EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

Mobile Robots

Summer Semester 2013
Assignment 6

due by: 11.06.2013, presentation: 18.06.2013
email submissions and questions: volker.grabe@tuebingen.mpg.de

Exercise 1 - Camera Calibration (10 Points)

As mentioned in class, a reliable calibration is necessary to perform research with cameras. In order to extract the required parameters, most people rely on freely available toolboxes. A common toolbox for MatLab has been provided by Jean-Yves Bouguet from CalTech and can be found here: `http://www.vision.caltech.edu/bouguetj/calib_doc/`. You will use it to calibrate a camera.

(a) The calibration is usually done by photographing a checker-board pattern with known dimensions as its corners are very regular and relatively easy to detect. The images provided on the website where taken using a MatrixVision USB2 mvBlueFOX-MLC200w G camera and an unknown lens with low distortions. Please search the web for the product website and familiarize yourself with the camera. What is the resolution and frame rate of this camera? Compute the maximum data rate you can expect to receive from the camera (1 pixel = 1 byte). (1 Point)

(b) Download the toolbox and unpack it on your computer. The checker-board pattern in the sample images consists of 10 by 6 inner squares as the outer row of squares is usually ignored. Each square measures 20 mm in length. In MatLab, you might want to add the folder of the toolbox to your path: right-click onto the folder and select "add to path". Run the toolbox by calling `calib_gui` and choose the standard method. Navigate to the folder with the images and load them using the `Image names` button. Assuming an ideal camera and lens, where would you expect the principle point to be? (1 Point)

(c) Start the interactive calibration process by clicking the `Extract grid corners` button. Follow the steps explained in the "first example" on the website of the toolbox to process all images. Use the default settings, but don't forget to specify the correct parameters for the checker-board. You probably won't need to specify initial distortion parameters due to the low distortions of this lens. Afterwards, click the button `Calibration` to run the calibration process. Report the resulting camera matrix. The reported focal length already corresponds to (k_u, k_v) in the slides. What is the pixel error? (4 Points)

(d) Compare the found principle point with your expectation. What could be the reason for the difference? (1 Point)

(e) You can try to further improve the calibration and thus reduce the pixel error. For this, you might want to use the `Analyse error` function to identify images with strong outliers. Suppress these images. Which images did you suppress for what reason? Furthermore, you can recompile the extracted corners and run the calibration again. Consult the example on the web for explanations if necessary. How much did the pixel error decrease in total? (2 Points)

(f) Although the lens has only minor distortions, the toolbox should have found at least the first two distortion parameters to be different from zero. In MatLab, you can call the script `visualize_distortions` to see how much a pixel is displaced due to lens distortion in the different regions of the image. What is the biggest displacement for the radial and the tangential distortion? (1 Point)

Exercise 2 - Towards Stereo Vision (10 Points)

You will now use the camera parameters you found before to do some stereo vision. In case you had troubles to calibrate the camera yourself, you may use the set of approximate variables which was provided with the calibration images.

(a) The toolbox provides functions to both project a point in camera frame into the pixel space and its inverse. Call `project_points2([x;y;z],[0;0;0],[0;0;0],fc,cc,kc)` to compute the pixel into which a point $[x, y, z]^T$ in space would project. Compute the pixel into which the following points would project: $[0, 0, 2]^T$, $[1, 0, 2]^T$, $[0, -1, 2]^T$, $[0, 1, -2]^T$. What did you notice? (2 Points)

(b) The inverse function `normalize([u;v],fc,cc,kc,alpha_c)` computes the x and y component of a point on a plane in 1 m distance which would project into the pixel (u, v) (as would any other point on the ray between the camera and this point). This function is relatively slow when called in a loop. Can you imagine why? Use this function to compute the corresponding point on the 1 m distance plane for the following points in pixel space: $[0, 260]^T$, $[350, 0]^T$, $[752, 260]^T$,$[751, 260]^T$, $[350, 480]^T$. (2 Points)

(c) From these results, what is the field-of-view (in degrees) of this camera setup in x and y dimension? Is is symmetric? You will need some basic trigonometry. (2 Points)

(d) Imagine a stereo camera setup of two of these cameras with a baseline of 0.1 m. What would be the closest distance for which 3D information could theoretically be obtained? (2 Points)

(e) With this stereo rig, you found a correspondence for a feature at $(248, 298)$ in the right and at $(269, 298)$ in the left image. Compute the location of the feature in camera frame. (2 Points)