Edge-aware Bilateral Filtering

Reducing across-edge blurring for the bilateral filter

Glenn Weeland

Supervisor(s): Elmar Eisemann, Mathijs Molenaar

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Glenn Weeland
Final project course: CSE3000 Research Project
Thesis committee: Elmar Eisemann, Mathijs Molenaar, Jing Sun

Abstract

The bilateral filter is a popular filter in image processing and computer vision. This comes from the fact that it is able to blur images while keeping the structure intact. However, the bilateral filter allows for blurring to happen across edges. This can result in halo-like effects around the edges of structures if both sides are made up of different intensities. In this paper, we propose an extension to the bilateral filter that reduces this phenomenon of blurring across edges. By giving the filter knowledge of the edges beforehand, it is possible to prevent the filter from blurring past them. When we filter a pixel, its surrounding area within the kernel is checked for edges. If a pixel within this area lies on or beyond an edge, its weight for blurring is reduced. As a consequence, pixels that lie past an edge have less influence on blurring. We show that this new edge-aware bilateral filter reduces across-edge blurring compared to the standard bilateral filter. Furthermore, when we allow a bigger range of intensities to mix, the new filter is also able to prevent the filtered image from appearing washed out, unlike the bilateral filter.

1 Introduction

The bilateral filter is an edge-preserving image filter [15]. It is used extensively in computer graphics and computer vision for a multitude of applications such as denoising, creating cartoon renditions [17] and enhancing low-light photography [7]. The bilateral filter works by mixing pixels in a specific region around the target pixel with weights that depend on how close their intensities are to each other. The closer the intensity value of a neighboring pixel is to the target pixel, the higher its weight will be, and vice versa. This is then combined with a weight specified by the spatial distance to the target pixel to create bilateral filtering. This filter blurs the image while keeping the structures in the image mostly intact [15].

However, the simplicity of the bilateral filter comes with a caveat: pixels across edges can still be mixed by the filter, resulting in blurring across them. This happens because the bilateral filter only takes the spatial distance and the color difference between a neighbor and the target pixel into account. The structure of the image itself is not considered while filtering, allowing the bilateral filter to sample from pixels across edges, as long as their spatial distance and color difference are not too high. This phenomenon is demonstrated in Figure 1, where the resulting image after bilateral filtering is shown in Figure 1b. The filtered image contains halo effects around the edges of the black lines in between the grayscale planes in the background. This, in turn, creates a gradient-like area around these edges, instead of preserving the exact gray levels seen in the original image in Figure 1a. For these areas close to the edges, the pixels on both sides are mixed a little because the intensities and distances are close enough to influence each other while filtering. In some situations, this works in favor of the filter as it will be able to sample colors from more pixels, while in other situations like in Figure 1, it does not and produces undesired effects. Giving the bilateral filter knowledge of the edges in the image would help prevent these cases and may improve the performance of the filter in terms of conserving edges and colors on a local level.

In this paper, we propose extensions that combine the knowledge of the edges or segments in an image with the bilateral filter. These aim to prevent the previously discussed across-edge blurring of the filter. We use two types of edge detection to segment the image: one which produces hard binary edges and the other creating softer gradient-like edges. Interpolating between the results of both edge detectors allows us to create a combination of both. We then reduce the influence of pixels that lie beyond those edges for bilateral filtering. Depending on the chosen mix of the detected edges, the result can include harder, more defined edges or softer gradient-like edges. This new edge-aware bilateral filter keeps the edge-preserving smoothing and emphasizes it by preventing the filter from blurring beyond the edges. This results in better-defined edges and less washed-out colors.

To address the findings in this paper, first the related work is discussed, where more background information on the bilateral filter and its applications can be found. Next, the methodology to create the edge-aware bilateral filter is explained. Following that, the results of the experiments are presented and interpreted. The found conclusions about the edge-aware extensions to the bilateral filter are then discussed after. Finally, this paper ends with a section about responsible research and possible ideas for future work.

2 Related work

This section will briefly explain some of the background this paper builds upon, combined with related research, starting with the definition of the bilateral filter and how it works. After which a couple of relevant optimizations, extensions and use cases for it are explored. Lastly, various approaches for image segmentation and edge detection are discussed with their specific use cases and strengths and weaknesses.

2.1 The bilateral filter

The bilateral filter was first introduced by Aurich, et al. [3], which they called the ”nonlinear Gaussian filter”. It was later rediscovered by Tomasi and Manduchi [15] who called it the ”bilateral filter”, which it is still named, to this day. The filter works by taking a weighted average over the close neighbors of a pixel. This weight depends on two different kernels: the first is the distance between the target pixel and the neighbor, decreasing the further the neighbor lies from the target pixel (the spatial kernel). The second is the intensity difference between the target pixel and the neighboring pixel, which also decreases when the difference becomes higher (the range kernel). These filter functions are chosen as Gaussian functions, resulting in the following formulas for the bilateral filter:

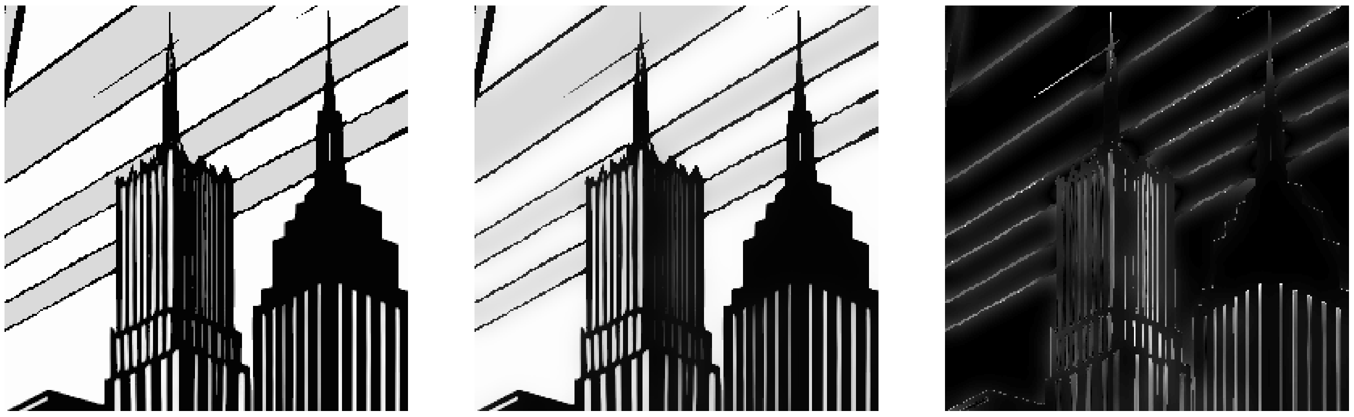

(a) (b) (c)

Figure 1: An example of across-edge blurring with the bilateral filter. (a) The original image. (b) The image after bilateral filtering with $\sigma_s = 10$ and $\sigma_r = 80$. (c) The absolute difference between (a) and (b) boosted $\times 3$.

$$I^{BF}(x) = \frac{1}{W_b} \sum_{p \, \epsilon \, \Omega} I(p) G_s(|x-p|) G_r(|I(x) - I(p)|) \quad (1)$$

Where W_b is the normalization factor:

$$W_b = \sum_{p \, \epsilon \, \Omega} G_s(|x-p|) G_r(|I(x) - I(p)|) \quad (2)$$

Here in both Equation 1 and Equation 2, G_s stands for a Gaussian spatial kernel with standard deviation s, G_r a Gaussian range kernel with standard deviation r, $I(\alpha)$ for the intensity of pixel α, Ω the kernel window centered on target pixel x and p for a pixel which lies within this kernel window.

Most of the work on improving the bilateral filter is related to runtime performance, as the filter is non-linear and thus difficult to optimize. Achieving faster runtime performance needs complex approximations via, for example, signal processing by Paris, et al. [11] or integral histograms by Porikili [13] to name a few, though the quality of the result can suffer depending on the type of approximation. Other research has focused on improving the bilateral filter for a specific use case, such as the joint or cross bilateral filter with a flash and no-flash photo in the works of Eisemann, et al. [7] and Petschnigg, et al. [12], texture filtering with conditional constraints [5], shorter runtime for displaying high dynamic range images [6] and many more that apply an altered or extended version of the bilateral filter.

Improvement of denoising with the bilateral filter has been reached by incorporating edge detection while filtering. Changing the spatial kernel of the filter depending on the orientation and anisotropy of the structures in the image as demonstrated by Venkatesh, et al. [16] has shown to improve PSNR metrics for denoising. Adding another kernel on top of the bilateral filter using the Roberts cross operator for edge preservation as suggested by Kaur, et al. [10] also has demonstrated to improve PSNR in denoising applications. There is also the guided filter proposed by He, et al. [9], which closely resembles the behavior of the bilateral filter and improves upon it in terms of runtime performance and preservation of edges and gradients.

2.2 Image segmentation and edge detection

Image segmentation and edge detection are very prevalent in modern computer vision applications. They help with identifying structures and can assist in recognizing specific objects in combination with other image filters.

Image segmentation attempts to find separate connected regions based on the structure of the image. One of the most popular and common methods to segment images is the SLIC algorithm proposed by Achanta et al. [1] as it is fast and allows to specify the amount of desired segments. This algorithm creates a grid of points on the image and then finds the connected regions by expanding the pixels into connected regions with k-means clustering. The resulting segmentation adheres to edges, but it also oversegments the image, as all original points in the grid create a separate segment centered on themselves. Felzenszwalbs graph-based approach for segmentation [8] prevents these unnecessary segments and creates less regular-sized segments. However, this comes at the cost of not being able to specify the exact amount of segments.

Unlike image segmentation, however, edge detection does not have to find separate connected regions out of the edges. A simple approach for finding edges is approximating the local gradient. The Sobel operator by Sobel et al. [14] is a popular method for this and approximates the gradient in both the vertical and the horizontal directions. The magnitude of these can then be calculated resulting in gradient-like edges with varying intensities. To find the most important edges in the image, the Canny edge detector [4] by Canny identifies the local maxima in the Sobel edge detection. It then removes unwanted edges depending on multiple thresholds, creating thin binary edges. Detecting edges can also be done with a physics-based approach. The Phase stretch transform [2] proposed by Asghari et al. which emulates light, can detect edges of different strengths similar to the Sobel operator. It is also able to include smaller details, where the gradient approximations fail to create significantly intense edges. Furthermore, the resulting edges can be thresholded to create binary edges just like the Canny edge detector.

3 Methodology and experiments

In this paper, we propose extensions to the bilateral filter which combine it with edge detection and segmentation to reduce across-edge blurring. This section describes the setup of the conducted experiments and the reasoning behind them. The version of the bilateral filter that we extend in this paper is the one proposed by Tomasi and Manduchi [15] defined by Equation 1 and Equation 2, which uses a Gaussian spatial and a Gaussian range kernel.

3.1 Range kernel based segmentation

Giving the bilateral filter the knowledge of the edges in the to-be-filtered image can be done in a multitude of ways. A simple approach to stop the filter from blurring across edges would be to find the surrounding connected segment for every single pixel. This method adds an extra kernel to the bilateral filter: one that specifies if pixels lie in the segment of the target pixel. If a pixel is not located in the connected region of the target pixel, it is not included for mixing intensities, even if they are located close to each other and have similar intensity values. To create this segment, a threshold value based on the amount of standard deviations in the range kernel is chosen. Every intensity value that lies beyond this threshold value, is seen as a border for the unique segment of the target pixel. The segment is then found by searching every directly reachable pixel from the target pixel which has an intensity that differs less from the target intensity than the chosen threshold. This method makes sure that pixels with intensity values close to the target, but lie beyond a detected border, are not included for filtering that specific target pixel. The segmentation can be included in the filter by an inclusion kernel as in the following formulas:

$$I^{RSBF}(x) = \frac{1}{W_{rsbf}} \sum_{p \in \Omega} I(p) G_s(|x-p|) \\ G_r(|I(x)-I(p)|) C_r(x,\ p,\ t) \tag{3}$$

$$W_{rsbf} = \sum_{p \in \Omega} G_s(|x-p|) G_r(|I(x)-I(p)|) \\ C_r(x,\ p,\ t) \tag{4}$$

$$C_r(x,\ p,\ t) = \begin{cases} 1 & if \quad p \in Connected(x,\ t \cdot \sigma_r) \\ 0 & else \end{cases} \tag{5}$$

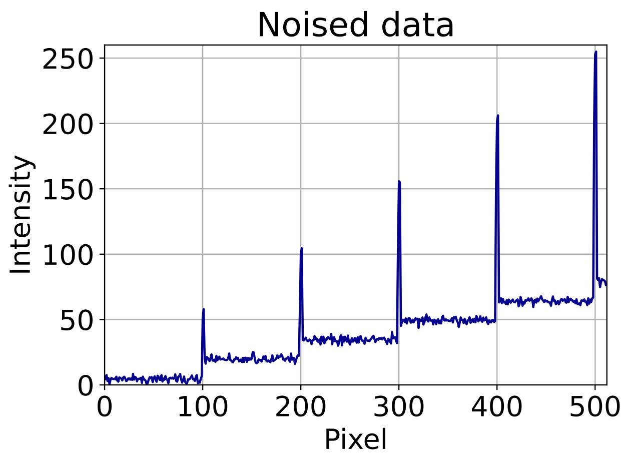
In these three equations for the range kernel segmented bilateral filter, all the previous variables from Equation 1 and Equation 2 are equivalent. The only changed part is the inclusion kernel: $C_r(x,\ p,\ t)$, which as specified in Equation 5, takes three parameters: x, the target pixel, p, the neighboring pixel and t, the amount of allowed standard deviations before thresholding. Furthermore, the $Connected(x,\ d)$ function creates a set of all pixels that are directly reachable from target pixel x, without crossing any pixels with an intensity difference higher than d compared to x. This kernel returns 1 if the intensity difference is lower than the specified threshold and is located within the connected segment, otherwise, it returns 0. This new inclusion kernel is then combined with the bilateral filter as seen in Equation 3 and Equation 4, where it decides if pixels are included for filtering.

A comparison of this thresholded bilateral filter with the standard one can be seen in Figure 2. Both the filters are able to reduce or even remove the noise from the one-dimensional data in Figure 2a. However, there are multiple visible differences between both filters which can be seen in the regions around the peaks at every one hundred pixels in Figure 2b. Here the bilateral filter samples from pixels on both sides of the peaks, creating gradients that approach the intensity levels beyond the peaks. The thresholded bilateral filter is not able to sample from both sides as this is prevented by the inclusion kernel. Another difference is that the bilateral filter attempts to smooth out the intensity of the peaks to their surroundings. The range segmented filter is not allowed to do that and preserves the original levels.

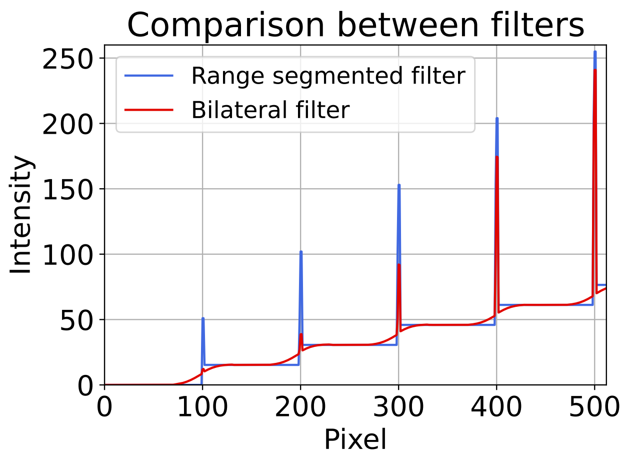
How well the structure of the original data is preserved, depends on threshold parameter t in Equation 5. A low threshold will result in smaller segments that have similar intensity values. This creates very localized blurring, which possibly retains noise from the original data. In the case that the threshold is set to 0, the image would only be blurred with pixels of the exact same intensity, resulting in the exact same image as the original. Increasing this threshold allows the filter to sample from more different pixels and therefore increases overall blur. This converges to the same result as the standard bilateral filter when $t \geq 3$ in $t \cdot \sigma_r$, because a threshold after three standard deviations in a Gaussian curve includes more than 99% of the possible values. Changing the threshold parameter allows to partially apply the bilateral filter: smoothing more with high values and retaining more structure with low values. An appropriate threshold can then be chosen to create the desired result.

A problem with this approach is that far outliers, in terms of intensity values, are treated as separate segments, which would result in no change after filtering. Forcing segments to have a minimum amount of pixels makes sure that outliers still mix with adjacent pixels even though the intensity difference is bigger than the threshold. If a segment does not reach the minimum size requirement, we expand the segment by also including all pixels directly adjacent to it. This process is then repeated until the segment is at least as big as the desired size that would reduce the outliers.

In Figure 3, the behavior of the filter on an image with outliers is shown. Without a minimum segment size, the filter is not able to smooth out the noise as seen in Figure 3b, because of the reasons discussed earlier. A forced minimum segment size results in the outliers being filtered to an intensity closer to their surroundings as demonstrated in Figure 3c. Pixels that do meet the minimum size requirement do not mix with the outliers, as their segments are not altered.

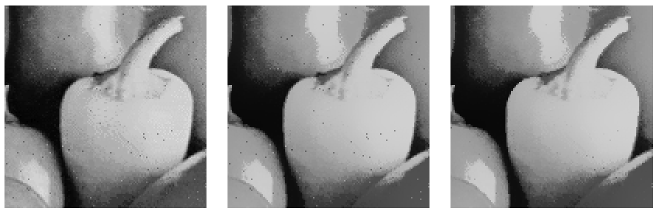


(a)



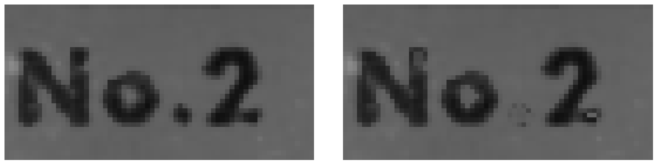
(b)

Figure 2: (a) One dimensional data of multiple increasing intensity levels with increasing peaks in between them. Gaussian noise has been added with a standard deviation of 2. (b) The data from the left figure filtered with the bilateral filter (red) and the thresholded bilateral filter (blue). For both filters the parameters are: $\sigma_s = 15$, $\sigma_r = 50$ and $t = 0.5$.



(a) (b) (c)

Figure 3: Filtering outliers with the range kernel segmented bilateral filter. (a) Image with 0.5% salt and peppers noise. (b) Image filtered with the range kernel segmented filter. (c) The same filter, but with a minimum segment size of 15 pixels. The parameters of the filter are: $\sigma_s = 10$, $\sigma_r = 50$ and $t = 0.5$.



(a) (b)

Figure 4: (a) Filtering with no minimum size. (b) Details are filtered with their surroundings as the minimum segment size (15 pixels) is too high.

While solving the outlier issue, this approach introduces another problem: the chosen minimum size needs to be big enough to reduce outliers, but also not too big, as it would then start blurring small details as seen in Figure 4. For some images, no minimum segment size needs to be specified, as there are no outliers in the image. Others will have outliers, but no small details that need to be conserved, so an appropriate minimum segment size can be chosen to filter them appropriately. Images that have both details that need to be conserved and outliers that need to be reduced, do not work well with this approach of reducing across-edge blurring of the bilateral filter.

3.2 Global image segmentation

Finding a unique segment per individual pixel is not the only method for adding the knowledge of edges or segments to the bilateral filter. Performing edge detection or segmentation of the image beforehand can also yield the desired behavior of the bilateral filter not blurring across edges. Segmentation algorithms find certain connected regions depending on the implementation, which then can be used by the bilateral filter to only mix pixels from the same connected region as the target pixel. This is unlike the thresholded filter discussed earlier, where pixels were allowed to mix with pixels that had different unique segments. Another difference is that segmentation algorithms do not produce overlapping regions, which the previous method did allow.

$$C(x,p) = \begin{cases} 1 & if \quad S(x) = S(p) \\ 0 & else \end{cases} \tag{6}$$

Again, for this approach, we add an additional kernel to the standard bilateral filter. This kernel as seen in Equation 6 takes the target pixel x and another pixel p from within the kernel window. The $S(\alpha)$ function takes a pixel α and returns the segment in the image it belongs to. If both segments match, 1 is returned, otherwise it returns 0. Now any segmentation algorithm works with our new filter.

A popular approach to segment images for computer vision applications is using superpixels to create connected regions in an image [1, 8]. This creates segments that have similar intensity values. The downside is that they also create a lot of extra edges in the segmentation that do not exist in the original image. This is a consequence of segments not being allowed to overlap and not being centered on every pixel, unlike the range kernel based segmentation discussed before. Using this global segmentation for filtering then creates visible boundaries in the resulting image, which are not present with the standard bilateral filter. Other global segmentation approaches suffer from the same problem, where they create segments that do not exist in the original image. An example of using the k-means based SLIC algorithm [1] for the new filter is demonstrated in Figure 5. The borders of the resulting segmentation are depicted in Figure 5a. This then creates artifacts after using the new filter as shown in the background of Figure 5b.

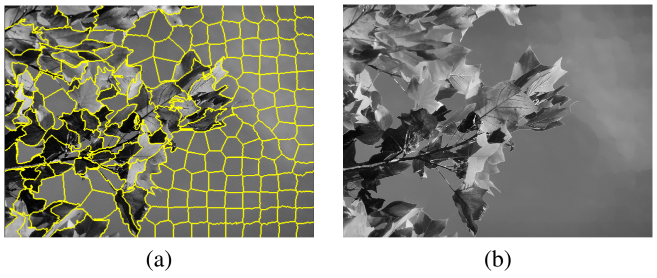


Figure 5: (a) Image segmentation using SLIC, boundaries of the found segments are colored yellow. (b) The SLIC segmentation combined with the proposed filter. The chosen amount of segments is 250.

Another part where segmentation algorithms fail is including smaller details. If the parameters are set to creating smaller segments, even more segments start appearing, where there should be none. The other way around is also not satisfactory, as opting for bigger segments reduces the amount of oversegmentation, but in turn, removes a lot of the details. Both situations for Felzenszwalb segmentation [8], are shown in Figure 6.

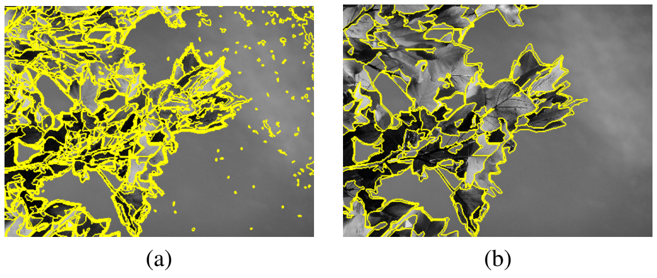


Figure 6: Felzenszwalb segmentation with the boundaries of the found segments colored yellow. (a) Low scale parameter, resulting in a lot of small and extra segments. (b) High scale parameter, producing bigger but less detailed segments.

Another problem for global segmentation as a whole is that the borders in image segmentation are all treated equally resulting in harsh edges, while softer edges might be more preferable. Depending on how the image is segmented, this increases the staircasing effect of the standard bilateral filter, as pixels can only sample from their own segment, averaging the segments' colors. This increased staircasing effect with oversegmentation is demonstrated in Figure 5b.

3.3 Edge detection

The final method for preventing across-edge blurring in this paper uses edge detection, which combines the bigger segments from the global segmentation approach with the less defined and overlapping boundaries of the range kernel based segmentation. Edge detection does not have to create fully separate regions like the segmentation algorithms. It also allows for more and less defined edges in the segmentation, which enables the possibility of blurring along the edges, smoothing them in the resulting image. This reduces possible artifacts from hard edges as seen in Figure 5b.

A simple approach for edge detection is the Sobel operator, also known as the Sobel filter [14]. This filter approximates the local gradient in the image, producing smooth, soft edges. Applying this Sobel filter to an image results in another image where the intensity of a pixel corresponds to its gradient as depicted in Figure 7. This type of edge detection is able to generate much more edges than segmentation algorithms, while not producing any extra edges. But just like the global segmentation it has difficulties with detecting small details in the image, as the faces of the people seen in Figure 7a are hardly visible in Figure 7b.

To amplify the most important edges of the Sobel filter, local maxima need to be found and the rest of the output needs to be reduced. This is exactly what the widely used Canny edge detector does [4]. It suppresses and thresholds the gradients into thin lines and turns them into binary edges. These edges are similar to the type of edges seen in Figure 6, but missing the extra undesired edges while including more details as seen in Figure 7c. This approach still has the same problem as global segmentation, in the sense that segments are defined by binary edges. Another pitfall of this approach is that the edges often contain small gaps, which produces bigger segments than would seem at first glance. This is demonstrated in the zoomed-in part of Figure 7c where the connected region is shown. Finally, the Canny edge detection uses the Sobel operator under the hood, which results in it missing the same small details in the edge detection. A different way of finding hard binary edges, that does not rely on gradient approximation, can be done with the Phase stretch transform [2] which is shown in Figure 7d. The edges it produces are thicker and capture more details than those of the Canny edge detector, but this approach also has problems with creating a lot of gaps in the found edges.

For filtering an image with more defined edges, like the comic in Figure 7a, hard edges are preferred to preserve the structure. Images with more gradients like the pepper in Figure 3, would need softer edges to reduce the stair casing effect. Images with both defined and gradient-like edges need something in between. Using this information, we can add the following extra kernel defined by Equation 7 to the standard bilateral filter:

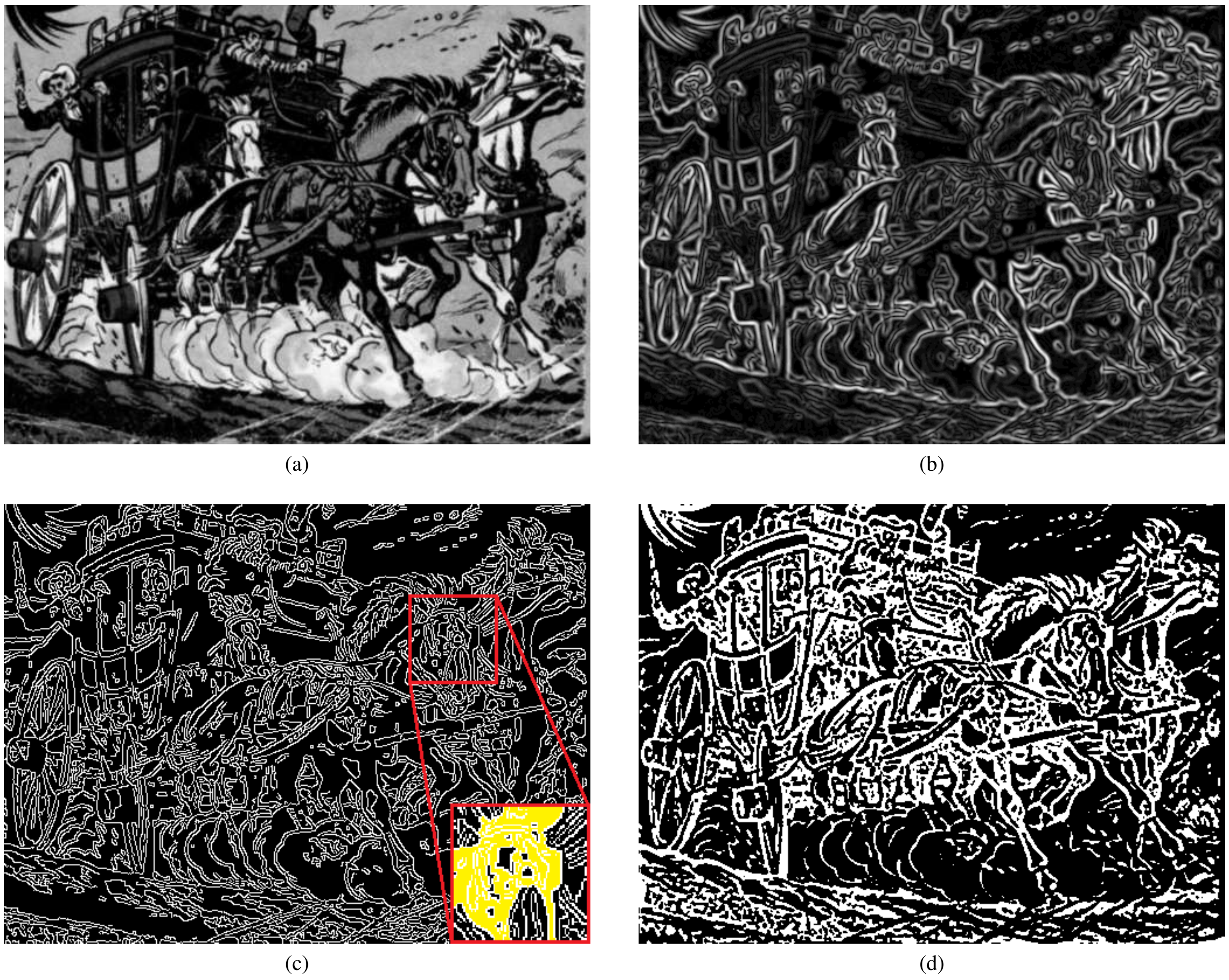


Figure 7: Multiple edge detection methods on a scan of a comic. (a) The original image. (b) The output of the Sobel operator where the result is normalized to values in between 0 and 1. (c) Canny edge detection with a lower threshold of 100 and a higher threshold of 200. A part of the image is zoomed in where a connected region is colored yellow. (d) The phase stretch transform with a phase strength of 3.

$$C(x,p,e) = \begin{cases} 1 - Cost(x,p,e) & if \quad Cost(x,p,e) \ < \ 1 \\ 0 & if \quad Cost(x,p,e) \to too\ far \\ 0 & else \end{cases} \tag{7}$$

This new inclusion kernel takes the target pixel x, a neighboring pixel p that lies within the specified kernel window, and e, the edge detection image. The $Cost(x,p,e)$ function returns the minimal cumulative amount of traversed changes in intensity on the edges image to reach pixel p from the target pixel x. For example: the target pixel has an edge intensity of 0.2. To reach p, intensities 0.3 and 0.6 have to be crossed where p also has an edge intensity of 0.2. The Cost function will then return the cumulative change: $(|0.2-0.3|)+(|0.3-0.6|)+(|0.6-0.2|) = 0.8$ setting the inclusion weight to 0.2. If pixel p is reachable from target pixel x by traversing less total change in edge intensity, then that lower cost will be the new return value of the cost function for pixel p. If the path to a pixel takes more steps than the diameter of the kernel window, the pixel is considered too far to reach and is set to 0. Applying this extended bilateral filter with the Sobel edge detection is demonstrated in Figure 8.

This kernel window in Figure 8b shows that the approach of including certain amounts of edge gradients prevents the across-edge blurring without introducing a complete cutoff at the edges in the image like binary edge detection or global segmentation. It also creates the smooth edges that the range kernel-based segmentation would create. Now we combine the soft edges of the Sobel operator for smoothing and the hard edges from the Phase stretch transform to preserve details. We do this by linearly interpolating in between both edge detectors, the ratio between them being the desired smoothness. A smoothness of 0 will result in the hard edges of the Phase stretch transform, values between 0 and 1 create interpolations between both, until a smoothness of 1, where we get the soft edges of the Sobel operator.

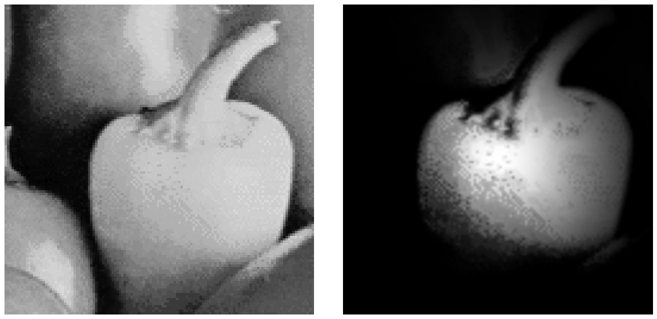

(a) (b)

Figure 8: Demonstration of the edge-aware bilateral filter kernel with Sobel edge detection. (a) The original image. (b) The kernel window of weights of the edge-aware bilateral filter applied on the center pixel of (a) with $\sigma_s = 30$ and $\sigma_r = 30$.

4 Results and Discussion

We test the new edge-aware bilateral filter on two types of images: one is a scan of a comic demonstrated in Figure 9, which contains hard edges with well-defined boundaries. The other image is the full picture of the peppers, which has more soft and gradient-like edges shown in Figure 10. Filtering the comic with the bilateral filter as seen in Figure 9b removes the texture from the ground and smooths the details in the background. It also includes a lot of halo effects around the edges, which makes the image appear washed out. The edge-aware bilateral filter reduces these halo effects for all smoothness values. A smoothness value of 0 in Figure 9c creates grainy edges after filtering and mostly preserves the original intensities from Figure 9a. By increasing the smoothness parameter, the edges gradually become softer and more details get mixed with their surroundings. With higher smoothness values, segments with similar colors start blending more. A maximum smoothness of 1 as seen in Figure 9f, produces almost the same amount of blur between similar intensity levels as the bilateral filter, but still removes most of the halo effects.

Similar effects can be seen for the peppers in Figure 10. Low smoothness values create hard edges, which make the image appear cartoonish in Figure 10c. These hard edges become less visible with higher smoothness values as shown in both Figure 10d and Figure 10e. While most of the across-edge blurring is reduced by the edge-aware filter, the stem of the left pepper with a smoothness of 0 in Figure 10c has a halo around it. This effect then gradually disappears with higher smoothness values.

The proposed edge-aware filter does show promising results. It is able to reduce the across-edge blurring of the standard bilateral filter and is better at preserving the intensities of the original image with low smoothness values. The binary edges generated with a smoothness of 0, create hard borders in the resulting image, this can be beneficial if the original image only contains these types of edges, but it does not look that good for most images. Choosing low smoothness values still creates well-defined edges and blurs them slightly to make them smoother. This works great for images with hard edges like the comic as seen in Figure 9d, where intensities are preserved and the edges are still smooth. Higher smoothness values work better with images containing softer edges as the peppers in Figure 10. The gradients are preserved better and no extra borders like those in Figure 10c are created.

All of this does not mean the edge-aware bilateral filter works for any use case. The edge detection can miss details that might need to be preserved. With low smoothness parameters, the binary edge detection decides most of the boundaries of the filter. Both the Phase stretch transform and the canny edge detector sometimes leave gaps in between the edges, allowing the filter to still blur across an edge. If the amount of gaps is large enough, the halo effect from the normal bilateral filter appears like in Figure 10c. Furthermore, the edge-aware filter is very slow in terms of runtime, taking more than a minute to filter an image of 512x512 pixels. This makes the filter less practical for real-time use cases and filtering images with higher pixel counts.

5 Conclusions

In this paper, we have presented an extension to the bilateral filter, which is able to reduce the phenomenon of across-edge blurring. This extension takes the structure of an image into account while filtering to prevent mixing pixels across edges. It adds an extra kernel to the standard bilateral filter using two edge detection algorithms, where one produces smooth gradient-like edges and the other hard binary edges. We choose how smooth we want the edges to be after blurring by interpolating between the two types of edge detection. The additional kernel then reduces blurring across the chosen edges. We show that this new edge-aware bilateral filter is indeed able to reduce the across-edge blurring of the standard bilateral filter and is able to smooth the boundaries where blurring is prevented. An additional effect of the new filter is its ability to also preserve the edges themselves, preventing washed-out-looking results.

6 Future work

In this paper we only explore the edge-aware bilateral filter for grayscale images, but it can easily be expanded to color images as well. The proposed edge-aware bilateral filter is also quite computationally expensive, reducing this complexity with better approaches to the edge detection and the cost function are potential research problems. Furthermore, a better method for choosing softer or harder edges would improve the performance of the filter and could make it more intuitive to use.

7 Responsible Research

Ethics

All the images used in this paper were either made specifically for this research or fall under the public domain to not violate intellectual property rights. Furthermore, to keep transparency about the conducted research, all the source code containing the described algorithms and the visualization methods are made publicly available online[1].

Reproducibility

To make the results as reproducible as possible, all the implementations of the edge-aware bilateral filters are described in detail. Parameters for shown images are included in their description to make them easier to reproduce. Additionally, the source code is available as discussed in the section above. The code makes use of OpenCV-4.9.0 library for filtering images which also hosts the standard bilateral filter implementation that was used for comparisons. Both the code and openCV were compiled with C++ version 23 and the MSVC compiler version 19.39.33523 for x64 architecture on the Windows 10 operating system.

[1]https://gitlab.ewi.tudelft.nl/cse3000/2023-2024-q4/Eisemann_Molenaar/gweeland-Image-Processing-with-the-Bilateral-Filter

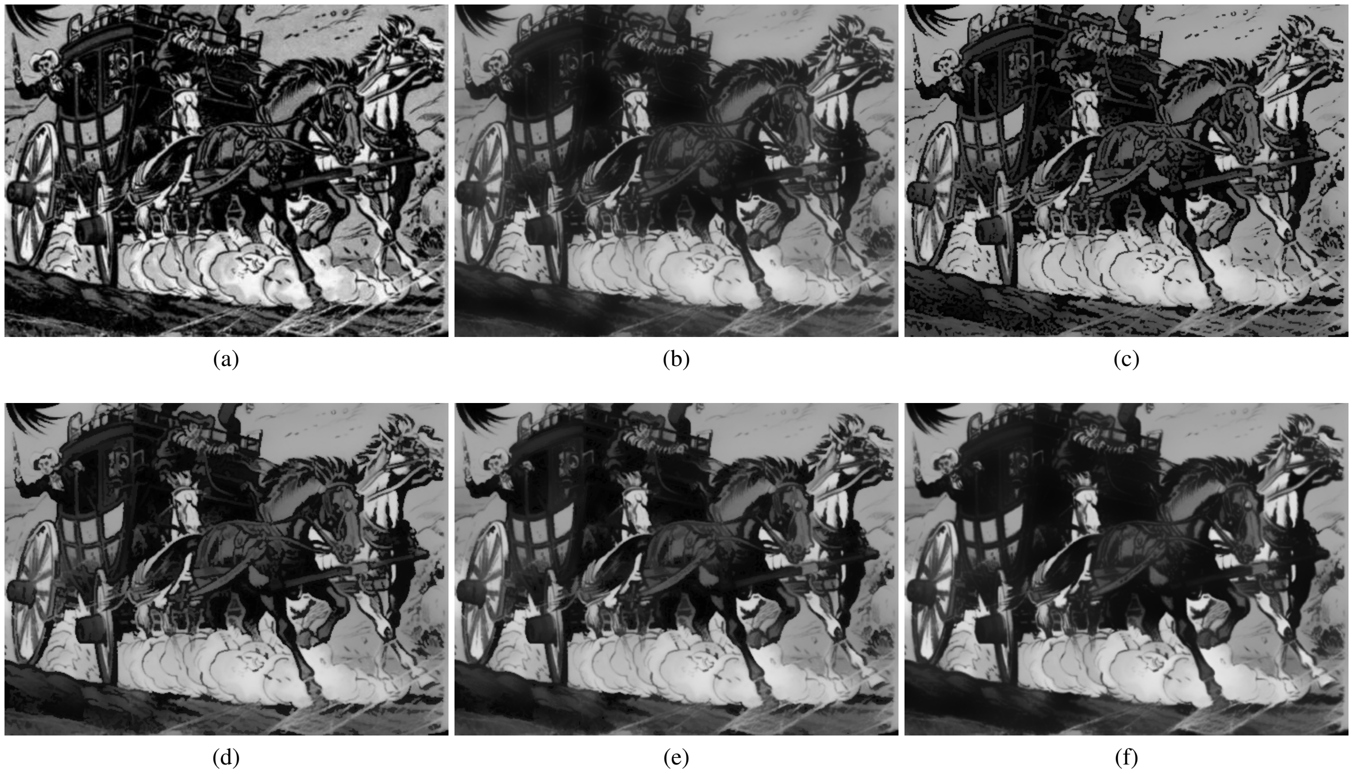


Figure 9: Results of applying the edge-aware bilateral filter on a scan of a comic compared to the standard bilateral filter and the original image. (a) The original image. (b) The bilateral filter. (c) A smoothness of 0. (d) A smoothness of 0.2. (e) A smoothness of 0.5. (f) A smoothness of 1. All filters have the parameter values: $\sigma_s = 15$ and $\sigma_r = 70$. The edge-aware filter uses the Phase stretch transform for hard edges and the Sobel operator for soft edges.

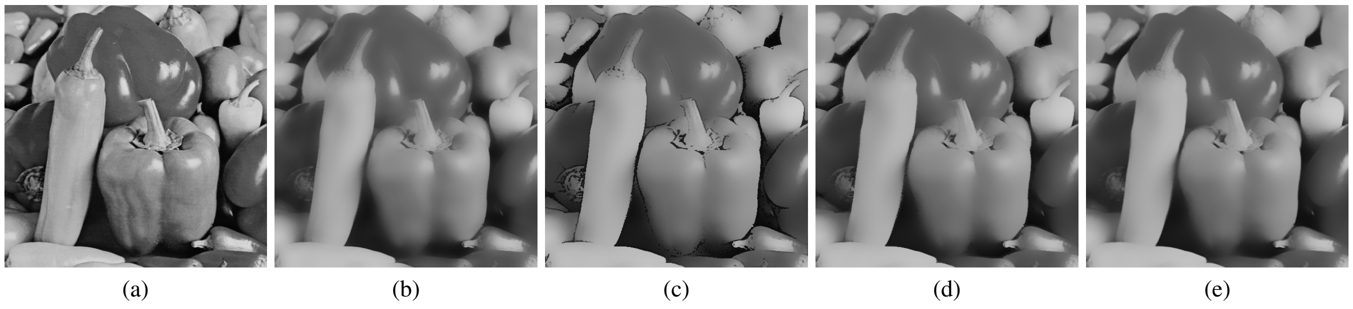


Figure 10: Results of applying the edge-aware bilateral filter on an image of peppers compared to the standard bilateral filter and the original image. (a) The original image. (b) The bilateral filter. (c) A smoothness of 0. (d) A smoothness of 0.5. (e) A smoothness of 1. All filters have the parameter values: $\sigma_s = 15$ and $\sigma_r = 70$. The edge-aware filter uses the Phase stretch transform for hard edges and the Sobel operator for soft edges.

Integrity

The images used for demonstrating the edge-aware bilateral filter contain different types of edges and gradients. This is done to show the behaviour of the filters with different types of inputs and create less biased results. Furthermore, both the strengths and weaknesses of all the demonstrated filters are discussed. We do this to ensure that all these filters are fairly evaluated compared to each other.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274 – 2281, 2012.

[2] Mohammad H. Asghari and Bahrain Jalali. Physics-inspired image edge detection. In *2014 IEEE Global Conference on*

Signal and Information Processing (GlobalSIP), pages 293–296, 2014.

[3] Volker Aurich and Jörg Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995, 17. DAGM-Symposium, Bielefeld, 13.-15. September 1995, Proceedings*, Informatik Aktuell, pages 538–545, 1995.

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679 – 698, 1986.

[5] W. Cao, S. Wu, J. Wu, Z. Liu, and Y. Li. Edge/structure-preserving texture filter via relative bilateral filtering with a conditional constraint. *IEEE Signal Processing Letters*, 28:1535–1539, 2021.

[6] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *ACM Transactions on Graphics*, volume 21, pages 257–266. Association for Computing Machinery (ACM), 2002.

[7] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM transactions on graphics*, 23(3):673–678, 8 2004.

[8] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

[9] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2013.

[10] A. Kaur, H. Singh, and D. Arora. An efficient approach for image denoising based on edge-aware bilateral filter. In *4th IEEE International Conference on Signal Processing, Computing and Control, ISPCC 2017*, volume 2017-January, pages 56–61. Institute of Electrical and Electronics Engineers Inc., 2017.

[11] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81(1):24–52, 2009.

[12] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics (TOG)*, 23:664–672, 08 2004.

[13] Fatih Porikli. Constant time o(1) bilateral filtering. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[14] Irwin Sobel and Gary Feldman. A 3×3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, pages 271–272, 01 1973.

[15] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998.

[16] M. Venkatesh and C. S. Seelamantula. Directional bilateral filters. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 2015-August, pages 1578–1582. Institute of Electrical and Electronics Engineers Inc., 2015.

[17] H. Winnemöller, S. Olsen, and B. Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, 2006.