Mathematisch-Naturwissenschaftliche
Fakultät

Fachbereich Informatik
Arbeitsbereich Visual Computing

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Bildverarbeitung I (Prof. Schilling)

WS 2022/2023

Assignment 5

Remarks

Please submit your exercises in ILIAS before 23:55 on the closing date. At least one member of the group must be present at our biweekly tutorial, beeing prepared to explain *each* exercise. Random groups will be asked to present their solutions.
Reminder: If there is a built-in function for an algorithm, you are supposed to implement, do *not* use it. If you are asked to explain your results, hand in as separate pdf-document containing your answers.
Attention: This assignentment requires you to install the `python`-package `scikit-image` via the command line with `pip install scikit-image` or via the graphical user interface.

Exercise 10: Edge Detection [5 points]

The code for this exercise is located in `exercise_10.py`.

a) Sobel operator [1 point]: Complete the three functions `sobel_x_filter`, `sobel_y_filter` and `sobel_combine`. The functions `sobel_x_filter` and `sobel_y_filter` apply a horizontal and vertical Sobel operator to an image. The function `sobel_combine` return computes the combined magnitude

$$G = \sqrt{G_x^2 + G_y^2}$$

from the horizontally (G_x) and vertically (G_y) filtered image.

b) Laplacian of Gaussian [1 point]: Complete the function `laplace_filter` that approximates a Laplacian of Gaussian filter using the difference of Gaussians.

c) Marr-Hildreth filter [2 points]: Complete the function `get_zero_crossing` that finds the zero crossings in a Laplace-filtered image to implement the Marr-Hildreth edge detector. Implement the function's behaviour according to its docstring. Think about edge cases!

d) Canny edge detector [1 point]: The function `canny` applies a Canny filter to an image. Name differences (at least the most significant one) between the results of the Canny edge detector and the Marr-Hildreth edge detector which let you recognize which edge detection was used on an image.

Exercise 11: Hough transformation [6 points]

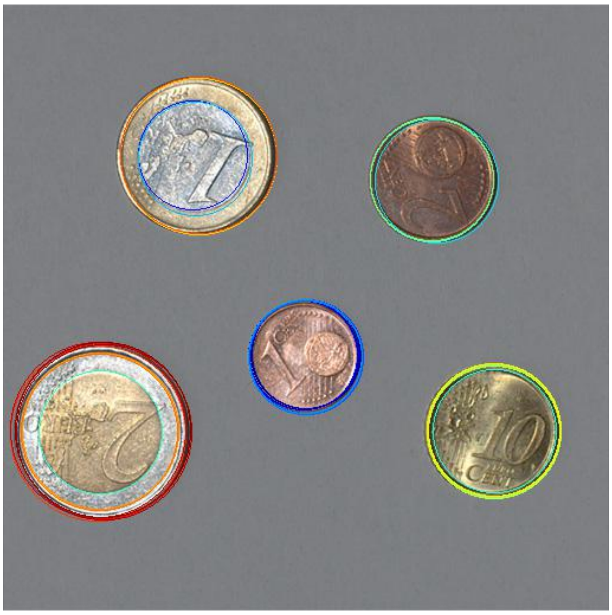

Figure 1: Circles around coins detected by Hough transformation.

a) [3 points]: Implement a function `circular_hough_transform` that transforms a binary image (1 if there is an edge, 0 otherwise) into Hough space by using an appropriate parametrization for circles. The list `radiuses` defined in the main method contains radiuses that are likely to appear in the given image. `circular_hough_transform` should return an array that contains the Hough transformations for all entries of its argument `radiuses`. *Hint:* Use fancy indexing to avoid loops. You may import and use the function `get_zero_crossing` from the previous excersise to compute the coordinates of circle points.

b) [3 points]: Complete the function `detect_circles` that detects circles in the input image by looking for accumulation points in Hough space. The result should look similar to Figure 1. *Hint:* The function `np.argpartition` may be useful to find the arguments corresponding to the k largest values in each layer of the Hough space.