Mathematisch- Naturwissenschaftliche
Fakultät

Fachbereich Informatik
Arbeitsbereich Visual Computing

Eberhard Karls
Universität
Tübingen

Bildverarbeitung I (Prof. Schilling)
WS2017/2018

Assignment 5, Due: January 11th, 2018

Remarks

You are allowed to use the built-in functions `conv2` for the following exercises. Use grayscale images for the edge detection exercises.

Exercise 11: Edge Detection [5 points]

a) Laplace operator [1 point]: Implement a function `my_laplaceEdge` that applies a Laplace operator to an image and returns the filtered image.

b) Sobel operator [1 point]: Implement a function `my_sobelEdge` that applies a horizontal and vertical Sobel operator to an image. The function should return three images: horizontally (G_x), vertically (G_y) filtered and combined magnitude ($G = \sqrt{G_x^2 + G_y^2}$).

Write a function `my_plotSobel` that creates a new figure and displays the original image and all three filtered versions of the image in one plot (with labels).

c) Marr Hildreth filter [2 points]: Implement a function `my_marrhildrethEdge` that applies a Marr Hildreth operator to an image and returns the filtered image.

The filter is composed of three steps:

- First, a difference of Gaussians is applied to the image (a fast approximation for a Laplacian of Gaussian). Accordingly, you have to implement a function `my_differenceOfGaussians` that applies a DoG to an image. (To get similar results as in the literature, you might need to multiply your Kernel by 255, because the Matlab Gaussian kernel is normalized to 1.)
- Second, the final output image is created by finding zero crossings in the DoG filtered image (areas that are no zero crossings should be black).
- Third, the image is thresholded to remove noise.

Select suitable values for `sigma0`, `sigma1`, `threshold`.

Write a function `my_plotMarrhildreth` that creates a new figure and displays all four steps (original, filtered, zero crossings, thresholded image) in one plot (with lables).

d) Canny edge detector [1 point]: Implement a function `my_cannyEdge` that applies a Canny filter to an image. You may use built-in functions for the Canny filter (e.g., `edge` with 'canny' as an argument)! Find appropriate values for the high and low thresholds.

Hint: *Based on your implementation your results may differ a little bit from the provided ground truth results. Anyway the basic characteristics of the different filters should be the same as in the provided ground truth results.*

Exercise 12: Hough transformation [6 points]

a) [1 point]: Implement a function `my_edgeDetection` that uses an edge detection algorithm of your choice to create a gradient image for the image above. The returned image should be binary (1 if there is an edge, 0 otherwise).

b) [2 points]: Implement a function `my_houghTransform` that transforms the filtered image into Hough space by using an appropriate parametrization for circles. Find circle sizes that are likely to appear in the given image and modify the array `circleSizes` directly in the script. `my_houghTransform` should return a matrix that contains the Hough transformations for all entries of the array `circleSizes`.

c) [1 point]: `my_plotHoughSpaces` should plot all of the just created Hough spaces in a plausible way.

d) [2 points]: Implement a function `my_detectCircles` that detects circles in the input image by looking for accumulation points in Hough space. Transform the found circles back into image space and show them in the input image.

HINT: *The calculation of the Hough spaces can be really time consuming. For testing your code you should comment out the solution call (but don't forget to un-comment them again).*
HINT: *Again, based on your implementation the results may differ. See the solutions as a hint, where the results should look similar. In the end your algorithm should identify similar (but maybe not all) circles as the provided solution.*
HINT: *Have a look at the Mathworks examples for shapeInserter objects from the computer vision system toolbox. It might come in handy when drawing circles.*