Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021-23185

Edge-aware filter based on adaptive patch variance weighted average

FERNANDO J. GALETTO[1], GUANG DENG[1], MUKHALAD AL–NASRAWI[2], and WASEEM WAHEED[1].

[1]Department of Engineering, La Trobe University, Melbourne, VIC 3086, Australia.
[2]Al-Furat Al-Awsat Technical University, Technical College of Al-Mussaib, Iraq.

Corresponding author: Fernando J. Galetto (e-mail: f.galetto@latrobe.edu.au).

ABSTRACT Edge-aware smoothing is an essential tool for computer vision, graphics and photography. In this paper, we develop a new and efficient local weighted average filter for edge-aware smoothing. The proposed filter can use guidance information which permits an iterative filtering process. Since the weights of the proposed filter depend on the local variance, the implementation requires linear filters only, leading to $\mathcal{O}(N_{pix})$ computational complexity. We also present statistical analysis and simulations which provides new insights into its computational efficiency and its relationship with the bilateral filter. The performance of the proposed filter is comparable to those state-of-the-art filters in many applications including: edge-preserving smoothing, compression artifact removal, structure separation, edge extraction, non-photo realistic image rendering, salience detection, detail magnification and multi-focus image fusion.

INDEX TERMS Edge-aware smoothing, Bilateral filter, Guided filter, Detail magnification, Multi-focus image fusion.

I. INTRODUCTION

Filters which smooth image details are essential tools for many low-level vision applications [1]. Such filters have been historically developed based on linear time-invariant (LTI) systems [2]. LTI filters, such as the widely used Gaussian filter, do not consider edges or boundaries of objects. As a result of this obliviousness to edges, LTI filters produce artifacts such as halo and blurriness. To tackle this problem, edge-aware filters (EAF) [3] which are based on non-linear techniques have emerged as powerful tools for a wide range of applications in image processing and computer vision [4]–[10]. Typical applications include: texture smoothing [7], [9]–[12], non-photo-realistic rendering [4], [6], [9], clip-art compression artifact removal [6], [11], flash/no-flash image denoising [8], detail magnification [6], [11], [13], edge extraction [6], [7], [14], tone mapping [4], [6], multi-focus image fusion [15], [16], salient object detection, texture enhancement [7], and image vectorization [7]. Since there are many publications on EAF, we will adopt and enhance some of the previously suggested categorizations in the literature [17]. More specifically, EAF can be classified into four categories: local, global, transform domain, and data-driven.

Local filters model an output pixel as a weighted average of the surrounding pixels as follows:

$$J(m) = \frac{1}{\sum w(n,m)} \sum_{n\in\Omega_m} w(n,m)I(n) \tag{1}$$

where $w(n,m)$ is a weight function measuring the similarity between two pixels at locations n and m, $I(n)$ is the pixel intensity at the location n of the input image I, and Ω_m denotes the set of pixels in the neighbourhood of pixel $I(m)$. Generally speaking, local filters differ in the way their weight functions are defined. Many filters fall in this category, including the bilateral filter (BF) [2], the guided filter (GF) [8] and its many variants [18]–[23], and the Bayesian model averaging filter (BMA) [24]. Some local filters such as the guided filter are popular due to their computational efficiency [17]. On the other hand, other local filters such as the bilateral filter are computationally expensive due to the non-linearity in the weight.

Global filters are usually the product of minimizing an image level objective function of the following form:

$$J = \arg\min_{J} \mathcal{D}(I,J) + \mathcal{R}(J) \tag{2}$$

where $\mathcal{D}(I,J)$ is a measure of the distance between the input image I and the output image J, and $\mathcal{R}(J)$ is the regularizer that encodes the filter designer's knowledge about the expected output image J. Global filters can be formulated through either a variational model or a Bayesian estimation model. The optimization problem can be convex or non-convex, continuous or discrete. Example of global filters in the literature include the weighted-least squares filter (WLS) [4], L_0 filter [6], relative total variation (RTV) filter [7], the region-covariance filter (RC) [25], the static-dynamic filter [26], and the iterative global optimization filter (IGO) [27].

Transform domain filters can be summarized in the following general model:

$$J = \mathcal{T}^{-1}\left\{f(\mathcal{T}\{I\})\right\} \tag{3}$$

where $\mathcal{T}$ and $\mathcal{T}^{-1}$ are the forward transform and inverse transform operations, and $f : \mathbf{R} \to \mathbf{R}$ is usually a point-wise nonlinear function. Filters in this category include the classic Wiener filter [3], edge-avoiding wavelets [28], guided wavelet shrinkage [29], mixed-domain filter [30], and the domain transform filter [5]. A related type of filters are based on multi-resolution techniques. Examples of such filters include the local-Laplacian filter [31] and mixed-Domain filter [30].

A more recent attempt has been towards taking a machine learning approach [32] in which the image filtering process is formulated as a parametric model in the form of a deep neural network. Parameters of the model are learned from a huge data set of image patches. The filtering techniques in this group can generally be sub-categorized into two groups: learned priors [33], [34] and end-to-end learned models [35], [36].

This work is partly motivated by recent developments in edge-aware filtering, especially those based on the idea of guided filter [8]. An intriguing question is: can we develop a new local weighted average filter in the form of equation (1)? The new filter should retain the computational efficiency of the guided filter, and should avoid the computational complexity of the bilateral filter. The main contributions of this paper summarized below aim to answer this question.

- We develop a new statistically motivated local weighted average filter. The intuition behind the proposed filter is that a larger weight should be given to a pixel in a flat area, while a smaller weight should be given to a pixel in an edge or highly textured area. The variance of the patch can be used to measure the flatness and the weight is thus defined as a decreasing function of the patch variance. We then adapt this filter to use the bilateral weight, guidance image [37], and the idea of rolling guidance [10].
- We show that the proposed filter is related to the bilateral filter [2] in that the range weight is calculated by the patch variance. The patch variance is used to measure the similarity between two pixels.
- We further show that the proposed filter not only retains the same $\mathcal{O}(N_{pix})$ computational complexity as that of the guided filter, but also produces comparable or better results in a wide range of applications where edge-aware filters are required.

The organization of this paper is as follows. We first review the basic idea of the guided filter in Section II, which provides a foundation for the development of the proposed filter. In Section III, we present the theoretical development of the proposed filter as well as a detailed discussion on its properties such as the computational complexity and its relationship with the bilateral filter. In Section IV, we present examples of typical applications of the proposed filter including: texture-structure separation, detail magnification, multi-focus image fusion, edge-detection, compression artifact removal, and salient object detection. We also compare the performance of the proposed filter with related state-of-the-art filters [4]–[11], [18], [26], [27]. In Section V, we present a summary of the main idea of the proposed filter and its applications.

II. THE GUIDED FILTER AND ITS WEIGHTED VERSION

We briefly review the main idea of the guided filter and its weighted version in this section. We use the following notation. The image to be processed and the guidance image are represented as I and G, respectively. An image patch consists of pixels from a square neighborhood. The set of location indices for pixels in the patch centered at location k is denoted Ω_k. The number of pixels in the patch is $N = |\Omega_k|$. A pixel in the patch is denoted $I(n_k)$ where $n_k \in \Omega_k$ is the location index within the patch. We also define the following mean-notation:

$$\mu_I(k) = \frac{1}{N}\sum_{n_k=1}^{N} I(n_k) \tag{4}$$

Similarly, we define $\mu_G(k)$ as the corresponding patch mean for image G, $\mu_{GI}(k)$ as the patch mean for the pixel-wise product image $G(n_k)\times I(n_k)$, and $\mu_{GG}(k)$ as the patch mean of the pixel-wise square of image $G(n_k) \times G(n_k)$.

Using these notations, we can define the patch variance as

$$\sigma_G^2(k) = \mu_{GG}(k) - \mu_G^2(k) \tag{5}$$

and define the patch covariance as

$$cov_{GI}(k) = \mu_{GI}(k) - \mu_G(k)\mu_I(k) \tag{6}$$

There are two main ideas in the original guided filter: patch modelling and model averaging. In patch modelling, a linear model is assumed for pixels in the patch

$$J(n_k) = a(k)G(n_k) + b(k) \tag{7}$$

The two model parameters $a(k)$ and $b(k)$ are determined through a regularized optimization

$$\{a(k), b(k)\} = \min_{\{a(k),b(k)\}} \sum_{n_k=1}^{N} \left(I(n_k) - J(n_k)\right)^2 + \epsilon a^2(k) \tag{8}$$

It can be shown that the results are

$$a(k) = \frac{cov_{GI}(k)}{\sigma_G^2(k) + \epsilon/N} \tag{9}$$

and

$$b(k) = \mu_I(k) - a(k)\mu_G(k) \tag{10}$$

Once the model parameters are determined, we can use the model to generate the filter result. The key point is that a pixel $I(m)$ belongs to N patches. For example, let $m \in \Omega_k$. The kth patch model generates a result

$$J_k(m) = a(k)G(m) + b(k) \tag{11}$$
$$= \mu_I(k) + a(k)G(m) - a(k)\mu_G(k) \tag{12}$$

Therefore, there are N results due to N patch models. Let Ω_m denote the set of patch indices such that the pixel $I(m)$ is in the patch $k \in \Omega_m$. Model averaging is a principled technique for combining these results. The original guided filter takes the simplest form of model averaging by taking an average of the results

$$J(m) = \frac{1}{N}\sum_{k=1}^{N} J_k(m) \tag{13}$$
$$= \bar{a}G(m) + \bar{b} \tag{14}$$

where $k \in \Omega_m$, $\bar{a}$ and $\bar{b}$ are the average of $a(k)$ and $b(k)$ over N patch models.

There have been several recent attempts to improve the performance of the original guided filter. The main idea behind these attempts is to replace the simple average operation by the weighted average in $k \in \Omega_m$ [18]:

$$J(m) = \frac{1}{\sum_{k=1}^{N} w(k)} \sum_{k=1}^{N} w(k) J_k(m) \tag{15}$$

One proposal is to define the weight as a function of the patch variance

$$w(k) = f(\sigma_G(k)/\sigma_r) \tag{16}$$

where σ_r is a user defined scale parameter and $f(x)$ is a decreasing function of $|x|$.

III. THE PROPOSED FILTER

A. THE BASIC IDEA

We consider the case in which ϵ (in equation (8)) is set to a huge number such that the patch parameter a_k can be considered as almost the same value for all patches. Let us denote this value α. We can thus write the weighted guided filter output as

$$J(m) = \frac{1}{\sum_{k=1}^{N} w(k)} \sum_{k=1}^{N} w(k) J_k(m) \tag{17}$$
$$= \tau_I(m) + \alpha(G(m) - \tau_G(m)) \tag{18}$$

where $\tau_I(m)$ and $\tau_G(m)$ are weighted average of N pixels in the patches centered at location of m for the mean images μ_I and μ_G as follows:

$$\tau_I(m) = \frac{\sum_{k=1}^{N} w(k)\mu_I(k)}{\sum_{k=1}^{N} w(k)} \tag{19}$$

and

$$\tau_G(m) = \frac{\sum_{k=1}^{N} w(k)\mu_G(k)}{\sum_{k=1}^{N} w(k)} \tag{20}$$

B. A PATCH VARIANCE WEIGHTED AVERAGE (VWA) FILTER AND EXTENSIONS

In this paper, we only focus on a special case where $\alpha = 0$ (corresponding to $\epsilon \to \infty$). Let us first consider the simplest case $I = G$. The filter will be called the patch variance weighted average (VWA) filter which can be written as

$$J(m) = \tau_I(m) \tag{21}$$

1) The bilateral VWA filter

We can change the filter by replacing $\mu_I(k)$ in (19) with the original image $I(k)$ such that the new filter is in the form of local average:

$$J(m) = \frac{\sum_{k=1}^{N} w(k)I(k)}{\sum_{k=1}^{N} w(k)} \tag{22}$$

where $k \in \Omega_m$ is the location index of a pixel in the patch center at location m. We should point out that the index k is now used to represent the pixel location rather than the patch index. The weight is calculated as

$$w(k) = \frac{1}{1 + (\sigma_I^2(k)/\sigma_r^2)^2} \tag{23}$$

where $\sigma_I^2(k)$ is the variance of the patch centered at location k and σ_r is a user defined scale parameter. We further define this scale parameter as the following

$$\sigma_r^2 = s \times \bar{\sigma}_I^2(k) \tag{24}$$

where

$$\bar{\sigma}_I^2(k) = \frac{1}{N}\sum_{k=1}^{N} \sigma_I^2(k) \tag{25}$$

and s is a user defined scale parameter which controls the scale of smoothing.

Inspired by the bilateral filter, the VWA filter is further extended to have the following form

$$J(m) = \frac{\sum_{k=1}^{N} w_1(k)w_2(k)I(k)}{\sum_{k=1}^{N} w_1(k)w_2(k)} \tag{26}$$

where

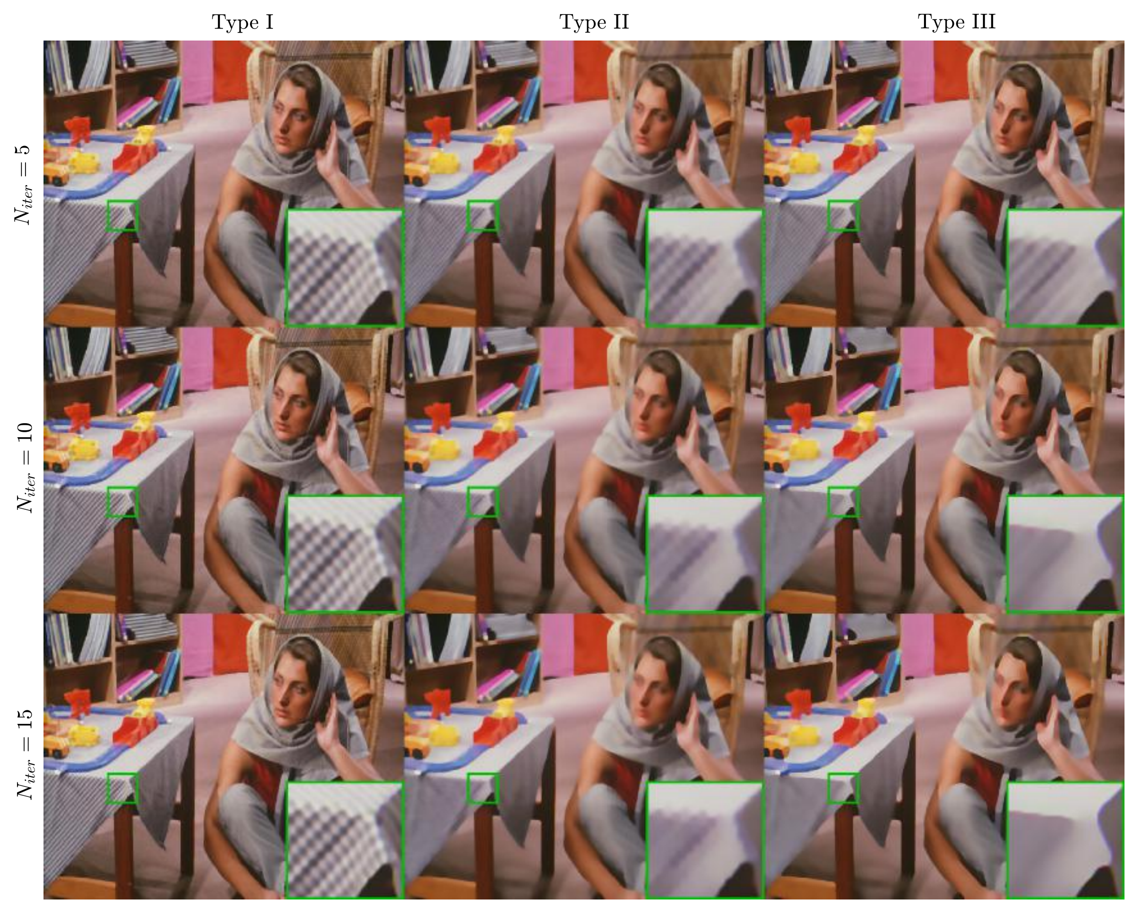


FIGURE 1: Rolling GVWA comparison using self guidance with fixed $\sigma_s = 1$ and $s = 0.75$.

$$w_1(k) = e^{-||m-k||_2^2/\sigma_s^2} \tag{27}$$

and σ_s is a user defined spatial scale parameter and $||m - k||_2^2$ is the Euclidean distance between the two pixels. We set $w_2(k) = w(k)$. While the filtering effect of the VWA filer is completely controlled by the patch size, the smoothing effect of the bilateral VWA filter is controlled by adjusting σ_s and s.

2) Guided VWA (GVWA) filter and rolling guidance filter

We now consider the case $I \neq G$ which leads to the use of guidance image. There are two main ideas in using the guidance image: (1) using the guidance image to calculate the weights, e.g., in the joint-bilateral filter [1], and (2) iterative guidance filtering which is called the rolling guidance filter [10]. Following the first idea we can easily extend the VWA filter by using the guidance image to calculate the weights, i.e.,

$$w(k) = \frac{1}{1 + (\sigma_G^2(k)/\sigma_r^2)^2} \tag{28}$$

Where σ_r^2 is also calculated using the guidance image:

$$\sigma_r^2 = s \times \bar{\sigma}_G^2(k) \tag{29}$$

The resulting filter will be called guided VWA (GVWA) filter which is mathematically represented as

$$J = \text{GVWA}(I, G; \theta) \tag{30}$$

where J, I, and G present the output, input and guidance images, respectively. The symbol θ represents the collection of all user defined parameters $\theta = \{\sigma_s, s, N_{iter}\}$, where N_{iter} is the number of iterations to be discussed next.

Following the second idea, the rolling guidance filter can be defined as one of the following iteration methods which stops when the maximum number of iteration is reached ($n = N_{iter}$).

- Type-I: Fixed input I, rolling guidance G, where $G^{(n)} = J^{(n-1)}$ and $J^{(0)} = G$.

$$J^{(n)} = \text{GVWA}(I, G^{(n)}; \theta) \tag{31}$$

- Type-II: Fixed guidance G, rolling input I, where $I^{(n)} = J^{(n-1)}$ and $J^{(0)} = I$.

$$J^{(n)} = \text{GVWA}(I^{(n)}, G; \theta) \tag{32}$$

- Type-III: Rolling input I and guidance G, where $I^{(n)} = G^{(n)} = J^{(n-1)}$, $I^{(0)} = I$ and $G^{(0)} = G$.

$$J^{(n)} = \text{GVWA}(I^{(n)}, G^{(n)}; \theta) \tag{33}$$

For all three types we can use the special self-guided case, i.e., $G = I$. In Fig. 1 we show the smoothing effect of the 3 types of rolling filters using the original image as a guidance and varying the number of iterations. We can see that Type-I has the least smoothing power, while Type-II and Type-III produce similar results which have stronger smoothing effect than that of Type-I. The weights are calculated each time the guidance image is updated, so Type I and Type III filters are computationally more expensive than Type-II as it requires the weights to be calculated only once. In the following we will focus on using the Type-II iteration only.

C. DISCUSSIONS

1) Implementation and computational complexity

We can see from section III-B1 that the computational complexity of the proposed filter is the same as that of the guided filter which is $\mathcal{O}(N_{pix})$. This is verified by a brute-force MATLAB implementation of the algorithm GVWA Type-II and Type-III which are applied to images of different sizes. Type-I was omitted because its running time is the same as Type-III method.

Fig. 2 shows that the computational time is a linear function of the size of the image and that Type-II has a lower running time than Type-III, this is because the number of iterations used on this experiment is $N_{iter} = 10$, so the weights need to be calculated 10 times when using GVWA Type-III and only once when using GVWA Type-II. Updating the weights requires the use of 3 linear filters (f_{mean}) and two element-wise multiplications and one element-wise division which significantly increase the running time when performed on each iteration. Such calculations require significantly more computation time when an application requires a large number of iterations. Since GVWA Type-II is more efficient, for the rest of the paper we only use this method for applications.

Fig. 2 also shows the running time of the guided filter (GIF[1]) [8] when applied for 10 iterations to the same image sizes. We can see that our filter is faster than the GIF, this is because our method does not need to compute the patch covariance.

The steps of the implementation of the GVWA filter are summarized in Algorithm 1, where ".$*$" is the symbol for the element-wise multiplication operation, "./" is the symbol for the element-wise division operation, and f_{mean} and f_{gauss} are a mean and a Gaussian filter respectively. The size of the filter kernel (patch size) is determined by the parameter σ_s. In our implementation, the patch size is defined as: patchSize $=$ floor$(4\sigma_s)+1$. For a color image, the weight for the {R,G,B} components is the same and is defined as:

$$w(k) = \frac{1}{1 + (\sigma^2(k)/\sigma_r^2)^2} \tag{34}$$

where the patch variance $\sigma_I^2(k)$ is replaced by the maximum of the three component: $\sigma^2(k) = \max\{\sigma_R^2(k), \sigma_G^2(k), \sigma_B^2(k)\}$. A MATLAB implementation can be found in Appendix and in the project's GitHub repository[2].

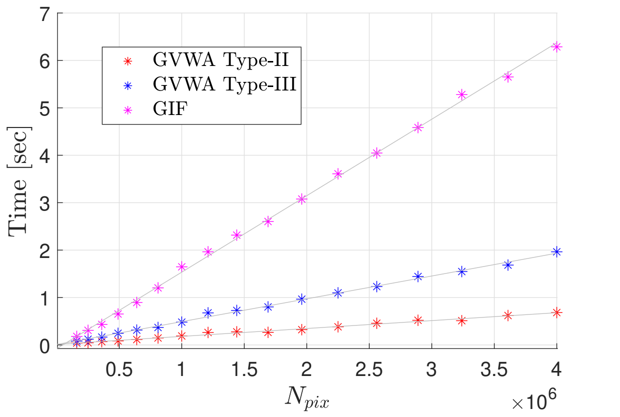


FIGURE 2: Running time of the proposed filter implemented in MATLAB. N_{pix} is the number of pixels in an image. The computer used to obtain these results is with an Intel Core i7 3930K processor and 32GB RAM. MATLAB is running in Ubuntu 16.04 LTS.

Algorithm 1: Pseudo code of the GVWA filter

```
Input: Filtering input image $I$, guidance image $G$,
       spatial scale $\sigma_s$, scale of smoothing $s$
Output: Filtering output J
$meanG = f_{mean}(G)$
$corr_G = f_{mean}(G. * G)$
$var_G = corr_G - mean_G. * mean_G$
$var_r = s. * f_{mean}(var_G)$ // Eq.29
$w = 1./(1 + (var_G/var_r)^2)$ // Eq.28
$J = f_{gauss}(w. * I)$
/* $f_{mean}$ and $f_{gauss}$ are a mean and a Gaussian filter
   respectively. */
```

2) Relationship with the bilateral filter

The bilateral filter is defined as:

$$J(m) = \frac{\sum_{k\in\Omega_m} w_s(k)w_r(k)I(k)}{\sum_{k\in\Omega_m} w_s(k)w_r(k)} \tag{35}$$

[1]http://kaiminghe.com/eccv10/

[2]https://github.com/fergaletto/GVWA

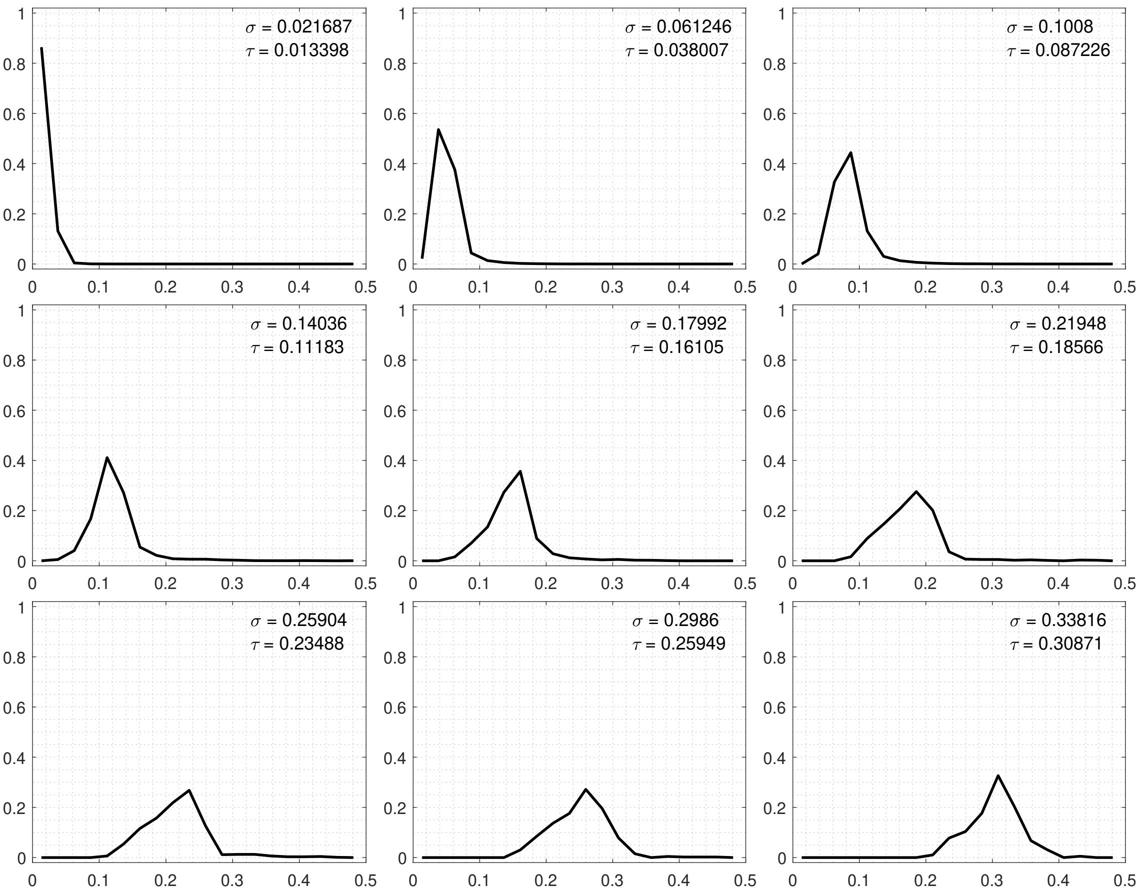


FIGURE 3: Approximation accuracy of the local absolute difference using the standard deviation. Each sub-figure is a histogram of the absolute difference of (horizontal axis) all pairs of pixels within a patch conditioned on a standard deviation value. The histograms are calculated from an image ("peppers.png"in MATLAB). σ and τ are the patch standard deviation and histogram's sample mean, respectively. There is a high probability that the absolute difference is less than $\kappa \times \sigma$ where $\kappa \leq 2$

where Ω_m is the set of pixel indices of a patch centered at $I(m)$. The spatial weight $w_s(k)$ and the range weight $w_r(k)$ with user defined scale parameters σ_s and σ_r are defined as:

$$w_s(k) = e^{-||m-k||_2^2/\sigma_s^2} \tag{36}$$

$$w_r(k) = e^{-(G(m)-G(k))^2/\sigma_r^2} \tag{37}$$

The bilateral filter is computationally expensive because the range weight involves the pixel to be processed. As a result, a double for-loop is required for a brute-force implementation. The outer loop runs through all pixels of the input image, while the inner loop runs through a patch of pixels centered at pixel $I(m)$. Let the number of pixels of the image be N_{pix} and the number of pixels in the patch be M, the computational complexity is $\mathcal{O}(N_{pix}M)$.

Comparing the proposed filter with the bilateral filter, we can see that the spatial weight is the same: $w_1(k) = w_s(k)$. The difference is in the way the range weight is calculated. In the following, we show that the proposed bilateral VWA filter is related to the bilateral filter in the following sense

$$(G(m) - G(k))^2 \to \sigma_G^2(k) \tag{38}$$

where $\sigma_G^2(k)$ is the variance of a patch centered at $G(k)$ and $k \in \Omega_m$. We use the symbol "$\to$" to indicate that the left term can be approximated/replaced by the right term. The proposed filter avoids the computationally expensive calculation of $(G(m) - G(k))^2$ by replacing it with $\sigma_G^2(k)$.

What is the justification for the relationship shown in (38)? For simplicity, let us assume pixels in the patch follow an independent and identical distribution with mean $\mu_G(k)$ and variance $\sigma_G^2(k)$. The difference $G(m) - G(k)$ is then zero mean with standard deviation $\sqrt{2}\sigma_G(k)$. For a normal distribution, the probability of the difference lies within two standard deviations, i.e., $|G(m) - G(k)| < 2 \times \sqrt{2}\sigma_G(k)$ is

$$p(|G(m) - G(k)| < 2 \times \sqrt{2}\sigma_G(k)) = 0.95 \tag{39}$$

As such, we can say that $2 \times \sqrt{2}\sigma_G(k)$ is the estimate of the worst case of the absolute difference. In other words, such estimate is an over estimate with probability 0.95 and is an under estimate with probability 0.05. If we instead use one standard deviation as the estimate, then

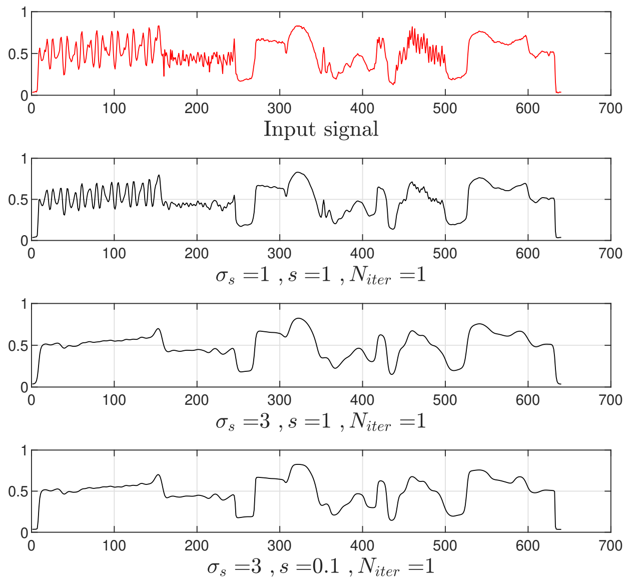


FIGURE 4: Filtering a 1D signal using the proposed GVWA filter with different parameter settings and no iteration.

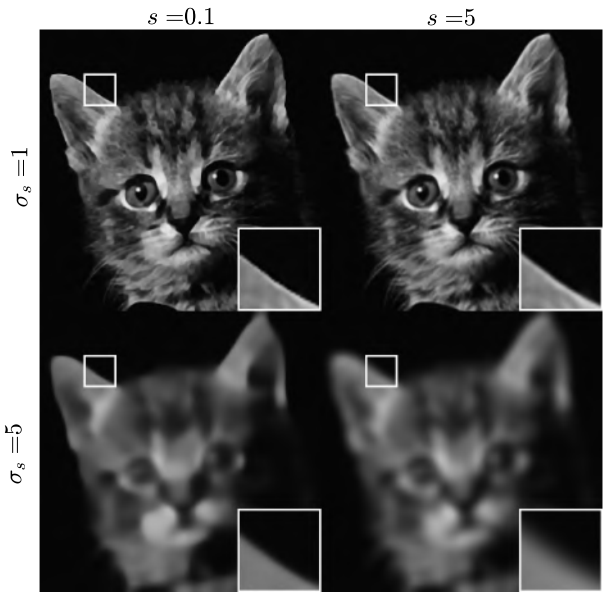


FIGURE 5: Image smoothing using the proposed filter (N_{iter} = 1) under different combinations of parameter settings. Increasing σ_s leads to smoothing out small scale details. Using a smaller s value results in sharper edges for large scale objects.

$$p(|G(m) - G(k)| < \sqrt{2}\sigma_G(k)) = 0.65 \tag{40}$$

In other words, the standard deviation is an over estimate of the absolute difference with probability 0.65 and is an under estimate with probability 0.35. Therefore, we can replace the range weight by the approximation:

$$\hat{w}_r(k) = e^{-(\kappa\times\sqrt{2}\sigma_G(k)/\sigma_r)^2} = e^{-2\kappa^2\sigma_G^2(k)/\sigma_r^2} \tag{41}$$

where κ is a scale parameter set by the user to control the number of standard deviations used for the approximation. In addition, the proposed filter replaces the exponential operation with the division operation which achieves further savings in computational time.

To empirically validate the estimation accuracy, we calculated the standard deviation for all the patches in an image (the image in MATLAB "peppers.png'). We also calculate the pixel absolute difference within all patches. We created a 2D histogram of the distribution of the patch standard deviation versus the absolute difference. The standard deviation is quantified into 9 bins, the absolute difference is quantified into 20 bins. For each standard deviation, we can plot a distribution of the absolute difference. The results are demonstrated in Fig. 3 which are presented as 9 sub-figures rather than a 2D histogram for easy visualization. Each sub-figure represents the distribution of the absolute difference for a given standard deviation indicated by σ in the figure. The sub-figures are organized in the order of increasing standard deviation. The mean of the absolute difference is indicated by τ. We can clearly see that the mean of the absolute difference roughly follows that of the standard deviation and for all cases there is a high probability that the absolute difference is less than $\kappa \times \sigma$ where $\kappa \leq 2$. These observations provide concrete evidence for the above statistical analysis. Thus, the proposed filter is justified from results of statistical analysis and simulation using a real image.

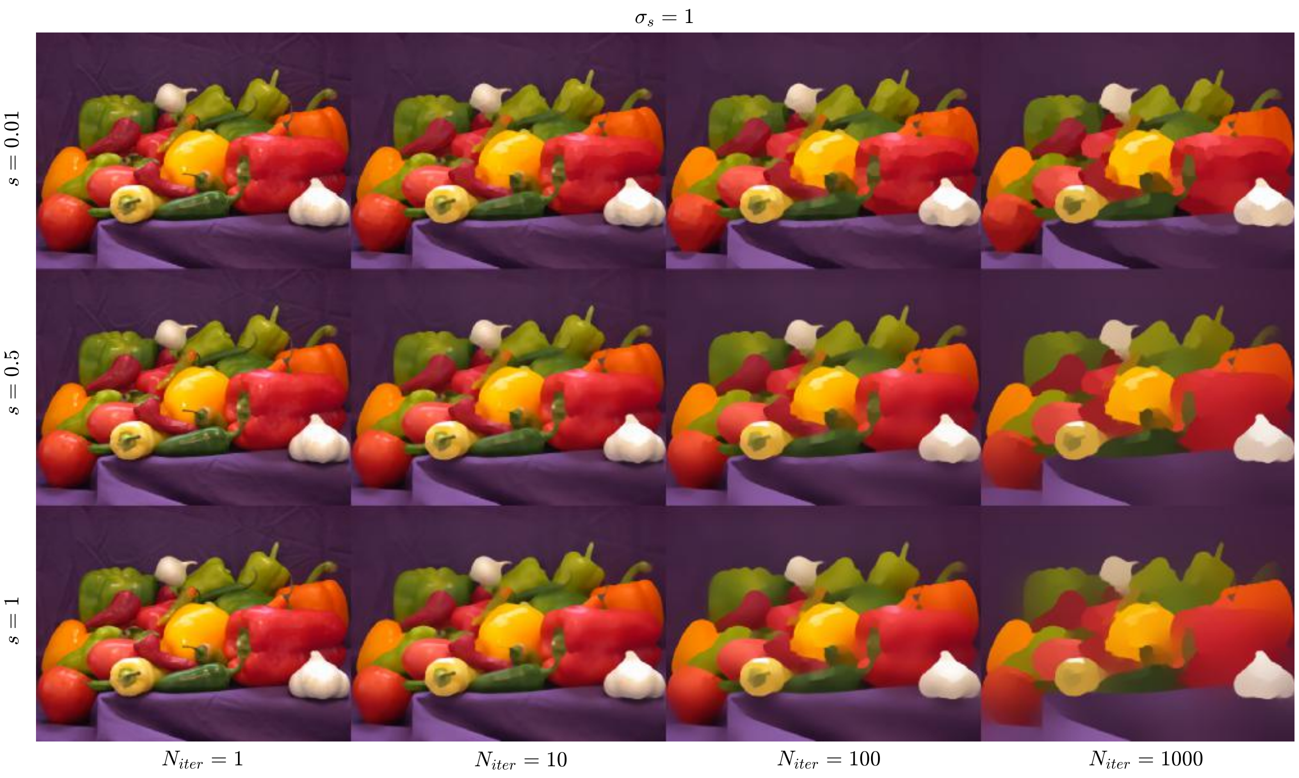
IV. SIMULATION RESULTS AND COMPARISONS

A. PARAMETER SETTINGS

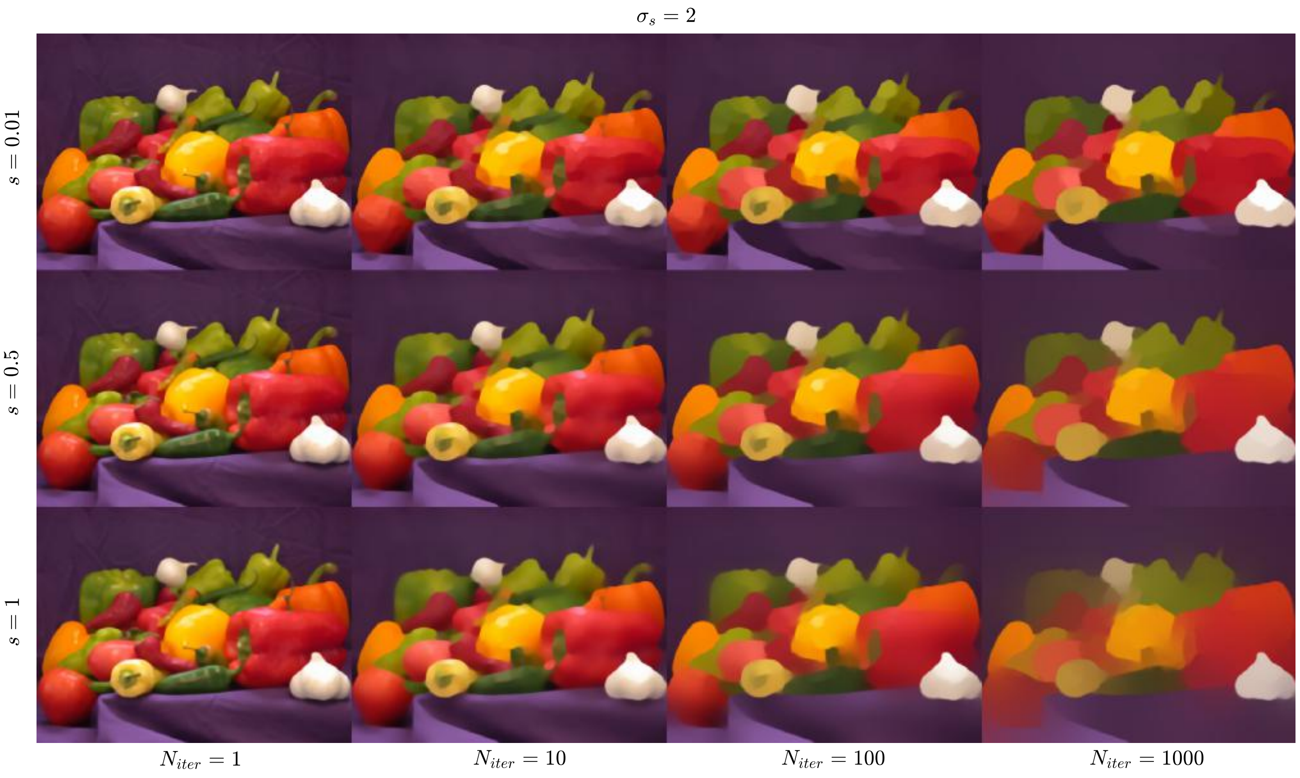
The proposed filter has three user defined parameters: (1) σ_s defines the patch size which controls the smoothing effect, (2) s controls the sharpness of the result, and (3) N_{iter} is the number of iterations when the filter is used in iterative mode. In this section, we study the effects of these three parameters.

1) The two scale parameters

To demonstrate the edge-preservation ability of the proposed filter in the non-iterative operation (N_{iter} = 1), we start by filtering a 1D signal which provides a demonstration of the filter performance. In Fig. 4, we can see that the GVWA filter reduces the effect of small-scale oscillations/edges while keeping the large-scale edges. In Fig. 5, we observe that increasing σ_s leads to an increased smoothing in the small details of the image. In addition, setting s a small value has two different consequences: increasing the smoothing effect on small oscillations and producing sharper edges for large scale objects.



(a) Smoothing effects for fixed $\sigma_s = 1$ under different combinations of s (row-wise) and N_{iter} (column-wise).



(b) Smoothing effects for fixed $\sigma_s = 2$ under different combinations of s (row-wise) and N_{iter} (column-wise).

FIGURE 6: Smoothing effects for $\sigma_s = 1$ and $\sigma_s = 2$ under different combinations of s (row-wise) and N_{iter} (column-wise).

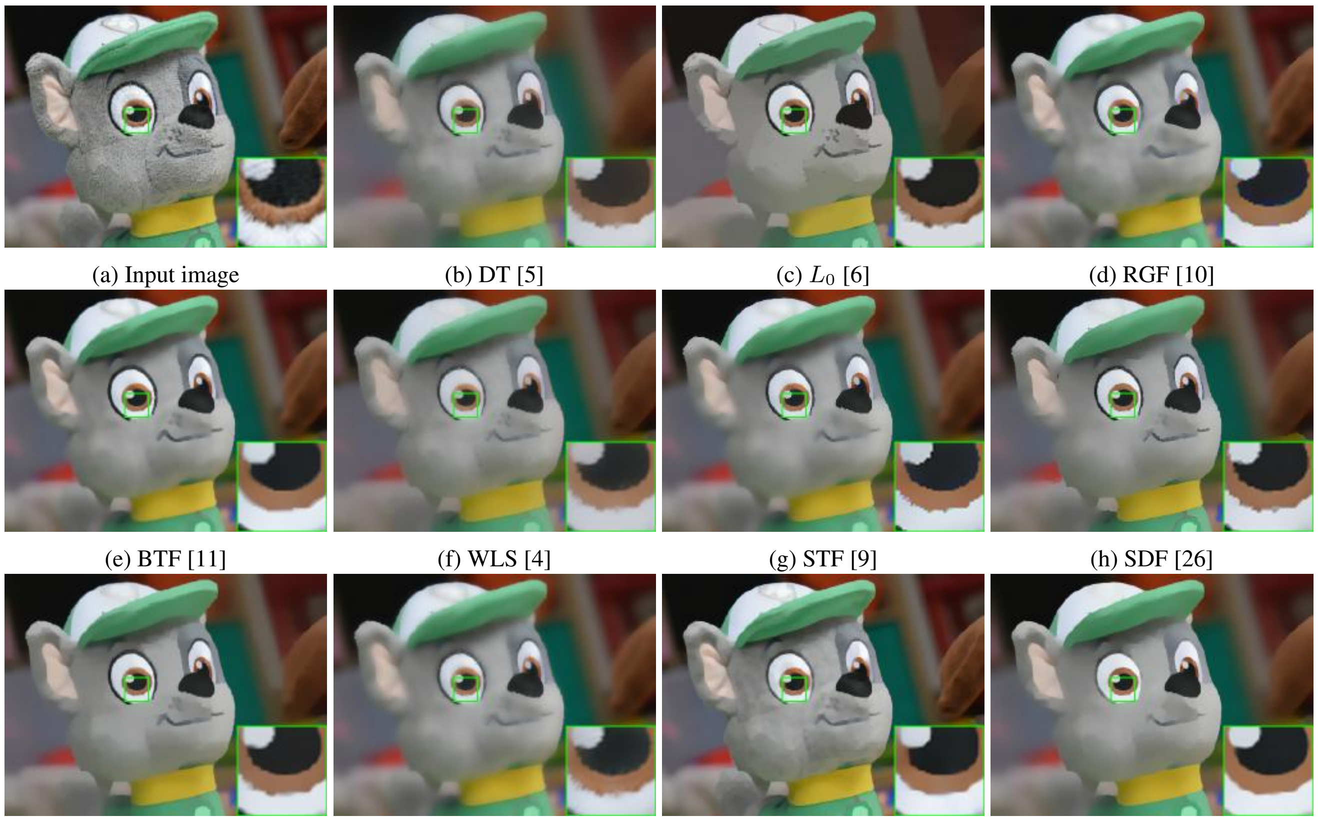

(a) Input image (b) DT [5] (c) L_0 [6] (d) RGF [10]
(e) BTF [11] (f) WLS [4] (g) STF [9] (h) SDF [26]
(i) RTV [7] (j) WGF [18] (k) AnisGF [23] (l) Proposed filter

FIGURE 7: A comparison of edge-preserving image smoothing with a group of state-of-the-art algorithms. (a) Original image, (b) DT ($\sigma_s = 30, \sigma_r = 0.7, N_{\text{iter}} = 5$), (c) L_0 ($\lambda = 0.02, \kappa = 2, N_{\text{iter}} = 1$), (d) RGF ($\sigma_s = 5, \sigma_r = 0.1, N_{\text{iter}} = 20$), (e) BTF ($k = 5, N_{\text{iter}} = 5$), (f) WLS, ($\lambda = 1, \alpha = 1.2$), (g) STF ($\sigma_s = 3, \sigma_r = 0.08, \sigma = 0.01, N_{\text{iter}} = 4$), (h) SDF ($\sigma_g = 50, \sigma_u = 400, \lambda = 30, nei = 1$), (i) RTV ($\sigma = 3, \lambda = 0.015, \epsilon = 0.02, N_{\text{iter}} = 4$) (j) WGF ($\sigma = 1, \epsilon = 0.01, N_{\text{iter}} = 10$) (k) AnisGF ($\sigma = 1.5, \epsilon = 1$) (l) Proposed rolling GVWA Type-II filter ($\sigma_s = 1.5, s = 0.75, N_{\text{iter}} = 20$).

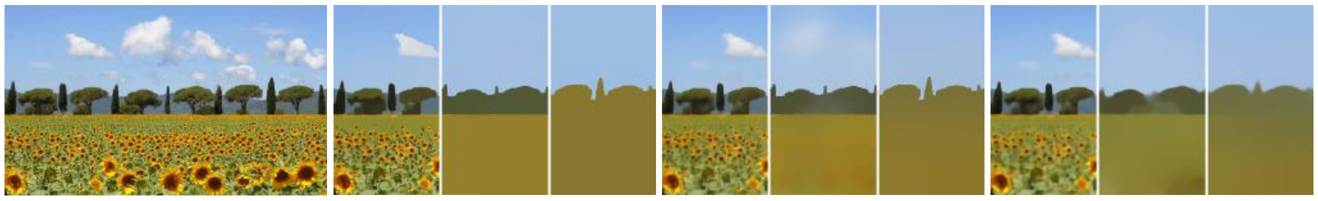

(a) Input image (b) LADR [38] (c) muGIF [39] (d) Ours

FIGURE 8: Visual comparison of scale-space representation, (a) Input image, (b) LADR filter, (c) muGIF, (d) Our method.

2) Iterative filtering

We study the Type-II iteration algorithm and the effects of different settings of σ_s and s under different number of iterations, we run simulations on the image "peppers.png" (an image in the MATLAB Image Processing Toolbox). Results are shown in Fig. 6. We make the following observations. For a fixed σ_s and s, the smoothing effect increases with the number of iterations. Indeed, when $N_{iter} = 100$ or 1000, the image is "flattened" in that almost all details have been removed. For a fixed σ_s, increasing the scale parameter s will lead to smoother results. This is more evident when the number of iterations increases. Comparing the two figures, we can see that for the same setting of s and N_{iter}, the bigger value of σ_s leads to a smoother result.

B. APPLICATIONS AND COMPARISONS

1) Edge-preserving image smoothing

We start by demonstrating the performance of the proposed filter in edge-aware filtering. Fig. 7 shows the edge-preserving properties of the filter compared with the do-

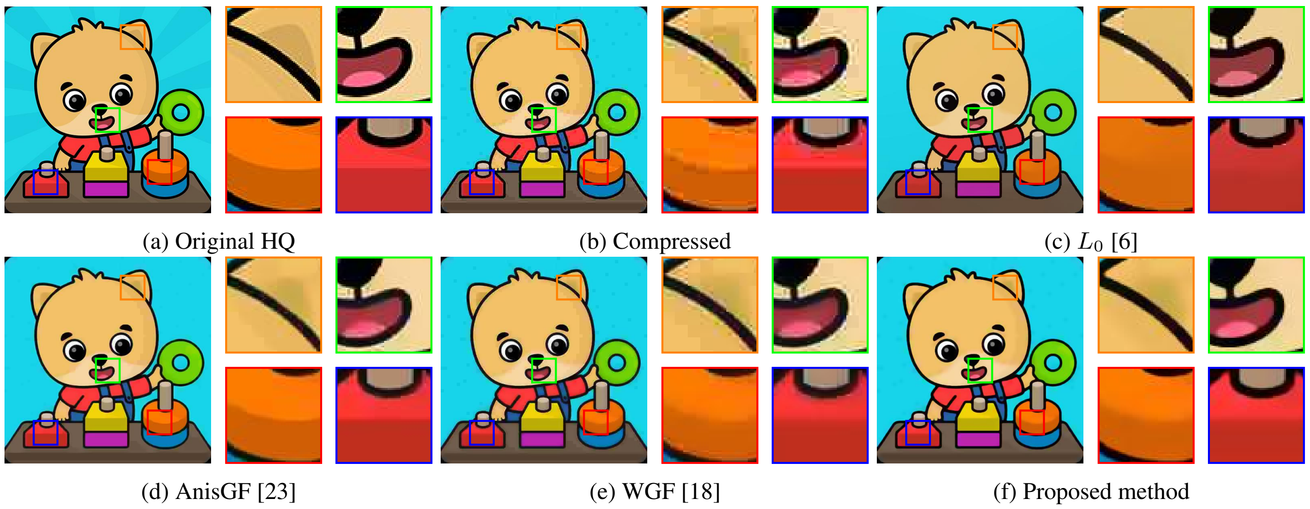


FIGURE 9: JPEG compression artifact removal (a) Original image, (b) Compressed image (Quality = 10%), (c) L_0 ($\lambda = 0.02, \kappa = 1.5$), (d) WGF ($\sigma = 1.5, \epsilon = 0.01, N_{\text{iter}} = 3$), (e) AnisGF ($\sigma = 2.5, \epsilon = 0.01$), (f) Proposed filter ($\sigma_s = 0.75, s = 0.5, N_{\text{iter}} = 20$). We can clearly see that the proposed filter outperforms L_0, WGF and AnisGF filter in preserving features and removing compression artifacts (marked by square boxes).

main transform filter (DT[3]) [5], L_0 filter (L_0[4]) [6], rolling guidance filter (RGF[5]) [10], bilateral texture filter (BTF[6]) [11], weighted least-squares (WLS[7]) [4], tree filter (STF[8]) [9], static-dynamic filter (SDF[9]) [26] and relative total variation (RTV[10]) [7]. To make comparison with other weighted versions of the guided filter, we wrote our own code for Weighted guided image filtering (WGF) and Anisotropic guided filtering (AnisGF). Parameter settings for all methods are provided in the figure caption. We can see that results produced by the proposed filter are similar to those produced by other methods.

The proposed filter can also be used to selectively remove objects of different scales from an image. Fig. 8 (d) shows the result of applying the proposed filter to smooth objects of different sizes by preserving the structural information of the image. In this experiment we compared with two state of the art methods for scale-ware smoothing: Local activity-driven structural-preserving filtering (LADR) [38] and Mutually guided image filtering (muGIF) [39]. To selectively smooth objects from 3 different scales we use three different set of parameter. When applying LADR we set the λ parameter to 0.03, 0.08 and 0.3. On the other hand, when muGIF is used we vary the regularization parameter α_t to 0.001, 0.1 and 1 while keeping a fixed α_r, N_{iter} and mode equal to 1, 10 and 0 respectively.

As can be seen in Fig. 8 our method can successfully remove small, medium and large scale details depending on the settings, but it does not overpass the outstanding performance of LARD and muGIF at edge preservation when filtering large scale objects. Boundaries of meaningful objects at each scale are better preserved with LARD and muGIF.

[3]DT: https://www.inf.ufrgs.br/ eslgastal/DomainTransform/
[4]L_0 http://www.cse.cuhk.edu.hk/ leojia/projects/L0smoothing/
[5]RGF http://www.cse.cuhk.edu.hk/ leojia/projects/rollguidance/
[6]BTF https://github.com/JiaXianYao/Bilateral-Texture-Filtering
[7]WLS https://www.cs.huji.ac.il/ danix/epd/
[8]STF http://linchaobao.github.io/
[9]SDF https://github.com/bsham/SDFilter
[10]RTV https://github.com/yearway/RTV_Smooth

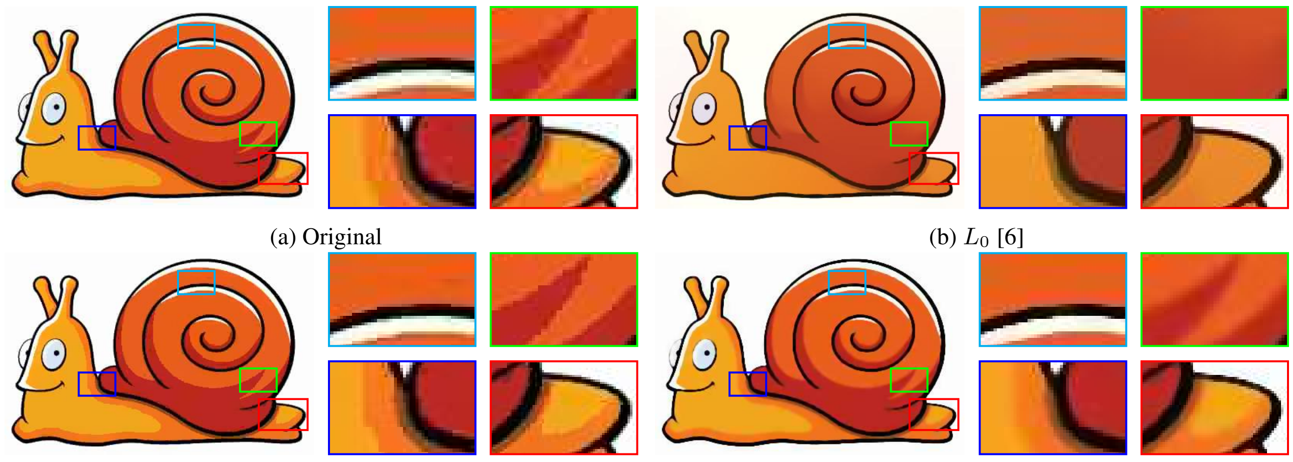
TABLE 1: MSE and PSNR for images in Fig. 9.

Image	MSE	PSNR
Compressed	0.0032	24.9877
L_0	0.0023	26.2988
AnisGF	0.0035	24.5728
WGIF	0.0032	24.9758
Proposed filter	**0.0020**	**26.9503**

2) Clip-art and JPEG compression artifact removal

To demonstrate the performance of the proposed filter on the task of removing JPEG compression artifacts, we first use a high quality image and compress it using the JPEG format with a compression quality factor of 10%. Then we filter the low quality image with compression artifacts using our filter rolling GVWA Type-II with $\sigma_s = 0.75, s = 0.5, N_{\text{iter}} = 20$. We compare the result using the L_0 smoothing filter [6] with $\lambda = 0.02, \kappa = 1.5$ and two weighted versions of the guided filter WGF [18] and AnisGF [23] with settings $\sigma_s = 1.5, \epsilon = 0.01, N_{iter} = 3$ and $\sigma_s = 2.5, \epsilon = 0.01$, respectively. Fig. 9 shows that our filter is able to remove all the color artifacts due to the low quality compression while keeping the edges and preserving the colors of the image. The proposed filter performs better than the L_0 smoothing in preserving some features of the image, e.g., the shade on the ear (marked by a red square in the figure). Also, the proposed filter removes all the artifacts due to the compression while WGF and AnisGF struggle to remove artifacts near high contrast edges.

Note: figure panel labels: (a) Original HQ, (b) Compressed, (c) L_0 [6], (d) AnisGF [23], (e) WGF [18], (f) Proposed method



(a) Original (b) L_0 [6] (c) IRWF [40] (d) Proposed method

FIGURE 10: Clip-Art compression artifact removal (a) Original image, (b) L_0 ($\lambda = 0.05, \kappa = 2$), (c) IRWF ($r = 8$), (d) Proposed filter ($\sigma_s = 0.5, s = 0.75, N_{\text{iter}} = 30$). We can clearly see that the proposed filter outperforms the L_0 filter in preserving features (marked by square boxes) of the original image.

Using the original high quality image as a reference, we calculated the mean squared error (MSE) and peak signal-to-noise ratio (PSNR) to perform a quantitative comparison. It can be seen in Table 1 that the MSE is improved from 3.2×10^{-3} to 2.0×10^{-3} using the proposed filter while with L_0 smoothing only improved to 2.3×10^{-3}. Similar results can be seen in the PSNR. On the other hand, AnisGF and WGF do not improve the MSE and PSNR with respect to the compressed image even though there is a notorious visual improvement.

To demonstrate the removal of compression artifact from clip-art images, a low quality clip-art image is processed by the proposed rolling GVWA Type-II filter with $\sigma_s = 0.5, s = 0.75, N_{\text{iter}} = 30$, L_0 smoothing [6] with $\lambda = 0.05, \kappa = 2$ and IRWF [40] with $r = 8$. We can see in Fig. 10 that the three methods successfully remove the artifacts, the proposed filter performs better in preserving the low contrast edges than L_0 and produce more natural looking edges than IRFW (see areas marked by squares).

3) Structure separation

Structure separation is a process of decomposing the overall image structures (meaningful information) from highly correlated background (texture/noise). In this application, we demonstrate the performance of the proposed filter in smoothing out small scale textures while maintaining the prominent structures. The proposed filter is able to extract the main structure from irregular textures because the small scale details will progressively vanish as the number of iterations increases. Results are shown in Fig. 11 which provides a comparison of the performance of the proposed filter with a group of state-of-the-art structure separation algorithms. We can clearly see that our filter can produce comparable results in terms of structure extraction. In fact, the proposed filter produces sharper edge boundaries, less blocking artifacts in texture areas, and better contrast in comparison with the other filters. In particular, as shown in Fig. 11 (a) and (b), we use a magnified box to highlight the performance of the filters on the fish eye where the main improvement of performance can be observed. When comparing Fig. 11 (b) with (c) we can see that our method under-performed RTV since it did not remove some pixels with high gradients from the background, although changing the filter parameters can smooth the background as RTV ($\sigma_s = 1, scale = 0.5, N_{\text{iter}} = 100$) we opted to prioritize sharpness and contrast at the cost of leaving those few pixels.

Table 2 presents a quantitative assessment using the entropy metric (T3SI[11]) [41]. This metric aims to measure the similarity between the original image and the processed image. A bigger value indicates higher similarity. We can see from Table 2 that the performance of the proposed algorithm produces comparable results and is closer to the average of the other filters (the average is 2.0851).

4) Edge extraction/enhancement example

Edge detection aims at finding the boundaries of the objects in the scene. The gradient is commonly used to extract the edges of an image I. For example, the magnitude of the gradients defined in (42) is used for edge extraction. However, gradient-based methods are greatly affected by noise or small details.

$$E = |\nabla I| = \sqrt{\nabla_x I^2 + \nabla_y I^2} \tag{42}$$

In Fig 12, we demonstrate the benefits of applying the proposed filter to smooth the image before calculating the edge gradient magnitude. We can see that pre-processing the

[11]T3SI: https://github.com/liuchong777/T3SI
[13]IGF: https://github.com/JunhoJeon/interval_gradient
[13]SAF: https://github.com/JunhoJeon/safiltering

TABLE 2: Texture smoothing quality assessment for results in Fig. 11.

Algorithm	RTV	RGF	STF	BTF	SDF	IGF	SAF	SND	Proposed
T3SI [41]	2.2080	1.9820	2.1787	2.0101	2.1483	2.1160	2.0753	1.9626	2.0334

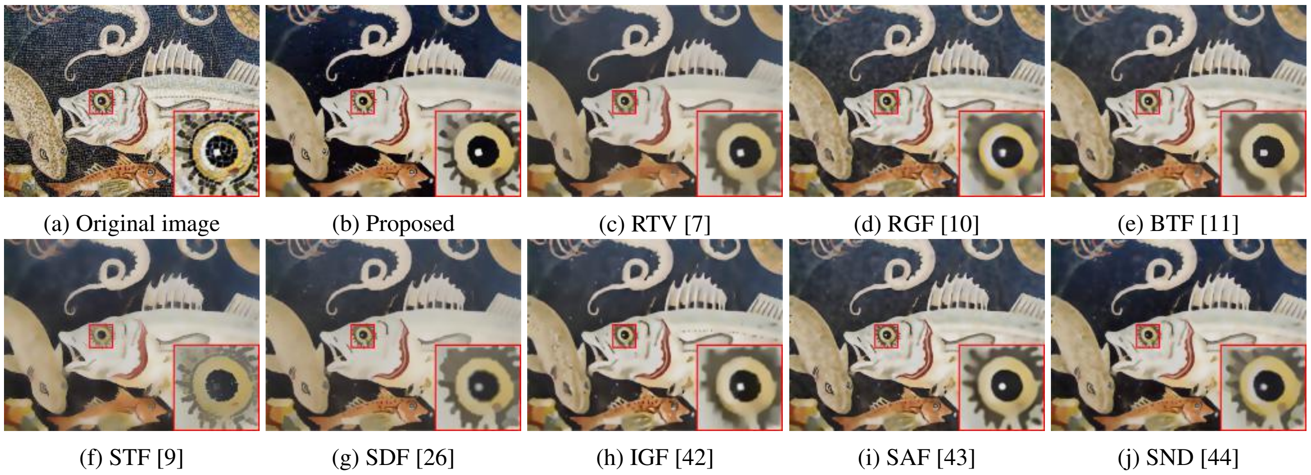


FIGURE 11: A comparison of structure separation results for the "Fish" image with a group of state-of-the-art algorithms. The proposed method produces an image of highest contrast compared with results from other methods. (a) Original image, (b) Proposed ($\sigma_s = 0.5, s = 0.5, N_{\text{iter}} = 500$), (c) RTV ($\sigma = 6, \lambda = 0.02, \epsilon = 0.02, N_{\text{iter}} = 4$), (d) RGF ($\sigma_s = 5, \sigma_r = 0.08, N_{\text{iter}} = 5$), (e) BTF ($\sigma = 5, N_{\text{iter}} = 5$), (f) STF ($\sigma_s = 5, \sigma_r = 0.05, \sigma = 0.02, N_{\text{iter}} = 5$), (g) SDF ($\sigma_g = 50, \sigma_u = 400, \lambda = 200, nei = 2$), (h) IGF[12]($\sigma = 4.3, \epsilon = 0.03^2$), (i) SAF[13]($\sigma = 4, \sigma_r = 0.1, N_{\text{iter}} = 5$), (j) SND ($\sigma = 0.04, \lambda = 0.25, N_{\text{iter}} = 19$).

image using the proposed filter leads to a cleaner and sharper edge map which not only preserves the main structure of the scene but also reduces the effect of the noise and textures.

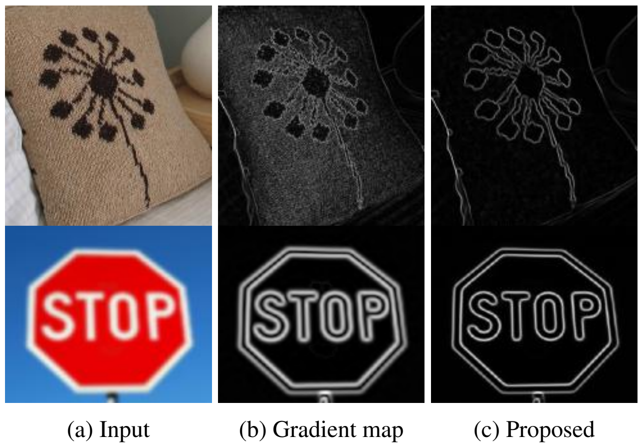


FIGURE 12: Edge extraction example. (a) Original image. (b) Gradient map of original image. (c) Gradient map of smooth image ($\sigma_s = 1, s = 0.5, N_{\text{iter}} = 15$).

5) Non-photo realistic rendering

In this application, we demonstrate the use of the proposed filter to produce non realistic versions of the image using the framework proposed in [14]. This method stylizes the image by first simplifying its content using a filter to blur small details and sharp edges. The high contrast details or edges are then magnified to further increase the visual abstraction. The luminance is quantized to add the cartoon appearance to the image. In this paper, we take a slightly different approach since we do not employ the luminance quantization.

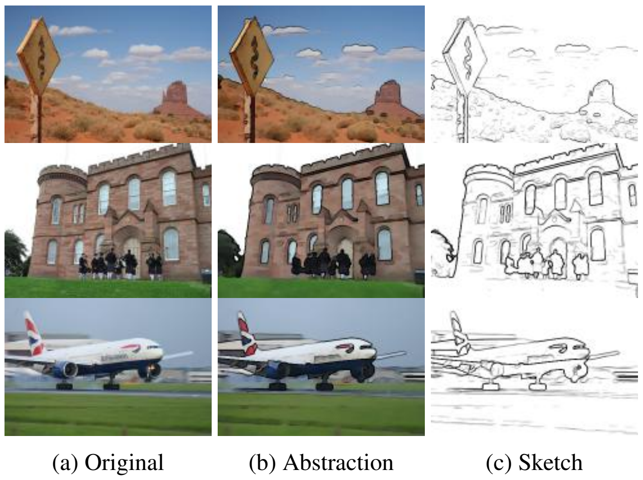


FIGURE 13: Non-photo realistic rendering, the proposed filter is used to produce two different artistic effects: (a) Original image, (b) Abstraction, (c) Sketch ($\kappa = 0.1, \zeta = 0.1$).

We use the proposed filter to blur the low contrast details while keeping the edges of the image. We then calculate the gradient map denoted $E = |\nabla B|$ for the filtered image denoted B to detect the edges. We further process the value E using the following method:

$$D(x,y) = \begin{cases} 1, & E(x,y) \geq \kappa \\ \frac{E(x,y)}{\kappa}, & E(x,y) < \kappa \end{cases} \tag{43}$$

where κ is a user defined parameter. We define $\zeta \in [0,1]$ as another user defined parameter to set to zero all the pixels that comply with $D(x,y) < \zeta$. The purpose is to remove all the small edges that weren't removed by smoothing the image and were amplified in the previous step. As a by-product of our approach, the gradient magnitude map of the filtered image represent only large and high contrast edges in the original image. As such, the processed edge map produces a sketch effect $S(x,y) = 1 - D(x,y)$. We then use the edge map $D(x,y)$ and the smooth image to produce the abstract image:

$$A(x,y) = (1 - D(x,y)) \times B(x,y) \tag{44}$$

Results in Fig. 13 show that our approach successfully produces an artistic abstraction and a sketch effect from the input images. Column (b) shows that the proposed filter preserves only strong edges and structure. Abstraction results are shown in column (c), as described in [14], these images look stylized since all low contrast details were blurred while strong edges were visually increased. The sketch images are shown in column (d) which show only large and high contrast edges of the original images.

6) Saliency object detection

Saliency detection aims at locating the structural information (objects/regions) in a natural scene without emphasizing unimportant details. This process is similar to human perception. In some images, the foreground and background are correlated which makes saliency detection a challenging task. To address this problem, edge-aware filters can be used as a pre-processing step to aid saliency detection. We employ our filter to abstract the object of interest by getting rid of the unwanted details while preserving meaningful structures. In [45], the authors smoothed out the input image prior to the application of the saliency map generation algorithm in [46]. We follow the same approach to generate the map. Results are shown in Fig. 14 in which it is clearly noticeable that our saliency map is more consistent and uniform than the algorithms in [45] and [46].

7) Detail magnification

Unsharp masking is an effective algorithm to enhance the details of an image. The algorithm is defined as:

$$U = J + \gamma(I - J) \tag{45}$$

where I is the input image, J, called base layer, is the result of a low-pass filter, and γ is the gain used to amplify the high frequency components $(I - J)$ called detail layer.

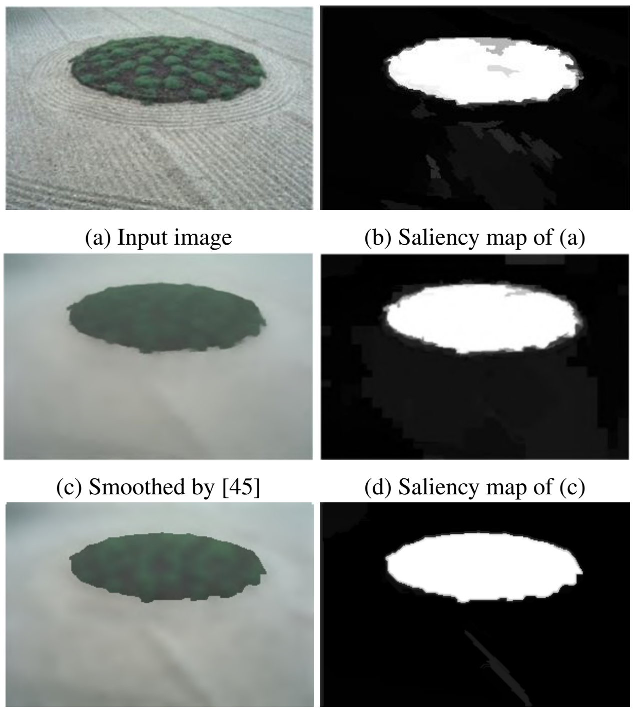

(a) Input image (b) Saliency map of (a)
(c) Smoothed by [45] (d) Saliency map of (c)
(e) Smoothed by our filter (f) Saliency map of our filter

FIGURE 14: A comparison of salient object detection. Top line: (a) Original image, (b) Saliency map of (a) using [46]. Middle line: (c) is smoothed by [45], (d) Saliency map of (b) using [46]. Bottom line: (e) Smoothed by proposed filter (Rolling GVMA Type-III with $\sigma_s = 0.5, s = 2.5, N_{iter} = 3$), (f) Saliency map of (c) using [46].

In Fig. 15, we demonstrate that the proposed filter can be used to produce J in the unsharp masking algorithm, we also compare the result with other 3 well-known methods for image sharpening and detail enhancement such as contrast adaptive sharpening (CAS)[14], generalized unsharp masking (GUM) [47] and a guided edge-aware smoothing-sharpening filter (SSIF[15]) [48]. The settings for each algorithm were selected to avoid over-sharpening so the resulting image has a natural appearance. We can see that our method is able to amplify the details of the scene without producing halo artifacts and its result is comparable to all 3 other algorithms.

8) Multi-focus image fusion

Multi-focus image fusion is a technique that blends two or more images which are only focused on certain objects of the same image. The fusion algorithm produces a new image in which all objects are in focus. The methods to perform multi-focus image fusion can be categorized into four categories: transform domain, spatial domain, combined transform, and deep learning methods [54], [55]. In this paper we modify two popular fusion methods by replacing the guided filter [8] by the proposed filter to investigate its performance in this

[14]https://www.amd.com/en/technologies/radeon-software-fidelityfx
[15]https://github.com/fergaletto/SSIF

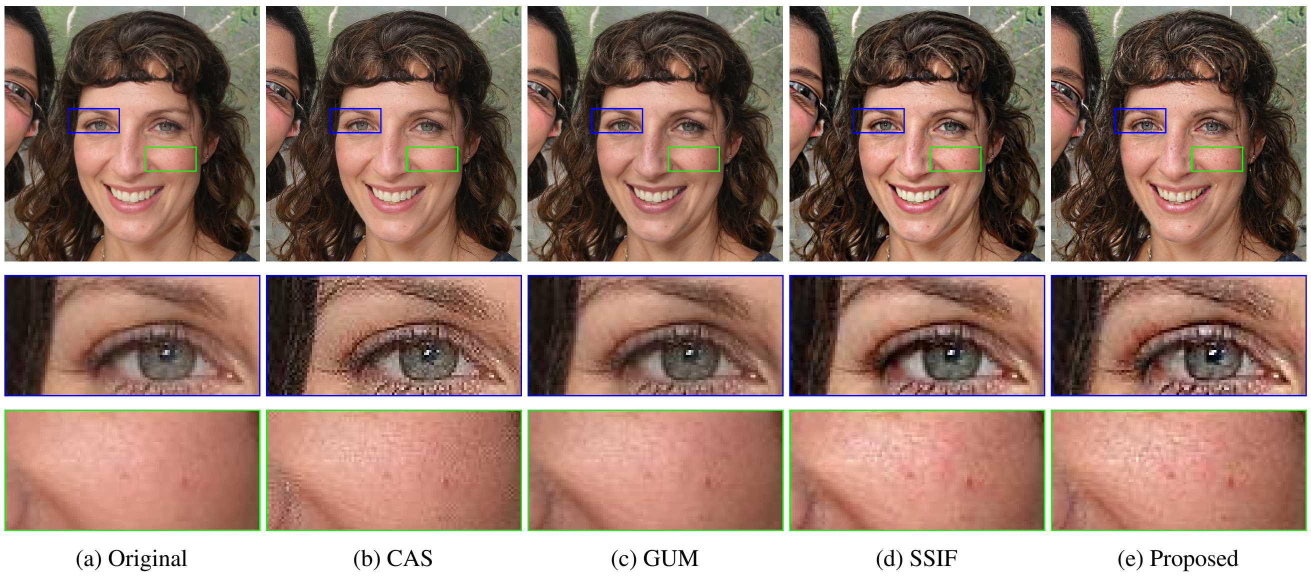


FIGURE 15: Detail enhancement. (a) Original image. (b) CAS results. (c) GUM results. (d) SSIF results. (e) Proposed filter results ($\sigma_s = 0.5, s = 0.01, N_{\text{iter}} = 10, \gamma = 2$).

application. Both qualitative and quantitative comparison are performed to validate the proposed filter.

The first method called GFF [30] decomposes each source image into a base layer and a detail layer. Base layers and detail layers of source images are then fused individually using a weighted average technique. The weight map is calculated based on the salience map which is refined by using the guided filter. The resulting base and detail layers are used to finally reconstruct the fused image.

The second method called GFDF [15] performs a pixel-based weighted linear combination of the source images. First, a rough focus map for each source image is estimated by subtracting the image from a filtering result. The rough map is then refined using the guided filter. A decision map is generated by applying a pixel-based maximum rule. It is also refined by using another instance of the guided filter. The refined decision map is used as the weight map for the linear combination that fuses the input images.

To demonstrate the performance of the proposed filter in this application, we implemented the GFF[16] and GFDF[17] filters in MATLAB. In our implementation, we replace the guided filter with the proposed filter. We use the terms Proposed filter (1) and Proposed filter (2) to represent the algorithms of GFF and GFDF which use the proposed filter, respectively. Figures 16 and 17 show the fusion results for the two pairs of images from the Lytro dataset [56]. Input A and Input B focus on the foreground and background, respectively. The fusion result applying the original GFF [16] and GFDF [15] are shown in columns (c) and (d) respectively. The result of using the proposed filter in the GFF and GFDF algorithms are displayed in columns (e) and (f). We can see that the proposed filter produces similar results to those of the guided filter.

We perform an objective comparison using five metrics to evaluate the quality of the fusion result without a reference as suggested in [15]. These metrics are:

- Q_G [49] evaluates the amount of edge information transferred from the source images to the fusion result.
- Q_P [50] measures the edge information transferred from the source to the fusion result by using phase congruence.
- Q_Y [51] measures the degradation of structural informa-

[16]https://github.com/funboarder13920/image-fusion-guided-filtering
[17]https://github.com/bitname/Multi-focus-image-fusion-GFDF

TABLE 3: Multi-focus image fusion quality measurement for Fig. 16 and Fig. 17.

Image	Fusion Method	Q_G [49]	Q_P [50]	Q_Y [51]	Q_{CB} [52]	Q_{FMI} [53]
Cookie	GFF	0.6886	0.8520	0.9630	0.7328	0.5035
	Proposed filter (1)	0.6905	0.8555	0.9643	0.7330	0.5073
	GFDF	0.7117	0.8635	0.9884	0.7944	0.5963
	Proposed filter (2)	0.7128	0.8631	0.9887	0.7939	0.5962
Book	GFF	0.6133	0.8764	0.9078	0.7418	0.5118
	Proposed filter (1)	0.6341	0.8829	0.9207	0.7429	0.5229
	GFDF	0.6799	0.8935	0.9855	0.8057	0.6309
	Proposed filter (2)	0.6827	0.8950	0.9855	0.8040	0.6296

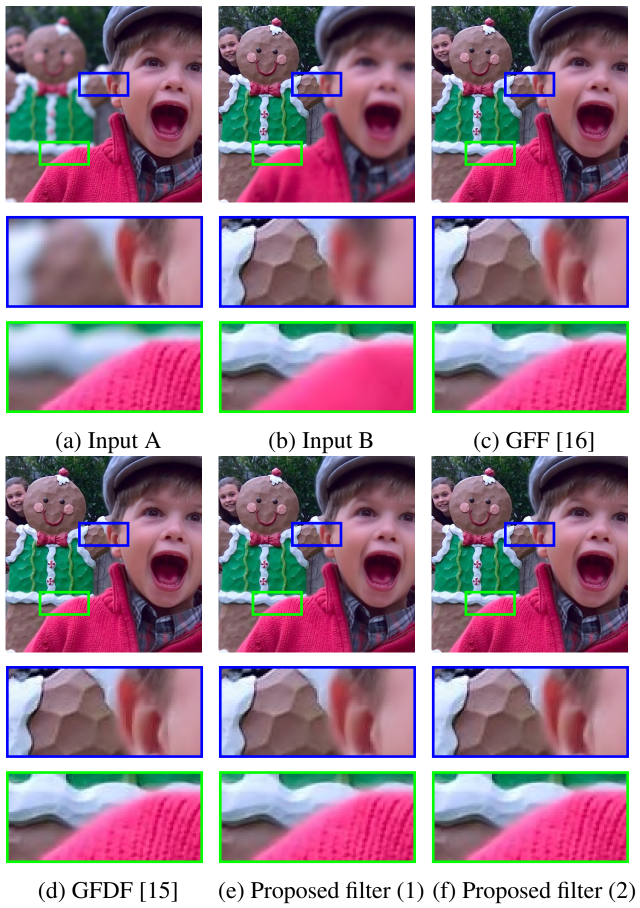


FIGURE 16: A comparison of multi focus image fusion with a group of state-of-the-art algorithms. (a) Input image A, (b) Input image B, (c) GFF ($r_1 = 45, \epsilon_1 = 0.3, r_2 = 7, \epsilon_2 = 10^{-6}$), (d) GFDF ($r = 5, \epsilon = 0.3, w = 7$), (e) Proposed filter (1) ($\sigma_{s1} = 3.5, s_1 = 10, N_{iter1} = 2, \sigma_{s2} = 3.5, s_2 = 10, N_{iter2} = 2$), (f) Proposed filter (2) ($\sigma_s = 5, s = 1, N_{iter} = 1$).

tion of an image with respect to another image by using the structural similarity [57] between the source images and the fusion result.

- Q_{CB} [52] performs a perceptual quality evaluation of the fusion result by using a local contrast and saliency map.
- Q_{FMI} [53] measures the mutual information between the feature map of the fusion image and the feature map of the source images with small windows and average all the results to get a single value.

Table 3 shows the results of the quantitative assessments of images shown in Fig. 16 and Fig. 17. For all five metrics higher values indicate higher fusion quality. We can see that, in general, using our filter produces similar values as those using the guided filter.

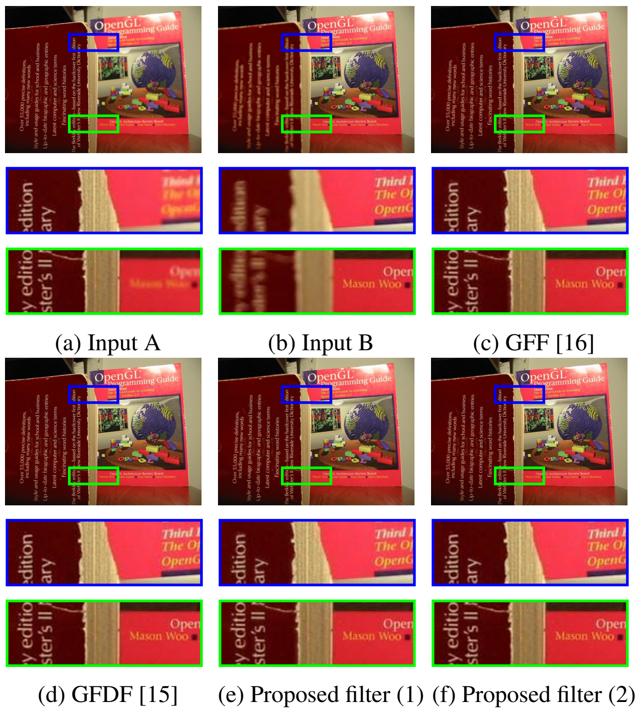


FIGURE 17: A comparison of multi-focus image fusion with a group of state-of-the-art algorithms. (a) Input image A, (b) Input image B, (c) GFF ($r_1 = 45, \epsilon_1 = 0.3, r_2 = 7, \epsilon_2 = 10^{-6}$), (d) GFDF ($r = 5, \epsilon = 0.3, w = 7$), (e) Proposed filter (1) ($\sigma_{s1} = 3.5, s_1 = 10, N_{iter1} = 1, \sigma_{s2} = 3.5, s_2 = 10, N_{iter2} = 1$), (f) Proposed filter (2) ($\sigma_s = 3.5, s = 1, N_{\text{iter}} = 1$).

V. CONCLUSION

In this paper, we have presented a new edge-preserving filter which is based on the local weighted averaging structure and statistics of the image. The new feature of this filter is the use of a decreasing function of the local variance as the weight. As a result, the filter has a computational complexity of $\mathcal{O}(N_{pix})$. We have motivated the development of this filter by taking an extreme parameter setting of the guided filter and have performed statistical analysis and simulations. Results not only show the connections between the proposed filter, the bilateral filter and the guided filter, but also provides new insights into the edge-preserving ability and the computational complexity of the proposed filter. In addition, we have presented extensions to the proposed filter using the ideas of bilateral weight, guidance information and iteration.

The edge-preservation performance of the proposed filter has been demonstrated in many applications including: edge-preserving smoothing, non-photo realistic image rendering, compression artifact removal, detail magnification, edge extraction, multi-focus image fusion, structure separation, and salience detection. We have shown by using many images and objective evaluation metrics (where they are available) that the performance of the proposed filter is comparable or superior to state-of-the-art filters. Therefore, the proposed filter is a new tool for tackling a wide range of image processing problems.

APPENDIX. IMPLEMENTATION

MATLAB implementation of the proposed filter. We have also released the code and applications on Github[18].

```
function [J,w] = GAVWA(I,G,SigmaS,scale)

%input: I -- image to be processing I \in [0,1]
%        G -- guidance image G \in [0,1]
%        SigmaS -- bilateral spacial parameter,
%        scale -- bilateral range parameter,

% patch size
patchSize = floor(4*SigmaS) + 1;
if mod(patchSize,2) == 0
   patchSize = patchSize + 1;
end
N = patchSize*patchSize;

% normalized average kernel
h = ones(patchSize)/N;
% Gaussian Kernel
g = fspecial('gaussian',patchSize,SigmaS);

% patch mean of G
muG = imfilter(G, h,'symmetric');
% patch mean of G.*G
muGG = imfilter(G.*G, h,'symmetric');

% patch var of G
SigmaG= max(max(0,muGG - muG.*muG),[],3);

% weight calculation
SigmaR = scale*mean(SigmaG(:)); %eq. (24-25)
w = 1./(1+(SigmaG./SigmaR).^2); %eq. (28)

% normalization factor
nF = imfilter(w,g,'symmetric');

% for color images
if size(I, 3) == 3 % For color images.
    w = cat(3,w,w,w); %make it 3d
    nF = cat(3,nF,nF,nF);
end

% final output
J = imfilter(w.*I,g,'symmetric')./(eps+nF);%eq. 26
end
```

REFERENCES

[1] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, Bilateral filtering: Theory and applications. Now Publishers Inc, 2009.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in Proc. ICCV, pp. 839–846, 1998.

[3] L. P. Yaroslavsky, Digital picture processing. Springer Berlin Heidelberg, 1985.

[4] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," ACM Trans. Graph., vol. 27, no. 3, pp. 1–10, 2008.

[5] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," ACM Trans. Graph., vol. 30, no. 4, pp. 1–12, 2011.

[6] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," ACM Trans. Graph., vol. 30, no. 6, pp. 1–12, 2011.

[7] L. Xu, Q. Yan, Y. Xia, and J. Jia, "Structure extraction from texture via relative total variation," ACM Trans. Graph., vol. 31, no. 6, pp. 1–10, 2012.

[18]https://github.com/fergaletto/GVWA

[8] K. He, J. Sun, and X. Tang, "Guided image filtering," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 6, pp. 1397–1409, 2013.

[9] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang, "Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree," IEEE Trans. Image Process., vol. 23, no. 2, pp. 555–569, 2013.

[10] Q. Zhang, X. Shen, L. Xu, and J. Jia, "Rolling guidance filter," in Proc. ECCV, pp. 815–830, Springer, 2014.

[11] H. Cho, H. Lee, H. Kang, and S. Lee, "Bilateral texture filtering," ACM Trans. Graph., vol. 33, no. 4, pp. 1–8, 2014.

[12] M. Al-nasrawi and G. Deng, "Modified iterative guided texture filtering algorithm," Comput. Graph., vol. 79, pp. 81–100, 2019.

[13] W. Li, W. He, X. Ou, W. Hu, J. Wu, and G. Zhang, "Fast combination filtering based on weighted fusion," J. Vis. Commun. Image R., vol. 62, pp. 226–233, 2019.

[14] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," ACM Trans. Graph., vol. 25, no. 3, pp. 1221–1226, 2006.

[15] X. Qiu, M. Li, L. Zhang, and X. Yuan, "Guided filter-based multi-focus image fusion through focus region detection," Signal Process. Image Commun., vol. 72, pp. 35–46, 2019.

[16] S. Li, X. Kang, and J. Hu, "Image fusion with guided filtering," IEEE Trans. Image Process., vol. 22, no. 7, pp. 2864–2875, 2013.

[17] P. Milanfar, "A tour of modern image filtering: New insights and methods, both practical and theoretical," IEEE Signal Process. Mag., vol. 30, no. 1, pp. 106–128, 2013.

[18] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu, "Weighted guided image filtering," IEEE Trans. Image Process., vol. 24, no. 1, pp. 120–129, 2015.

[19] F. Kou, W. Chen, C. Wen, and Z. Li, "Gradient domain guided image filtering," IEEE Trans. Image Process., vol. 24, no. 11, pp. 4528–4539, 2015.

[20] B. Chen and S. Wu, "Weighted aggregation for guided image filtering," Signal Image Video Process., vol. 14, no. 3, pp. 491–498, 2019.

[21] H. Yin, Y. Gong, and G. Qiu, "Side window guided filtering," Signal Process., vol. 165, pp. 315–330, 2019.

[22] Z. Sun, B. Han, J. Li, J. Zhang, and X. Gao, "Weighted guided image filtering with steering kernel," IEEE Trans. Image Process., vol. 29, pp. 500–508, 2020.

[23] C. N. Ochotorena and Y. Yamashita, "Anisotropic guided filtering," IEEE Trans. Image Process., vol. 29, pp. 1397–1412, 2020.

[24] G. Deng, "Edge-aware BMA filters," IEEE Trans. Image Process., vol. 25, no. 1, pp. 439–454, 2016.

[25] L. Karacan, E. Erdem, and A. Erdem, "Structure-preserving image smoothing via region covariances," ACM Trans. Graph., vol. 32, no. 6, pp. 1–11, 2013.

[26] B. Ham, M. Cho, and J. Ponce, "Robust image filtering using joint static and dynamic guidance," in Proc. CVPR, pp. 4823–4831, 2015.

[27] Z. Zhou, B. Wang, and J. Ma, "Scale-aware edge-preserving image filtering via iterative global optimization," IEEE Trans. Multimedia, vol. 20, no. 6, pp. 1392–1405, 2018.

[28] R. Fattal, "Edge-avoiding wavelets and their applications," ACM Trans. Graph., vol. 28, no. 3, pp. 1–10, 2009.

[29] G. Deng, "Guided wavelet shrinkage for edge-aware smoothing," IEEE Trans. Image Process., vol. 26, no. 2, pp. 900–914, 2016.

[30] X.-Y. Li, Y. Gu, S.-M. Hu, and R. R. Martin, "Mixed-domain edge-aware image manipulation," IEEE Trans. Image Process., vol. 22, no. 5, pp. 1915–1925, 2013.

[31] S. Paris, S. W. Hasinoff, and J. Kautz, "Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid," ACM Trans. Graph., vol. 30, no. 4, pp. 81–91, 2011.

[32] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models," Acta Numerica, vol. 28, pp. 1–174, 2019.

[33] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in Proc. ICCV, pp. 479–486, 2011.

[34] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in Proc. CVPR, pp. 9446–9454, 2018.

[35] L. Xu, J. S. J. Ren, Q. Yan, R. Liao, and J. Jia, "Deep edge-aware filters," in Proc. ICML, pp. 1669–1678, 2015.

[36] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1256–1272, 2017.

[37] E. Eisemann and F. Durand, "Flash photography enhancement via intrinsic relighting," in Proc. ACM SIGGRAPH, pp. 673–678, 2004.

[38] L. Zhao, H. Bai, J. Liang, A. Wang, B. Zeng, and Y. Zhao, "Local activity-driven structural-preserving filtering for noise removal and image smoothing," Signal Processing, vol. 157, pp. 62–72, 2019.
[39] X. Guo, Y. Li, J. Ma, and H. Ling, "Mutually guided image filtering," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 3, pp. 694–707, 2018.
[40] L. Zhao, H. Bai, A. Wang, and Y. Zhao, "Iterative range-domain weighted filter for structural preserving image smoothing and de-noising," Multimedia Tools and Appl., vol. 78, no. 1, pp. 47–74, 2019.
[41] C. Liu, C. Yang, M. Wei, and J. Wang, "Texture smoothing quality assessment via information entropy," IEEE Access, vol. 8, pp. 88410–88421, 2020.
[42] H. Lee, J. Jeon, J. Kim, and S. Lee, "Structure-texture decomposition of images with interval gradient," Comput. Graph. Forum, vol. 36, pp. 262–274, 2017.
[43] J. Jeon, H. Lee, H. Kang, and S. Lee, "Scale-aware structure-preserving texture filtering," Comput. Graph. Forum, vol. 35, no. 7, pp. 77–86, 2016.
[44] M. Al-Nasrawi, G. Deng, and W. Waheed, "Structure extraction of images using anisotropic diffusion with directional second neighbour derivative operator," Multimedia Tools Appl., vol. 78, no. 5, pp. 6385–6407, 2019.
[45] L. Xu, F. Wang, L. Dempere-Marco, Q. Wang, Y. Yang, and X. Hu, "Path-based analysis for structure-preserving image filtering," J. Math. Imaging Vis., vol. 62, no. 2, pp. 253–271, 2020.
[46] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li, "Salient object detection: A Discriminative Regional Feature Integration Approach," in Proc. CVPR, pp. 2083–2090, 2013.
[47] G. Deng, "A generalized unsharp masking algorithm," IEEE Trans. on Image Process., vol. 20, no. 5, pp. 1249–1261, 2010.
[48] G. Deng, F. Galetto, M. Al–nasrawi, and W. Waheed, "A guided edge-aware smoothing-sharpening filter based on patch interpolation model and generalized gamma distribution," IEEE Open Journal of Signal Processing, vol. 2, pp. 119–135, 2021.
[49] C. S. Xydeas and V. S. Petrovic, "Objective pixel-level image fusion performance measure," in Proc. Sensor Fusion: Architectures, Algorithms, and Applications IV, vol. 4051, pp. 89–98, International Society for Optics and Photonics, 2000.
[50] J. Zhao, R. Laganiere, and Z. Liu, "Performance assessment of combinative pixel-level image fusion based on an absolute feature measurement," Int. J. Innov. Comput. I., vol. 3, no. 6, pp. 1433–1447, 2007.
[51] C. Yang, J.-Q. Zhang, X.-R. Wang, and X. Liu, "A novel similarity based quality metric for image fusion," Inform. Fusion, vol. 9, no. 2, pp. 156–160, 2008.
[52] Y. Chen and R. S. Blum, "A new automated quality assessment algorithm for image fusion," Image Vision Comput., vol. 27, no. 10, pp. 1421–1432, 2009.
[53] M. Haghighat and M. A. Razian, "Fast-FMI: non-reference image fusion metric," in Proc. AICT, pp. 1–3, IEEE, 2014.
[54] S. Li, X. Kang, L. Fang, J. Hu, and H. Yin, "Pixel-level image fusion: A survey of the state of the art," Inform. Fusion, vol. 33, pp. 100–112, 2017.
[55] Y. Liu, L. Wang, J. Cheng, C. Li, and X. Chen, "Multi-focus image fusion: A Survey of the state of the art," Inform. Fusion, vol. 64, pp. 71–91, 2020.
[56] M. Nejati, S. Samavi, and S. Shirani, "Multi-focus image fusion using dictionary-based sparse representation," Inform. Fusion, vol. 25, pp. 72–84, 2015.
[57] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. Image Process., pp. 600–612, 2004.

FERNANDO J. GALETTO received his B.Eng. degree in electronics engineering from National Technological University, Cordoba, Argentina in 2015, and the M.Eng degree in electronics engineering in 2019 from La Trobe University, Melbourne, Australia, where he is currently working toward the Ph.D. His research interests include computer vision and underwater image processing.

GUANG DENG is currently a Reader and Associate Professor of electronic engineering with La Trobe University, Melbourne, VIC, Australia. His current research interests include Bayesian signal processing, lossless image compression and generalized linear systems.

MUKHALAD AL-NASRAWI received his B.S. and M.E. degrees in University of Baghdad, Iraq, in 2002 and 2008 respectively, and the Ph.D degree from Department of Electronic Engineering, La Trobe University, Melbourne, Australia, in 2018. His research interests include signal and image processing, machine learning, edge-aware smoothing, and computer vision.

WASEEM WAHEED received the B.Sc. degree from the Department of Electrical Engineering, Salahaddin University, Erbil, Iraq, in 2009, and the master's and Ph.D. from La Trobe University, Melbourne, VIC, Australia, in 2014 and 2020, respectively. His research interests include signal and image processing, computer vision, deep learning, and optimization techniques.