Assignment 3

Syed Zaidi

February 2023

1 Part 1: Filtering

1.1 Gaussian Kernel

This implementation creates a $3sigmax3sigma$ 2D Gaussian kernel using np.meshgrid to create a grid of coordinates and np.exp to compute the Gaussian values. It then normalizes the kernel to ensure that its values sum to 1 using the $h/=h.sum()$ expression.

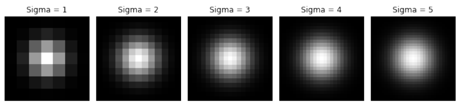

Figure 1: GaussKernal with different values for Sigma

1.2 Convolution

The function takes as input the grayscale image I and the convolution filter h. It first computes the dimensions of the input image and the filter. Then it calculates the amount of padding needed on each side of the input image to handle boundary cases. It adds zero-padding to the input image using np.pad. It initializes an output image of the same size as the input image.

The function then loops over all the pixels in the input image and extracts the patch of the input image that the filter overlaps. It computes the element-wise product of the patch and the filter and sums the products to get the value of the output pixel at that location. Finally, it returns the output image.

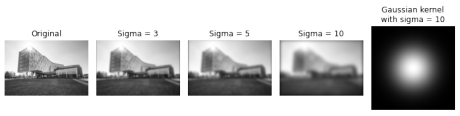

Figure 2: filtering “Iribe.jpg” image using Gaussian kernels with standard deviations 3, 5, and 10.

1.3 Filters

The filter $h1$ is a simple averaging filter, which has the effect of smoothing the image and reducing the amount of high-frequency noise. The resulting image appears more blurry and less detailed.

The filters $h2$ and $h3$ are both gradient filters that can be used to detect edges in an image. The filter $h2$ detects horizontal edges, while the filter $h3$ detects vertical edges. The resulting images show the edges in the image in the horizontal and vertical directions, respectively. The edges are represented by bright or dark pixels, depending on the direction of the edge and the orientation of the filter.

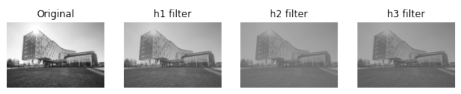

Figure 3: Applying the provided filter (H1, H2, and H3) to the original image

2 Part2: Edge Detection Canny Filter

The Canny filter is a popular method for detecting edges in images. It involves several steps, as described below.

2.1 Step 1: Apply a Gaussian filter to the image with standard deviation σ

The first step in the Canny filter is to apply a Gaussian filter to the input image to smooth out noise. The degree of smoothing is controlled by the standard deviation σ of the Gaussian. This step is performed using the `myImageFilter` function, which applies a filter to an input image. The Gaussian kernel used in the filtering is obtained using the `gausskernel` function. The resulting smoothed image is referred to as $I_{smoothed}$.

2.2 Step 2: Compute the gradient magnitude and angle

The next step is to compute the gradient magnitude and angle of the smoothed image. This is done using two Sobel filters: one for computing the horizontal gradient (S_x), and another for computing the vertical gradient (S_y). The gradient magnitude is then calculated as the Euclidean norm of the two gradients: $|\nabla I| = \sqrt{I_x^2 + I_y^2}$. The gradient angle is computed as the arctan of I_y over I_x. To simplify the computation, the angle is quantized to four possible values: 0, 45, 90, or 135 degrees, according to which quadrant it falls in. The resulting images are referred to as I_x, I_y, $|\nabla I|$, and θ.

2.3 Step 3: Edge thinning / Non-maximum suppression

The next step is to thin the edges to a single pixel width. This is done by checking the gradient magnitude at each pixel and comparing it to the magnitudes of the neighboring pixels along the gradient direction. If the magnitude at the current pixel is larger than both of its neighbors, the pixel is considered a local maximum and retained as an edge pixel. All other pixels are set to zero. The resulting image is referred to as I_{nms}.

2.4 Step 4: Hysteresis thresholding

The final step is to apply hysteresis thresholding to the thinned edge image to obtain the final edge map. This involves setting two threshold values: a low threshold t_{low} and a high threshold t_{high}. Pixels with gradient magnitude below t_{low} are considered non-edges and are discarded. Pixels with gradient magnitude above t_{high} are considered strong edges and are retained. Pixels with gradient magnitude between t_{low} and t_{high} are considered weak edges. These pixels are retained only if they are connected to a strong edge pixel. This is done by labeling the connected components in the strong edges image, and checking whether any weak edges are connected to each component. The resulting image is the final edge map.

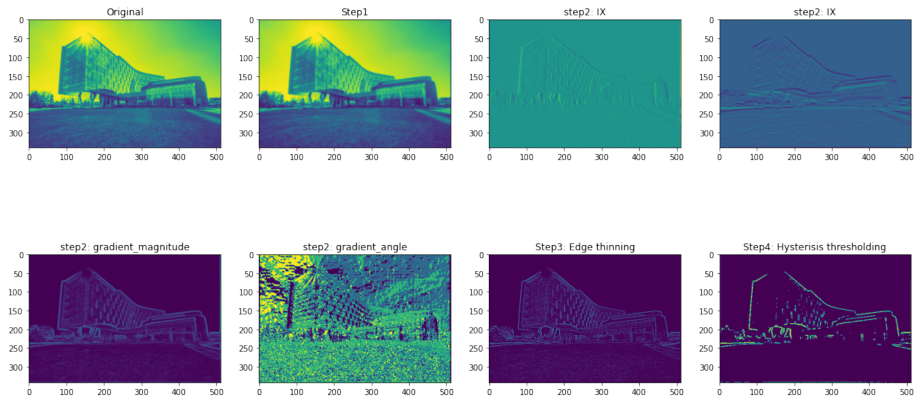

Figure 4: Steps of the Canny Filter.

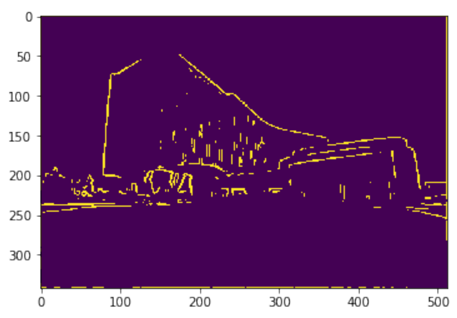

Figure 5: Final result after applying the canny filter.

Overall, the Canny filter is a powerful method for detecting edges in images, and is widely used in computer vision and image processing applications.

3 Testing

We will try these different values for testing:

- $sigmas$ = [0.25, 0.5 ,1]
- t_{low} = [0.2, 0.4, 0.6]
- t_{high} = [0.6, 0.8, 1]

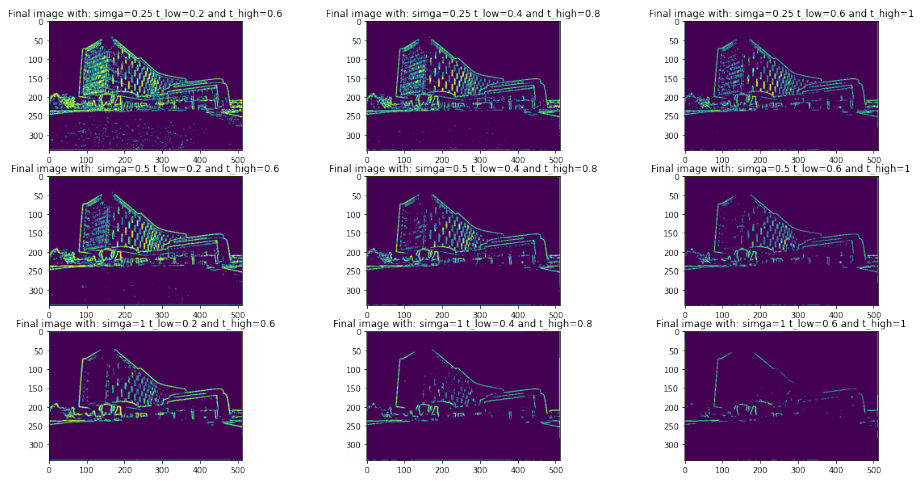
Final image with: simga=0.25 t_low=0.2 and t_high=0.6
Final image with: simga=0.25 t_low=0.4 and t_high=0.8
Final image with: simga=0.25 t_low=0.6 and t_high=1
Final image with: simga=0.5 t_low=0.2 and t_high=0.6
Final image with: simga=0.5 t_low=0.4 and t_high=0.8
Final image with: simga=0.5 t_low=0.6 and t_high=1
Final image with: simga=1 t_low=0.2 and t_high=0.6
Final image with: simga=1 t_low=0.4 and t_high=0.8
Final image with: simga=1 t_low=0.6 and t_high=1

Figure 6: Trying different values of $Sigma, t_{high}, andt_{low}$.

we can notice that:

- Increasing the value of sigma will result in a smoother image, which can help to reduce the effect of noise and produce more accurate edges. However, a large value of sigma may cause the edges to be blurred, and thus may miss some of the edges in the image.
- We can notice that increasing the values of t_{low} and t_{high} will result in fewer edges detected, while decreasing the values of t_{low} and t_{high} will result in more edges detected. Setting t_{low} too low may result in detecting noise as edges, while setting it too high may miss important edges. Setting t_{high} too low may cause the algorithm to miss some of the real edges, while setting it too high may cause multiple edge segments to be detected.

4 Extra Credit:

The code combines two input images to create a hybrid image by taking the low frequency components from one image and high frequency components from the other image. The code first reads in two input images and resizes them to have the same dimensions. It then converts the images to numpy arrays, computes the Fourier transforms of both images, shifts the zero frequency component to the center of the spectrum, computes the magnitudes of the Fourier transforms, and combines the frequency components using a weighted sum. The resulting hybrid spectrum is then transformed back into the image domain and normalized to the range [0, 255]. Finally, the resulting hybrid image is displayed

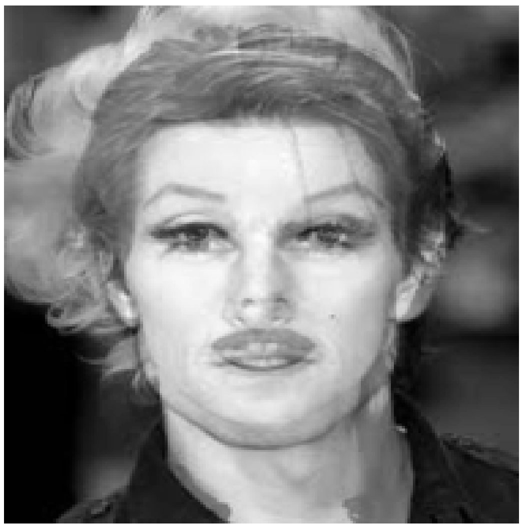

Figure 7: The combined image of Marilyn Monroe(Low frequency) and Tom Cruise(High frequency)