Desktop-based Toolkit for Image Processing and Basic Computer Vision Techniques

Team 07

Faculty of Engineering - Cairo University

Dr. Ahmed Badawy

Eng. Laila Abbas

Collaborators

Name	Section	B.N
Amir Hesham Khalil	1	10
Omar Mohamed Ahmed	1	46
Ahmed Emad	1	3
Hazem Zakaria	1	16
Omar Tarek	1	42

Introduction

In today's digital age, where images play a crucial role in communication, documentation, and creative expression, the need for efficient and accessible image processing tools is ever-growing. Our project introduces a desktop application focused on fundamental image processing techniques, catering to the diverse needs of users across various domains. This application serves as a comprehensive solution, offering a user-friendly platform for tasks such as noise addition, edge detection, thresholding, and filtering.

By integrating common filters including average, median, gaussian, low and high-pass filters, our application empowers users to enhance and manipulate their images with ease. Whether it's removing unwanted noise, accentuating edges for better visualization, or refining image details through precise filtering, our application provides the tools necessary for effective image processing.

Furthermore, our application goes beyond mere processing by enabling users to generate image histograms, facilitating detailed pixel intensity analysis. This feature not only enhances the understanding of image characteristics but also assists in making informed decisions regarding image adjustments and enhancements.

Through this report, we aim to provide a comprehensive overview of our image processing application, covering its features, architecture, implementation, and future prospects. By offering an accessible and intuitive tool for image manipulation and analysis, we seek to empower users from diverse backgrounds to harness the full potential of digital imagery in their respective fields.

Experiments and Results

1. Noise Addition:

Noise addition is a fundamental process in computer vision, aimed at simulating real-world conditions or introducing controlled disturbances to digital images. Noise, in this context, refers to unwanted variations or artifacts that distort the original image, often stemming from factors such as bad camera sensor, dirty camera lens, electronic interference, etc.

Here we are showcasing the different noise types supported by our software and the effect of their corresponding parameters on the results:

A) Uniform Noise:

0% Noise Strength Applied

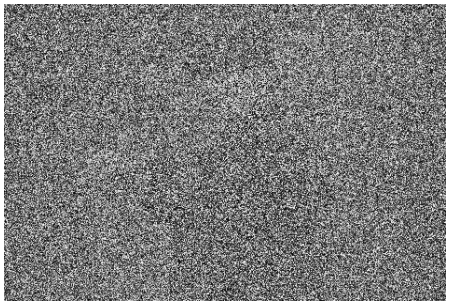
50% Noise Strength Applied

90% Noise Strength Applied

B) Gaussian Noise:

0 Mean - 0 Standard deviation

25 Mean - 30 Standard deviation

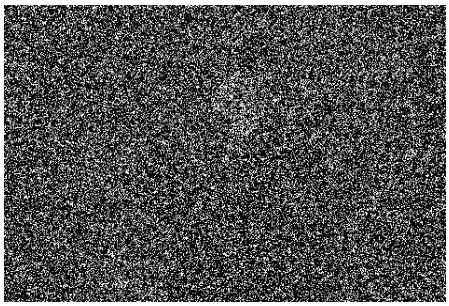


50 Mean - 75 Standard deviation

C) Salt and Pepper Noise:

0.0 Salt and Pepper ratio

0.15 Salt and Pepper ratio



0.55 Salt and Pepper ratio

2. Filters:

In the realm of image processing, filters play a pivotal role in modifying images to enhance their quality or extract valuable information.

A. Average Filter:

The average filter, also known as the box filter, computes the average intensity of the pixels in the neighborhood of each pixel. The size of the kernel is denoted by NxN.

Effect of Kernel Size:

- Small Kernel Size: As the kernel size decreases, the smoothing effect becomes less pronounced. Edges and finer details in the image are preserved, but noise reduction may be insufficient.

Image showing the effect of applying a kernel size of 5x5 on the original image.

- Large Kernel Size: Conversely, enlarging the kernel size leads to stronger smoothing. Fine details are blurred out, but noise reduction is more effective.

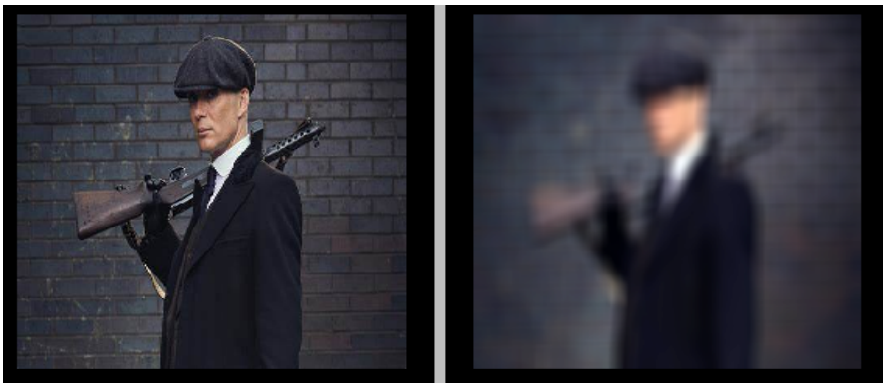


Image showing the effect of applying a kernel size of 21x21 on the original image.

B. Median Filter:

The median filter replaces each pixel's value with the median value of its neighborhood. It is particularly useful in removing salt-and-pepper noise while preserving edges.

Effect of Kernel Size:

Similar to the average filter, altering the kernel size impacts the degree of noise reduction and edge preservation. Smaller kernel sizes offer less noise reduction but preserve finer details better.

C. Gaussian Filter:

The Gaussian filter applies a convolution operation using a Gaussian kernel, which emphasizes central pixels and diminishes those farther away, akin to a bell curve.

Effect of Kernel Size and Sigma (Standard Deviation):

- Kernel Size: Increasing the kernel size broadens the smoothing effect, leading to more pronounced blurring.

Image showing the effect of applying a kernel size of 47x47 ,Sigma of 4 on the original image.

- Sigma: A larger sigma value increases the spread of the Gaussian function, resulting in smoother images.

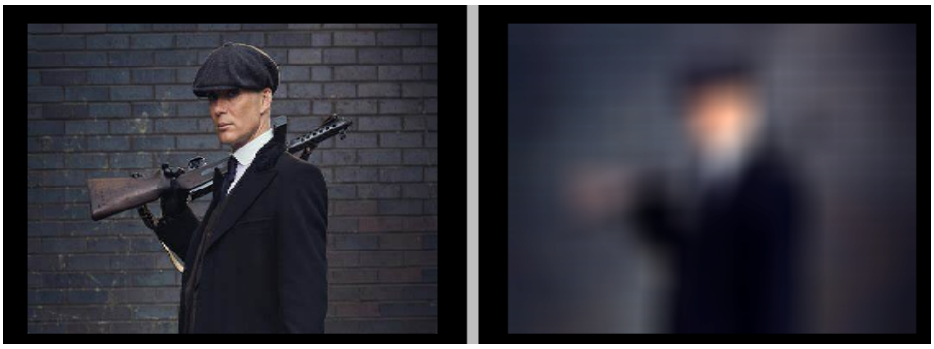


Image showing the effect of applying a kernel size of 47x47 ,Sigma of 47 on the original image.

D. Gaussian Filter:

A low-pass filter attenuates high-frequency components of the image while allowing low-frequency components to pass through, effectively smoothing the image.

Effect of Cutoff Frequency:

Lowering the cutoff frequency results in more aggressive filtering, reducing both noise and high-frequency details.

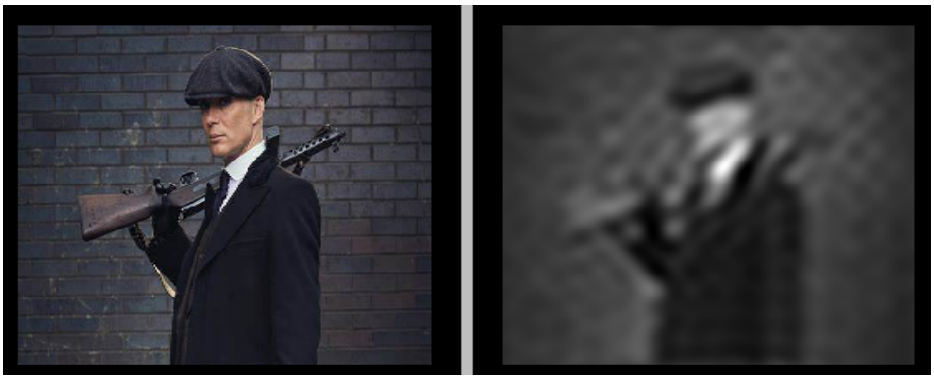


Image showing the effect of applying a cutoff frequency of 13 on the original image.

Raising the cutoff frequency preserves more high-frequency information, but noise and finer details might remain prominent

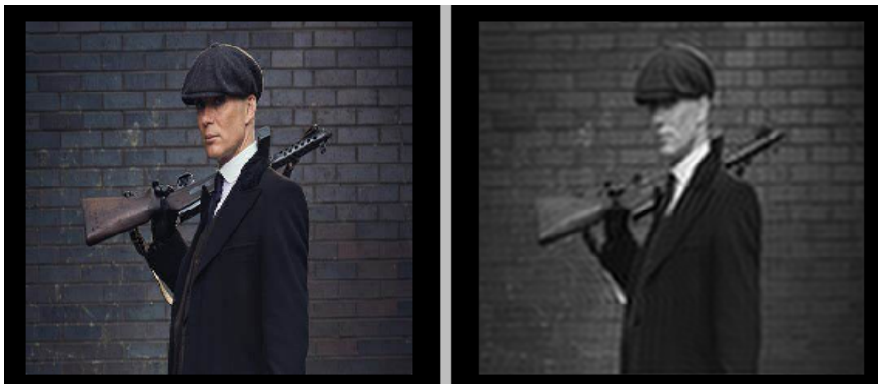


Image showing the effect of applying a cutoff frequency of 30 on the original image.

E. High-Pass Filter:

Conversely, a high-pass filter highlights high-frequency components, such as edges and fine details, while suppressing low-frequency components.

Effect of Cutoff Frequency:

Decreasing the cutoff frequency diminishes the prominence of edges and fine details, leading to a smoother image.

Image showing the effect of applying a cutoff frequency of 9 on the original image.

Increasing the cutoff frequency accentuates edges and enhances image details, albeit at the risk of amplifying noise.

Image showing the effect of applying a cutoff frequency of 60 on the original image.

3. **Edge Detection:**

Edge detection is a fundamental process in image processing that aims to identify boundaries within images. It plays a crucial role in various applications such as object detection, image segmentation, and feature extraction. Here, I'll provide an overview of the Sobel, Prewitt, Canny, and Roberts edge detection methods, outlining their principles, strengths, and weaknesses.

1. **Sobel Operator:**

- The Sobel operator is a popular edge detection technique that emphasizes edges based on the gradient magnitude.
- It applies two 3x3 convolution kernels (one for detecting horizontal changes and one for vertical changes) to the image.
- The gradient magnitude and direction are computed using these kernels, highlighting edges where the magnitude exceeds a specified threshold.
- Strengths:
    - Simple and computationally efficient.
    - Effective at detecting edges even in noisy images.
- Weaknesses:
    - Sensitive to variations in lighting and contrast.
    - May produce thick edges.

Image after applying Sobel operator for edge detection.

2. **Prewitt Operator:**

- Similar to the Sobel operator, the Prewitt operator also uses convolution kernels to compute gradients for edge detection.
- It consists of two 3x3 convolution kernels for detecting horizontal and vertical edges.
- The gradient magnitude is computed using these kernels, and edges are identified based on predefined thresholds.
- Strengths and weaknesses are similar to the Sobel operator.

Image after applying Prewitt operator for edge detection.

3. **Canny Edge Detector:**

- The Canny edge detector is a multi-stage algorithm that is widely used due to its robustness and accuracy.
- It involves several steps: smoothing the image with a Gaussian filter to reduce noise, computing gradients using Sobel operators, performing non-maximum suppression to thin edges, and applying hysteresis thresholding to detect strong and weak edges.

- Canny edge detection provides thin, well-defined edges and suppresses noise effectively.
- Strengths:
    - High accuracy in detecting edges.
    - Low susceptibility to noise.
    - Produces thin edges.
- Weaknesses:
    - Computationally more expensive compared to simpler methods like Sobel and Prewitt.
    - Requires tuning of parameters.

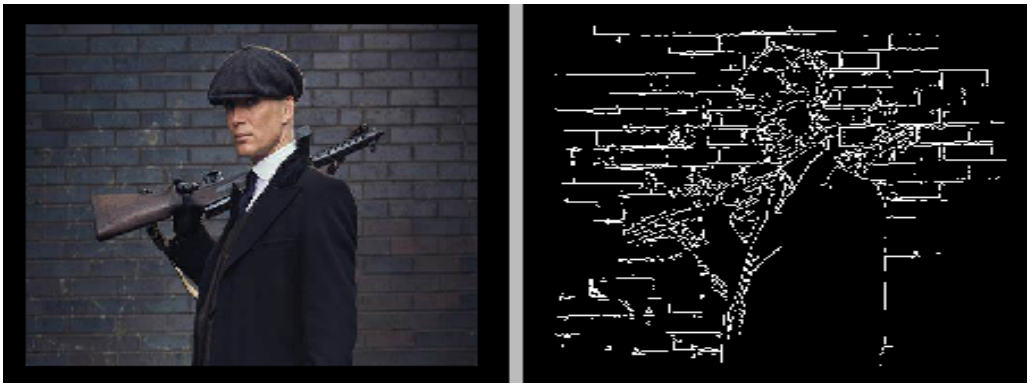


Image after applying Canny operator for edge detection.

4. Roberts Cross Operator:

- The Roberts operator is one of the simplest edge detection methods, based on computing the gradient using a pair of 2x2 convolution kernels.
- These kernels detect diagonal edges by computing the difference in pixel intensity between neighboring pixels.

- The gradient magnitude is obtained by combining the squared responses from both kernels.
- Strengths:
    - Simple and computationally efficient.
    - Effective for detecting diagonal edges.
- Weaknesses:
    - Sensitive to noise.
    - May miss edges that are not aligned with the diagonal.

Image after applying Roberts edge detector.

4. Thresholding:

Thresholding is a fundamental technique in image processing used to segment images into regions based on intensity levels.

A) Local Thresholding:

Local thresholding, also known as adaptive thresholding, adjusts the threshold value for each pixel based on its local neighborhood. This method is particularly useful for images with varying illumination levels or contrast.

Effect of Block Size and Effect of C-Value:

A smaller block size results in finer local adjustments to the threshold, which may capture subtle intensity variations. Adjusting the C-value allows for fine-tuning the thresholding process, especially in regions with uneven illumination or significant intensity variations.

Image showing the effect of applying a block size of 7x7 , C-value of 5 on the original image.

Conversely, larger block sizes provide a more generalized thresholding approach, suitable for smoothing out noise and handling gradual illumination changes**.**

Image showing the effect of applying a block size of 49x49 , C-value of 5 on the original image.

B) Global Thresholding:

Global thresholding sets a single threshold value for the entire image. While simpler than local thresholding, it is effective for images with uniform illumination and contrast.

Effect of Threshold Value:

Lowering the threshold value classifies more pixels as foreground, resulting in a larger segmented region. This is suitable for images with darker foreground objects or lower contrast.

Image showing the effect of applying a threshold value of 15 on the original image.

Conversely, increasing the threshold value reduces the foreground region, useful for images with lighter foreground objects or higher contrast.

Image showing the effect of applying a threshold value of 65 on the original image.

5. Equalization - Normalization:

1. **Equalization:** Image Equalization is a histogram-based technique that aims to adjust the contrast of an image by redistributing pixel intensities.

Equalization Process:

- Calculate the cumulative distribution function (CDF) of the image histogram.
- Normalize the CDF to stretch the intensity values over the entire range (typically 0 to 255 for 8-bit images).
- Use the normalized CDF to transform the pixel values of the image.

Image showing the effect of applying equalization on the original image.

2. **Normalization:** Image Normalization is the process of adjusting the pixel values of an image to conform to a predefined scale or range.

Normalization Process:

- Calculate the minimum and maximum pixel values (min_val & max_val) in the image.
- Map the pixel values to the desired range (e.g., 0 to 1 or 0 to 255) using a linear transformation: normalized_value = (pixel_value - min_val) / (max_val -min_val)

Image showing the effect of applying normalization on the original image.

6. Hybrid Images:

Hybrid images are composite images created by combining a low-pass filtered version (blurred) of one image with a high-pass filtered version (sharpened) of another image.

Image showing the effect of applying a low-pass filter on the dog's image and a high-pass on Murphy's image

Image showing the effect of applying a low-pass filter on Murphy's image and a high-pass on the dog's image

7. Histograms and CDFs of Gray-scaled Images:

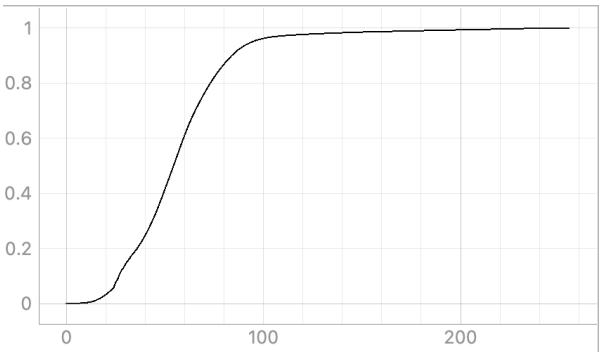
Plotting the Cumulative Distribution Function (CDF) and Histogram of an image holds significant importance, offering invaluable insights into the distribution of pixel intensities and aiding in various analysis and enhancement tasks.

Below is a demonstration of the CDF and Histogram plotting for gray scale images:

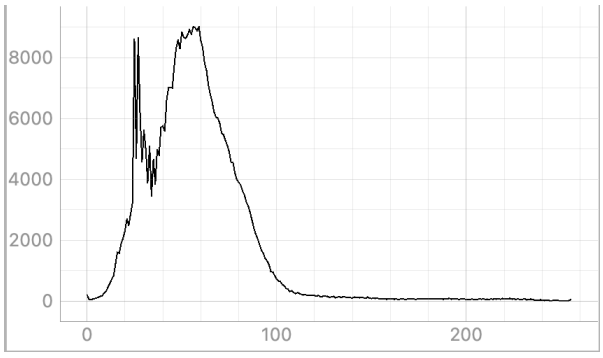
Original Image

Gray Scale

CDF

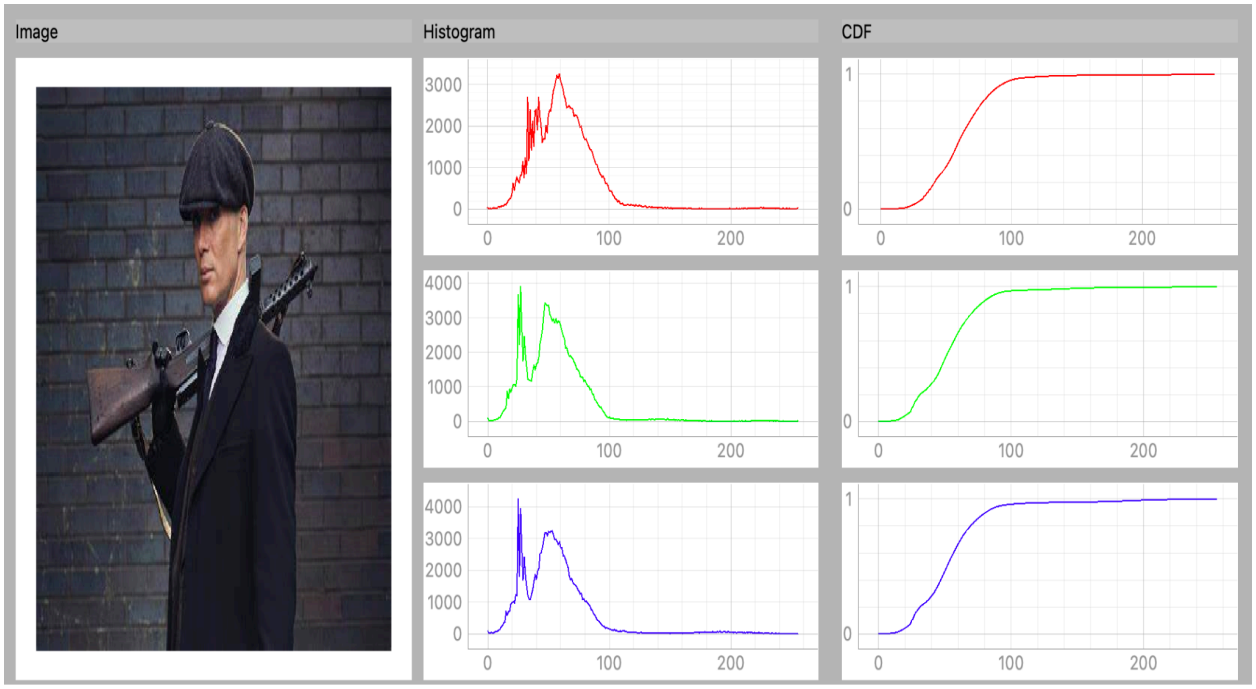


Histogram



8. Histograms and CDFs of RGB Images:

Here you will find that we have the original image alongside 6 plotting channels (2 plot for each image data channel). That means we have an independent histogram and CDF for Red channel, independent histogram and CDF for Green channel, and independent histogram and CDF for Blue channel.



A. Software Design and Architecture

The application is built upon a robust software design architecture that leverages the Python programming language, PyQt framework, Qt Designer for UI design, and follows the Model-View-Controller (MVC) pattern for efficient organization and separation of concerns.

1. Model-View-Controller (MVC) Pattern

- **Model**: The Model component represents the data and business logic of the application. In our context, it encapsulates image processing algorithms, data structures for image representation, and manipulation functions.

  The model in our case is represented in Image processing algorithms, data structures, and functions reside within the Model component, ensuring encapsulation and modularity.

- **View**: The View component is responsible for presenting the user interface to the user. Qt Designer is utilized to design the graphical user interface (GUI) elements, including buttons, sliders, image displays, and histograms.

  The view in our case encompasses the graphical user interface designed using Qt Designer. It includes various widgets for user interaction and image display.

- **Controller**: The Controller acts as an intermediary between the Model and the View, handling user input and triggering appropriate actions. It binds GUI events to corresponding image processing functions, ensuring seamless interaction between the user and the application.

  The controller in our case bridges the gap between the Model and the View, handling user events and orchestrating image processing tasks based on user input.

2. Used Technologies and Tools

- **Python**: Python serves as the primary programming language for our application, offering simplicity, readability, and extensive libraries for image processing tasks.
- **PyQt**: PyQt is a set of Python bindings for the Qt application framework. It enables the integration of Qt functionalities within Python applications, providing access to a vast array of GUI components and features for building cross-platform desktop applications.
- **Qt Designer**: Qt Designer is utilized for designing the graphical user interface of our application. It offers a drag-and-drop interface for creating and arranging UI elements, streamlining the GUI design process. Designed UI files (.ui) created in Qt Designer are seamlessly integrated into the Python codebase using PyQt, allowing for rapid development and modification of the user interface.

B. <u>Goals</u>

- **Modularity**: The MVC pattern facilitates modular design, enabling independent development and testing of Model, View, and Controller components.
- **Separation of Concerns**: MVC separates the presentation logic (View) from the application logic (Model), enhancing maintainability and scalability.
- **Ease of Development**: Python, PyQt, and Qt Designer collectively provide a conducive environment for rapid application development, allowing us, as developers, to focus on functionality rather than low-level details of GUI implementation.