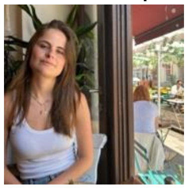
Neta Nakdimon
322216128
BS.c Computer Science



Edge Detection – Ex3

Bilateral Filter:

The bilateral filter is a non-linear technique that smooths and image while maintaining edges.

A. bilateral_one_pixel method:

For a given pixel (x, y) the filter computes a weighted average of its surrounding pixels (its neighborhood). The weights are calculated based on two factors:

1. **Spatial Closeness:** How far a neighboring pixel is from its center
2. **Intensity Similarity:** How similar the intensity of a neighboring pixel is to the central pixel.

The final value of the filtered pixel is determined as follows:

filtered_pix = $\frac{1}{Wp}\sum_{i=-d}^{d}\sum_{j=-d}^{d} S(i,j,\sigma_S)\cdot R\left(I_{neighbor}, I_{center}, \sigma_r\right)\cdot I_{neighbor}$

Where:

1. **filtered_pix:**
   The computed intensity value for the pixel at (x,y) in the filtered image.
2. $\boldsymbol{Wp}$**:**
   Normalization factor that ensures the weights sum to 1 – we need it to prevent the filtered pixel's intensity from becoming artificially brightened or darkened.
3. $\boldsymbol{d}$:
   The size of the neighborhood around the pixel. The filter considers a square window of size $(2d+1)\ \cdot (2d+1)$.
4. $\boldsymbol{S(i,j,\sigma_S)}$**:**
   Spatial weight, calculated as:
   $$S(i,j,\sigma_S) = \exp(-\frac{i^2+j^2}{2\sigma_S^2})$$
   This ensures that pixels further away from the central pixel contribute less.

5. $\boldsymbol{R(I_{neighbor}, I_{center}, \sigma_r)}$:
   Range weight, calculated as:

$$R(I_{neighbor}, I_{center}, \sigma_r) = \exp\left(-\frac{\left(I_{neighbor} - I_{center}\right)^2}{2\sigma_r^2}\right)$$

   This ensures that pixels with similar intensity to the central pixel contribute more.

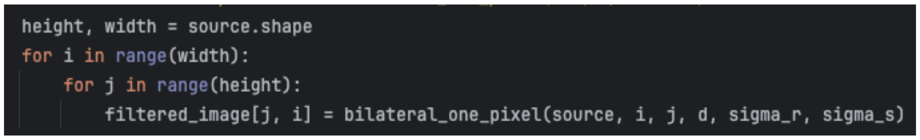
6. $\boldsymbol{I_{neighbor}}$:
   The intensity of a neighboring pixel located at $(x + i, y + j)$
7. $(\boldsymbol{i}, \boldsymbol{j})$:
   The relative position of the neighboring pixel within the filter window.

The computation steps in my code are:

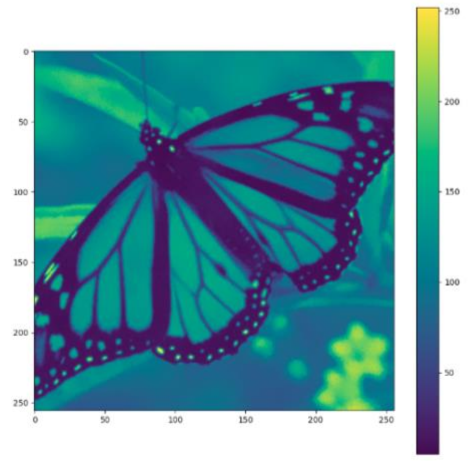
i. Iterate through neighbors: Loop through all the pixels in the square window centered at (x, y) with a size of (2d + 1) X (2d + 1)
ii. Compute weights:
   a. Spatial weight based on the geometric distance
   b. Range weight based on the intensity difference.
iii. Sum weighted intensity: Multiply the neighbor's intensity with its combined weight and add it to the sum.
iv. Normalize: Divide the summed weighted intensity by Wp to compute the final filtered value.

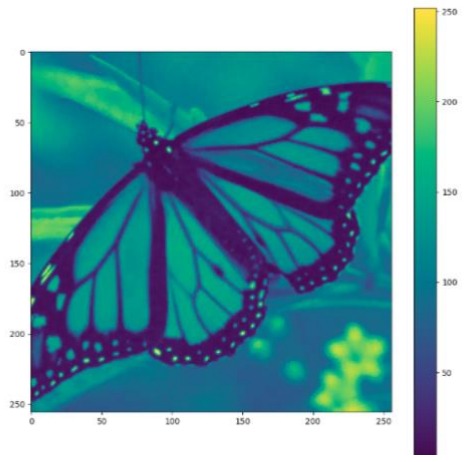
B. bilateral_filter method:

Run the bilateral_one_pixel method on each pixel in the input image:


```
height, width = source.shape
for i in range(width):
    for j in range(height):
        filtered_image[j, i] = bilateral_one_pixel(source, i, j, d, sigma_r, sigma_s)
```

- **Comparing the results of CV2.bilateral_filter and my bilateral_filter:**
  As can be seen, the results are similar.



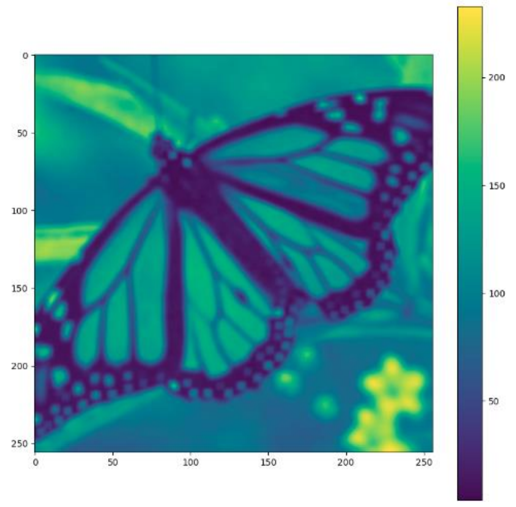


My bilateral_filter results

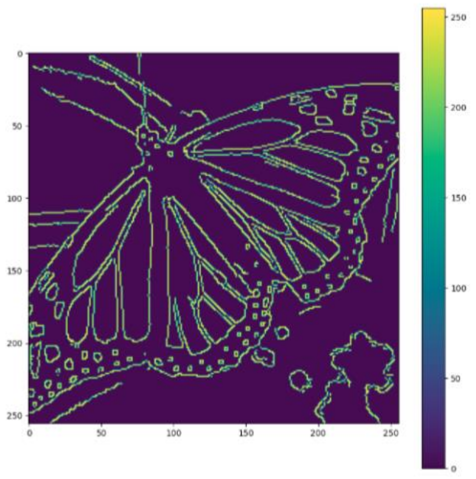
cv2.bilateral_filter results

- **Comparing to Gaussian blur:**

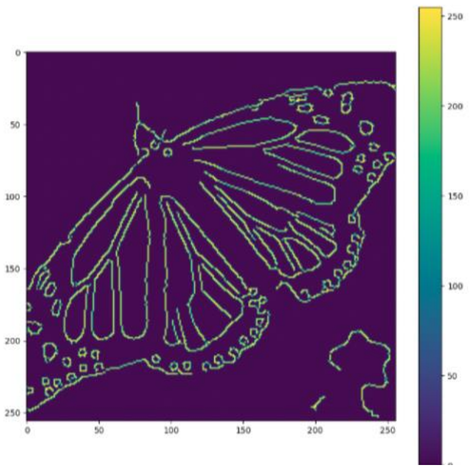
Gaussian blur applies uniform weighting based only on the physical distance between pixels, resulting in an overly smooth image where important details, especially **edges**, can become indistinct. In contrast, the bilateral filter incorporates both spatial closeness and intensity similarity when determining pixel contributions. This approach means that neighboring pixels that are not only nearby but also have comparable intensity values play a greater role in the filtering process. As a result, the bilateral filter effectively smooths the image while maintaining the clarity of edges, offering a more refined output compared to Gaussian blur.



- **Comparing Canny between the filtered images:**
  As explained above, the Canny result with the bilateral filtered image is more detailed since the bilateral preserves edges.



Canny with **bilateral** filter result



Canny with **Gaussian blur** filter result