Image Processing Project

Name	ID
Nour El-Din Tarek	20210426
Abo Bakr Khaled Amin	20210004
Aly Khaled Fekry	20210244
Aly Mohamed Hussian	20210248
Mahmoud Mamdouh Abdelaziz	20210376

Edge Detection using Sobel Filters

Abstract

This report presents an implementation of edge detection using Sobel filters. The algorithm is applied to an input image, and the resulting edge magnitude image is displayed. The report outlines the problem definition, objectives, and methodology used to achieve edge detection.

Problem Definition

Edge detection is a fundamental task in computer vision, aiming to identify the boundaries or edges within an image. Edges are significant features that help in understanding the structure and content of an image. The problem involves developing an algorithm that can accurately detect edges in a given image.

Objectives

The primary objectives of this project are:

1. To implement an edge detection algorithm using Sobel filters.
2. To apply Gaussian blur to the input image to reduce noise.
3. To calculate the gradient magnitude using the Sobel kernels.
4. To display the original image, grayscale image, blurred image, and the resulting edge magnitude image.

Methodology

The methodology used to achieve edge detection involves the following steps:

1. Image Loading and Preprocessing:
    - Load the input image using *cv2.imread*.
    - Convert the image to RGB format using *cv2.cvtColor*.
    - Convert the image to grayscale using a weighted sum of RGB channels.
2. Gaussian Blur:
    - Apply Gaussian blur to the grayscale image using a kernel of specified size.
    - The *gaussian_blur* function creates a Gaussian kernel and convolves it with the input image.
    - Sobel Edge Detection:
    - Use the *get_sobel* function to obtain the Sobel kernels for x and y directions.
    - Convolve the blurred image with the Sobel kernels to calculate the gradient in x and y directions (Ix and Iy).

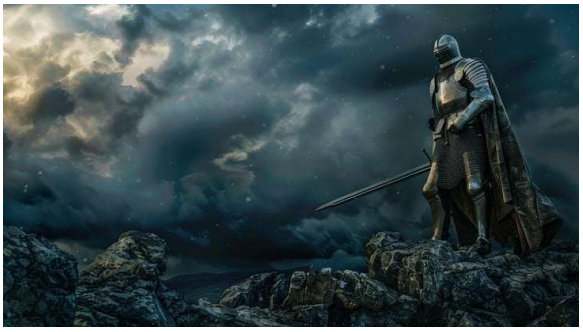
- Calculate the gradient magnitude (*G*) using the *calc_magnitude* function, which computes the hypotenuse of *Ix* and *Iy*.

3. Displaying Results:
    - Display the original image, grayscale image, blurred image (if *show_steps* is True), and the resulting edge magnitude image using *cv2.imshow*.
    - Optionally save the edge magnitude image to a file if *save_result* is True.

The implementation is done in **Python**, utilizing **OpenCV** and **NumPy** libraries for image processing and numerical computations. The *edge_detection* function encapsulates the entire process, taking the image path, blur size, and optional flags for displaying intermediate steps and saving the result.

Results

Before



After