HW1 Computer Vision Ivan Hinz

The code is structured in 5 functions corresponding to each part of the homework.

Firstly, func1 is called.

Func1:

Reads an image from the given file path.

Displays the image in the window.

Wait for 5 seconds before closing the window.

Original image (portrait)

Portrait photo (one of the presidents of USA) is chosen. Portraits are typically well-lit and have a clear subject, making them suitable for testing preprocessing techniques like grayscale conversion, histogram equalization, and edge detection.

Secondly, func2 is called.

Func2:

Reads an image from the given path (path to original image is given).

Converts the RGB image to Grayscale format.

Converts the RGB image to HSV format.

Save both Grayscale and HSV images.

Displays the Grayscale and HSV images for 5 seconds.

Compute brightness histograms for:

The original image.

The Grayscale image.

Plot histograms for comparison.

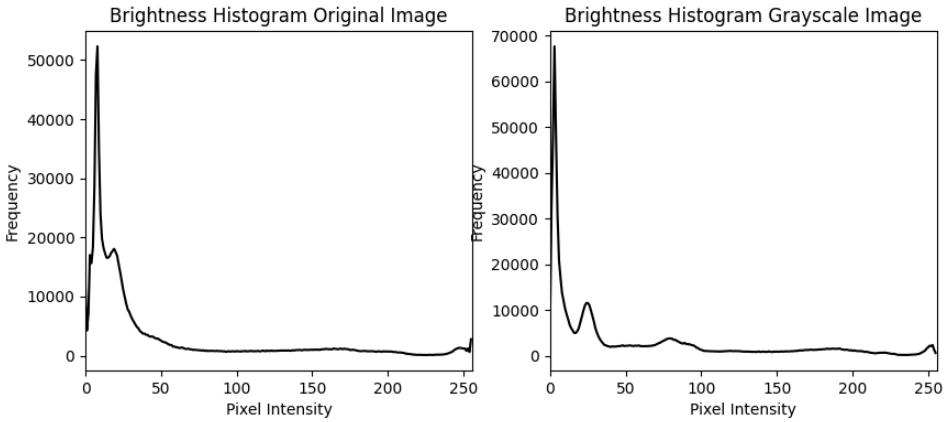
Results:

Grayscale image

HSV image

HSV image - variations in hue create unnatural colors, especially in skin tones and background elements. In some areas, the saturation is high, which is why colors appear exaggerated. Shadows or highlights in the original image appear as bright or dark spots in the HSV image.

Grayscale image - use only one channel instead of three (RGB), meaning colors are represented solely by their intensity. Darker areas (like a suit) become nearly black.
Bright regions (like the forehead or highlights) become white.



Analysis of the brightness histogram of the **original image** - there is a **sharp peak near the lowest pixel intensity** (close to 0), which suggests that the image contains a large number of dark pixels (shadows or dark background), and overall for this portrait photo it is true, because of the hair, background, costume.

After this initial spike, the frequency drops sharply and shows smaller peaks around low to mid intensities, indicating that the image contains a moderate amount of mid-tone details. The right side of the histogram (high pixel intensity) is relatively flat, indicating that there are fewer bright regions in the image.

Analysis of the brightness histogram of the **Grayscale image** - the overall shape of the histogram is similar to the original, but the peak at low intensity is even more pronounced. The grayscale conversion seems to have increased the concentration of dark pixels while slightly reducing mid-tone details.

After that, func3 is called.

Func3:

Reads an image from the given path (path to Grayscale image is given).

Defines 2 different kernel sizes - one is 7 and one is 3(affects the filtering strength) .

Applies 3 different filters:

Median filter: Removes noise by replacing pixels with the median of their neighbours.

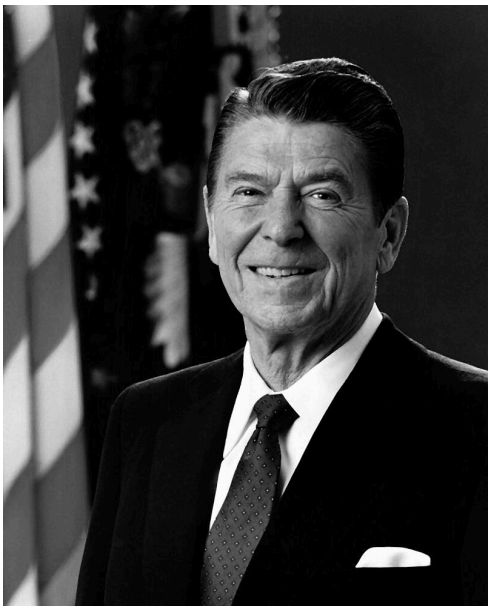
Gaussian blur: Smooths the image using a Gaussian-weighted kernel (kernel size = 7 is used). Different sigma's are used - one is normal = 5, another one is small = 0,001.

Laplacian filter: Enhances edges by detecting intensity changes.

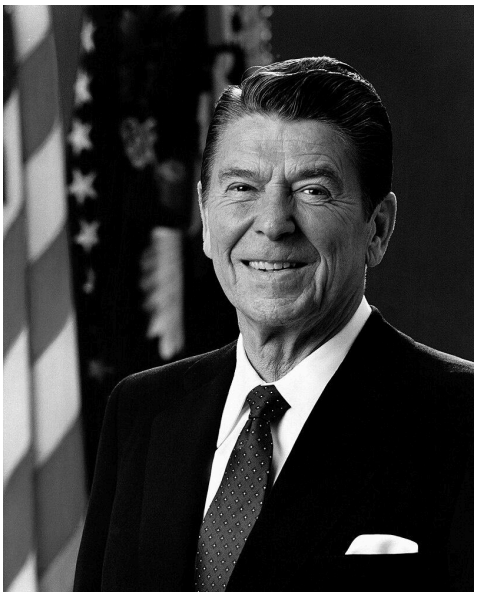
Displays the processed images after filters for 5 seconds.

Results:

Grayscale after median filter, kernel = 7

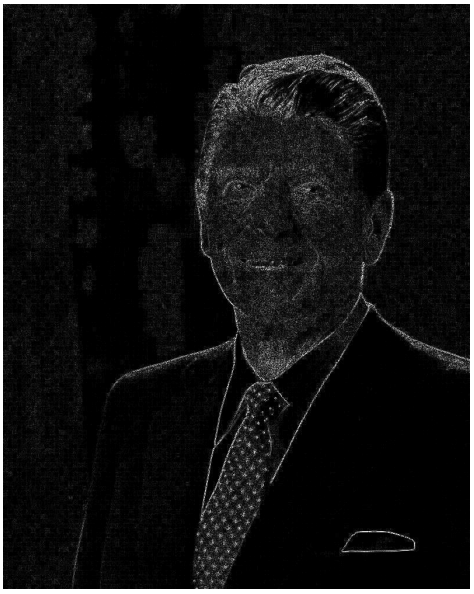


Grayscale after median filter, kernel = 3



Gaussian smoothing, sigma = 0,001

Gaussian smoothing, sigma = 5



Laplacian filter

Grayscale after **Gaussian smoothing** - while the original grayscale image has sharp edges, meaning transitions between different intensity levels (e.g., between the suit and the background or facial features) are clear, the Gaussian-smoothed image smooths these edges, reducing high-frequency components and making contours appear less distinct.

Gaussian smoothing (sigma = 5):

- The high value of sigma causes significant blurring.
- The details of the face, clothing, and background are heavily smoothed out, creating a soft and hazy look.
- Edges are softened, and fine details (like hair, wrinkles, and texture) are lost.
- Large σ leads to more weight given to neighboring pixels, which results in blending over a larger area.

Gaussian smoothing (sigma = 0.001):

- A very small sigma means that the output image is essentially unchanged since the effect of the smoothing is negligible.
- The details (like the texture of the tie and facial features) remain sharp and crisp.

Grayscale after **median filter** - the median filter works by replacing each pixel's value with the **median** value of the pixel intensities within the kernel window. The median filter is highly effective in removing such noise while preserving edges, unlike Gaussian smoothing, which introduces blurring. The median-filtered image smooths out small variations, which can remove unwanted noise but may also reduce small details (skin texture or fabric patterns).

For Kernel Size = 3:

- A **small kernel size** (3x3) means that the filter considers only the closest neighboring pixels.
- Since the window is small, the median calculation only removes small-scale noise while preserving sharp edges and fine details.
- The image of the president remains **sharp** because the localized nature of the filtering prevents over-smoothing.

For Kernel Size = 7:

- A **larger kernel size** (7x7) means that the filter considers more neighboring pixels, increasing the smoothing effect.
- The larger window allows for better noise reduction but causes loss of detail and blurring of edges.
- Fine textures and sharp features are averaged out, making the image appear **smoother and less detailed**.

Grayscale after **Laplacian filter** - the Laplacian-filtered image highlights only areas where intensity changes rapidly (edges), turning most of the image black with only white outlines where edges are detected. Makes large uniform regions (e.g., suit, background) appear black while emphasizing only edges.

After that, func4 is called.

Func4:

Reads the image.

Applies Sobel filters to detect:

Horizontal edges (sobel_x)

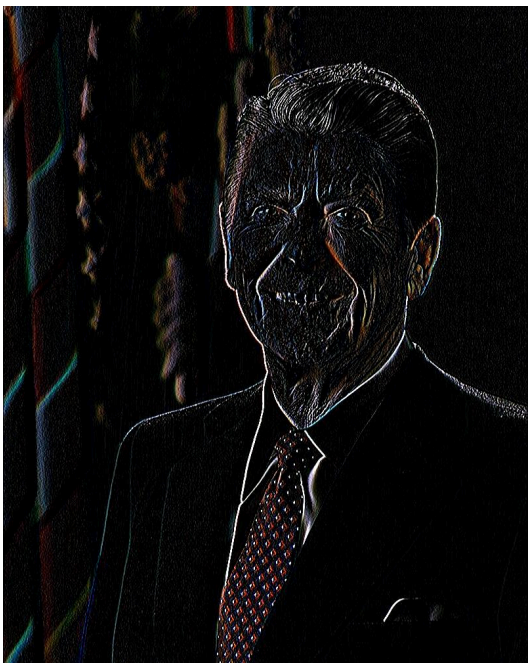
Vertical edges (sobel_y)

Applies Canny edge detection to highlight strong edges.

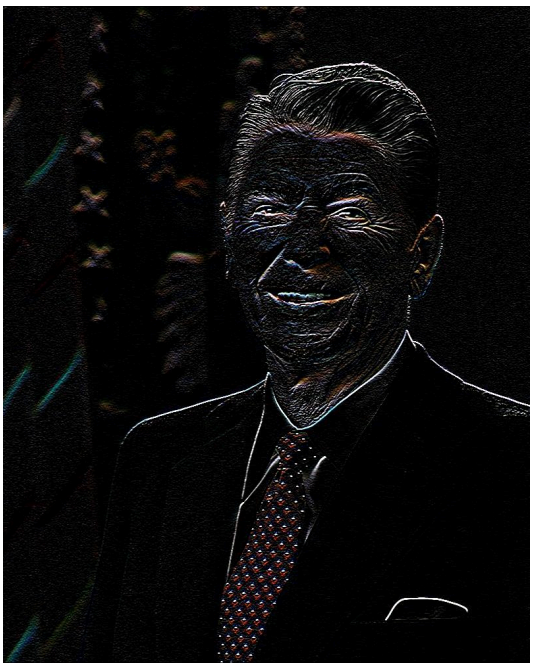
Use Harris Corner Detection algorithm to find corners in the Grayscale image.

Display the processed images for 5 seconds.

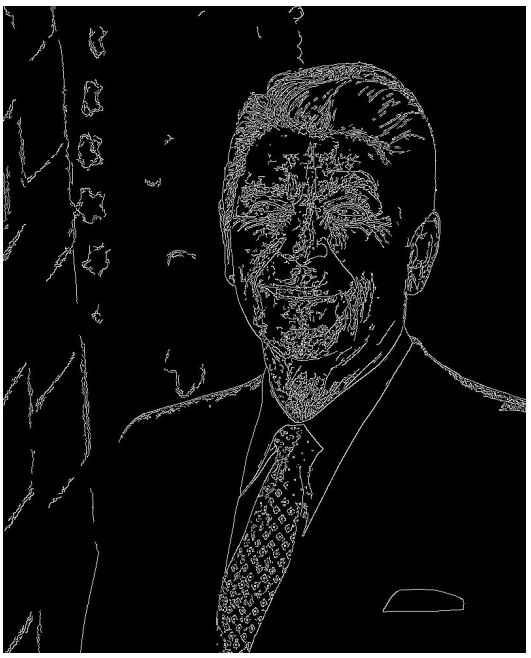
Results:



Horizontal edges (after Sobel operator)



Vertical edges (after Sobel operator)



After Canny edge detection algorithm

After Harris corner detection algorithm

The Sobel-x filter detects only horizontal intensity changes, meaning that horizontal edges (e.g., boundaries between light and dark areas aligned horizontally) appear bright (white). Vertical edges and uniform regions become dark (black or near zero).

The Sobel-y filter detects only vertical intensity changes, meaning that vertical edges (e.g., boundaries between light and dark areas aligned vertically) appear bright (white). Horizontal edges and uniform regions become dark (black or near zero). As a result of Sobel-x and Sobel-y strong edges are visible around the face, suit, and background details.

Unlike Laplacian or Sobel filters, which highlight many edges, Canny edge algorithm refines edge detection by removing weak or irrelevant edges. Result: clean, sharp lines around facial features and clothing.

Harris measures changes in intensity in **both x and y directions** to find corners (high gradient changes).

Corners are typically detected at:

- **Eyes** (contrast between whites and pupils)
- **Junctions** between suit and shirt
- **Hairline and ears**

And we see the most corners at eyes, near ears and clothing (especially the tie).

Lastly, func5 is called.

Func5:

Reads an image from the given path (path to Grayscale image is given).

Apply binary thresholding:

Pixels above 200 are set to 255 (white).

Pixels below 200 are set to 0 (black).

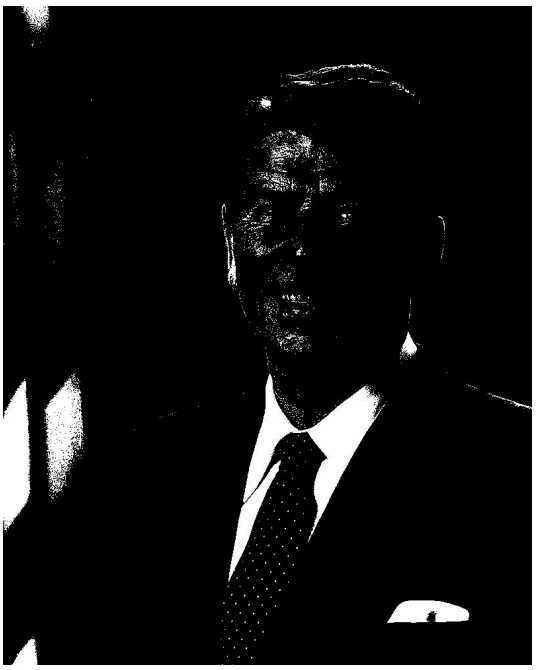
Define 3x3 kernel.

Applies Erosion operation.

Applies Dilation operation.

Display the processed images for 5 seconds.

Results:



Binarized grayscale image

Binarized grayscale image after erosion

Binarized image after dilation

Binarized image - converts the grayscale image to binary based on a threshold of **200**:

- Pixels > 200 -> White
- Pixels ≤ 200 -> Black

Since 200 is a high value, only the brightest areas (like highlights on the face) turn white, which leads to almost everything becoming black.

Binarized grayscale image after erosion - erosion works by removing white pixels at the edges of white regions, causing the white areas to shrink. After **10 iterations**, erosion aggressively reduces the size of white areas, and small white regions disappear completely. This explains why the eroded image looks almost completely black — most small bright details were eroded away. As we can see, from white regions only the shirt collar remained.

Dilation - works by adding white pixels around the edges of white regions, causing them to grow. After **5 iterations**, the white regions become thicker and more prominent.
The shape of the white regions is preserved but expanded, making the white areas appear larger compared to the original binary image. As we can also see, the checkerboard on the tie becomes very large.