ICBV241 HW 2

Release Date: 21/02/2024

Submission Deadline: 06/03/2024, 23:59

Instructions

Please read and follow these instructions carefully.

- To be able to submit your work you must first enroll in a group on Moodle.
- The assignment contains both programming tasks and a written part. The written part needs to be submitted in a single PDF file and the programming tasks must be answered in the provided Jupyter notebook. You need to submit a single zip file where:
  - The name of the file is "Assignment2-Group#.zip", where # should be replaced by your group number.
  - In the zip file put the written answers as a PDF file and the Jupyter notebook (.ipynb file) along with any additional files used by your code.
- The written assignment must be typed on the computer. Handwritten scanned documents **will not** be accepted.
- Make sure that the Jupyter notebook can be run. Before submitting, restart runtime and run all cells. In Google Colab: 1) *Runtime > Restart and run all;* 2) *File > Download .ipynb*. In addition, make sure you submit any additional files used by your code (usually image files).
- Both the PDF containing the written answers and the Jupyter notebook must contain the id numbers and the names of the submitting students.

Question 1 – Edge Detection

Section A

You are given the following grayscale image:

$$I(x, y) = x + sin\, y$$

Let E_1 be an edge detector based on finding the local maximum of $|\nabla I|$ and E_2 be an edge detector based on finding the zero-crossing of the Laplacian ΔI. Write the mathematical expressions of the edge points detected by each of the operators. Are the results identical?

Guidance: Compute analytically the edge points detected by each of the edge detectors when applied to the image. For E_1 this means you should first find the expression of $|\nabla I(x, y)|$ (the gradient's magnitude), and then find the local maximum points of $|\nabla I(x, y)|$. Recall that in order to find extremum points you should compute the first derivative of the expression you found for $|\nabla I(x, y)|$ and equate it to zero. Then, you compute the second derivative of the expression you found for $|\nabla I(x, y)|$, and compute its value at the extremum points you previously found. If the value of the second derivative at this point is greater than 0, then it is a local minimum. If the value of the second derivative at this point is smaller than 0, then it is a local maximum. For E_2 this means you should first find the expression of the Laplacian $\Delta I(x, y)$. Then, you should find all the points where $\Delta I(x, y) = 0$.

Section B

In practical session #6 you have been shown an implementation of the LoG (this implementation is also provided in the attached Jupyter notebook). Locating edges via finding values close to zero in the Laplacian is too noisy. Hence, you will implement an edge detector that uses a stricter criterion of zero-crossing. Let LoG denote the Laplacian of Gaussian for an image I. For some threshold values $a \geq 0 \geq b$, a pixel (p_x, p_y) is considered a zero-crossing pixel if one of the following is true:

- $LoG[p_x, p_y] > a$ and a pixel (p'_x, p'_y) exists in the 8-environment of (p_x, p_y) so that $b > LoG[p'_x, p'_y]$.
- $b > LoG[p_x, p_y]$ and a pixel (p'_x, p'_y) exists in the 8-environment of (p_x, p_y) so that $LoG[p'_x, p'_y] > a$.

Your goal is to detect the edges of the coins present in the attached **coins_edges.jpg** image. Experiment with the values of *a, b* and the parameters of the Gaussian used for smoothing, to find the values that generate the best result on the provided coins' image (evaluate the quality of the result by its appearance). Plot the following computational steps (i.e., a total of 4 plots):

1. The original image in grayscale.
2. The smoothed image (i.e., the image after applying the Gaussian smoothing).
3. The image after applying LoG.
4. The image of the detected edges after applying zero-crossing.

Note: The 8-environment of a pixel (p_x, p_y) is defined as: $(p_x - 1, p_y)$, $(p_x - 1, p_y - 1)$, $(p_x, p_y - 1)$, $(p_x + 1, p_y - 1)$, $(p_x + 1, p_y)$, $(p_x + 1, p_y + 1)$, $(p_x, p_y + 1)$, $(p_x - 1, p_y + 1)$.

Section C

In this section you are provided with the same coins' image with Gaussian noise added to it (in the attached Jupyter notebook the noisy image is called '*noisy_coins*'). Find the edges of the coins in this image using the zero-crossing method you created in section B, and using OpenCV's implementation of Canny's algorithm. Try to find good parameters for both methods. Display, side by side, the best result you achieved on the noisy image using your method and Canny's algorithm. Which method produced better results? Why do you think this is the case?

Question 2 – Parametric Curves

Section A

In class you saw how a parametric curve can be described as mapping from an interval to the image plane:

$$\alpha(t) = \big(x(t), y(t)\big)$$

A 3D curve can be described in the same way, as mapping from an interval to a 3D space:

$$\alpha(t) = \big(x(t), y(t), z(t)\big)$$

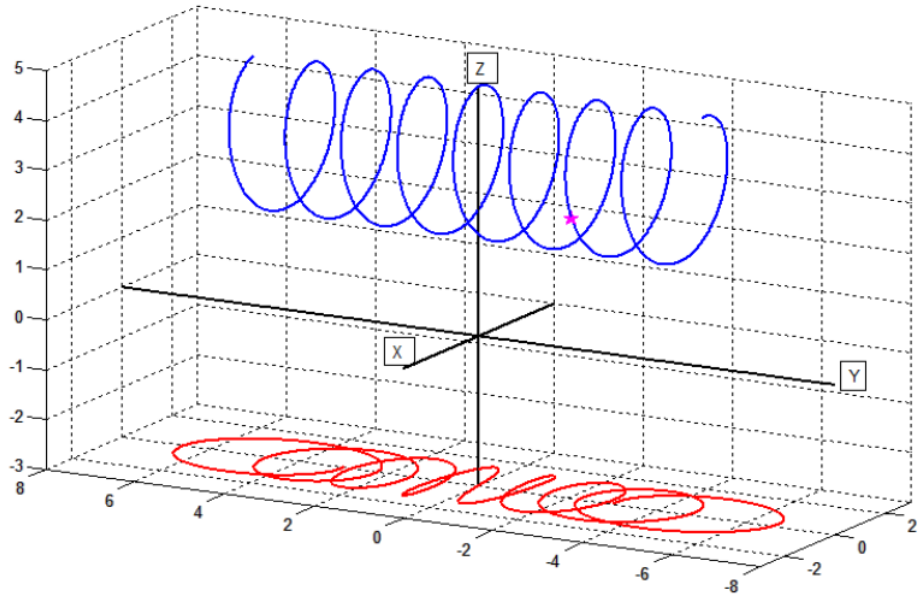
Use this description to prove that a perspective image of a straight line in the 3D world is a straight line in the image plane.

Reminder:

A parametric description of a line in the plane is $\alpha(t) = (x_0, y_0) + t \cdot (a_1, a_2)$, and a parametric description of a line in space is $\alpha(t) = (x_0, y_0, z_0) + t \cdot (a_1, a_2, a_3)$.

Section B

The image plane of a pinhole camera is parallel to the XY plane, its pinhole is located at the world's origin, and its focal length is $f = 3$. The following graph presents an object in the world (painted in blue) which is the trace of a Helix curve $\alpha(s)$, and one of its points is marked by a pink star. In addition, the graph presents the trace of the curve $\beta(s)$ (painted in red), which is the perspective projection of $\alpha(s)$ on the image plane. Examine the object and its projection, and answer the following questions:



1. Mark on top of the graph (using a star) the point in the image plane which is the projection of the object point marked by a pink star.
2. Describe the object as a parametric curve $\alpha(s)$ for $s \in [-8\pi, 8\pi]$, while taking in consideration the following additional information:
   - $\alpha(0) = (0,0,5)$
   - $\alpha(-8\pi) = \left(0, \frac{-8\pi}{5}, 5\right)$
   - The z-coordinate of the object points which are closest to the image plane is $z = 2$.
   - The z-coordinate of the object points which are furthest from the image plane is $z = 5$.
3. Compute a parametric description for $\beta(s)$, based on $\alpha(s)$ you defined.

Question 3 – Hough Transform

In this question you will implement a python code which uses Hough Transform to detect circles in the provided **coins_Hough.png** image. Your code should perform the following computations:

1. **Produce an edge map from the image using an edge detector of your choice.** The edge map is a binary image, the same size as the 'coins_Hough' image, with a value of one in each detected edge pixel and a value of zero anywhere else. Choose the best algorithm and parameters for the edge detection (in your opinion), and explain (in written response) your choice. Plot the edge map in the provided Jupyter notebook.
2. **Detect all the circles in the image using Hough transform.** Define the appropriate Hough space and explain your choice (that is, explain the parameters of the Hough space, and why you chose this representation). You should define the bucket size for each of the Hough transform parameters (i.e., the number of values you scan in the range of each parameter). Explain your

choices and the considerations behind it (in written response). Detect the circles, i.e., the local maximum points in the Hough space. Use a method of your choice for detecting the local maximum points. Explain the method you chose and any parameters you had to set for it (in written response).

3. **Plot the detected circles on top of the original coins image** in the provide Jupyter notebook.

Follow the following guidelines when answering this question:

- You cannot use any built-in OpenCV method to perform the Hough transform, you must implement your own version of the Hough transform for circle detection.
- You may use convolution to count the number of votes for each possible circle. It is not obligatory to use convolution, but it may significantly reduce the runtime of your algorithm.
- Note that using a constant threshold for the number of votes (to isolate maximum points) will most probably not work. You will have to find another solution for detecting the maximum points in the Hough space.
- All the plots you were asked to draw should be presented in the provided Jupyter notebook. In addition, you should provide in the PDF your written responses explaining the choices you made throughout your implementation (the methods and parameters you chose in the different steps).