Bachelor in Computer Science

COMP 425: Computer Vision

Homework 1 Report

Student Name: Alejandro Leonardo García Navarro

Student ID: 40291834

Email: al.garcia636@gmail.com

Instructor: *Yang Wang*

Academic Year: 2023/24

Date: February 20th, 2024

Contents

1 Introduction

This report encapsulates the outcomes and insights gained from the completion of the first homework assignment for the course COMP 425 (Computer Vision). The assignment is structured into four distinct parts: Image Filtering, Edge Detection, Corner Detection, and Image Subsampling, each designed to deepen our understanding and practical skills in fundamental computer vision techniques.

In fulfilling the requirements of this homework, a series of image processing algorithms were implemented and applied to various images, as specified in the assignment instructions. Part 1 focuses on 2D image filtering, emphasizing the implementation of custom filters and gradient calculation methods. Part 2 and Part 3 delve into edge and corner detection, respectively, challenging us to not only apply but also to understand the underlying principles of these critical computer vision tasks. Part 4 explores the concept of image subsampling, illustrating the effects of different downsampling techniques on image quality.

This report not only presents the results obtained through the implementation of these algorithms but also reflects on the learning process, highlighting both the technical challenges encountered and the solutions devised to overcome them.

To ensure clarity and ease of understanding, all code has been meticulously commented, and the report is organized to systematically present the approach, results, and analysis for each part of the assignment.

2 Part 1 (Image Filtering)

In this very first part of the project, there were two tasks that had to be carried out:

1. Implement a function that performs 2D filtering. For this, it is important to recall from the lecture notes that 2D filtering consists on filtering with an upright kernel (contrary to convolution).
2. Implement two functions (partial_x and partial_y) that calculate the gradient image along the x and y directions.

The code can be checked in the files uploaded upon submission, but in the report it is required to explain the filter used in each of the functions that calculate the partial derivatives. In this case, the chosen filter was the **Sobel filter**.

At its core, the Sobel filter is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. The technique is used to emphasize regions of

high spatial frequency that correspond to edges. Typically, it is applied separately to the X and Y axes of an image to find edges in both horizontal and vertical directions. It is represented as follows:

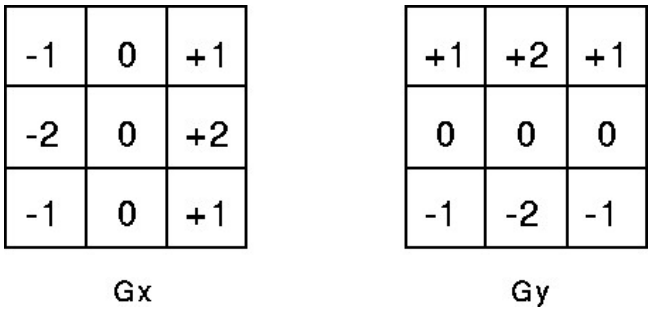

Figure 1: Sobel Filter

One of the reasons why I chose this filter is because it is simple to understand and implement. However, above all, the main reason is that it is robust to noise. Compared to simpler filters, Sobel includes a smoothing effect in its operation due to the coefficients in its kernel. This makes it more robust against noise in the image, as it performs a sort of localized averaging before calculating the derivative.

3 Part 2 (Edge)

Moving on with **Part 2**, we had to implement an edge detector. This was easily done following these steps:

1. Load the desired image.
2. Smooth the image with the Gaussian kernel.
3. Compute the x and y derivative on the smoothed image.
4. Compute the gradient magnitude.
5. Visualize the results.

These were applied to the image of an iguana, and the results from Figure 2 were obtained:

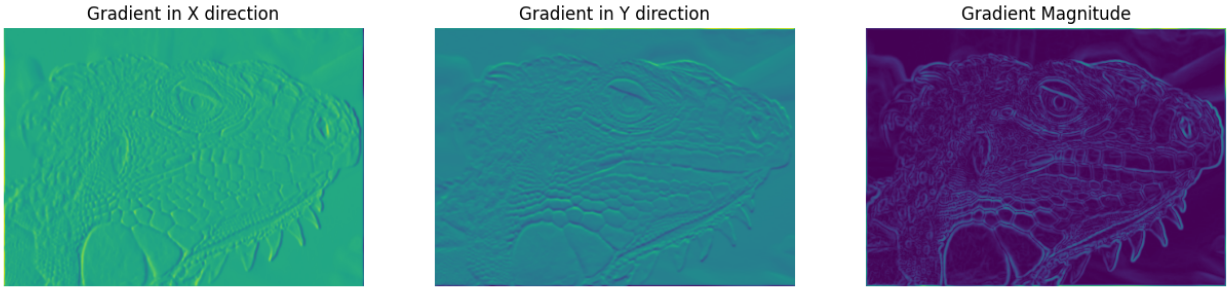

Figure 2: Edges from the iguana picture

4 Part 3 (Corner)

Constituting the third part of the project, the task involved filling the missing code in the "corner.py" script to implement the Harris corner detection algorithm. This was later applied to an image named "building.jpg," focusing on identifying and visualizing corner points within the image.

The Harris corner detector is a method in computer vision for detecting corners, which are points in the image with significant variation in intensity in all directions. The implementation was structured around these fundamental steps:

1. Compute x and y derivative of the image.
2. Compute products of derivatives at every pixel.
3. Apply a filter (uniform for simplicity) to weighted derivatives to obtain a smooth representation of gradient changes over the window.
4. Compute the Harris corner response at each pixel. The formula assesses the likelihood of a pixel being a corner based on the local gradient distribution.
5. Threshold on response.
6. Perform non-max suppression by finding peak local maximum.
7. Visualize results.

After performing these steps to the "building.jpg" image, the results from Figure 3 were obtained:

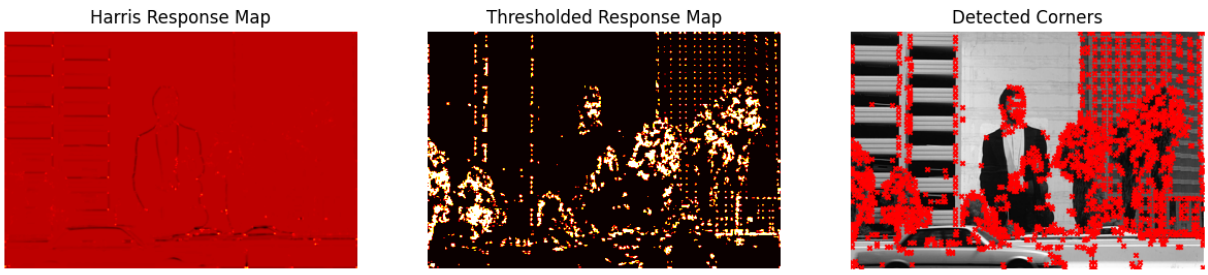

Figure 3: Corners from the building picture

5 Part 4 (Image Subsampling)

This last segment of the project focused on modifying the ”downsample.py” script to execute image downsampling on the image ”paint.png.” The objective was to reduce the image size by a factor of 2 across 5 consecutive levels, both with and without addressing the issue of aliasing.

Image downsampling is a common operation in image processing, used to decrease the resolution of an image. Two approaches were explored: naïve downsampling, which directly reduces the image size without additional processing, and anti-aliased downsampling, which incorporates a smoothing step to prevent aliasing effects.

- **Naive Downsampling:** As mentioned, the naïve downsampling approach straightforwardly reduces the resolution of the image by selecting every other pixel in both the horizontal and vertical dimensions. This method is fast but introduces aliasing. This occurs when high-frequency details in the original image are not adequately represented in the downsampled image, leading to distortion.
- **Anti-Aliased Downsampling:** To mitigate the aliasing observed in the naïve downsampling method, a Gaussian filter is applied to the image before downsampling. The Gaussian filter smooths the image, effectively removing high-frequency components that cannot be accurately represented at lower resolutions. This pre-processing step is crucial for preserving the visual integrity of the image as its size is reduced.

Finally, the results of both downsampling methods were visualized in a single figure comprising two rows of subplots (Figure 4). The first row showcases the outcomes of naïve downsampling, and the second row presents the results of anti-aliased downsampling.

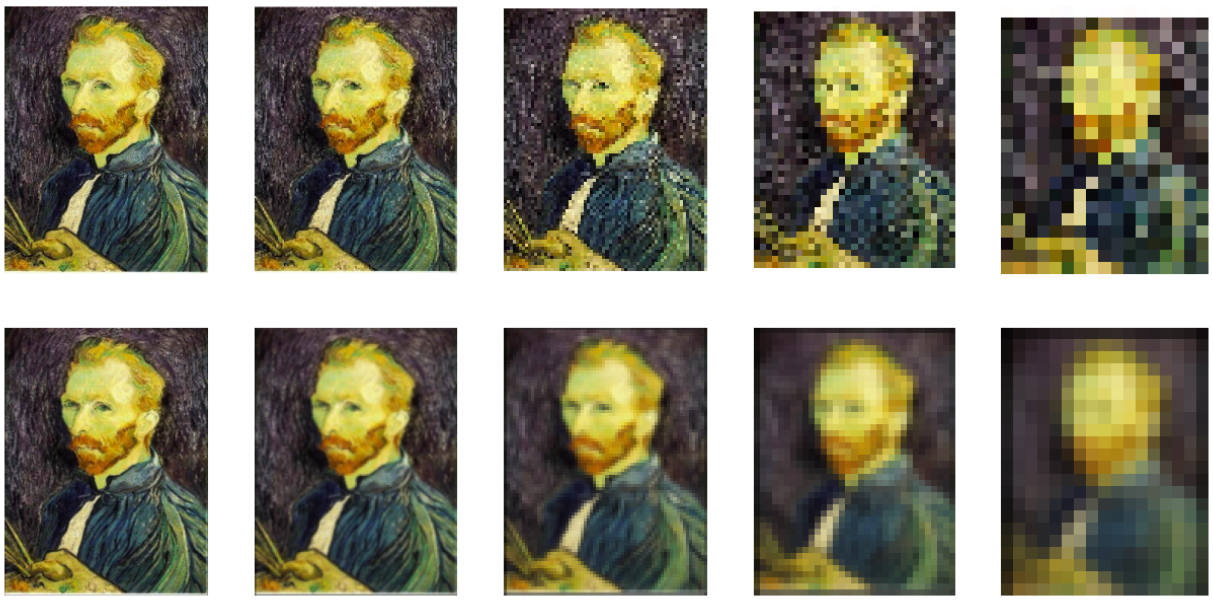

Figure 4: Downsampling methods

6 Conclusions

The completion of this first homework offered a deep dive into essential computer vision techniques, spanning image filtering, edge and corner detection, and image subsampling.

Implementing these algorithms provided practical insights into their operational intricacies and their vital roles in analyzing and interpreting visual data. The tasks underscored the importance of preprocessing, such as noise reduction and anti-aliasing, in enhancing algorithmic outcomes.

Challenges encountered along the way were key in refining problem-solving skills and deepening the understanding of theoretical concepts through hands-on application. The visual results not only demonstrated the effectiveness of each method but also highlighted the importance of careful technique selection based on the desired outcome.