



XDoG: advanced image stylization with eXtended Difference-of-Gaussians

Presented By: Big Dippers

December 4th, 2023

A Traditional Approach to Image Stylization

The Difference-of-Gaussians (DoG) operator can be used to yield aesthetically pleasing edge lines for an image despite not being an edge detector, produced as the difference of two DoG-filtered images with differing parameters. With reparametrization, this can be further used for the implementation of various artistic and stylistic applications.

How to extend DoG?

- The DoG (difference of Gaussian) operator is an efficient way of computing LoG (Laplacian of Gaussian)
- We extend the DoG by including more parameters (τ , ε etc) to achieve a wider range of stylistic effects. This is called Extended DoG (or XDoG).

$$S_{\sigma,k,p}(x) = (1 + p) \cdot G_{\sigma}(x) - p \cdot G_{k\sigma}(x) \text{ (reparameterised)}$$

(k is scaling factor for σ)

(p makes it possible to control the strength of the edge sharpening effect without influencing any other aspects of the filter)

How to extend DoG?

$$T_{\varepsilon, \varphi}(u) = \begin{cases} 1 & u \geq \varepsilon \\ 1 + \tanh(\varphi \cdot (u - \varepsilon)) & \text{otherwise.} \end{cases}$$

The thresholding function used is described by the formula above. It is a continuous ramp function.

Thresholding

Parameters Used

- k -> scaling factor for the standard deviation of the DoG
- σ -> standard deviation of the gaussian filter that is used
- p -> controls strength of edge sharpening effect
- size -> size of gaussian kernel
- ϵ -> thresholding value
- ϕ -> threshold parameter for ramp-based thresholding

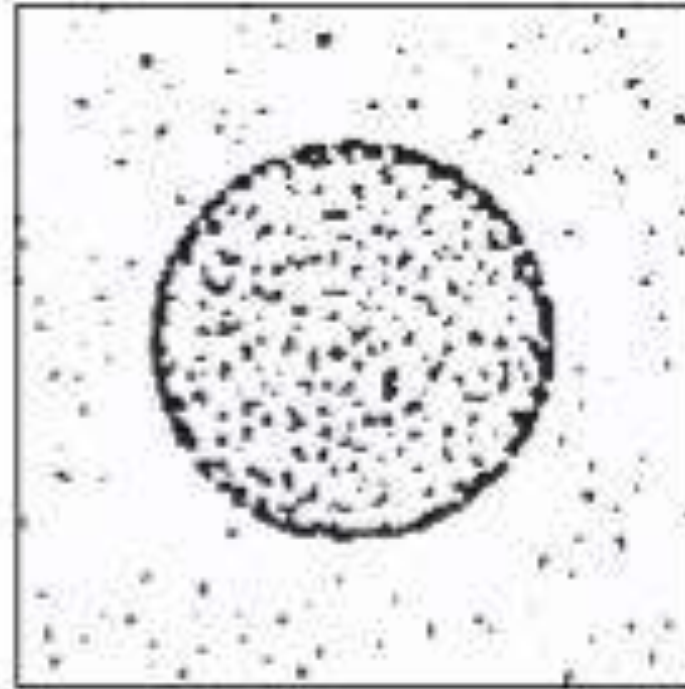
Introduction to FDoG

- Adapt the filter according to the approximated edge orientations. The idea behind these approaches is to first respond to changes in luminance that occur across edges and then to smooth those responses using an edge aligned blur.
- Helpful to remove noise that occurs in some images due to XDoG
- The local edge orientations are obtained using a *smoothed structure tensor (SST)*. The eigenanalysis of this calculated at each point is called the *edge tangent flow (ETF)*

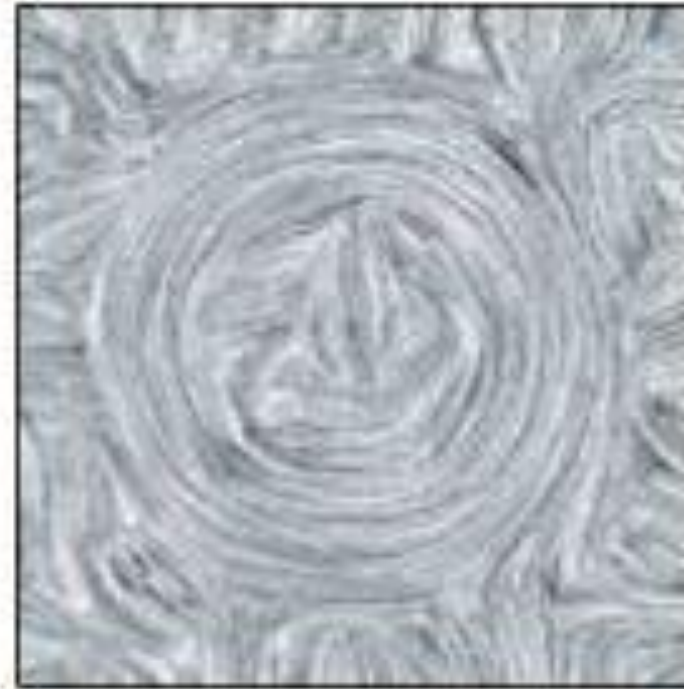
Edge Tangent Flow



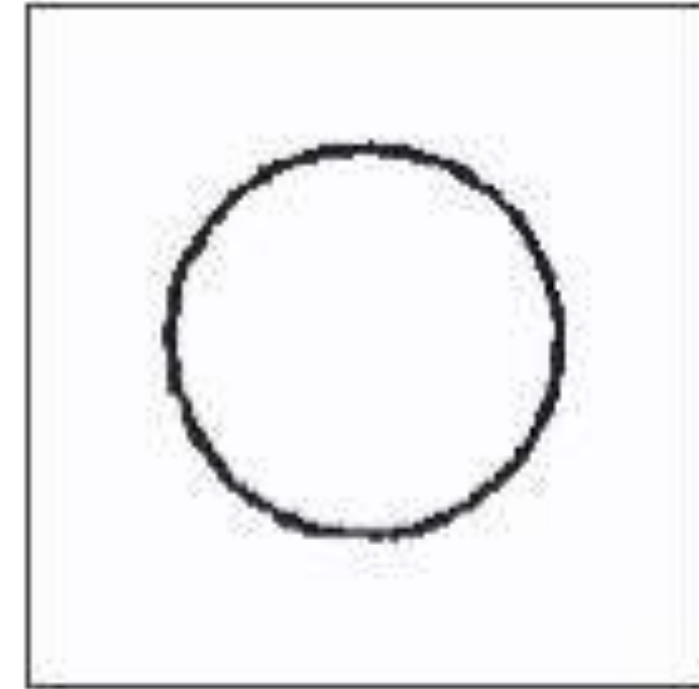
Input



DoG



ETF



FDoG

How FDoG helps reduce noise

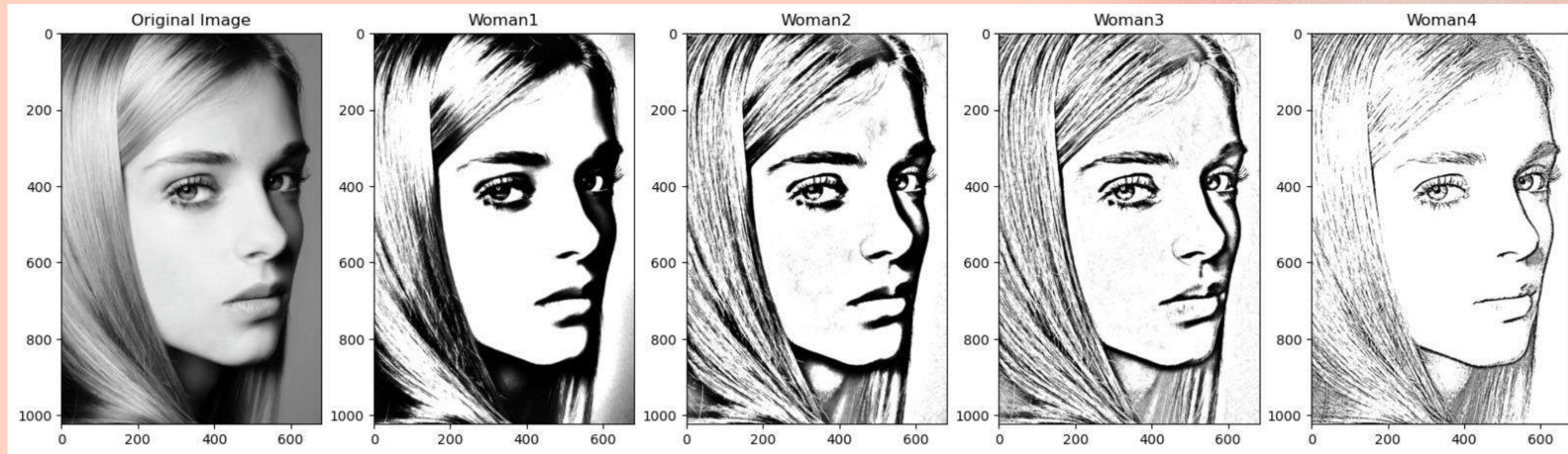
Flow-based DoG (FDoG)

Extending the basic DoG to FDoG is done by reparametrization: replacing the single parameter σ by three separate parameters, which are:

- σ_c : parameter for gaussian blur which is used to blur the structure tensor
- σ_e : parameter for the gradient aligned gaussian blur
- σ_m : parameter for the edge tangent aligned line integral convolution

Implementation

With approach & Results



Base Code

increasing sigma - increasing blurriness (lesser spots)

increasing p - increasing white spots

increasing epsilon - less bright

Source: Add your references here.

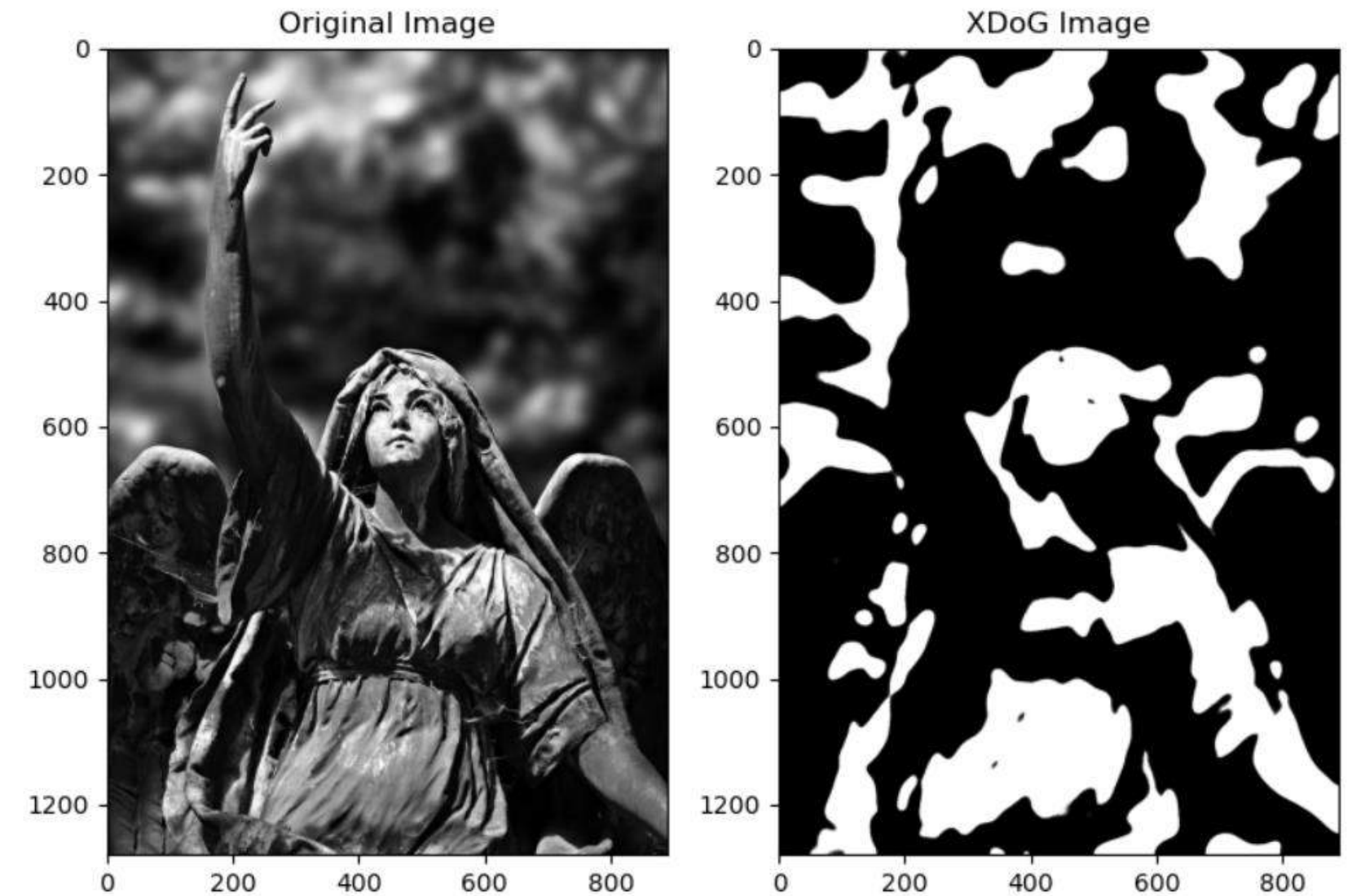
F-DoG

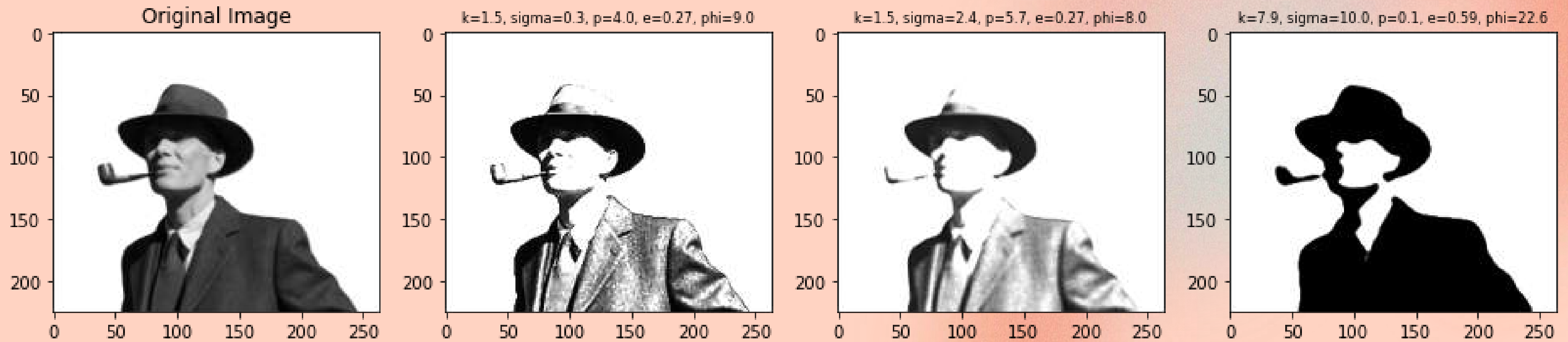
- As mentioned before, FDoG is used to remove the influence on noise in XDoG outputs by blurring only along the edges.
- This is achieved using methods discussed previously (reparametrization of sigma and such)



Abstraction

$$\begin{aligned} S_{\sigma,k,p}(x) &= \frac{D_{\sigma,k,p}(x)}{\tau - 1} = G_{\sigma}(x) + p \cdot D_{\sigma,k}(x) \\ &= (1 + p) \cdot G_{\sigma}(x) - p \cdot G_{k\sigma}(x) \end{aligned}$$



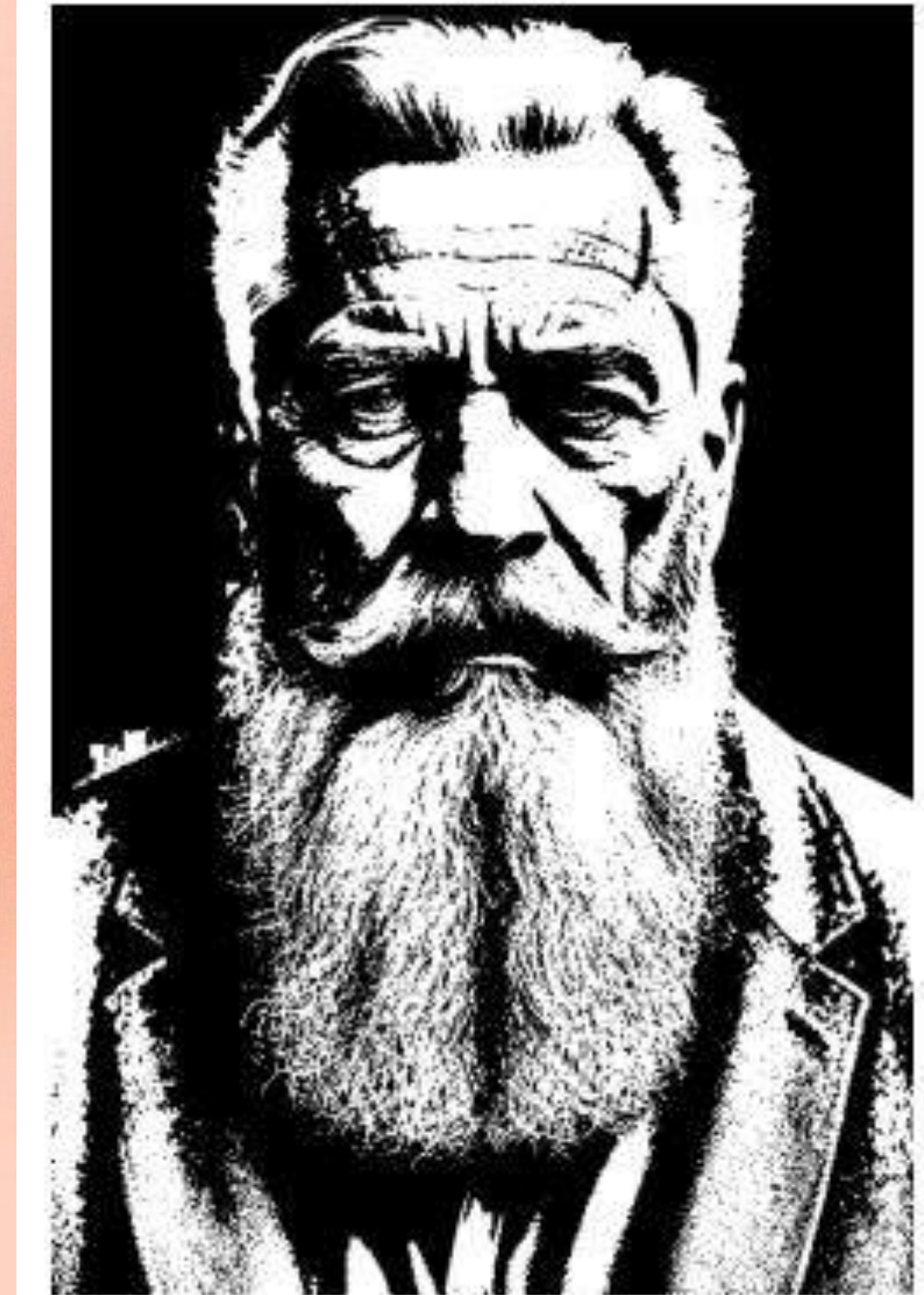


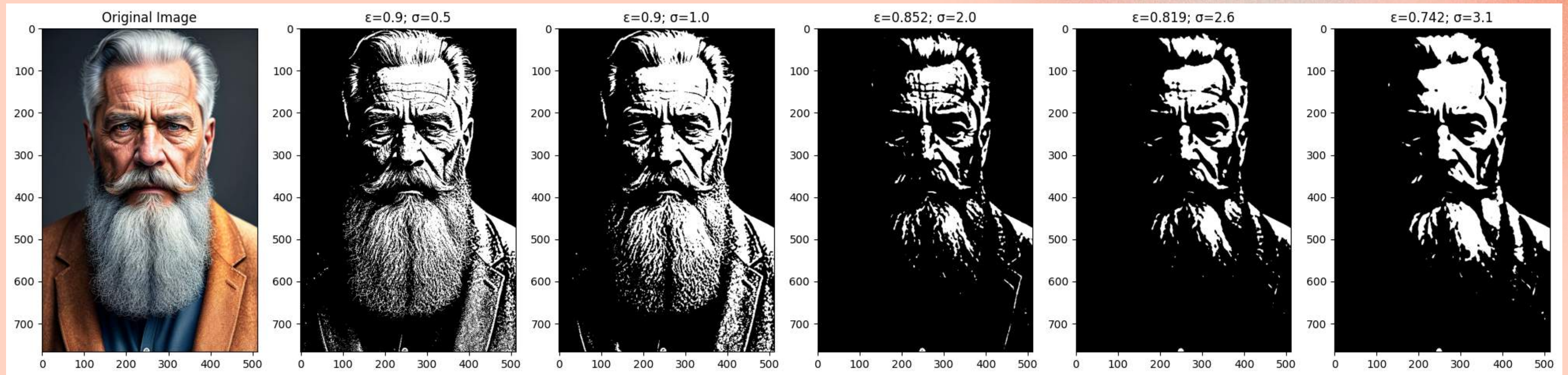
Abstraction

- Abstraction is achieved by varying the sigma parameter
- As sigma increases, the image becomes more blurred, thus removing fine details and resulting abstraction.

Thresholding

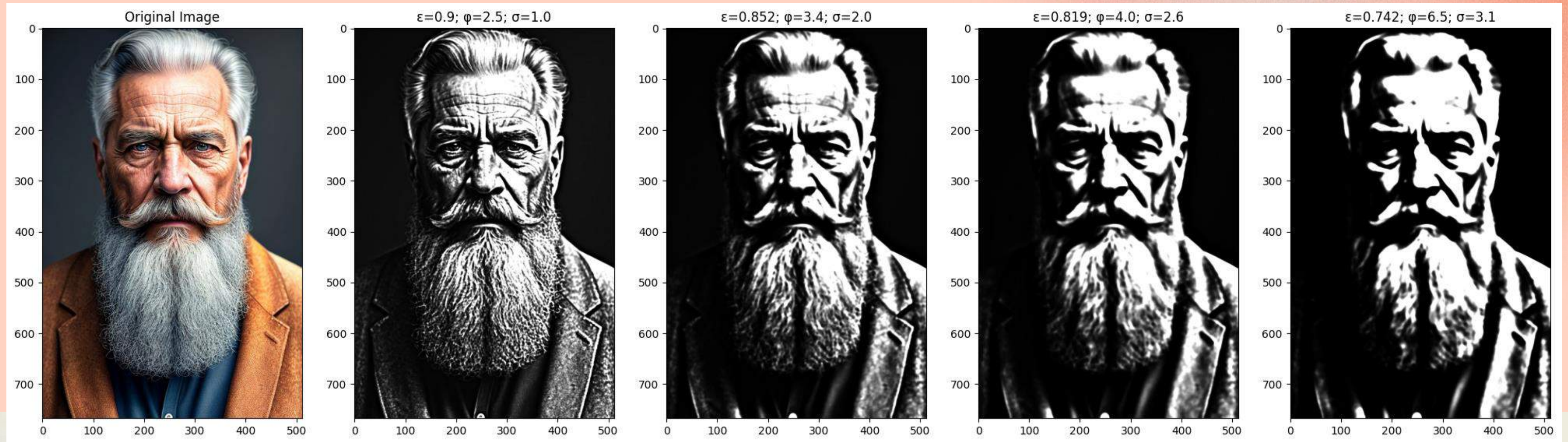
$$\begin{aligned} S_{\sigma,k,p}(x) &= \frac{D_{\sigma,k,p}(x)}{\tau - 1} = G_{\sigma}(x) + p \cdot D_{\sigma,k}(x) \\ &= (1 + p) \cdot G_{\sigma}(x) - p \cdot G_{k\sigma}(x) \end{aligned}$$





2 Tone Thresholding

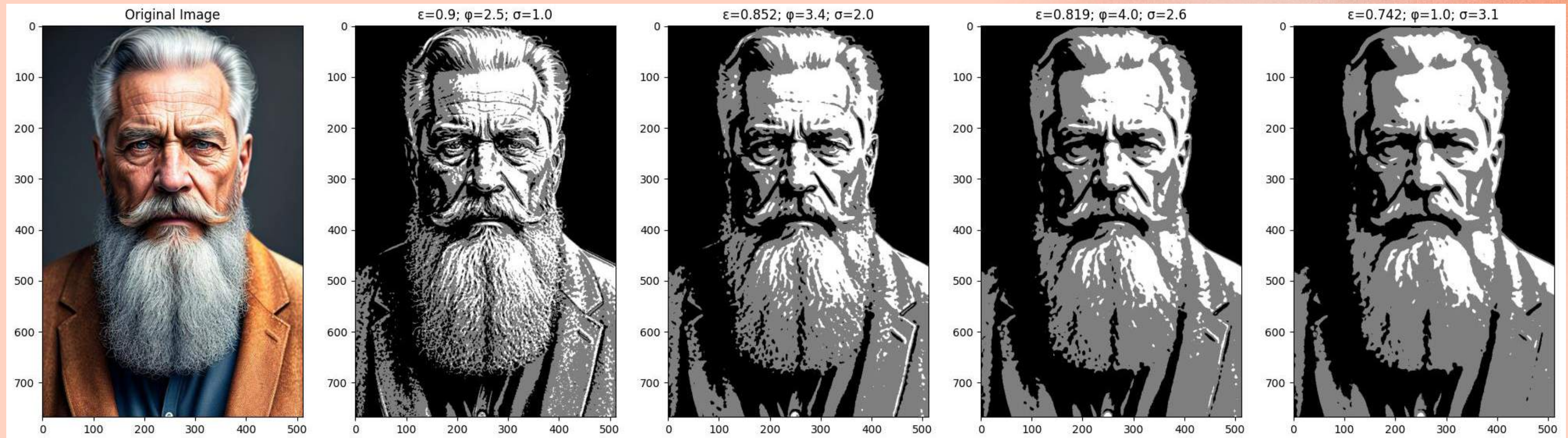
- As $\sigma \uparrow$, the finer details in the image are lost (blurred out)
- ϵ is the threshold value above which certain pixels are set to 0 and 1



Gradient Thresholding

- As $\varphi \uparrow$, the hyperbolic tangent curve starts to become a binary threshold function

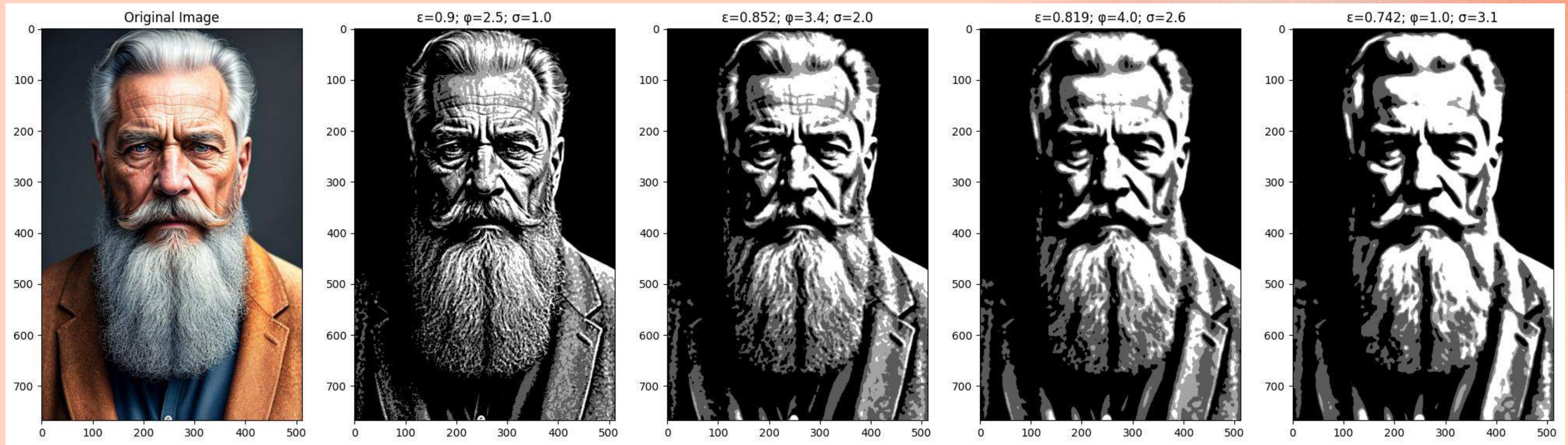
$$T_{\varepsilon, \varphi}(u) = \begin{cases} 1 & u \geq \varepsilon \\ 1 + \tanh(\varphi \cdot (u - \varepsilon)) & \text{otherwise.} \end{cases}$$



3 Tone Thresholding

$$\frac{\lfloor color * (n - 1) + 0.5 \rfloor}{n - 1}$$

- For best 3-tone thresholding for this image, a color quantization function from [1] is used

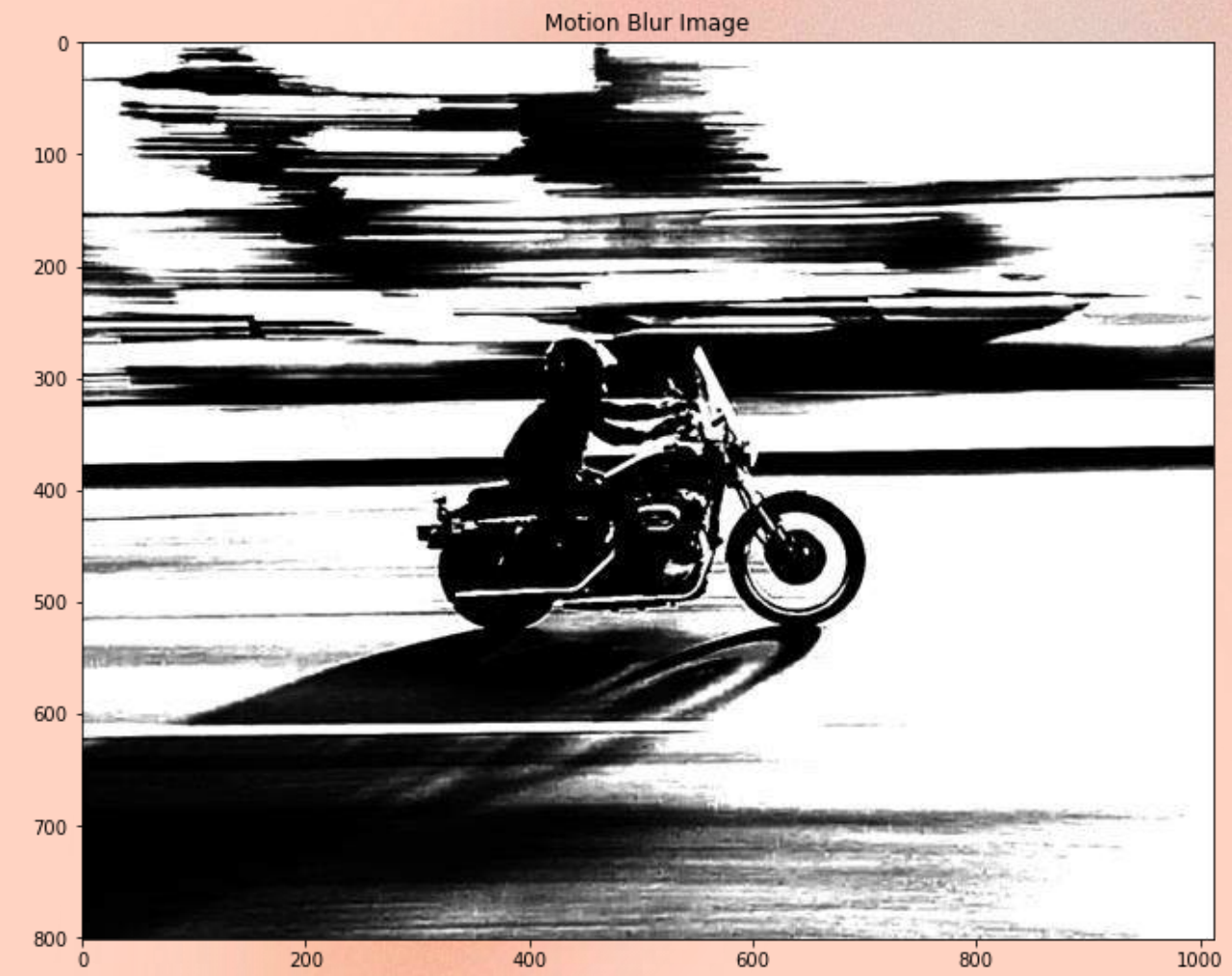


4 Tone Thresholding

Two intermediate values ϵ_1 and ϵ_2 ; 0.6465 and 0.355 are used here

Motion Blur





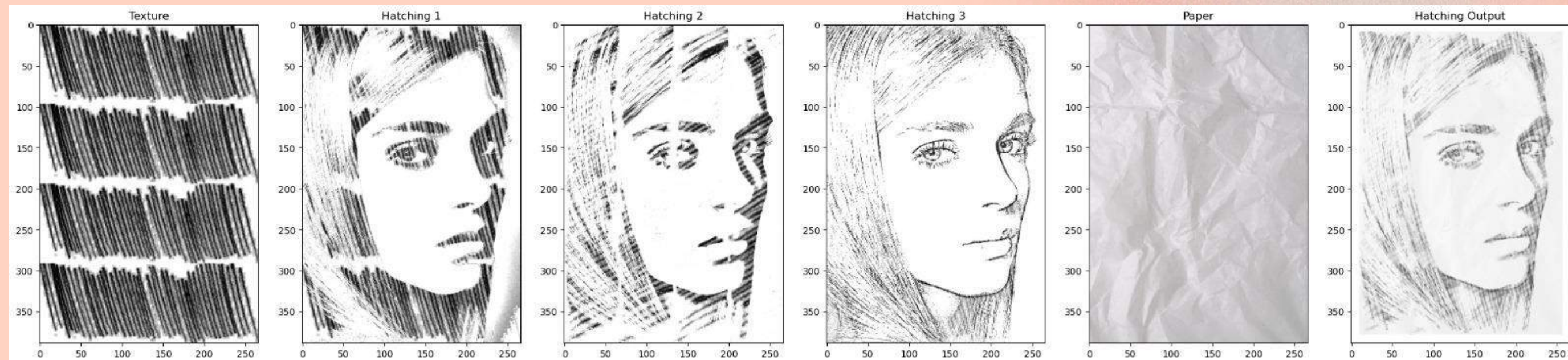
Motion Blur

- It exaggerates the lines and removes the unnecessary low frequency data from the image giving sharp lines (seen as speed lines)

Cross-Hatching

- The hatching is based on the concept of tonal art maps, where layers of strokes add up to achieve a desired tone.
- While simple to implement and efficient to compute, the hatching style retains many of the XDoG features (clean edges, tone control, negative edges) and produces high-quality results.



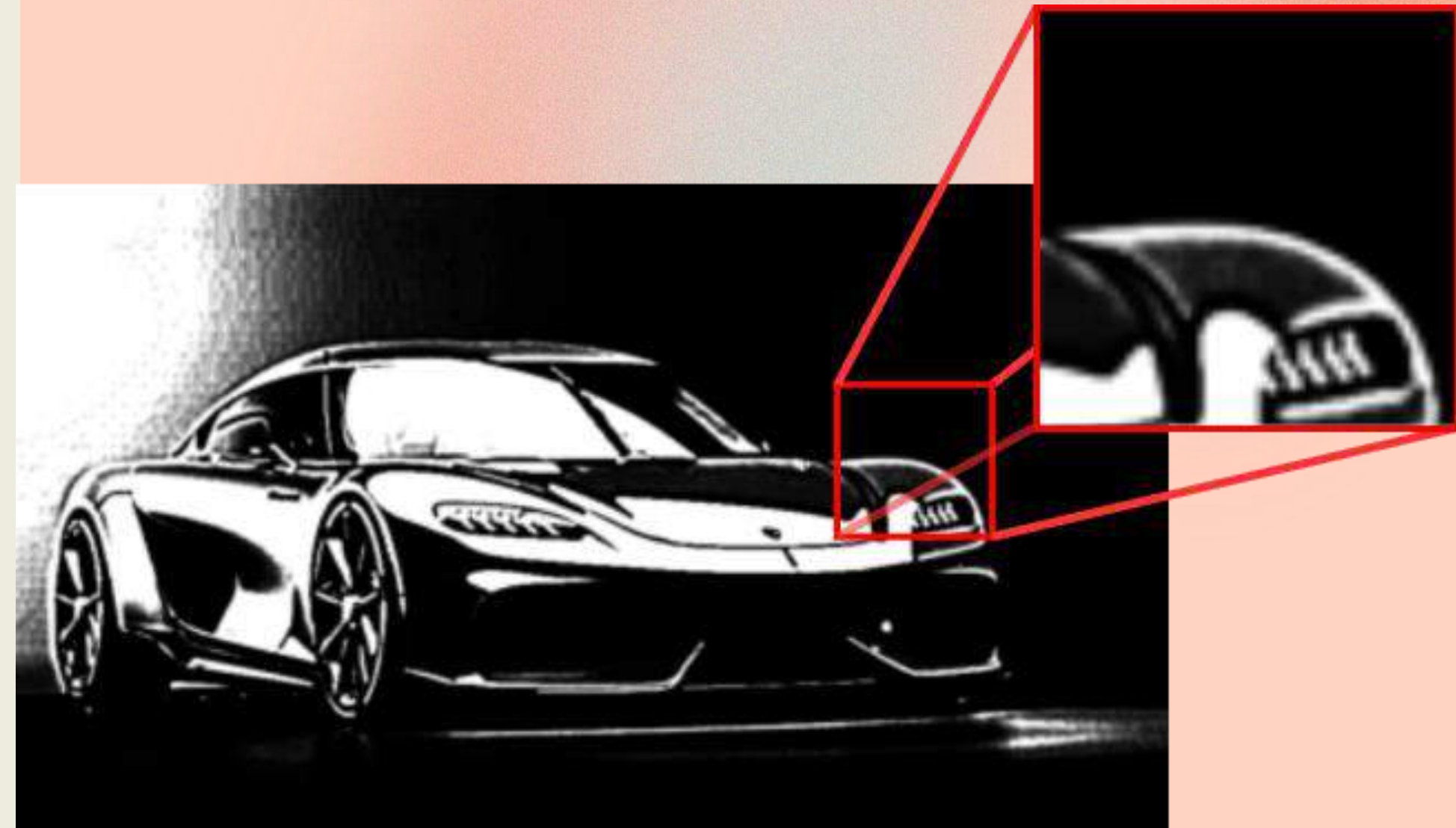


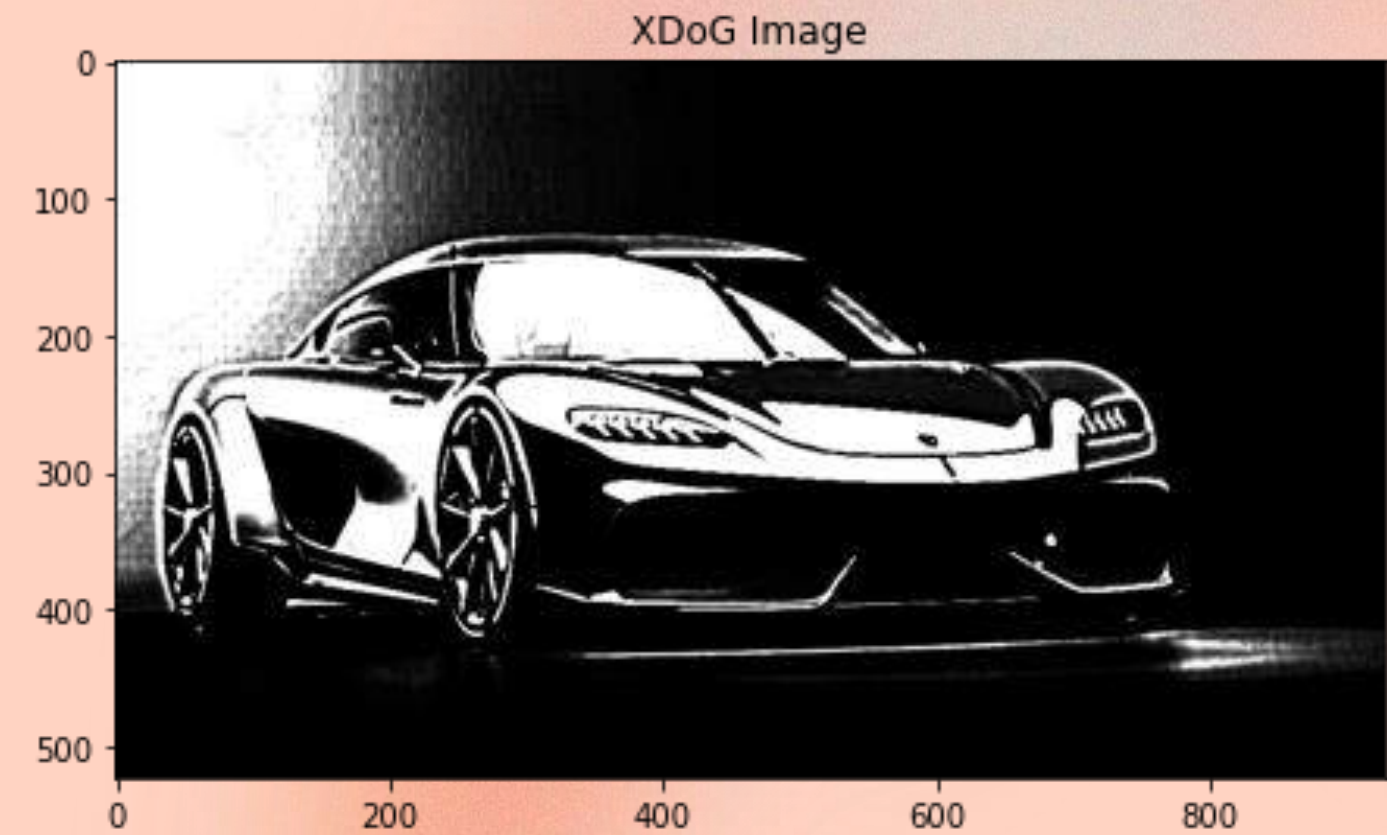
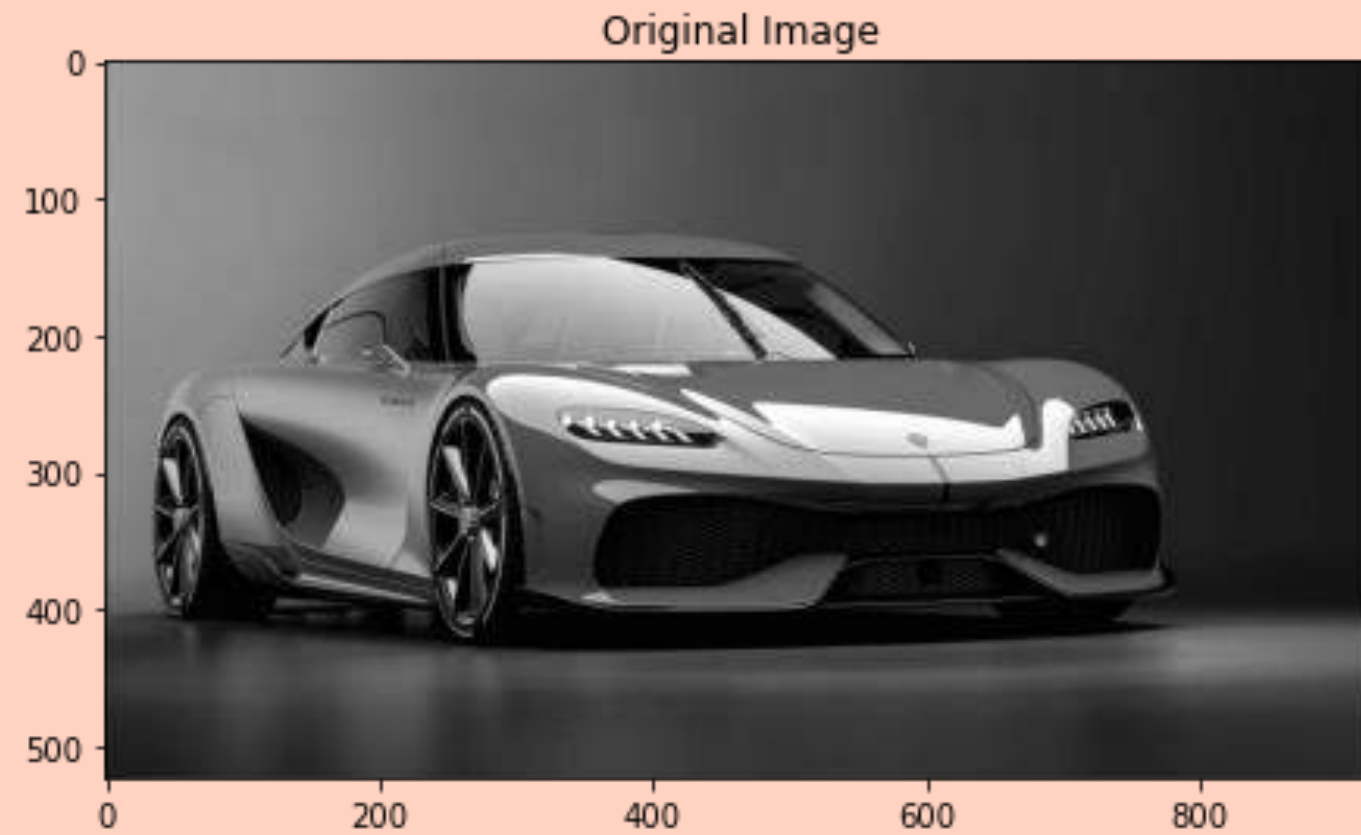
Cross-Hatching

- First, we compute a standard DoG edge image. We then create two high contrast XDoG images by setting $\phi \gg 0.01$.
- Individual images are multiplied together to compute the final image. For added effect, the hatching output can be composited onto a paper texture

Negative Edges

- If a strong positive DoG response occurs in a dark region of the image: white edge line
- If a negative DoG response occurs in a bright region of the image: black edge line
- Thus, both positive and negative edges can be achieved using XDoG



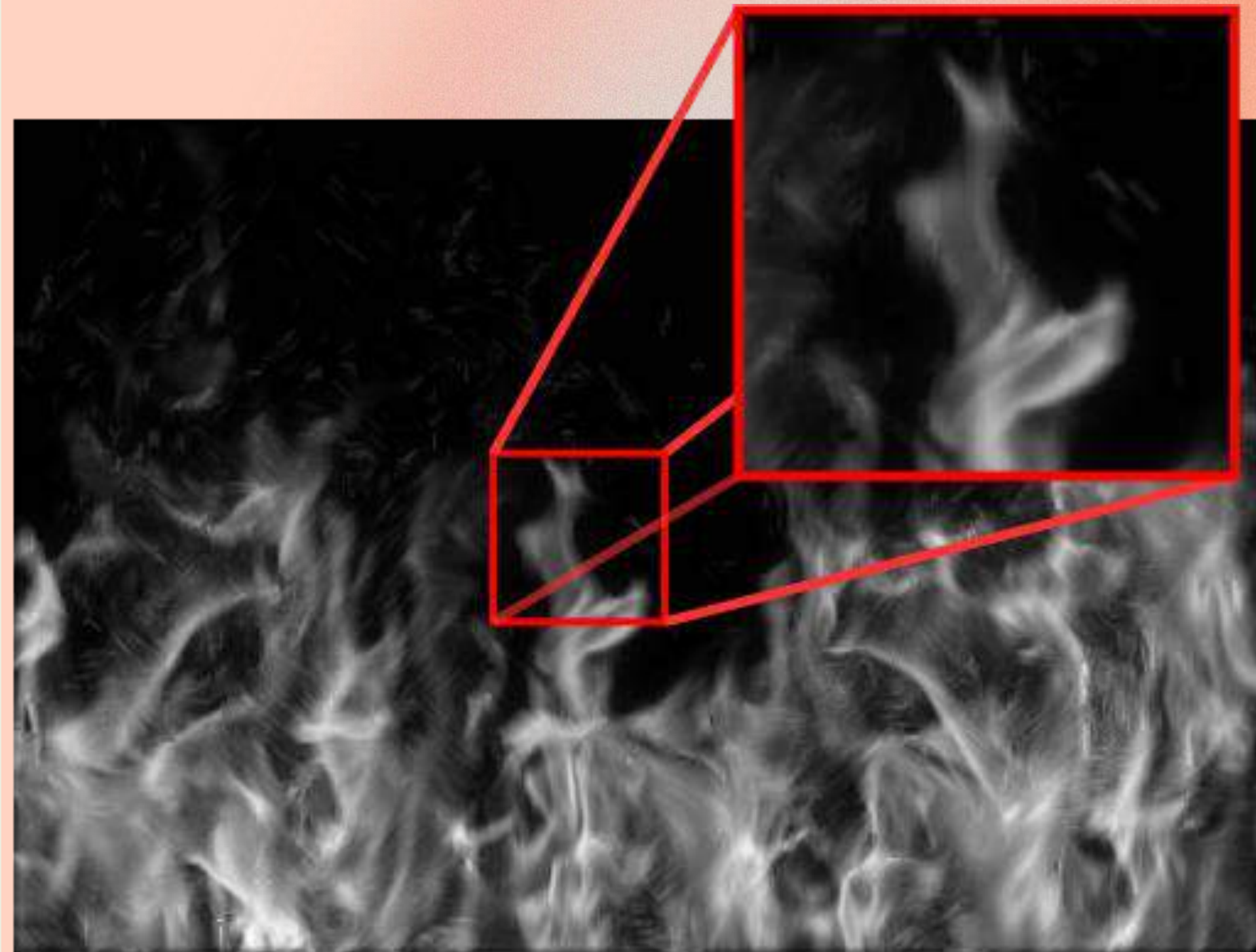


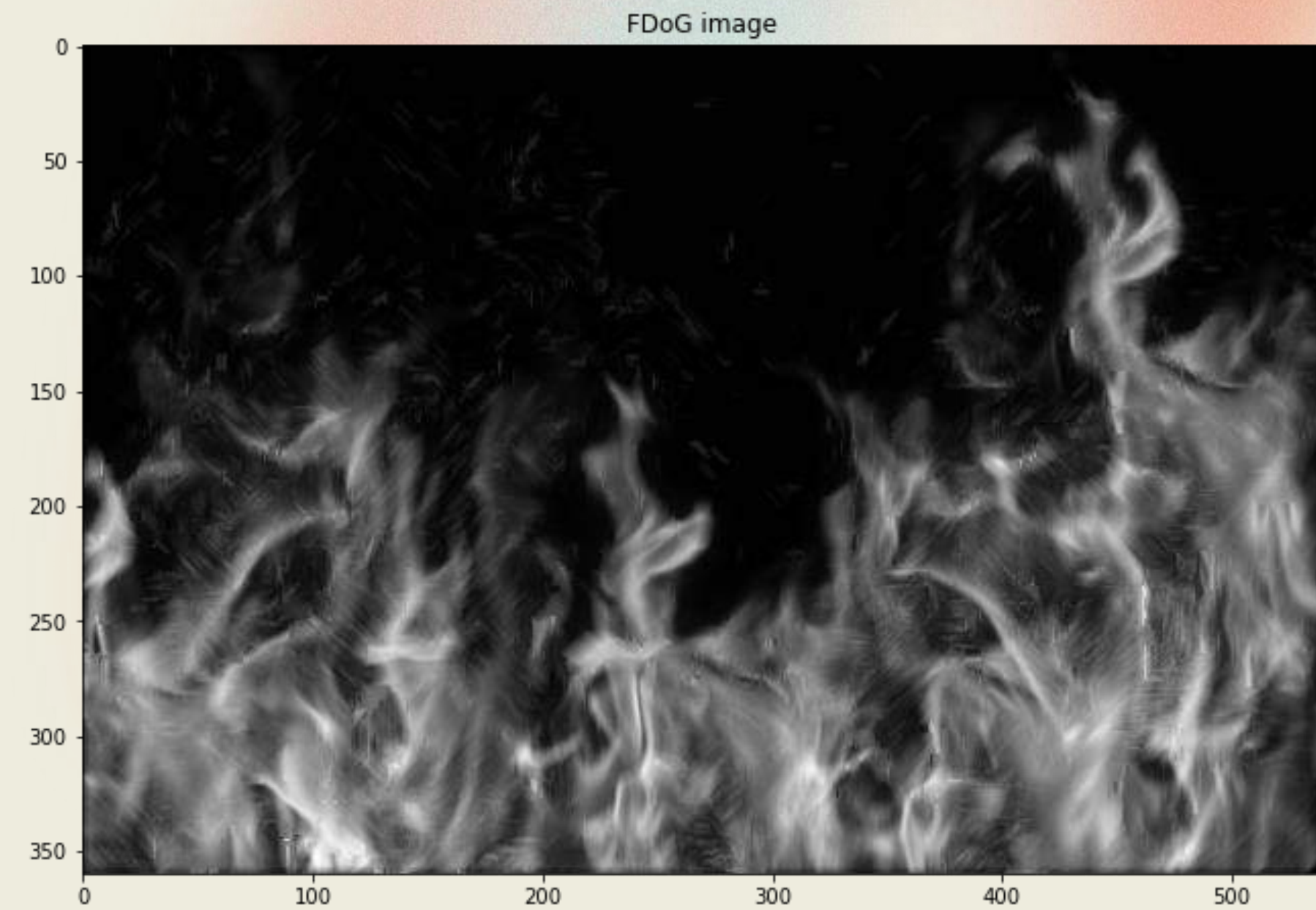
Negative Edges

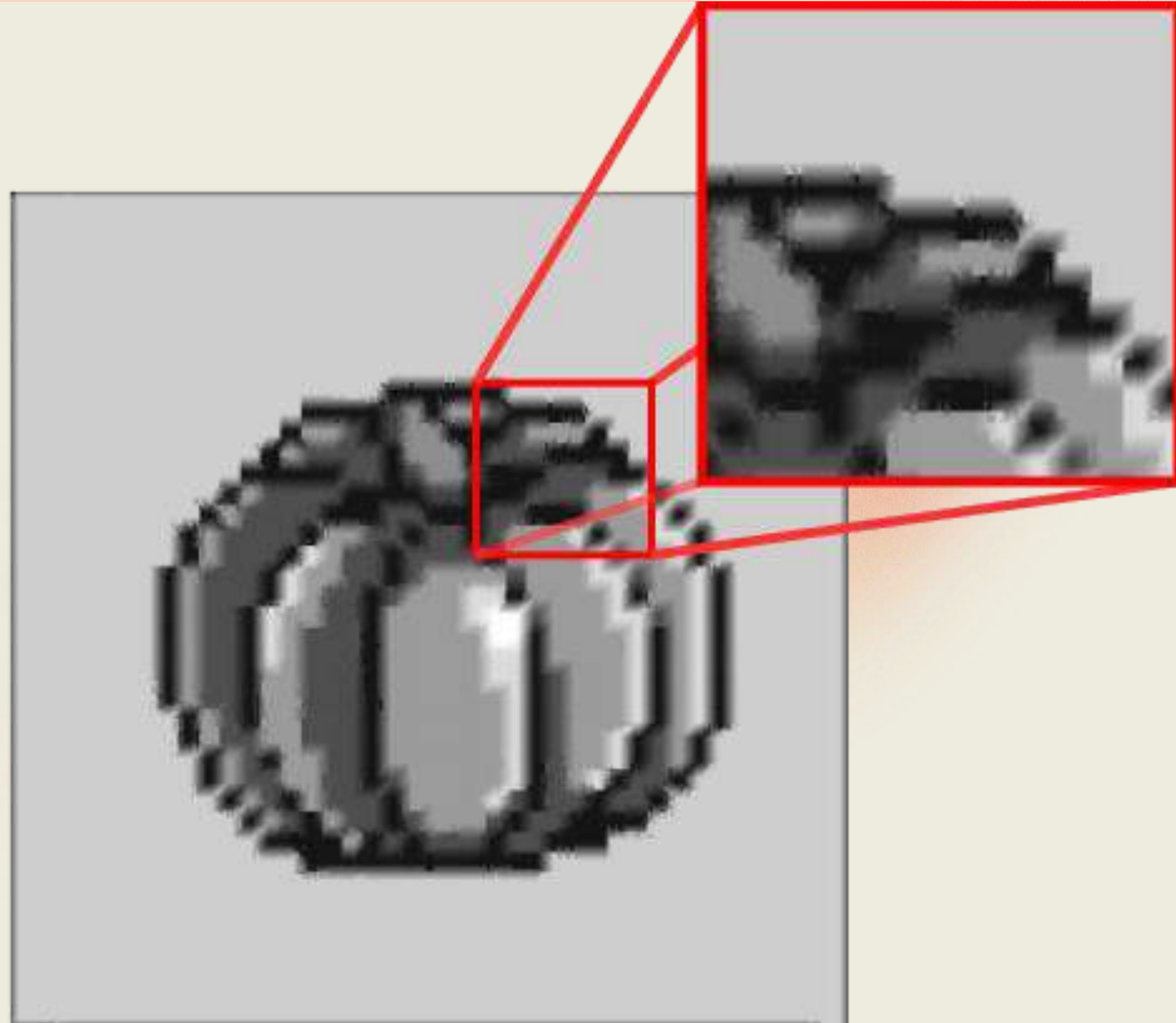
Using Negative Edge on an image - original vs. xDoG image

Anti-Aliasing

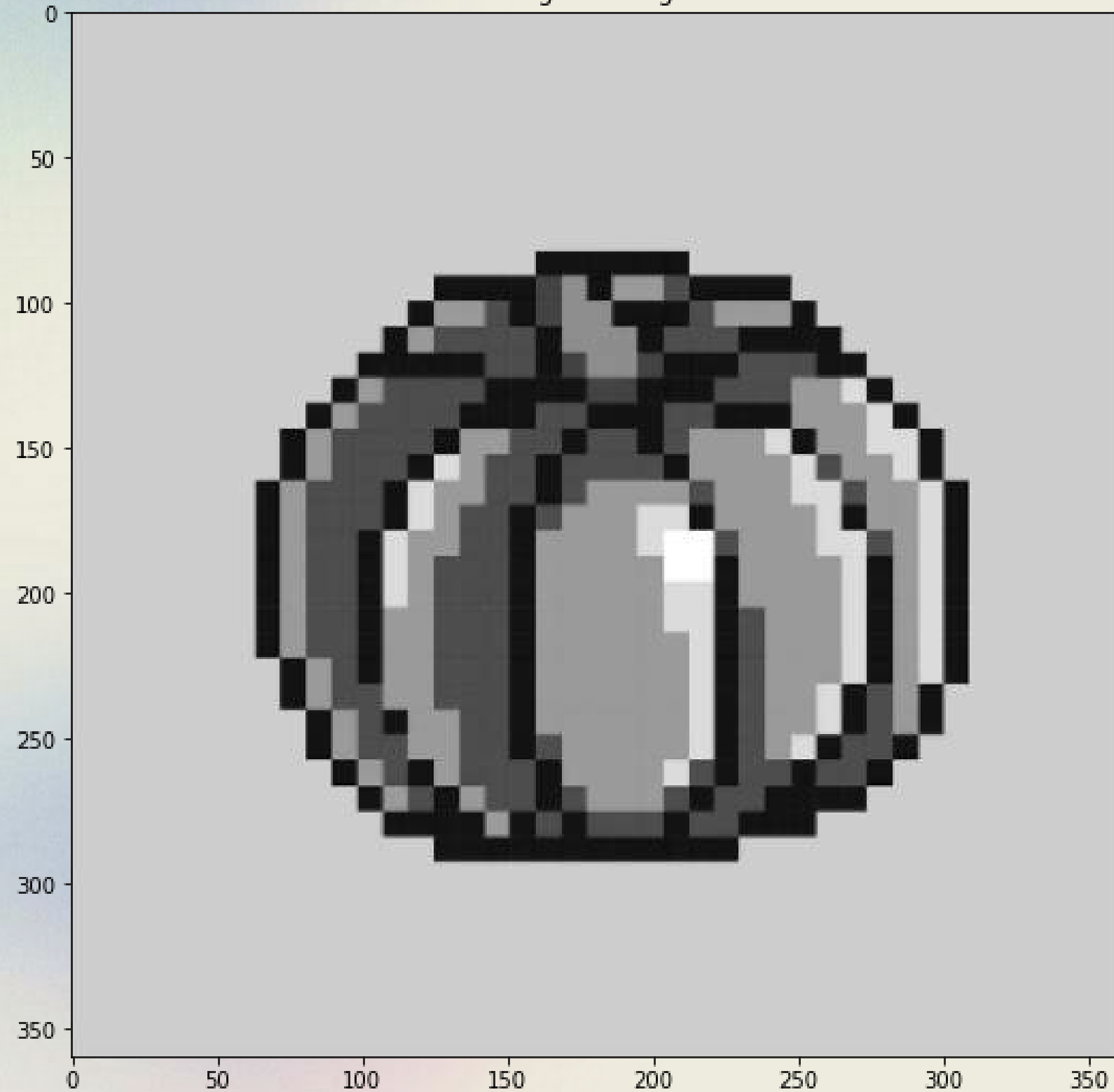
The Edge Tangent Flow that is used to compute the coherent edges can be reused to apply a very small line integral convolution along the field. This produces image-coherent and visually pleasing anti-aliasing, which is implemented here.



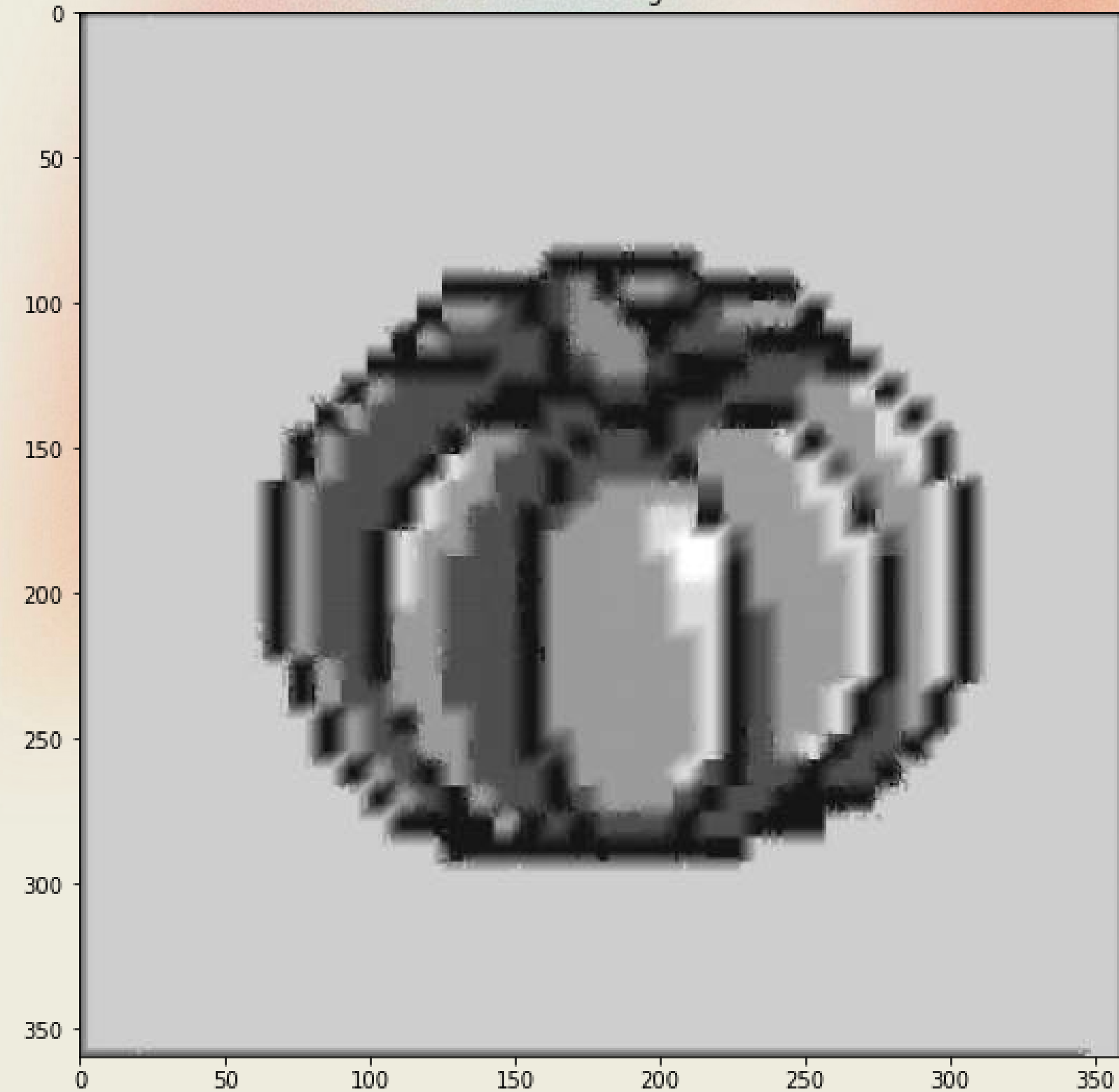




Original Image

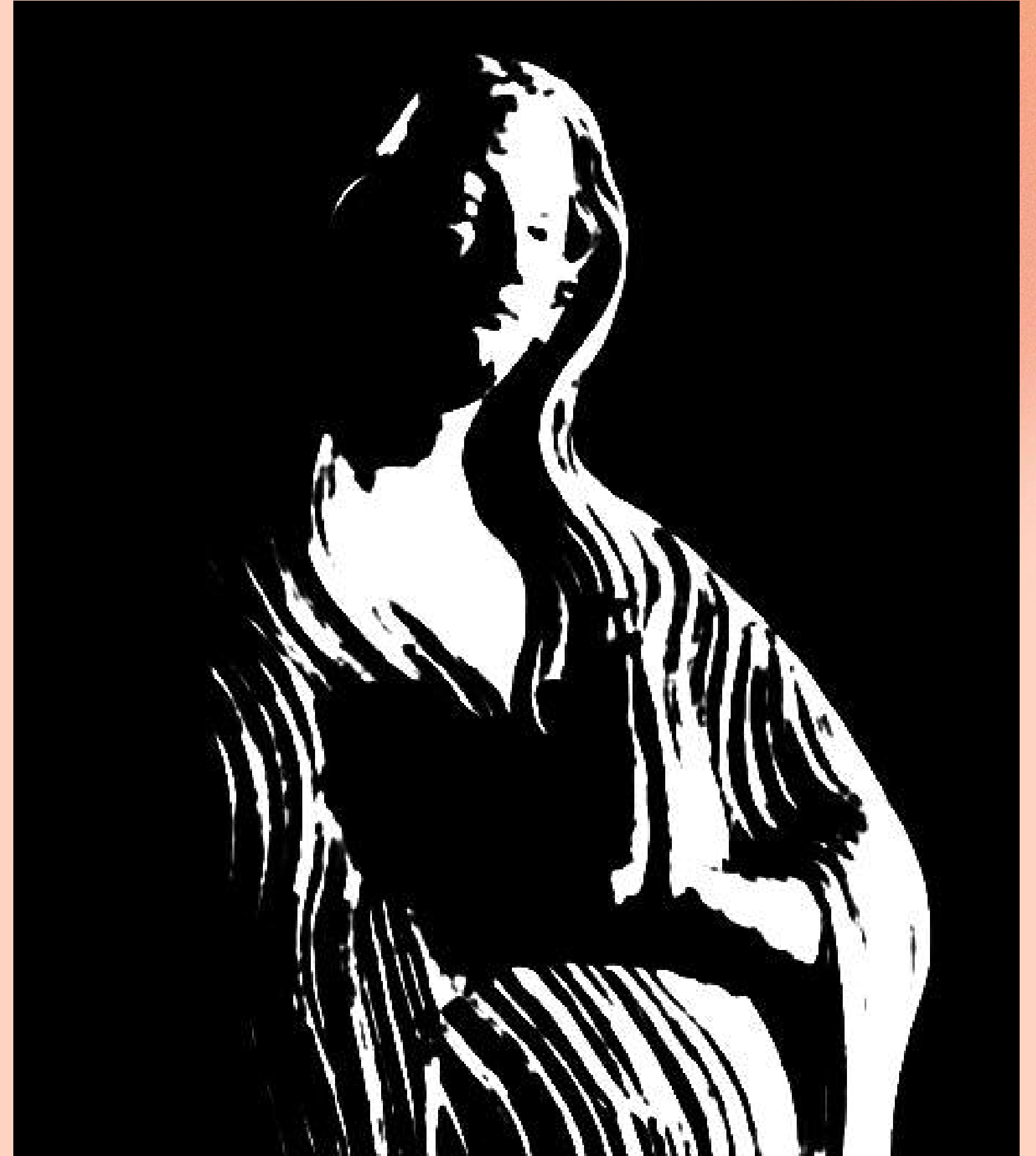


FDoG image



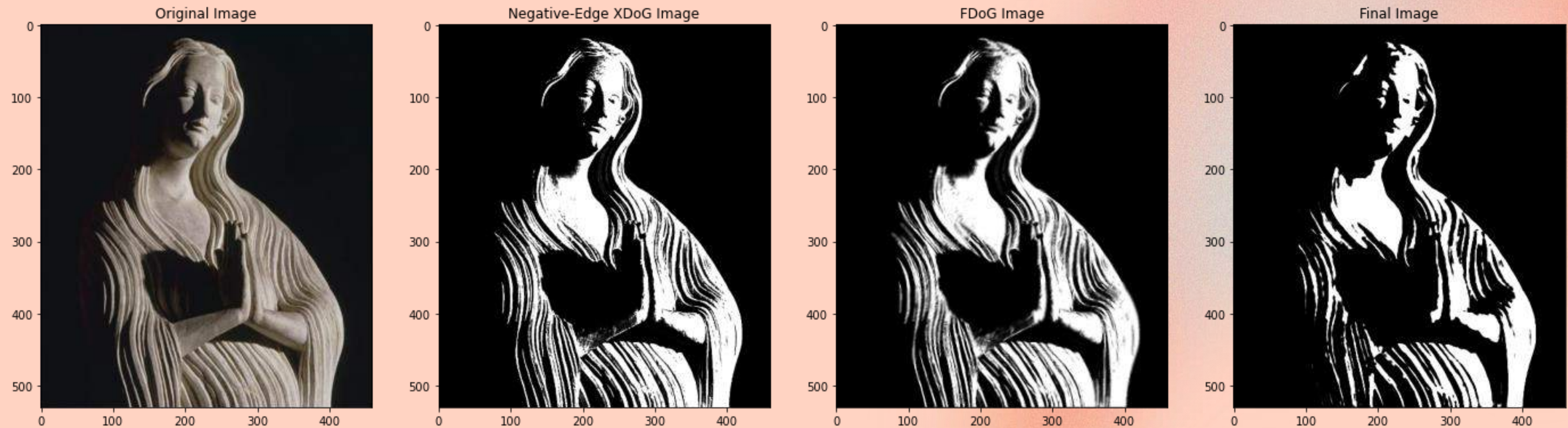
Woodcut

- We use very extreme edge emphasis and aggressive flow-based blurring to produce the desired results.
- $p = 100$ and $\sigma_c = 5$





Digital Image Processing



Woodcut

- Comparing the FDoG output with the woodcut output

Charcoal

- We decrease σ_c (gaussian applied on the structure tensor) and greatly increase the σ_m value for anti aliasing along the edge.
- This results in subtle stroke flows along the edge, giving us an effect of charcoal strokes
- Finally, we decrease the ϕ value to give a gradual gradient of white point of strokes.

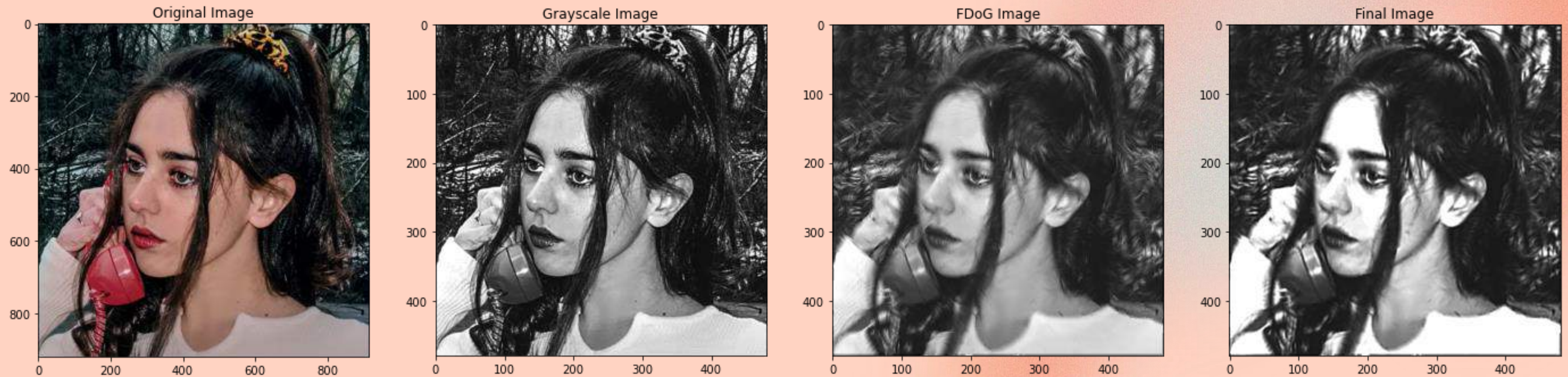


F-DoG Step





Digital Image Processing



Charcoal Effect

- The charcoal-like appearance can be attributed to larger spatial support; achieved by changing the sigma parameters. The FDog kernel is applied along the flow-direction, inducing an effect similar to charcoal on canvas.

Image Quantization Abstraction



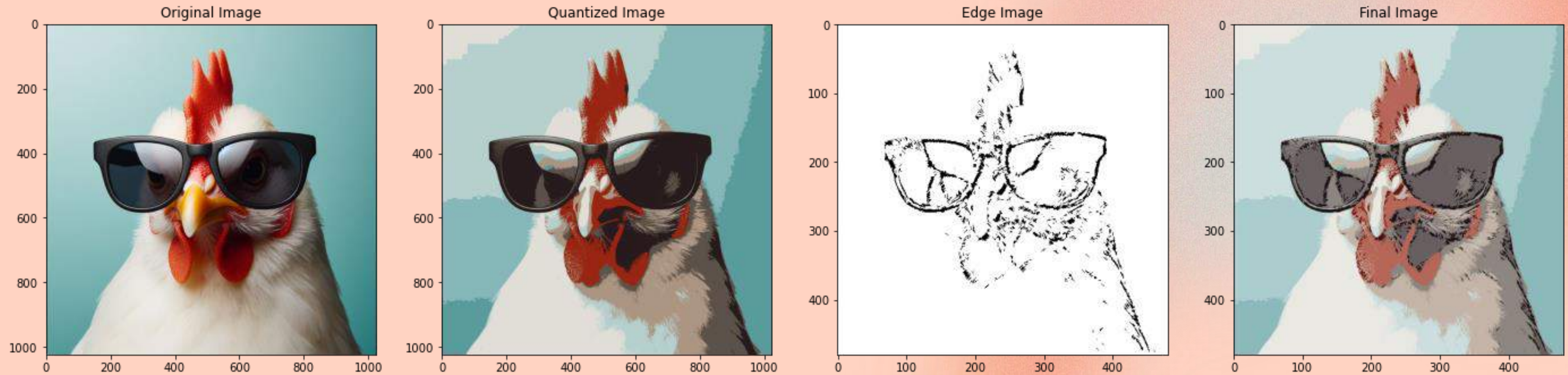
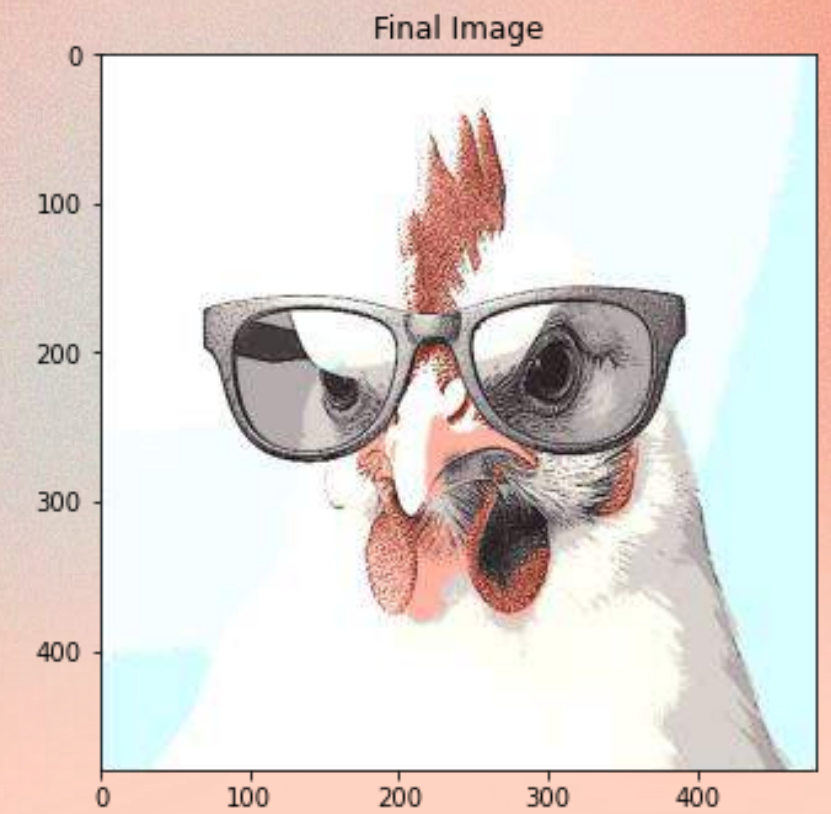
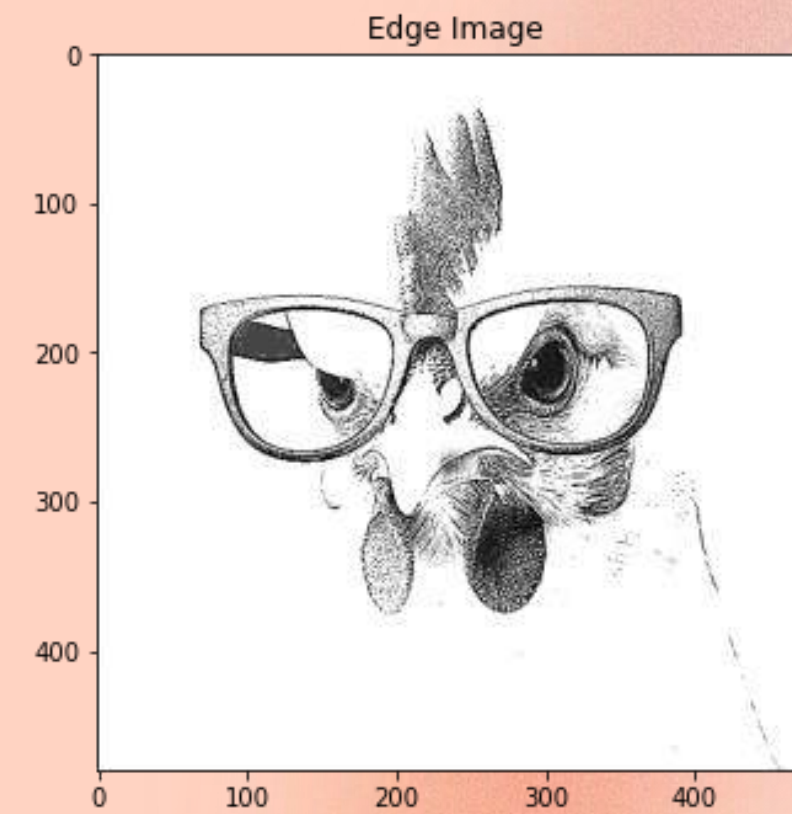
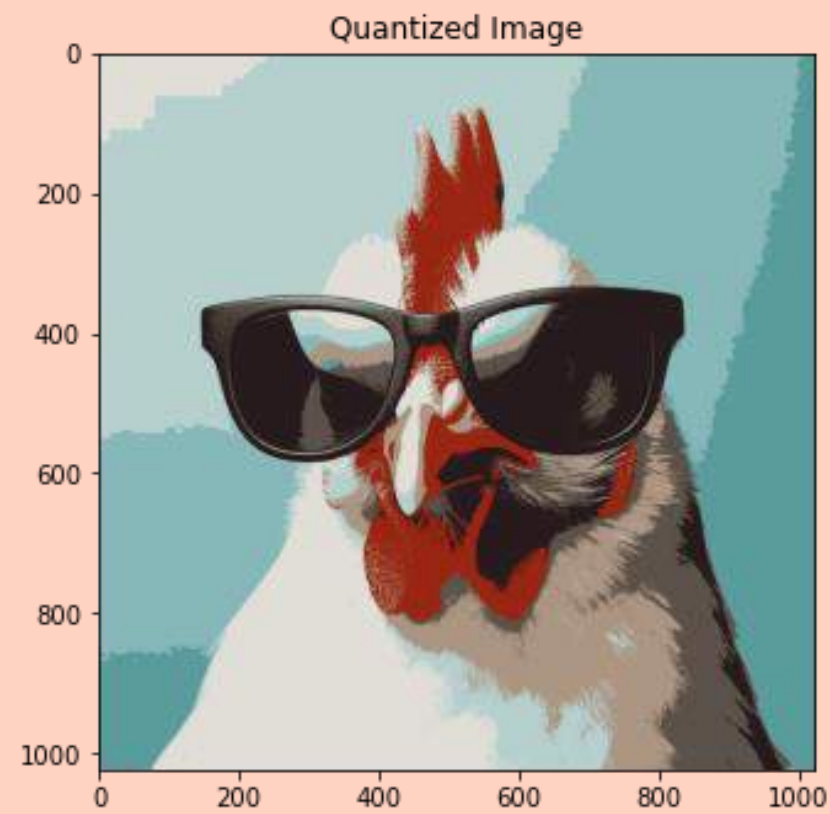
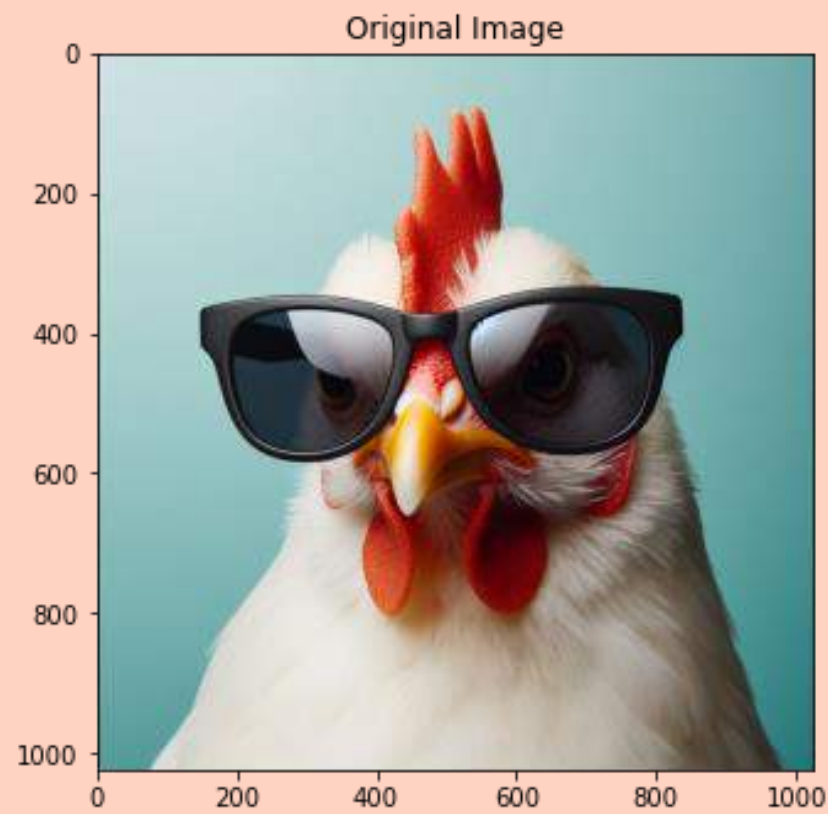


Image Quantization Abstraction

The ETF of the quantised image is computed and then we apply FDoG to it



Digital Image Processing



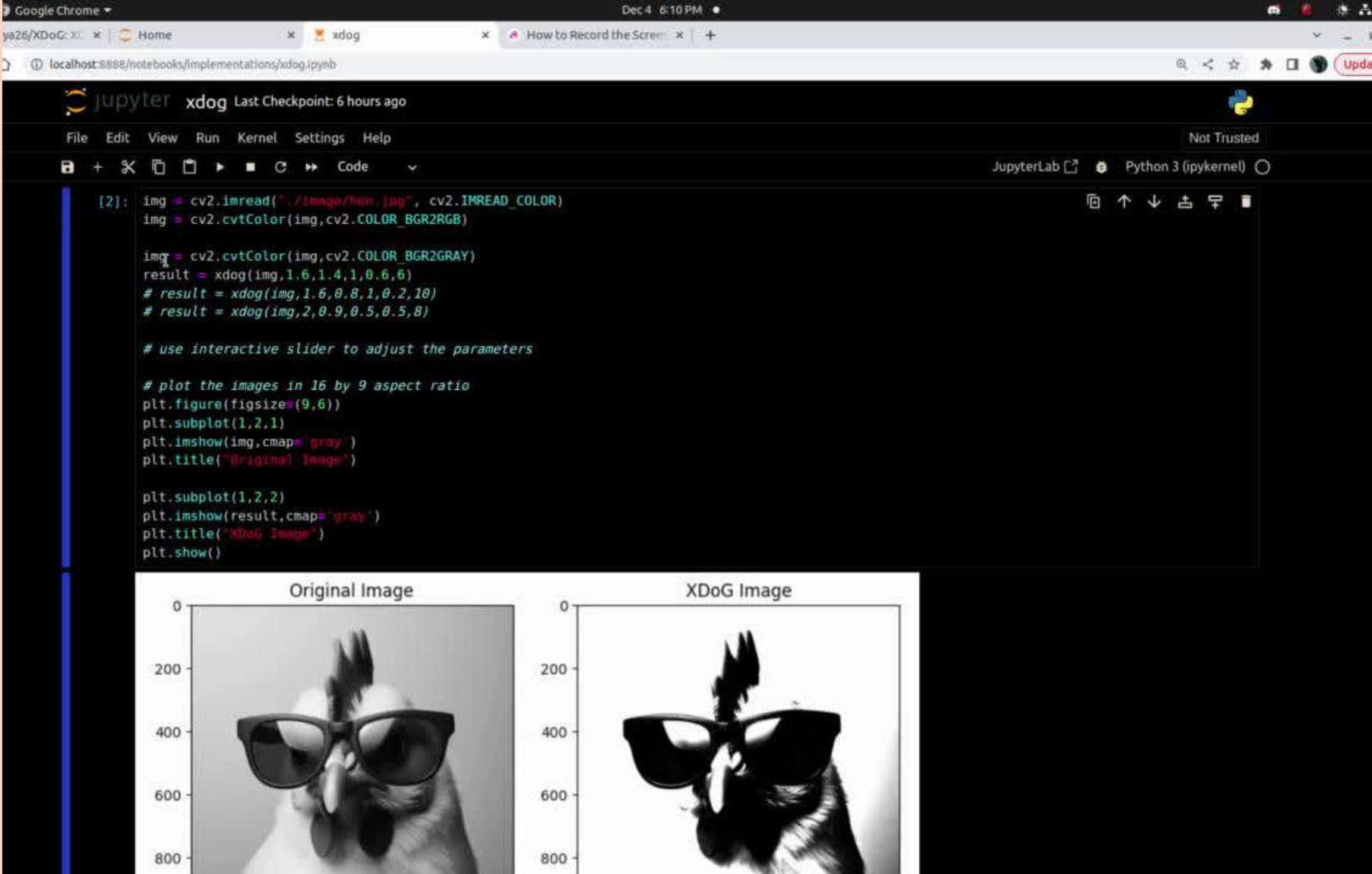
Pencil Sketch

Pencil Sketch

- Using a very high value of σ_e , we can isolate high frequency data like texture and edges.
- This can be blended with quantization abstraction to create pencil sketch image.



Real-time XDoG Implementation



```
[2]: img = cv2.imread("../image/han.jpg", cv2.IMREAD_COLOR)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
result = xdog(img, 1.6, 1.4, 1, 0.6, 6)
# result = xdog(img, 1.6, 0.8, 1, 0.2, 10)
# result = xdog(img, 2, 0.9, 0.5, 0.5, 8)

# use interactive slider to adjust the parameters

# plot the images in 16 by 9 aspect ratio
plt.figure(figsize=(9,6))
plt.subplot(1,2,1)
plt.imshow(img, cmap="gray")
plt.title("Original Image")

plt.subplot(1,2,2)
plt.imshow(result, cmap="gray")
plt.title("XDoG Image")
plt.show()
```


End of Presentation