Intelligent Systems Group WS 18/19
Universität Duisburg-Essen 2017-11-30
Prof. Dr. Josef Pauli / Stefan Lörcks

Exercises Computer/Robot Vision - 07

PROGRAMMING (7 points)

In this exercise we will turn our smartphone/camera into a ruler using camera calibration techniques provided by MATLAB. The aim is be able to measure objects in Millimeters. The necessary functions are part of the computer vision toolbox, which is not part of the student license. This means the exercises have to be done in the exercise class or in one of the computer centers provided by ZIM (e.g. BA 028, MA 425, LC 036). MATLAB on that computers contains the computer vision toolbox.

15. **Camera calibration** (2 points)
    In this exercise we will perform camera calibration using MATLAB. The aim is to determine the intrinsic parameters and the distortion. We will use the results in the next exercises.

    a) Watch the video 'Camera Calibration with MATLAB' [1]. Further information on camera calibration using MATLAB is available [2], [3]. Note that the notation differs from the lecture (e.g. 'focal length' in the video corresponds to 'effective focal length' in the lecture).

    b) Print a checkerboard pattern. One is provided in the moodle course. You can also use the one provided by MATLAB or create a pattern yourself. Measure the size of the checkerboard squares carefully. (Printed patterns will also be provided in the lecture and in the exercise class.)

    c) Take 10-20 pictures of the pattern from different angles. The distance between camera and pattern should be roughly the same as in the later experiments. Figure 1 shows examples of pictures taken for calibration.

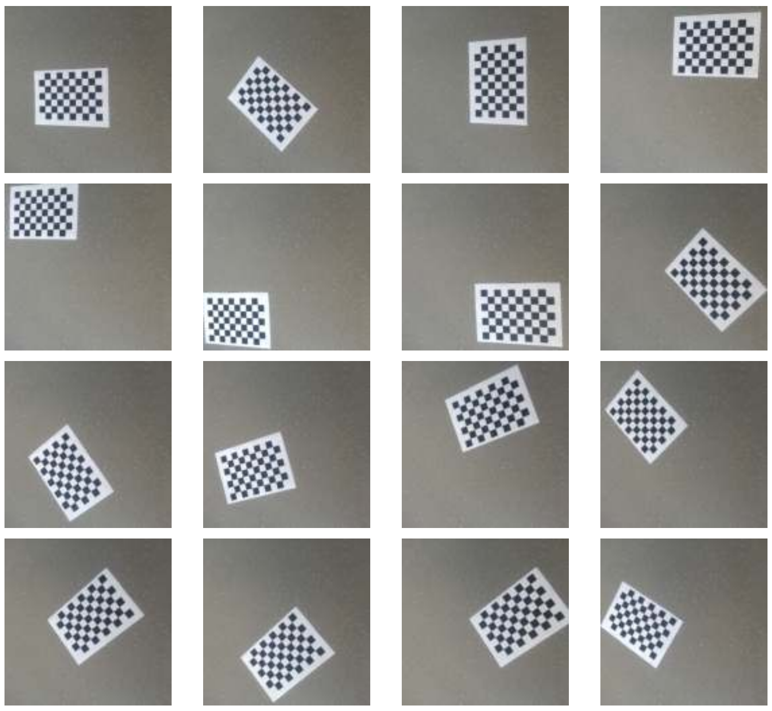

Figure 1: Example of pictures used for calibration

d) Perform camera calibration using the CameraCalibrator App as shown in the video.

   i. 'Add Images'.

   ii. Enter square size in the popup window.

   iii. 'Calibrate'

   iv. Delete images, which result in outliers, and recalibrate if necessary.

   v. 'Export Camera Parameters'

e) Save the workspace containing the variable cameraParams into a file named 'WScameraparameters.mat'

16. **Ruler** (2 points)
In this exercise we will turn the camera/smartphone into a ruler. The aim is to measure objects in world units (Millimeters). Using MATLAB we will process a image to extract distances between two points in world units. The intrinsic camera parameters are taken from the previous exercise. The extrinsic parameters of the camera depend on the current camera position and orientation. That is why the checkerboard pattern is part of the image.

a) Take a picture containing the checkerboard pattern and a few objects you want to measure. To rate the accuracy it is a good idea to include a ruler (or a folding rule, set square, measuring tape, etc.). Copy the picture to your working directory and rename it to 'objectsToMeasure'.

b) Download the script

   `CRV_16.m`

   from moodle, execute it and select the picture.

c) After the extrinsic parameters are calculated and the picture has been rectified it is shown in a figure. In the middle two draggable blue points are shown. The distance between these points in world units is given in the title. You can move the points around with the mouse and watch the distance change. Measure some objects by dragging the points on them.

d) Save the figure as a 'Ruler.png' (This can be done via the GUI: 'File'-> 'Save As...'!). An example is shown in figure 2.

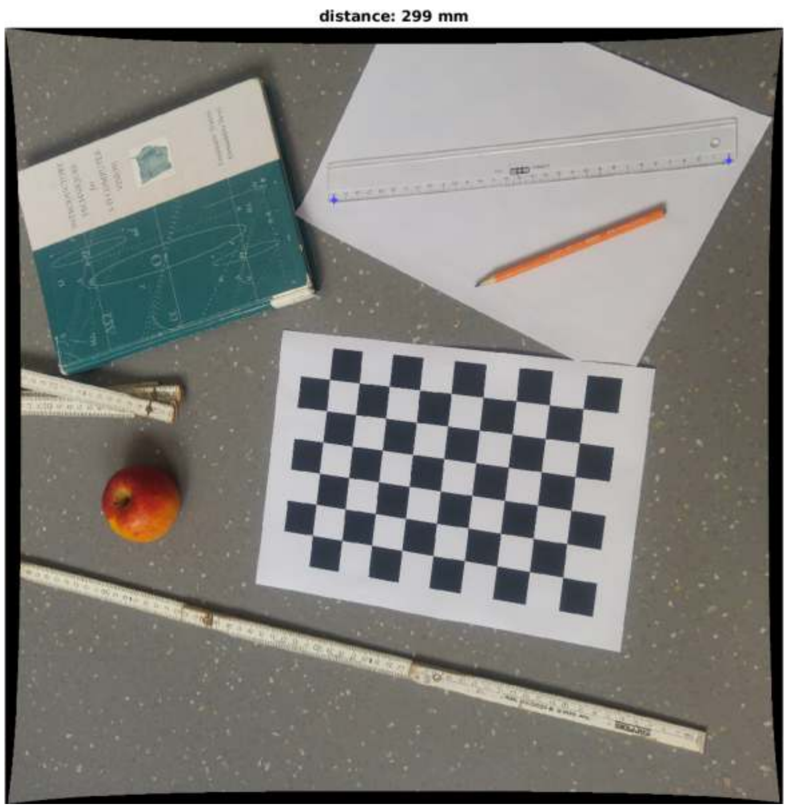

Figure 2: An example of the image 'Ruler.png'. Here the 30 cm ruler has been measured. Therefore, the blue points have been set on the 0 and 30 marker. The determined distance is 299 mm.

17. **Coin classifier** (3 points)
In this exercise we will use the calibrated camera to classify coins. We will apply circle hough transform on the rectified image to detect the coins. Based on the results we can determine two points on the border of the coin, calculate their distance in world units and classify the coin based on this. As this approach only works for coins with a suitable high difference in diameter we will only use 1 Eurocent, 10 Eurocent and 50 Eurocent coins. The aim is to count how many coins of each category are in the image.

   a) Create a script named

      `CRV_17.m`

      based on the previous exercise. Of course again, fill in your name and student number!

   b) Use circle hough transform to detect the c[illegible] in the rectified image. Visualize the results in a figure.
      Useful commands: `imfindcircles`

   c) For each circle determine the image coordinates of two points, whose distance is the diameter.

   d) Convert the coordinates of the points to world units and calculate the distance. This distance is the diameter d.
      Useful commands: `pointsToWorld`

   e) Classify the detected circles based on the diameter d according to

$$\text{CoinClassifier}(d) = \begin{cases} \text{1 Eurocent} & \text{if } d \in (14.5\text{ mm } | 17.5\text{ mm }) \\ \text{10 Eurocents} & \text{if } d \in (17.5\text{ mm } | 21.5\text{ mm }) \\ \text{50 Eurocents} & \text{if } d \in (21.5\text{ mm } | 28\text{ mm }) \end{cases}$$

      Change the ranges if necessary.

   f) Count how many coins are in each category and give the result in the title of the figure. Save the figure as 'Coins.png'. An example is shown in fig. 3.

References

[1] MATLAB. *Camera Calibration with MATLAB.* URL: `https://www.youtube.com/watch?v=g8SyzR0jZOA`.

[2] MathWorks Matlab Help. *What Is Camera Calibration?* URL: `https://de.mathworks.com/help/vision/ug/camera-calibration.html` (visited on 11/29/2017).

[3] MathWorks Matlab Help. *Single Camera Calibration App.* URL: `https://de.mathworks.com/help/vision/ug/single-camera-calibrator-app.html` (visited on 11/29/2017).

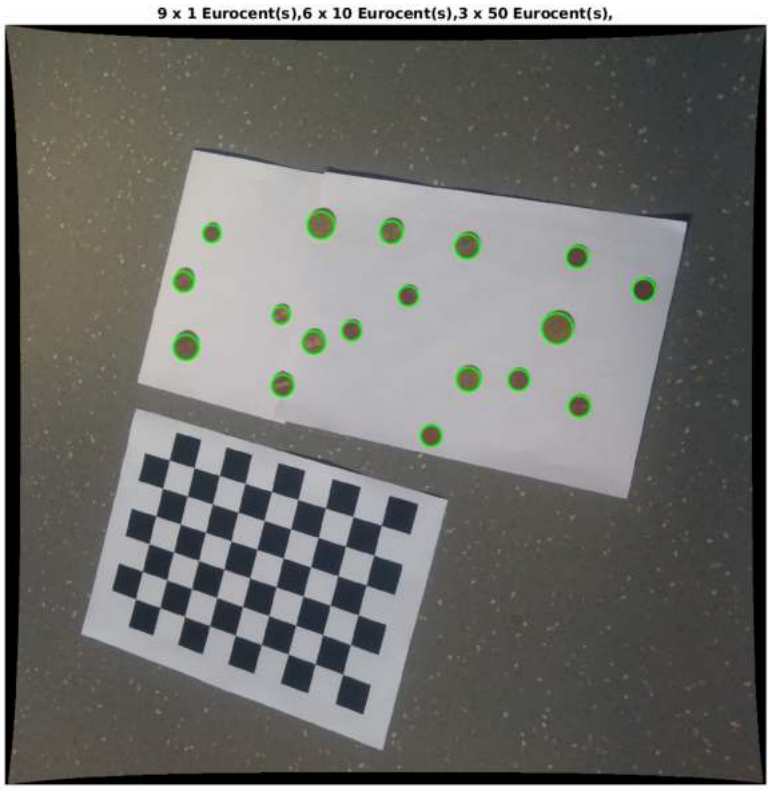

Figure 3: An example of the image 'Coins.png'.