of the calibration images, or should it be an image of a different scene? Does it need to be an actual image from the camera? (The data folder contains an additional image "sample_image.jpg", which may be useful.)

(b) The undistortion functions in Matlab and OpenCV let you configure how the output should be scaled or cropped (if at all), and whether the output is allowed to contain invalid pixels. What configuration do you find gives a good indication of the uncertainty in the distortion coefficients?

(c) Is the uncertainty in the distortion coefficients visually noticeable? Is it more noticeable in some parts of the image? If so, can you relate that to your answer in Task 3.2?