

INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 4 - Edges and Hough Transform

Submission Date: 1.1.2023

Introduction

In this assignment, we will learn about edge detection and the hough transform. Although MATLAB has many built-in functions, our goal is to learn image processing by doing things ourselves. Please avoid using built-in MATLAB functions unless clearly stated otherwise. However, feel free to compare your own functions to the built-in functions.

Before you start, please read the submission instructions at the end of the exercise and follow them during your work. For any questions regarding this assignment please refer to the course forum on the moodle web site, for personal questions **only** please email shaked0@post.bgu.ac.il

1 Edge Detection (40%)

In this section, we will play with some common edge detectors.

1.1 Reading the Image

Read the image named *cameraman.tif*. Normalize the image between $[0, 1]$. This image is built-in into MATLAB, you can simply read it by writing the line `imread('cameraman.tif')`. For those of you who will not be able to do it, the image is added to the exercise file.

1.2 Prewitt Edge Detector

The Prewitt edge detector was developed to solve some of the problems of the Roberts detector. Its derivative kernels are defined as:

$$G_{Px} = \frac{1}{6} \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}, G_{Py} = \frac{1}{6} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

1. Write your own function named `dip_prewitt_edge(img,thresh)` that will apply the Prewitt edge detector on `'img'` and output the edge image with the same size as the input image. The `'thresh'` parameter will determine the gradient magnitude cutoff threshold. Note: You don't have to deal with the image boundaries, you can use the `conv2()` function with the parameter `'same'`.

2. Display 2 edge images generated using `dip_prewitt_edge(img,thresh)` with 2 different thresholds. explain the differences between the results.

1.3 Canny Edge Detector

The Canny edge detector is the most commonly used edge detector.

1. Read about the MATLAB function `edge(I,'Canny')`. What are the optional parameters of this function? What are the default values assigned to these parameters?
2. Use the MATLAB functions `edge(I,'Canny')` with two sets of parameters: the default set and another set of your choosing. Explain the difference. Which set is better in your opinion? Explain.
3. If the answer to the question above was "default options are better" try to change the parameters for a better result, in your opinion. *Hint:* you can capture a certain parameter used by `edge` as a second output argument (which one?).

2 Hough Transform (60%)

In this exercise we will learn the Hough transform both for line and circles detection. You **may NOT** use the following MATLAB functions: `hough()`, `houghlines()`, `imfindcircles()`.

1. Hough line transform

- (a) Read the *floor.jpg* image (enclosed to this exercise), convert it into grayscale image and normalize to $[0,1]$.
- (b) Extract the edges using MATLAB's `BW=edge(I)`. Which edge detector is used by default? What are its default parameters? (parameter meanings, and their values).
- (c) Write your own `dip_hough_lines(BW,R0,Θ0)` function that calculates the Hough Matrix for finding **lines**. Use 2 different quantization of R and Θ : $(R_0, \Theta_0) = (1, 1)$, $(R_0, \Theta_0) = (5, 4)$ as explained in the abstract algorithm mentioned below this section.
- (d) For each of the results in the previous question display the Hough matrix using `imshow(M,[])` and explain the result.
- (e) Find the 4 most significant lines in the BW image, and plot them on the grayscale floor image for each set of (R_0, Θ_0) . Use MATLAB's `houghpeaks` function to find the peaks of your Hough matrix.
- (f) Explain the results. In your explanation, consider the advantages and disadvantages of high values of R_0, Θ_0 vs. low values.

2. Hough circle transform

- (a) Read the *coffee.png* image (enclosed to this exercise), convert it into grayscale image and normalize to $[0,1]$.
- (b) Extract the edges using MATLAB's `BW=edge(I)`.

- (c) Write your own `dip_hough_circles(BW,R0,Θ0)` function that calculates the Hough Matrix for finding **circles**. Use 2 different quantization of R and Θ : $(R_0, \Theta_0) = (1, 1), (R_0, \Theta_0) = (4, 10)$ as explained in the abstract algorithm mentioned below this section. *Hint:* you should bound your search for radius values $80 < R < 100$.
- (d) Measure the Run-Time of your function using `"tic; dip_hough_circles(BW, R0, Θ0); toc; "` for $(R_0, \Theta_0) = (1, 1)$. Try to find (R_0, Θ_0) pair that has the same accuracy of $(1, 1)$ but runs faster. What is the $speedup = \frac{T_{old}}{T_{new}}$?
- (e) This time your Hough matrix is 3D, display one slice (2D image) using `imshow(M,[])` for $(R_0, \Theta_0) = (1, 1)$, $(R_0, \Theta_0) = (4, 10)$, and the set of (R_0, Θ_0) you found in the previous question.
- (f) Write you own `dip_houghpeaks3d(HoughMatrix)` function to find the 5 most significant circles of different cups of coffee in the BW image. Plot them on the grayscale coffee image for each quantization (as in the previous question). See the note at the end of the question.
- (g) Explain the results. In your explanation, consider the advantages and disadvantages of high values of R_0, Θ_0 vs. low values.

In order to construct the Hough matrix follow these steps:

1. For BW of size $M \times N$, create an empty matrix of size $|R| \times |\Theta|$ where $|R|$ denotes the length of the vector $R = -\sqrt{M^2 + N^2} : R_0 : \sqrt{M^2 + N^2}$, and $|\Theta|$ denotes the length of the vector $\Theta = -90 : \Theta_0 : 90$.
 - (a) For every white pixel in the image, $x, y \in BW$:
 - i. For every value in $\theta \in \Theta$:
 - A. $r = x \cos(\theta) + y \sin(\theta)$
 - B. Add 1 to the Hough matrix at the location corresponding to r, θ . Note: you will need to round r for an available $r \in R$

In order to construct the Hough matrix **for circles** follow these steps:

1. For BW of size $M \times N$, create an empty matrix of size $M \times N \times |R|$ where $|R|$ denotes the length of the vector $R = R_{min} : R_0 : R_{max}$
 - (a) For every white pixel in the image, $x, y \in BW$:
 - i. For every value of $r \in R$:
 - A. For every value of $\theta \in 0 : \Theta_0 : 360$:
 - $a = x - R \cos(\theta)$
 - $b = y - R \sin(\theta)$
 - Add 1 to the Hough matrix at the location corresponding to (a, b, R) . (Don't forget to round the values)

Note: This time your Hough matrix is 3D, so `houghpeaks` may not work. To find the 5 peaks you may use this piece of code instead:

```

peaks = zeros(5,3);
for i = 1:5
    [val,idx] = max(H(:));
    [idx1,idx2,idx3] = ind2sub(size(H),idx);
    peaks(i,:) = [idx1,idx2,idx3];
    H(idx1,idx2,idx3) = 0;
end

```

3 Bonus

Use the Generalized Hough Transform (GHT) in a creative way. Take two images, one is a binary image of an interesting shape, and the other is an image of your liking. Use the GHT to find the shape of the first image in the second image and show the results. **Be creative! One of the images will be chosen by the course staff and it's authors will receive one bonus point to the final grade.** The staff will judge by the visual result, originality and the code.

In order to implement the GHT, you can use the added *Generalized_hough_transform_standalone.m* function or any other implementation you find online.

Submission Instructions

The following instructions are mandatory and will be checked and graded by the course staff. Failing to follow these instructions **will** reduce points from you grade.

The assignment is to be done in MATLAB, preferably with MATLAB notebook. **Note:** We recommend submitting Ex*.mlx with the exported PDF (generated by the .mlx). Submit your assignment to the course moodle page in the form of a *.zip (**not RAR**) containing *Ex3.mlx* or *Ex3.m* - the main MATLAB file, other *.m files and images along with a report in the form of a PDF file (NOT .doc). **Both the PDF and ZIP file names should be the initials and ID of both of the team members ex. 'SC-1234567_YE-7654321.pdf' and 'SC-1234567_YE-7654321.zip', respectively.**

Academic integrity: the originality of the submitted exercises **will be checked.**

Document Instructions

- Only one of the team members should submit the file
- The report should be written in Hebrew or English.
- Each section should have the relevant title as is in this document.
- Every image should be accompanied with the relevant explanation.
- The displayed images should be large enough for us to see them.
- The document should be organized and readable.

Code Instructions

- Use MATLAB version 2014b or later. If you don't have one on your computer, you can work from the computer laboratories in building 33 using VPN.
- A **main** function should call all the section functions in the correct order and should be named *Ex4.mlx* or *Ex4.m*.
- The first line of *Ex4.mlx* / *Ex4.m* should print the full names and IDs of all team members. Use MATLAB's *disp()* function.
- Write modular functions for the subsections and reuse those functions throughout your code whenever possible.
- Every *.m file should start with a comment containing the full names and IDs of all team members.
- Use meaningful names for all functions and variables.
- Try to avoid overriding variables.
- Write comments for every line of code that is not completely self explanatory.
- For every image displayed give a meaningful title using MATLAB's *title()* function.
- Use subplots whenever possible.
- All paths to files should be relative paths. If you are using subfolders use MATLAB's *fullfile()* function to construct the path to the file. Do not hard code '/' or '\' in the paths.
- The code should run completely without errors. A project with errors **will not be checked!**