Review of "Bilateral Filtering For Gray and Color Images"

Concordia University

COMP 478/6444 Final Report, 2022

Pablo Arevalo Escobar
40081955

Jahrel Stewart
40115728

Pablo Arevalo Escobar 40081955
Jahrel Stewart 40115728

Motivations and Contribution

Bilateral filtering is a solution to edge blurring, a key flaw of the traditional, linear, smoothing functions. Before bilateral filtering there existed iterative diffusion methods which could smooth while preserving edges; however, they came at a cost of stability and efficiency [1] . The bilateral filter proposed provided a non-linear, non-iterative approach that smooths regions while preserving edges.

Principle Approach

The idea is to consider and combine both the spatial domain and the color range of the image when filtering. Where the spatial filter has a bias to pixels closer to it, the range filter has a bias to pixels similar to it. The paper does not use the standard RGB color space, instead it makes use of the CIE-Lab color space. In this space colors that are closer together (euclidean distance) are strongly correlated perceptually, producing a more favorable bias function than RGB. The bilateral filter can be formalized as a filter **H** composed of: a normalizing term **W**, a photometric similarity component **s**, and a geometric closeness component **c**.

$$H(x) = \frac{1}{W(x)} \int\int f(x)c(\xi, x)s(f(\xi) - f(x))\, d\xi$$

$$W(x) = \int\int c(\xi, x)s(f(\xi) - f(x))\, d\xi$$

Components **c** and **s** each have their own standard deviation parameter, $\boldsymbol{\sigma}_c$ and $\boldsymbol{\sigma}_s$. Changing the value of $\boldsymbol{\sigma}_c$ will alter the strength of the blur, changing the value of $\boldsymbol{\sigma}_s$ will determine how similar two pixel colors have to be in order to be mixed.

Unlike range filtering, bilateral filtering is capable of local mappings of gray areas throughout different points of an image based on the context of neighboring pixels.

Critiques

The paper shows how range and domain filtering differ individually, and how they perform when combined as bilateral filtering. Results are also shown for how the tweaking of bilateral filtering parameters can alter the image. There are also appropriate illustrations that allow for a better understanding of the details that go into implementing bilateral filtering. The reader is left with a grasp on when using bilateral filtering is the most advantageous, how its components affect different parts of an image and what drawbacks it has under several conditions. On the other hand, the paper fails to adequately explore and explain the limitations of the algorithm. Bilateral filter side effects such as the staircase effect [5] and gradient reversal [6] are not mentioned. The paper also fails to provide any metrics for the running time and complexity of the algorithm. The bilateral filter is slow and many papers have since provided faster alternatives.

Modern Techniques

Fast bilateral filtering [2] provides an approximation of the range kernel that produces results without the time-consuming drawbacks seen in the original paper. The algorithm factorizes the range kernel as a product of exponential functions and approximates the last term; this transforms the bilateral filter to a linear combination of linear convolutions. The mean square error is then employed to find the optimal approximation of the range kernel. This method runs in O(1), [2] significantly outperforming the brute force approach and outperforming shiftable kernels based methods, such as Chaudhary [3] by a margin greater than 30%. The method is also more accurate than its shiftable kernel counterparts because of the use of the mean square error.

Implementation

Our implementation of bilateral filtering is based on the discrete formulation of the continuous equation described in the original paper. Therefore, our bilateral filtering implementation can be understood, mathematically, as:

$$H(x) = \frac{1}{W(x)} \sum I(x_i)c(||I(x_i) - I(x)|)s(||x_i - x||) \quad [4]$$

$$W(x) = \sum c(||I(x_i) - I(x)|)s(||x_i - x||) \quad [4]$$

Defined in Matlab as : bilateralF(Image, WindowSize, SigmaSpatial, SigmaRange)

This function is then split into bilateralFCol and bilateralFGray. We can understand this transformation function as a series of steps:

1. Compute spatial gaussian
2. Compute (bounded) kernel window
3. Bound the gaussian
4. Compute range (photometric) gaussian
5. Compute the weight
6. Normalize function and return the value

Note that for the color version of the algorithm, we first convert the image from rgb space to the lab color space. We then scale the sigma range value by the largest value in the lab color space (initially we did not do this and the filter did not transform the image as the sigma range value was too small). Then steps 2,4,5, and 6 are each done separately on each of the 3 LAB color channels; at the end we convert the entire image from the lab color space to rgb color space.The spatial kernel is initialized and computed before the main loop based on the window size parameter, initially this was not the case but the performance overhead was high. Moving the spatial kernel initialization out of the loop improved performance significantly.

In the main loop of the algorithm we first compute the bounds for the current window, any value in the window that falls outside of the image is ignored. Because the window is bounded to lie within the image borders, we also have to bound the gaussian to match the window's bounded dimensions. The photometric gaussian is then computed for each value in the bounded window, the result is then multiplied with the precomputed (bounded) spatial gaussian to get the weight, W. The result is then computed by taking the sum of the weight multiplied by the window and normalizing the value by the sum of the weight. The attached code is heavily commented and variables are named descriptively in the interest of clarity.

Table 1 demonstrates the edge preserving behavior of our bilateral filtering implementation. The images are all taken from the matlab image library except the cat image, which has been extracted from the original paper. The MATLAB image names (in order) are: eight.tif', 'cat.png', 'peppers.png', and 'toysflash.png'.

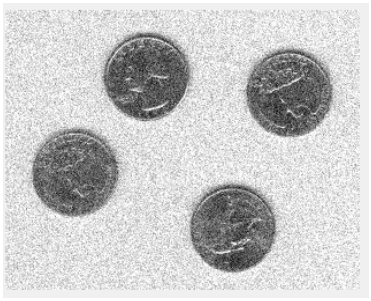
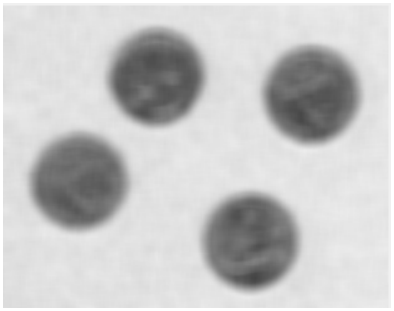
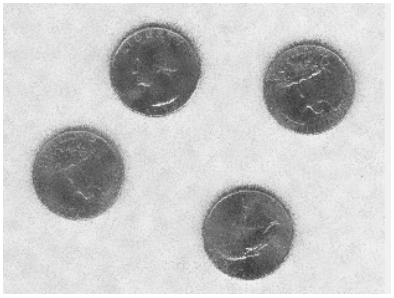
Original Image	(Gaussian) Smoothed	Bilateral Filtering
		
		
		
		

Table 1: Effects of the bilateral filter

Kernel size = 11, σ_c *=7,* σ_s *= 0.2*

Table 1 shows the behavior of our implementation at the constant parameter state described above. The gaussian function used has the same parameter set but without the range sigma (kernel 11, spatial sigma 7). The main item of note is that (at the same parameter state) gaussian smoothing performs better at smoothing noise in general when compared to the bilateral filter, this is most obvious in the coin image where the bilateral filter succeeds at preventing edge smoothing but at the cost of more leftover noise in the background of the scene. This behavior is a consequence of the range bias, the parameters need to be tuned on an image-by-image basis in order to provide the optimal result. In this case, if less noise was desired, that would involve increasing the range sigma or increasing the kernel size.

The second image (of the cat) was chosen to provide a direct comparison to the original paper; the bilateral filter implemented successfully preserves the eyes, nose, and ear edges of the cat. This is akin to the behavior demonstrated in the paper. The last two images were chosen because they are colored and because they contain a large amount of edges which we could use to judge the performance of the algorithm.

The edges on the pepper image are well preserved for the most part, however, the transformation is not consistent and certain edges are preserved more than others. This can be seen by comparing the edges of the green pepper in the back left against the garlic clove in the front right. This behavior is most likely due to the color difference (in LAB space) between the background and the vegetable, in the case of the garlic clove the difference is larger than the green pepper and therefore the edge separation is clearer than before.

The final image was chosen as it contains a variety of edge types. It is clear that our implementation successfully preserves the edges of the sphere and objects surrounding it. Small edges, such as the details of the face in the picture, are not preserved as well. This is likely due to the kernel size and the range sigma, decreasing either would result in a less blurred image with more details.

The behavior demonstrated by the bilateral filter in **Table 1** closely mirrors the behavior demonstrated by the algorithm in the original paper. This edge preserving behavior has applicability in image restoration and more. One possible scenario we considered is to apply the filter selectively in order to create an illusion of camera focus, we use the bilateral filter around the region of the desired object of focus and in every other region we can use standard, linear, gaussian smoothing.

The behavior of the bilateral filter is better understood as we vary the parameters, **Table 2** demonstrates how the filter behaves as we vary the spatial sigma ($\boldsymbol{\sigma}_s$) and the range sigma ($\boldsymbol{\sigma}_c$) on the standard test image of Lena Forsen.

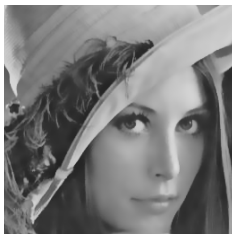
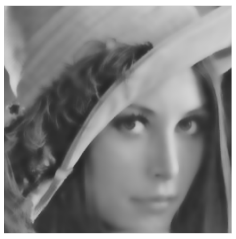
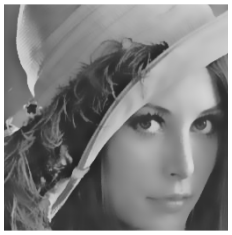
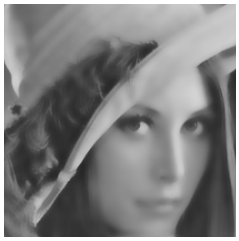
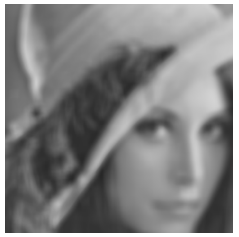
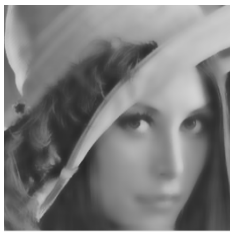
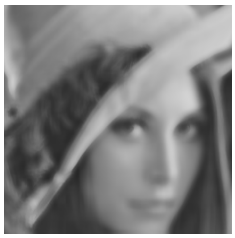
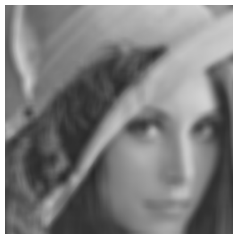
$\sigma_c = 3$				
$\sigma_c = 7$				
$\sigma_c = 11$				
	$\sigma_s = 0.1$	$\sigma_s = 0.25$	$\sigma_s = 0.5$	$\sigma_s = 1.0$

Table 2: Effects of changing the parameters at a constant kernel size (11)

We will consider first the effect of varying the sigma of the spatial gaussian and then that of the range/photometric gaussian. Examining the spatial sigma (σ_c) at each iteration at the extreme of the range sigma ($\sigma_s = 1.0$) provides a good understanding of its purpose in the function. By setting the spatial sigma to 3 we set a bias for pixels that are within 3 units of the center pixel, it is important to note that pixels that are more than three standard deviations away (in this case 9 units) will form less than 0.01% of the weight and can thus be ignored.

This is most evident in the rightmost image as the smoothing occurs across small distances, such that the edges are still clear. Once the spatial sigma is increased to 11, blurring occurs clearly across the face and edges are blurred away (the nose, lips, and eyes are clearest indicators of this).Leaving the spatial sigma at 11, as we gradually decrease the range sigma we start to see a higher degree of edge preservation. It is interesting to note that the difference in images between 0.5 and 1.0 (a sigma difference of .5) is subtle and not significant, the edges of the eyes are not blurred as much. However, the difference between 0.6 and 0.25 (a sigma difference of .15) is more drastic. The edges of the eyes are clearly preserved. This difference is even more drastic

between 0.1 and 0.25, where the eyes, lips and nose edges are clearly preserved. This demonstrates that the range sigma does not behave linearly, in fact, most of the desired parameter tuning (at least in our implementation) occurs in the lower 30% of the range of possible values (0.0-0.3).

To summarize, for high values of σ_s, the domain filter behaves like a Gaussian filter, with σ_s having minimal effect on σ_c, and for low values, it has a much greater effect that preserves edges. When both parameter values are high, a hazy effect ensues with no loss of detail. Whereas, when both values are relatively low, details are removed, but contours are preserved. Although the bilateral filter has a high computational cost, it is possible to achieve a lower cost by computing all values for the similarity function beforehand.

Comparison with baseline methods

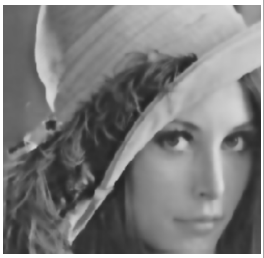
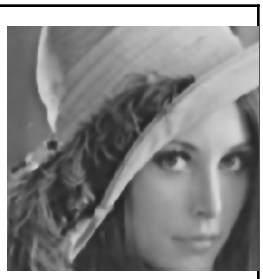
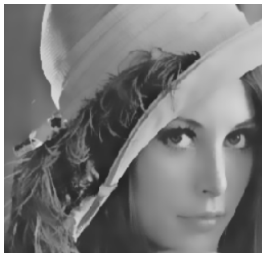
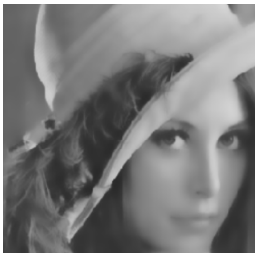
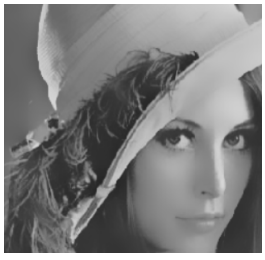
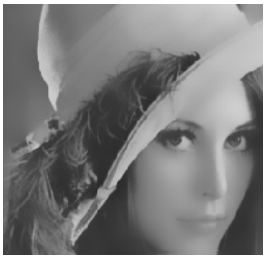
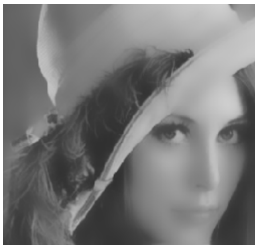
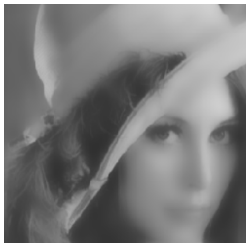
$\boldsymbol{\sigma}_c = 2$				
$\boldsymbol{\sigma}_c = 4$				
$\boldsymbol{\sigma}_c = 7$				
	$\boldsymbol{\sigma}_s = 0.01$	$\boldsymbol{\sigma}_s = 0.025$	$\boldsymbol{\sigma}_s = 0.05$	$\boldsymbol{\sigma}_s = 0.1$

Table 3: MATLAB's bilateral filtering function at various parameters at a constant kernel size (12)

The MATLAB function, *imbilatfilt(I,degreeOfSmoothing,spatialSigma)*, was used to implement bilateral filtering on all images shown above using various combinations of parameter values. In order to achieve a similar effect to our method, a different range of values for both $\boldsymbol{\sigma}_s$ and $\boldsymbol{\sigma}_c$ was required due to MATLAB's use of the parameters within their function. A noticeable difference from our results is the crispness of the edges which seems to persevere despite the increase in range sigma $\boldsymbol{\sigma}_s$. In addition, it can be seen that as the value of $\boldsymbol{\sigma}_c$ increases, the images become more flat, losing textures yet preserving contours. Ultimately, the results here are similar to ours and are indicative of the accuracy of our implementation of bilateral filtering.

Reflection:

The bilateral filter is a powerful, edge preserving, alternative to the traditional smoothing methods, however, because of its nonlinear nature it can be significantly slower than traditional smoothing. This becomes clearer as the size of the input image increases and the complexity grows asymptotically.

The trade off inherent to bilateral filtering and its linear smoothing counterparts is between the importance of edge preservation and the importance of speed. Methods such as the fast bilateral filtering discussed in the review address this issue by providing significant speed ups through different means of approximations. While these methods are approximations, optimizing techniques such as the mean squared error used in the fast bilateral filtering paper help provide highly accurate and fast methods. An implementation of the fast bilateral filtering algorithm was attempted for comparison's sake but proved to be out of the scope of this project given our commitments.

The implementation process for the algorithm was iterative, with the first iterations of the code being significantly slower than the end result. The first results of our implementation tended to blow out the image, increasing the brightness of most pixels while preserving edges. This was because the pixel values had to be normalized to fit within the range of [0,1] for the range gaussian to behave as expected, the solution was to convert the image to double format and divide it by 255 before starting the processing in the function.

The main difficulty to overcome in the implementation was speed, at first the algorithm would take around 20-40 seconds for gray values images and three times that for colored images, this made the process of algorithm testing incredibly tedious. Initially a nested for loop was used to create both the spatial and range gaussian kernels at each pixel, this was incredibly wasteful as the spatial gaussian kernel does not change once the kernel size is determined. Once we realized the issue, the spatial gaussian was initialized outside of the loop and (on average using the images mentioned in this report) the algorithm now took seconds to process and apply the filter.

Another key optimization choice had to do with the rgb2lab conversion, initially the color space conversion was happening at a pixel by pixel basis. Because the conversion process is slow, the algorithm for colored images took > 5 minutes to produce a result. The solution was simple, rgb2lab conversion was applied directly to the image before and after processing. The speed up was immediate and the colored images now processed in around 6 seconds.

There still exists further optimization potential (excluding non-brute force methods), currently our color bilateral filtering function is simply the gray filter function but performed in three different channels instead of one. It is likely possible to minimize the time spent on each individual channel and perform operations on all three channels at once. However, we concluded that for the scope of this project the speed of our implementation proved to be sufficient.

Overall we are satisfied with the performance of our implementation, however, certain design decisions have proved redundant in hindsight. The inclusion of a window size parameter could be completely removed, instead, the spatial sigma could be used to estimate the size of the kernel (three standard deviations).

References

[1] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images", Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India.

[2] Zhang X, Dai L. Fast bilateral filtering. *Electronics letters*. pp.:258-260, 2019

[3] Chaudhury, K.N.: ‘Acceleration of the shiftable O(1), algorithm for bilateral filtering and nonlocal means’, Trans. Image Process.,pp. 1291–1300, 2013,

[4] Banterle, F.; Corsini, M.; Cignoni, P.; Scopigno, R.. "A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain". *Computer Graphics Forum.* 2011

[5] Kornprobst, Pierre . "Limitations? - A Gentle Introduction to Bilateral Filtering and its Applications". 2007

[6] He, Kaiming & Sun, Jian & Tang, Xiaoou.. “Guided Image Filtering”. IEEE transactions on pattern analysis and machine intelligence. 2013