DIGITAL IMAGE PROCESSING COURSE - 505060
PRACTICE LABS

LAB 09. EDGE DETECTION AND HOUGH TRANSFORM

Requirements

(1) Follow the instructions with the help from your instructor.
(2) Finish all the exercises in class and do the homework at home. You can update your solutions after class and re-submit all your work together with the homework.
(3) Grading
Total score = 50% * Attendance + 50% * Exercises
Rules:
- If the number of finished exercises is less than **80% total number of excercises**, you will get **zero** for the lab.
- Name a source file as "**src_XX.py**" where XX is the exercise number, for ex., "src_03.py" is the source code for the Exercise 3.
- Add the text of your Student ID to each of the output image.
- Name an output image as "**image_XX_YY.png**" where XX is the exercise number and YY is the order of output images in the exercise, for ex., "image_03_02.png" is the second output image in the Exercise 3.
- Submit the source code and output image files directly to Google classroom assignment, donot compress the files.

If you submit the exercises with wrong rules, you will get **zero** for the lab or the corresponding exercises.

(4) Plagiarism check
If any 2 of the students have the same output images, then all will get zero for the corresponding exercises.

INTRODUCTION

In this Lab, you will learn how to
- Edge detection
- Line detection
- Circle detection

INSTRUCTIONS

---

1. Edge detection

Edges are characterized by sudden changes in pixel intensity. To detect edges, we need to go looking for such changes in the neighboring pixels. Let's explore the use of two important edge-detection algorithms available in OpenCV: Sobel Edge Detection and Canny Edge Detection.

Look at Python code in the reference link below to learn more about edge detection algorithms.

Reference:
Edge Detection Using OpenCV | LearnOpenCV #
Test image:
https://aishack.in/static/img/tut/sudoku-original.jpg

2. Line detection using Hough transform

The Hough Line Transform is a transform used to detect straight lines. To apply the Transform, first an edge detection pre-processing is desirable.

OpenCV implements two kinds of Hough Line Transforms: Standard and Probabilistic Hough Line Transform.

a. The Standard Hough Transform

It gives you as result a vector of couples (θ, rθ). In OpenCV, it is implemented with the function ***HoughLines***()

b. The Probabilistic Hough Line Transform

This is a more efficient implementation of the Hough Line Transform. It gives as output the extremes of the detected lines (x0,y0,x1,y1). In OpenCV, it is implemented with the function ***HoughLinesP***()

Look at Python code in the reference link below to learn more about Hough Line Transform.

Reference: OpenCV: Hough Line Transform

Test image: https://aishack.in/static/img/tut/sudoku-original.jpg

3. Circle detection using Hough transform

The Hough Circle Transform works in a *roughly* analogous way to the Hough Line Transform explained in the previous tutorial. Use the OpenCV function **HoughCircles()** to detect circles in an image. For sake of efficiency, OpenCV implements a detection method slightly trickier than the standard Hough Transform: The Hough gradient method, which is made up of two main stages. The first stage involves edge detection and finding the possible circle centers and the second stage finds the best radius for each candidate center.

Look at Python code in the reference link below to learn more about Hough Line Transform.

Reference: OpenCV: Hough Circle Transform

Test image:http://amroamroamro.github.io/mexopencv/opencv/hough_circles_demo_01.png

EXERCISE

Ex1. Save the output images named by the algorithm used respectively in the Instruction 1, 2, 3.

Ex2. SuDoKu Grabber in OpenCV

In this exercise, we 'll look at detecting a SuDoKu puzzle. Your task is to detect the grid by using hough line transform. This includes all preprocessing done on the image: filtering the image to ensure we're not affected too much by noise.

Follow the instructions in the attached PDF file:

"Lab9_Ex2_SuDoKu Grabber in OpenCV_ Grid detection - AI Shack.pdf"

Understand C++ code and then change into Python to finish the demo in the article.

Download the image via this link

Ex3. Detect and read barcodes with OpenCV in Python

See https://note.nkmk.me/en/python-opencv-barcode/

or Lab9_ex3_Detect and read barcodes with OpenCV in Python _ note.nkmk.me.pdf

https://docs.opencv.org/4.x/dc/df7/classcv_1_1barcode_1_1BarcodeDetector.html#a305fe500187fd79111f6542b39885a03

➔ detectAndDecodeWithType(…)