STUDENT'S NAME Shirke Aryan			TOTAL MARKS OBTAINED
CLASS	SUBJECT: IPCV		
ROLL NO. 21bcs111	DATE 3rd April		

Bilateral Filtering Pseudocode

1. Define Input.

Image (I)

Kernel size (d) - diameter of the neighborhood around the center pixel.

sigma_r (σr) · controls the influence of intensity similarity.

(σs) - controls the influence of spatial proximity.

2. Iterate over each pixel (i,j) in the image

Initialize output pixel intensity (I result [i,j] to 0)

Initialize normalization factor (W) to 0.

3 Iterate over neighbouring pixel (p,q) within the kernel window centered at (i,j).

Calculate spatial distance (σs) between (i,j) & (p,q).

Calculate intensity difference (σr) between (i,j) & I (p,q).

Calculate spatial weight (ws) using a Gaussian func^n with started deviation σs: ws = exp. ($-\sigma s^2 / 2 * \sigma r^2$)

Calculate range weight (wr) using a Gaussian function with standard deviation σr: wr = exp ($-\delta r^2 / (2 * \sigma r^2)$)

Combine weights : w = ws * wr.

Update Normalizaⁿ factor : W += w

Accumulate weighted intensity : I result [ij] += I(p,q) * w

GOLD LEAF

STUDENT'S NAME		TOTAL MARKS OBTAINED
CLASS	SUBJECT	
ROLL NO.	DATE	

4. Normalize Output.

I result [i,j] /= W (divide by the normalizaⁿ factor).

Experimenting with σr & Sigma s (σs)

1. Implement the bilateral filtering function with options to set σr and σs.

2. Run the function on your image with different combinations of σr and σs values (eg: low, medium, high values for each).

3. Observe the resulting images. Typically:
- Lower σr leads to sharper edge presentaⁿ but may not remove all noise.
- Higher σr removes more noise but can blur edges.
- Lower sigma s affects a smaller neighborhood potentially leading to patchy results.
- Higher Sigma s considers neighborhood, provide smoother filtering.

Comparing with OpenCV:

- Use OpenCV's cv2.bilateralFilter function on your image with similar settings for Kernel size, σr & σs as your implementaⁿ.
- Calculate the absolute difference between the intensity values of corresponding pixels in your results (I_result) & open CV result (I_inbuilt)

- Sum the absolute differences across all pixels to get the total error (abs. sum (I_inbuilt - I_result))

Conclusion.

Analyzing the visual quality of your bilateral filtering results with different parameter settings.

Compare the error between your implementation & OpenCV's function. A small error indicates similar filtering behaviour.

Consider the trade off between noise removal & edge preservation based on your image & application requirement.