Abstract

When applying a filter to an image, it often makes practical sense to maintain the local brightness level from input to output image. This is achieved by normalizing the filter coefficients so that they sum to one. This concept is generally taken for granted, but is particularly important where nonlinear filters such as the bilateral or non-local means are concerned, where the effect on local brightness and contrast can be complex. Here we present a method for achieving the same level of control over the local filter behavior without the need for this normalization. Namely, we show how to closely approximate the any normalized filter without in fact needing this normalization step. We derive a closed-form expression for the approximating filter and analyze its behavior, showing it to be easily controlled for quality and nearness to the exact filter, with a single parameter.

1. Introduction and Background

Edge-aware filters are constructed using kernels that are computed from the given image. The adaptation of the filters to local variations in the image is what endows them with the power and flexibility to treat different parts of the image differently. This adaptability, however, can not be arbitrary. In particular, the local brightness of the image must often be maintained in order to yield a reasonable global appearance. The standard way to this achieved is by normalizing the filter coefficients pointing to each pixel, so that they sum to 1. In this paper we propose a new and different way. Namely, we present a general method to approximate any normalized filter with one that does not require normalization. This produces a rather simpler filter structure, but with essentially the same functionality.

Approximation ideas centered around nonlinear filters are not new. In particular, the bilateral filter has been subject to various interesting algorithmic attacks [5] which have resulted in significantly improved computational complexity with almost no loss in quality. Current state of the art algorithms are able to do bilateral filtering efficiently enough that cost is roughly proportional to image resolution, especially when the support of the filter kernel is very large\(^1\) [1, 2]. Very recently, the idea of using an un-normalized form of the bilateral filter specifically was used in [3] to produce fast local Laplacian filters. As will become clear, [3] is a special case of the broader concept we propose here.

What we propose here is not another approximation to the bilateral filter. Our treatment works equally well for any normalized filter that has a well-defined kernel; bilateral, non-local means, etc. being just a few popular examples. Our filter avoids local normalization of the filter coefficients, while remaining close in its effect to the base filter. This approximation has several interesting properties. First, the fidelity of the approximation is guaranteed since it is derived from an optimality criterion; furthermore, this fidelity can be controlled easily with a single parameter regardless of the form of the base filter (e.g. bilateral, non-local mean, etc.) Second, the approximate filter is guaranteed to maintain the average gray level just as the base filter would, regardless of tuning. Finally, the approximate filter is easy to analyze and provides intuitively pleasing structure for understanding the behavior of general image-dependent filters. By way of practical motivation, the approximation allows us to start with an arbitrary (normalized) base filter and generate a one-parameter family of simpler nearby filters, which can locally modulate the effect of the base filter. This is different, and more flexible, than the typical approach where the base filters (bilateral, NLM, etc) are controlled with global smoothing parameters. For various applications such as texture-cartoon decomposition, guided filtering, and local (e.g. Laplacian) tone mapping, and even noise suppression, the additional flexibility afforded can be very useful.

Before moving forward, we establish our notation. Consider an image \(Y\) of size \(\sqrt{n} \times \sqrt{n}\) as the input, and the image \(Z\) as the output of the filtering process. We will scan these images into vectors with, say, a column-stacking or-

---

1 These approaches however, do not generalize easily to kernels beyond the bilateral.
Gaussian Filters \[11, 7\] Measuring only the Euclidean extend to any filter with an SPD kernel, some popular ex-
y color value (denoted by \(x\)), but more importantly, also using the gray or color value (denoted by \(y\)). While the results of this paper extend to any filter with an SPD kernel, some popular examples commonly used in the image processing, computer vision, and graphics literature are as follows:

**Gaussian Filters** \[11, 7\] Measuring only the Euclidean (spatial) distance between pixels, the classical Gaussian kernel is

\[
k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2}\right). \tag{1}
\]

These kernels lead to the classical and well-worn Gaussian filters (including shift-varying versions).

**Bilateral (BL)** \[10, 6\] and **Non-local Mean (NLM)** \[4, 8\]

These filters take into account both the spatial and value distances between two pixels, generally in a separable fashion. For BL we have:

\[
k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2}\right) \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2}\right), \tag{2}
\]

As seen in the overall exponent, the similarity metric here is a weighted Euclidean distance between the concatenated vectors \((\mathbf{x}_i, \mathbf{y}_i)\) and \((\mathbf{x}_j, \mathbf{y}_j)\).

The NLM kernel is a generalization of the bilateral kernel in which the value distance term (1) is measured patch-wise instead of point-wise:

\[
k_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{h_x^2}\right) \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2}\right), \tag{2}
\]

where \(\mathbf{y}_i\) and \(\mathbf{y}_j\) refer now to subsets of samples (i.e. patches) in \(\mathbf{y}\).

These affinities are not used directly to filter the images, but instead in order to maintain the local average brightness, they are normalized so that the resulting weights pointing to each pixel sum to one. More specifically,

\[
w_{ij} = \frac{k_{ij}}{\sum_{j=1}^{n} k_{ij}}, \tag{3}
\]

where each element of the filtered signal \(z\) is then given by

\[
z_i = \sum_{j=1}^{n} w_{ij} y_j.
\]

It is worth noting that the denominator in (3) can be computed by simply applying the filter (without normalization) to an image of all 1’s.

In matrix notation, the collection of the weights used to produce the \(i\)-th output pixel is the vector \([w_{i1}, \cdots, w_{in}]\), and this can in turn be placed as the \(i\)-th row of a filter matrix \(W\) so that

\[
z = Wy.
\]

We note again that due to the normalization of the weights, the rows of the matrix \(W\) sum to one. That is, for each \(1 \leq i \leq n\),

\[
\sum_{j=1}^{n} w_{ij} = 1.
\]

Viewed another way, the filter matrix \(W\) is a normalized version of the symmetric positive definite affinity matrix \(K\) constructed from the unnormalized affinities \(k_{ij}, 1 \leq i, j \leq n\). As a result, \(W\) can be written as a product of two matrices

\[
W = D^{-1} K, \tag{4}
\]

where \(D\) is a diagonal matrix with diagonal elements \(D_{ii} = \sum_{j=1}^{n} k_{ij} = d_i\). To avoid the normalization, we will replace the filter \(W\) with an approximation \(\tilde{W}\) that only involves \(D\) rather than its inverse. More specifically,

\[
\tilde{W} = I + \alpha(K - D). \tag{5}
\]

But why is this a good idea? In what follows, we will motivate and derive this approximation from first principles, while also providing an analytically sound and numerically tractable choice for the scalar \(\alpha > 0\) that gives the best approximation to \(W\) in the least-squares sense. Before doing so, it is worth noting some of the key properties and advantages of this approximate filter which are evident from the above expression (5).

- Regardless of the value of \(\alpha\), the rows of \(\tilde{W}\) always sum to one. That is, like its counterpart \(W\) constructed with \(D^{-1}\), the approximation \(\tilde{W}\), constructed with only \(D\), is automatically normalized. This can be easily seen by multiplying \(\tilde{W}\) on the right by a vector of ones, and observing that it returns the same vector back regardless of \(\alpha\).

- While the filter \(W\) is not symmetric due to the multiplicative normalization (see Eq. (4)), the approximant \(\tilde{W}\) is always symmetric, again regardless of \(\alpha\). The advantages of having a symmetric filter matrix are many, as documented in the recent work \[9\].

- The SPD affinity matrix \(K\) is typically also non-negative valued, leading to filter weights in \(W\) which are also in turn non-negative valued. The elements in
However, can be negative valued due to the term $K - D$. This means that the behavior of the approximate filter may differ from its reference value, and must be carefully studied and controlled. We will do this in this paper.

2. The Normalization-free Filter $\widehat{W}$

To derive the approximation promised in the previous section, we first note that the standard filter can be written as:

$$W = I + D^{-1}(K - D)$$

Comparing this form to the one presented earlier in (5), we note that the approximation is replacing the matrix inverse (on the right hand side) with a scalar multiple of the identity:

$$D^{-1} \approx \alpha I$$

As an illustration, an image containing the normalization terms $d_i$ (which comprise the diagonal elements of $D$) for the photo in Fig. 4, are shown in Fig. 2. The proposal, as we elaborate below, is to replace these normalization constants any given pixel of the output image. To construct the pixel $z_i$ of the output from the input image, the exact weights used (in the $j$-th row of $W$) are:

$$z_i = w_i^T y = \frac{1}{d_i} [w_{i1}^T, \cdots, w_{iN}^T] y$$

In contrast to this, the approximate filter uses the weights

$$\hat{z}_i = \hat{w}_i^T y = \alpha [k_{i1}, \cdots, k_{in}] y$$

Note that the center (self) weight corresponding to the position of interest $i$ has been changed most prominently, and the other weights are pushed in the opposite direction as the change in this weight in order to maintain the sum as 1. The center weight in fact can become negative, while the other weights must remain positive.

Another way to make the comparison is more illustrative. Define the shifted Dirac delta vector $\delta_i = [0, 0, \cdots, 0, 1, 0, \cdots, 0]$ where the subscript $i$ indicates that the value 1 occurs in the $i$-th position. We have

$$\hat{w}_i^T = \delta_i + \alpha [(k_{i1}, \cdots, k_{in}) - d_i \delta_i]$$

Rewriting this last expression we have a rather simple relationship between the exact and approximated filter coefficients:

$$\hat{w}_i^T - \delta_i = \alpha d_i (w_i^T - \delta_i)$$

For sufficiently large $n$, the terms $\text{tr}(D)$ and $\text{tr}(D^2)$ dominate the numerator and the denominator, respectively. Hence,

$$\hat{\alpha} \approx \frac{\text{tr}(D)}{\text{tr}(D^2)} = \frac{s_1}{s_2},$$

where

$$s_1 = \sum_{i=1}^{n} d_i,$$

$$s_2 = \sum_{i=1}^{n} d_i^2$$

This ratio is in fact bounded as $\frac{1}{n} \leq \frac{s_1}{s_2} \leq \frac{1}{d}$, which for large $n$ justifies a further approximation:

$$\hat{\alpha} \approx \frac{1}{d}$$

where $\overline{d} = \text{mean}(d_i)$.

3. Properties of $\widehat{W}$

3.1. Pixel- and Frequency-Domain Behavior

As a matter of practical importance, we look at how the approximation changes the weights applied for computing any given pixel of the output image. To construct the pixel $z_i$ of the output from the input image, the exact weights

$$\hat{\alpha} \approx \frac{\text{tr}(D)}{\text{tr}(D^2)} = \frac{s_1}{s_2},$$

where

$$s_1 = \sum_{i=1}^{n} d_i,$$

$$s_2 = \sum_{i=1}^{n} d_i^2$$

This ratio is in fact bounded as $\frac{1}{n} \leq \frac{s_1}{s_2} \leq \frac{1}{d}$, which for large $n$ justifies a further approximation:

$$\hat{\alpha} \approx \frac{1}{d}$$

where $\overline{d} = \text{mean}(d_i)$.
controls the difference between the exact and the approximated filter. In particular, if at pixel location \( i \) the normalization factor \( d_i \) is close to the mean \( \bar{d} \), then at that pixel, the approximation is nearly perfect. More generally, gathering all terms like (16), the respective filter matrices are related as

\[
(W - I) = R(W - I)
\]

where \( R = \alpha D \).

Canonically, if we consider the two filters \( W \) and \( \hat{W} \) as edge-aware low-pass (or smoothing) filters, then their counter-parts \( W - I \) and \( \hat{W} - I \) are high pass filters. In fact, these filters are directly related to different (but well-established) definitions of the graph Laplacian operator emerging from the same affinity matrix \( K \). Namely, \( \mathcal{L} = I - D^{-1}K = I - W \) is known as the random walk Laplacian \([9]\), whereas \( \hat{\mathcal{L}} = D - K = (I - \hat{W})/\alpha \) is known as the un-normalized graph Laplacian \([9]\).

So what does all this tell us about how approximating the filter distorts the output image? As (17) makes clear, the distortion is concentrated in the higher-frequency components of the output. Furthermore, the degree of distortion given pixel-wise by the ratio \( d_j/\bar{d} \). The overall distortion is small when the coefficients \( d_j \) are tightly concentrated around the mean \( \bar{d} \).

4. Two Interpretations of Approximate Filter

The approximate filter lends itself to various interpretations which can help us better understand its effect. The derivations follow directly from the definition.

- First, consider

\[
\hat{W} = I + \alpha(K - D) = I + \alpha D(W - I)
\]

Applying this filter to the image \( y \), we have

\[
\hat{W}y = y + \alpha D(Wy - y)
\]

Moving \( y \) to the left-hand side, we have

\[
\hat{W}y - y = \alpha D(Wy - y)
\]

This shows that the residuals of the two filtering processes (exact vs. approximate) are simply pixel-wise scaled versions of one another, by the diagonal matrix \( \alpha D \).

- A slightly different manipulation yields

\[
\hat{W}y = (I - \alpha D)y + \alpha DWy
\]

\[
\hat{z} = (I - \alpha D)y + \alpha Dz
\]

This shows that the un-normalized filter is a pixel-wise blending of the input image and the output of the normalized filter. The coefficients of this blend are \( 1 - \alpha d_i \) and \( \alpha d_i \), for each pixel. It is tempting to think of this as a convex combination. But while the numbers \( \alpha d_i \) are clustered around 1, they are not restricted to the range \([0, 1]\).

5. Effect of Approximation on Local Variance

So far, our concern has been with maintaining the mean brightness value in the output pixels of the filter. But we also expect that the approximate filter should affect the variance of the output pixels. Here we characterize this effect. Recall the pixel-wise expressions for the exact and approximate filter, respectively:

\[
z_i = \sum_{j=1}^{n} w_{ij} y_j, \quad \hat{z}_i = \sum_{j=1}^{n} \hat{w}_{ij} y_j
\]

The variance in the output pixel in terms of the variance in the input pixel is given by the sum-squared of the filter weights. That is,

\[
\text{var}(z_i) = \left( \sum_{j=1}^{n} w_{ij}^2 \right) \text{var}(y_i) = \nu_i \text{var}(y_i) \tag{24}
\]

\[
\text{var}(\hat{z}_i) = \left( \sum_{j=1}^{n} \hat{w}_{ij}^2 \right) \text{var}(y_i) = \hat{\nu}_i \text{var}(y_i) \tag{25}
\]

It is of interest to establish a relationship between the factors \( \nu_i \) and \( \hat{\nu}_i \). We proceed as follows:

\[
\hat{\nu}_i = \hat{w}_i^T \hat{w}_i
\]

\[
= (\delta_i + \alpha^2 d_i^2 (w_i - \delta_i))^T (\delta_i + \alpha^2 d_i^2 (w_i - \delta_i)) \tag{26}
\]

\[
= \alpha^2 d_i^2 \nu_i + (\alpha^2 d_i^2 - 2\alpha(1+\alpha)d_i + 1 + 2\alpha) \tag{27}
\]

\[
\approx (\alpha d_i)^2 \nu_i + (\alpha d_i - 1)^2 \tag{28}
\]

where the last approximation assumes \( \alpha \ll 1 \). Denoting \( \rho_i = \alpha d_i \) we recall that this number typically hovers around 1. We note that \( \frac{1}{n} \leq \nu_i \leq 1 \), but \( \hat{\nu}_i \) can grow larger than 1 if \( \rho_i > 1 \). To summarize, the two variance factors are linearly related as follows:

\[
\hat{\nu}_i \approx \rho_i^2 \nu_i + (\rho_i - 1)^2 \tag{29}
\]

The contour plot in Fig. 1, shows the values of \( \hat{\nu}_i \) as a function of \( \rho_i \) and \( \nu_i \).

6. Experiments

As an illustration of the concepts, we refer the reader to Fig. 4. At left, an image of an old man is shown (size
505 × 578) along with two (bilateral-) filtered versions. At center is the exact bilateral filtered image, whereas on the right is the approximated filter. Visually, we note the approximated filter produces an image that is less aggressively smoothed. To better see this difference, the (luma channel) difference between the two output images is shown at left in Fig. 5. In the center and right panels of the same figure, the residuals of each filtered image against the input are shown for reference. For this example, the estimated \( \hat{\alpha} = 0.0108 \), corresponding to \( \hat{\sigma} = 92.88 \). In Figure 2, we can observe the values of \( d_j \) for this image, where the vast majority are in fact close to the mean value. The more "rare" pixels (i.e. those with small total affinities \( d_j \)) appear as blue, and indeed correctly predict the locations where the largest differences between the exact and approximated filter occur (see left panel of Fig. 5). The weaker smoothing effect of the approximate filter is not always desirable. For instance, consider "cartoonizing" the image, where we explicitly need to suppress nearly all small details. We carried this out for the old man image using the method in [12]. In Fig. 6, the cartoonized image using the exact BL filter is shown on the left panel. The approximated filter with the optimal \( \hat{\alpha} \) is shown in the middle panel, where too many details are left behind to produce a good result. In order to get a better result, we can boost the value of \( \alpha \) as shown in the right panel. This shows both a weakness of the approximation, but also its flexibility since the shortcoming was relatively easy to overcome by tuning a single parameter \( \alpha \).

A second example will illustrate application to a noisy image. The noisy "Laundromat" image (of size 400 × 600) is shown in Fig. 7. In the top panel is the original image and the left and right panels contain the exact filter and the approximation, respectively. In this case we used a non-local means (NLM) filter. For this image \( \hat{\alpha} = 0.0132 \), corresponding to \( \hat{\sigma} = 76.02 \). The presence of noise gives a smaller average affinity. Figure 3 contains the values of \( d_j \) for this image. In this case we see that the affinity values are less well-concentrated about the mean. But the overall picture still correctly predicts differences between the exact and approximated filter (see left panel of Fig. 8).

In terms of accuracy of the approximation, as shown in Figs. 5 and 8 (left panels), the residuals between the base filtered images and the approximate filtered images are small, and also highly localized. Taken on average, the RGB values across these two images differ by 2/255 in the oldman image and 3.5/255 in the laundry image. These translate to PSNR values between the base-filtered and approximate-filtered images of 42.1 dB and 37.2 dB, respectively.

![Figure 1. Values of \( \hat{\sigma}^2 \) as a function of \( \rho \) (horizontal axis) and \( \nu^2 \) (vertical axis)](image1)

![Figure 2. Values of \( d_j \) for the old man photo. Large values shown in red indicate pixels that have many "nearest neighbors" in the metric implied by the bilateral kernel.](image2)

7. Conclusions and Future Work

We presented a conceptually simple method for approximating a class of normalized linear or non-linear filters with ones that avoid pixel-wise normalization. The approximate filters are easy to construct, and surprisingly accurate. We studied the behavior of the approximated filter and showed how it can be controlled with a single parameter for both quality and nearness to the base filter.

Here we focused narrowly on the conceptual side of developing the new framework and its properties. However, there may also exist computational savings to be realized by avoiding pixelwise normalization. But this will likely depend on the instruction set and the hardware platform used. Speculatively, on standard floating point arithmetic, the savings will be modest. There is also the possibility of computing normalized filters on integer valued images without
having to convert to floats or using integer division altogether. In any case, the concept does have the advantage of being completely general – it can be applied to any base filter where conventional normalization is employed. Whether any computational advantages broadly exist is work that still needs to be carried out.

To quantify the behavior of the approximate filter, we illustrated that the majority of the distortion it can cause is concentrated at high spatial frequencies and around two types of pixels: First are very "rare" pixels that do not enjoy the affinity of many similar pixels (i.e. have few near neighbors as measured by the affinity kernel). The second are very common pixels that have far more similar pixels than average. These typically will correspond to large constant or slowly changing areas of the image, and hence the distortions caused will generally not be visible. In either case, the locations where these distortions occur can be conveniently predicted by computing the sum of the (unnormalized) weights pointing to each pixel and comparing this to the average across the image.

Admittedly, our story is not complete in these few pages – some interesting questions remain open:

- Are there significant computational saving to be had by using the approximated filter?
- It would be interesting to characterize the set of images for which these approximations are most (and least) well suited. One would suspect the presence or absence of texture would play a role.
- The computation of the factors \( d \) can be carried out efficiently by applying the filter, without normalization, to an image of all 1’s. Still, our proposal does not get rid of the computation of \( d \)'s altogether. What it does is bind together the per-pixel normalization factors into one number. Perhaps there is a simpler way to compute \( \hat{\alpha} \) that altogether circumvents the explicit computations of the intermediate values \( d_i \). We conjecture that fixed values of \( \alpha \) would be sufficiently good for large classes of appropriately similar images.

References


Figure 4. (Left) Input $y$; (Center) exact BL filter $z$, and (right) approximate BL filter $\tilde{z}$

Figure 5. Luma channel residuals shown on a common grayscale range $[-35, 35]$: (Left) $z - \tilde{z}$, (Center) $y - z$, (Right) $y - \tilde{z}$

Figure 6. Cartoonized using the method of [12] with the exact filter (left), the approximated filter $\tilde{\alpha} = 0.01$ (center), and $\tilde{\alpha} = 0.03$ (right)
Figure 7. (Top) Input $y$; (left) exact NLM filter $z$, and (right) approximate NLM filter $\hat{z}$.

Figure 8. Luma channel residuals shown on a common grayscale range $[-35, 35]$: (Top) $z - \hat{z}$, (left) $y - z$, (right) $y - \hat{z}$. 