FAST HIGH-DIMENSIONAL FILTERING USING CLUSTERING

Pravin Nair and Kunal N. Chaudhury

Department of Electrical Engineering, Indian Institute of Science

ABSTRACT

Several useful algorithms for image filtering involve non-linear processing of high-dimensional data. Instances of these so-called *high-dimensional filters* are the bilateral, joint-bilateral, and non-local means filters. Real-time implementation of high-dimensional filters can be challenging. In this paper, we present a simple and fast algorithm for generic high-dimensional filtering. The algorithm is based on a *linearization* mechanism, which allows us to approximate the high-dimensional filtering using a series of spatial convolutions. We use clustering for the linearization, whereby we are able to exploit the strong correlation between the components of the high-dimensional image. The highlight of our method is that we can prove that the approximation error (the gap between the fast approximation and the exact filtering) vanishes with the increase in the number of clusters. To the best of our knowledge, this is the first algorithm for high-dimensional filtering that enjoys this theoretical guarantee. In fact, we provide empirical evidence which suggests that this basic requirement is not met by the state-of-the-art Adaptive Manifolds algorithm. We use the proposed algorithm for edge-preserving smoothing and denoising of color and hyperspectral images. The results demonstrate that our algorithm is competitive with existing fast algorithms.

Index Terms— High-dimensional filter, bilateral filter, non-local means, approximation, fast algorithm.

1. INTRODUCTION

High-dimensional filtering forms an important component of several useful algorithms in image processing and computer vision [1, 2, 3]. Consider a high-dimensional image $\boldsymbol{f} : \Omega \to \mathbb{R}^n$, where $\Omega \subset \mathbb{Z}^d$ is the domain of the image and n is the dimension of the intensity range. Specific instances are color images ($d = 2, n = 3$), color videos ($d = 3, n = 3$), flow fields ($d = 3, n = 3$), patch-based image representations ($d = 2$ and n is the patch size), and hyperspectral images ($d = 2$ and n can be of the order of tens or hundreds). The focus of this work is on high-dimensional filtering in which the output $\boldsymbol{g} : \Omega \to \mathbb{R}^n$ is given by

$$\boldsymbol{g}(\boldsymbol{i}) = \frac{1}{\eta(\boldsymbol{i})} \sum_{\boldsymbol{j} \in W} \omega(\boldsymbol{j})\, \phi(\boldsymbol{p}(\boldsymbol{i}-\boldsymbol{j}) - \boldsymbol{p}(\boldsymbol{i}))\boldsymbol{f}(\boldsymbol{i}-\boldsymbol{j}), \qquad (1)$$

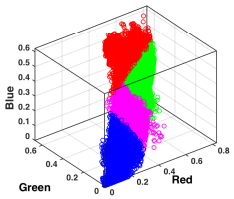
where

$$\eta(\boldsymbol{i}) = \sum_{\boldsymbol{j} \in W} \omega(\boldsymbol{j})\, \phi(\boldsymbol{p}(\boldsymbol{i}-\boldsymbol{j}) - \boldsymbol{p}(\boldsymbol{i})), \qquad (2)$$

and $\omega : \mathbb{R}^d \to \mathbb{R}$ and $\phi : \mathbb{R}^\rho \to \mathbb{R}$ are the *spatial* and *range* kernels. The image $\boldsymbol{p} : \Omega \to \mathbb{R}^\rho$ is referred to as the *guide* image. The aggregations in (1) and (2) are typically performed over a hypercube around the pixel of interest, that is, $W = [-S, S]^d$, where the integer S is the window size. Classical examples of high-dimensional filters include the bilateral, joint-bilateral, and non-local means filters [4, 5, 6]. For the bilateral and joint-bilateral filters, both ω and ϕ are usually Gaussian [4, 5]. For the non-local means filter [6], $\boldsymbol{p}(\boldsymbol{i})$ is a square patch of neighboring pixels around pixel $\boldsymbol{i}$ (more precisely, a patch vector), ϕ is a high-dimensional Gaussian, and ω is a box (constant profile) filter.

E-mail: {nairpravin,kunal}@ee.iisc.ernet.in. This work was supported by a Startup Grant from IISc Bangalore and an EMR Grant SERB/F/6047/2016-2017 from the Department of Science and Technology, Government of India.

(a) Dome [8].



(b) Range space.

Fig. 1. This shows the distribution of RGB values for a natural image, where the dynamic range in each dimension has been normalized to $[0, 1]$. Notice that the distribution is far from uniform and does not populate the entire dynamic range $[0, 1]^3$. Also shown are the four clusters (color coded) obtained using bisecting K-means [18].

The complexity of computing (1) and (2) is $O(nS^d)$ per pixel. In particular, real-time high-dimensional filtering is challenging when d and n are large. Several fast algorithms have been proposed for bilateral filtering of grayscale images ($n = 1$) that can get rid of the exponential term S^d. These fast algorithms are popularly referred to as constant-time or $O(1)$ algorithms in the literature [7]-[14]. Some of these $O(1)$ algorithms were later extended for performing bilateral filtering of color images [8, 15, 16]. The idea of high-dimensional filtering as a generalization of bilateral and non-local means filtering was explicitly put forward in [1, 2, 3]. Moreover, fast algorithms based on sophisticated data structures and query mechanisms were described in [1, 2], and a fast approximation based on adaptive manifolds was proposed in [3]. The latter is widely considered as the state-of-the-art in this area.

The proposed algorithm is motivated by [7, 9, 15], where the range space is quantized to approximate the filter using a series of fast spatial convolutions. The authors in [15] proposed to uniformly quantize the high-dimensional range space. However, as illustrated in Figure 1, this is clearly not optimal since the intensity values of a typical natural image do not fill the entire range space. Based on this observation, we propose to use clustering for quantizing (or binning) the data in high dimensions. This allows us to exploit the correlation between the components of the high-dimensional image. We demonstrate that this simple idea leads to an algorithm that is much less sophisticated than those in [1, 2, 3]. Nevertheless,

its performance for bilateral and non-local means filtering is at par with these methods. We note that the idea of clustering was used in [17] for fast bilateral filtering. The difference with the present proposal is that we use clustering to approximate $\boldsymbol{p}(\boldsymbol{i})$ in (1) and (2), whereas clustering is used in [17] to approximate the neighbouring pixels $\boldsymbol{p}(\boldsymbol{i}-\boldsymbol{j})$. This results in completely different algorithms. The highlight of our method is that we can prove that the gap between the proposed approximation and (1) vanishes with the increase in the number of clusters. This is the first algorithm for high-dimensional filtering that enjoys this guarantee. In fact, we provide empirical evidence which suggests that this basic requirement is not met by the Adaptive Manifolds algorithm. The rest of the paper is organized as follows. We present the proposed approximation and the resulting fast algorithm in Section 2. In Section 3, we use the fast algorithm for bilateral and non-local means filtering of color and hyperspectral images and analyze the results. We conclude the paper in Section 4.

2. PROPOSED ALGORITHM

We now develop a framework for approximating (1) and (2), which leads to a fast algorithm. Define the *range* of the guide $\boldsymbol{p}$ to be

$$\Theta = \{\boldsymbol{p}(\boldsymbol{i}) : \boldsymbol{i} \in \Omega\}. \tag{3}$$

We fix some $K \geq 1$ and partition the set Θ into subsets (clusters) $\mathscr{C}_1, ..., \mathscr{C}_K$; each point from Θ belongs to exactly one cluster. The clustering process will be described shortly. Let us denote the centre (representative) of cluster $\mathscr{C}_i$ by $\boldsymbol{\mu}_i$. The idea is to perform the high-dimensional filtering on a cluster-by-cluster basis. In particular, for $1 \leq k \leq K$, define

$$\boldsymbol{h}_k(\boldsymbol{i}) = \sum_{\boldsymbol{j}\in W} \omega(\boldsymbol{j})\ \phi(\boldsymbol{p}(\boldsymbol{i}-\boldsymbol{j}) - \boldsymbol{\mu}_k)\boldsymbol{f}(\boldsymbol{i}-\boldsymbol{j}), \tag{4}$$

and

$$\zeta_k(\boldsymbol{i}) = \sum_{\boldsymbol{j}\in W} \omega(\boldsymbol{j})\ \phi(\boldsymbol{p}(\boldsymbol{i}-\boldsymbol{j}) - \boldsymbol{\mu}_k). \tag{5}$$

That is, (4) is the numerator of (1), where we have used $\boldsymbol{\mu}_k$ in place of $\boldsymbol{p}(\boldsymbol{i})$. Similarly, (5) is (2) with $\boldsymbol{p}(\boldsymbol{i})$ replaced by $\boldsymbol{\mu}_k$. The point is that, unlike (1) and (2), we can express (4) and (5) as convolutions. Indeed, for $1 \leq k \leq K$, define $\boldsymbol{f}_k : \Omega \to \mathbb{R}^n$ and $\phi_k : \Omega \to \mathbb{R}$ to be

$$\boldsymbol{f}_k(\boldsymbol{i}) = \phi(\boldsymbol{p}(i) - \boldsymbol{\mu}_k)\boldsymbol{f}(\boldsymbol{i}) \quad \text{and} \quad \phi_k(\boldsymbol{i}) = \phi(\boldsymbol{p}(\boldsymbol{i}) - \boldsymbol{\mu}_k).$$

We can then write (4) and (5) as

$$\boldsymbol{h}_k(\boldsymbol{i}) = (\omega * \boldsymbol{f}_k)(\boldsymbol{i}) = \sum_{\boldsymbol{j}\in W} \omega(\boldsymbol{j})\boldsymbol{f}_k(\boldsymbol{i}-\boldsymbol{j}), \tag{6}$$

and

$$\zeta_k(\boldsymbol{i}) = (\omega * \phi_k)(\boldsymbol{i}) = \sum_{\boldsymbol{j}\in W} \omega(\boldsymbol{j})\phi_k(\boldsymbol{i}-\boldsymbol{j}). \tag{7}$$

We define the approximation of (1) to be the image $\hat{\boldsymbol{g}}_K : \Omega \to \mathbb{R}^n$ given by

$$\hat{\boldsymbol{g}}_K(\boldsymbol{i}) = \frac{\boldsymbol{h}_k(\boldsymbol{i})}{\zeta_k(\boldsymbol{i})} \qquad \big(\boldsymbol{p}(\boldsymbol{i}) \in \mathscr{C}_k\big). \tag{8}$$

In other words, for some fixed $1 \leq k \leq K$, we collect the pixels $\boldsymbol{i} \in \Omega$ for which the range value $\boldsymbol{p}(\boldsymbol{i}) \in \mathscr{C}_k$ and assign $\hat{\boldsymbol{g}}_K(\boldsymbol{i})$ to each of these pixels. The subscript K in (8) is used to emphasize that the approximation depends on the number of clusters; K can be interpreted as the approximation order. The intuition behind approximation (8) is that, if the range points in cluster $\mathscr{C}_k$ are closely approximated by $\boldsymbol{\mu}_k$, then we can expect $\hat{\boldsymbol{g}}_K(\boldsymbol{i})$ to be close to $\boldsymbol{g}(\boldsymbol{i})$. For example, if we define the representation error of the clustering to be

$$\mathcal{E}_K = \sum_{k=1}^{K} \sum_{\boldsymbol{p}(\boldsymbol{i})\in\mathscr{C}_k} \|\boldsymbol{p}(\boldsymbol{i}) - \boldsymbol{\mu}_k\|^2, \tag{9}$$

then one would expect (8) to be close to (1) if $\mathcal{E}_K$ is small. This is indeed confirmed by the following quantitative bound.

Theorem 2.1 (Approximation error)**.** *Assume that range kernel ϕ is Lipschitz continuous, that is, for some $L > 0$,*

$$|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \qquad (\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^\rho). \tag{10}$$

Then, for some absolute constant $C > 0$,

$$\sum_{\boldsymbol{i}\in\Omega} \|\hat{\boldsymbol{g}}_K(\boldsymbol{i}) - \boldsymbol{g}(\boldsymbol{i})\|^2 \leq CLn|W|^2\mathcal{E}_K, \tag{11}$$

where $|W|$ is the number of grid points in W.

We skip the proof of Theorem 2.1 due to space constraints. Instead, we make several remarks on the significance of the result. First, it can be verified that the Lipschitz property is satisfied by a wide range of kernels [19], including the Gaussian kernel (used in bilateral and non-local means filtering).

Proposition 2.2. *Let $\phi(\boldsymbol{x}) = \exp(-\lambda\|\boldsymbol{x}\|^2)$ where $\lambda > 0$. Then we have the bound in* (10) *with $L = \sqrt{2\lambda}\exp(-1/2)$.*

Proof. By the mean-value theorem,

$$\phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) = \nabla\phi(\boldsymbol{\xi})^\top(\boldsymbol{x} - \boldsymbol{y})$$

where $\boldsymbol{\xi}$ is some point on the line segment joining $\boldsymbol{x}$ and $\boldsymbol{y}$. Using the Cauchy-Schwarz inequality, we get

$$|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})| \leq \|\nabla\phi(\boldsymbol{\xi})\| \cdot \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Now, it can be shown that the maximum of $\|\nabla\phi(\boldsymbol{x})\|$ is $\sqrt{2\lambda}\exp(-1/2)$, which gives us the desired bound. □

The second remark is that for several clustering methods [18], it is known that $\mathcal{E}_K$ vanishes as $K \to \infty$. For any of these clustering methods, we have the following guarantee.

Corollary 2.3 (Arbitrary accuracy)**.** *If ϕ is Lipschitz continuous and $\mathcal{E}_K$ vanishes as $K \to \infty$, then for $\boldsymbol{i} \in \Omega$,*

$$\lim_{K\to\infty} \hat{\boldsymbol{g}}_K(\boldsymbol{i}) = \boldsymbol{g}(\boldsymbol{i}). \tag{12}$$

In other words, the proposed algorithm can approximate (1) *with arbitrary accuracy provided K is sufficiently large.*

We note that similar conclusions have not been reported in the literature for existing algorithms. In the present case, this is possible due to the simple and explicit nature of the approximation. Indeed, most of the existing fast algorithms are based on sophisticated approximations, which makes it rather difficult to perform a quantitative error analysis. Finally, it may seem possible that bounds similar to (11) do exist for these algorithms, although the derivation seems to be out of reach. In this regard, we will provide empirical evidence in Section 3 which suggests that this is probably not the case for the Adaptive Manifolds algorithm [3].

We propose to use bisecting K-means [18] for clustering Θ. Bisecting K-means is a hierarchical clustering method in which a selected cluster is split into two clusters using K-means [18]. For our

Input: $\boldsymbol{f} : \Omega \to \mathbb{R}^n$ and $\boldsymbol{p} : \Omega \to \mathbb{R}^\rho$, and kernels ω and ϕ;
Parameter: Number of clusters K;
Output: Approximation in (8);
Form Θ in (3) and cluster it using bisecting K-means;
Let $(\mathscr{C}_i, \boldsymbol{\mu}_i), 1 \le k \le K$, be the output of the clustering;
Initialize $\boldsymbol{h} : \Omega \to \mathbb{R}^n$ and $\zeta : \Omega \to \mathbb{R}$ with zeros;
for $k = 1, \ldots, K$ **do**
  $\boldsymbol{f}_k \leftarrow \phi(\boldsymbol{p} - \boldsymbol{\mu}_k) \otimes \boldsymbol{f}$;
  $\phi_k \leftarrow \phi(\boldsymbol{p} - \boldsymbol{\mu}_k)$;
  Set mask $m_k : \Omega \to \{0, 1\}$ using the rule:
  $$m_k(\boldsymbol{i}) \leftarrow \begin{cases} 1 & \text{if } \boldsymbol{p}(\boldsymbol{i}) \in \mathscr{C}_k, \\ 0 & \text{otherwise;} \end{cases}$$
  $\boldsymbol{h} \leftarrow \boldsymbol{h} \oplus (m_k \otimes (\omega * \boldsymbol{f}_k))$;
  $\zeta \leftarrow \zeta \oplus (m_k \otimes (\omega * \phi_k))$;
end
$\hat{\boldsymbol{g}}_K \leftarrow \boldsymbol{h} \oslash \zeta$.

Algorithm 1: Fast high-dimensional filtering.

algorithm, we split the cluster with the largest diameter at every iteration, which forces the clusters to shrink. As a result, the points within a given cluster are closely represented by the cluster center. Moreover, bisecting K-means satisfies the property desired in Corollary 2.3. The overall algorithm arising from the proposed approximation is described in Algorithm 1, where the symbols $\oplus, \otimes$ and $\oslash$ denote pixelwise addition, multiplication, and division of two images. The main computations are the $(n+1)K$ spatial convolutions (the cost of clustering is usually small compared to this cost). The computational advantage comes from the fact that, when ω is box or Gaussian, the spatial convolutions can be performed using $O(1)$ operations irrespective of the window size $(2S+1)^d$ [20, 21]. In this case, the overall per-pixel complexity (neglecting the cost of clustering) is $O(Kn)$, which should be compared with the per-pixel complexity of $O(nS^d)$ for the exact filtering in (1). Thus, the speedup is by a factor of S^d/K. This can be substantial for high-resolution images, since it is natural to set the width S of the spatial kernel in proportion to the spatial resolution. Similar to Adaptive Manifolds, the storage complexity of our algorithm is linear in n and the image size.

3. EXPERIMENTAL RESULTS

We now demonstrate the effectiveness of our algorithm by using it for bilateral and non-local means filtering of color and hyperspectral images. These are undoubtedly the two most popular forms of high-dimensional filters. Bilateral filter is mainly used for edge-preserving smoothing [4, 5], while non-local means is used for image denoising [6]. For the bilateral filter, $\boldsymbol{f}$ and $\boldsymbol{p}$ are identical, and the spatial and range kernels are Gaussian:

$$\omega(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2\sigma_s^2}\right) \quad \text{and} \quad \phi(\boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{z}\|^2}{2\sigma_r^2}\right), \quad (13)$$

where $\boldsymbol{x} \in \mathbb{R}^2$ and $\boldsymbol{z} \in \mathbb{R}^\rho$. The window size is set as $S = 3\sigma_s$. For color images $\rho = 3$, while ρ can be of the order of tens or hundreds for hyperspectral images. For our experiment, we have used a hyperspectral image with $\rho = 33$ frequency bands [22].

In Table 1, we have compared our algorithm with the Adaptive Manifolds (AM) algorithm for the task of bilateral filtering. The RGB image of Dome [8] was used for this experiment. To quantify the approximation error, we have used the root-mean-square error (RMSE) given by

$$\text{RMSE}^2 = \frac{1}{|\Omega|} \sum_{\boldsymbol{i} \in \Omega} \|\hat{\boldsymbol{g}}(\boldsymbol{i}) - \boldsymbol{g}(\boldsymbol{i})\|^2, \quad (14)$$

where $\boldsymbol{g}$ is the exact bilateral filter in (1) and $\hat{\boldsymbol{g}}$ is the approximation from the respective algorithms. Notice that the RMSE for our algorithm consistently decreases with increase in K (number of clusters). On the other hand, the RMSE for Adaptive Manifolds oscillates with increase in K (number of manifolds). In fact, we found that for certain images, the RMSE for Adaptive Manifolds tends to diverge when K is large.

In Tables 2 and 3, we have compared our algorithm with Adaptive Manifolds and the fast algorithm in [8]. We have used the publicly-available C++ implementations of the algorithms [23, 24]. The computations were performed on an Intel quad-core 3.4 GHz machine with 32 GB memory. We tuned the internal parameters of each algorithm to obtain the same range of RMSE. In Table 2, we have fixed σ_r and changed σ_s. Notice that our algorithm shows less dependence on σ_s compared to the other two algorithms. In particular, the timings

K	1	3	7	15	31	63	127	255	511
	$\sigma_s = 10, \sigma_r = 0.2$								
Proposed	13.14	7.10	2.28	1.74	1.00	0.72	0.55	0.38	0.28
AM [3]	3.45	1.74	1.79	1.20	1.04	0.95	0.96	0.97	0.98
	$\sigma_s = 50, \sigma_r = 0.8$								
Proposed	7.05	3.45	1.29	0.93	0.66	0.59	0.55	0.52	0.51
AM [3]	4.14	2.55	2.74	2.57	2.62	3.06	2.71	2.74	2.76

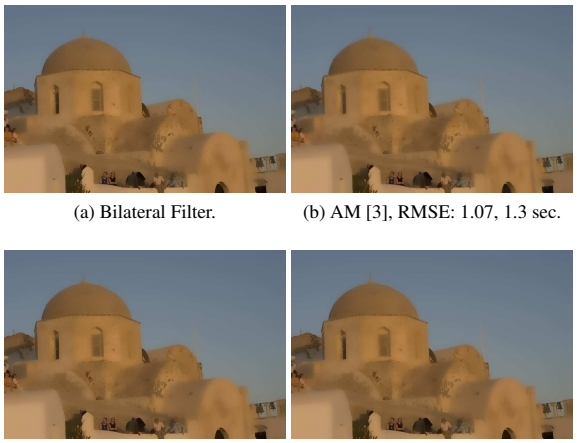
Table 1. Variation of RMSE with approximation order K (number of clusters for the proposed method and number of manifolds for AM).

σ_s	10	20	30	40	50	60	70	80
	Paris [8]							
Time	5.04	3.77	3.54	3.45	3.51	3.54	3.37	3.36
RMSE	1.88	2.31	2.52	2.70	2.94	3.28	3.43	3.62
	Adaptive Manifolds [3]							
Time	1.17	0.75	0.75	0.65	0.65	0.65	0.62	0.62
RMSE	0.92	1.05	1.19	1.30	1.32	1.36	1.84	1.92
	Proposed Algorithm							
Time (serial)	0.95	0.80	0.84	0.87	0.91	0.95	0.99	1.0
Time (parallel)	0.58	0.48	0.50	0.52	0.53	0.53	0.55	0.55
RMSE	0.91	1.44	1.48	1.45	1.42	1.38	1.38	1.38

Table 2. Timings (sec) and RMSE for the Dome image for $\sigma_r = 0.4$.

σ_r	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
	Paris [8]									
Time	20.41	8.89	5.83	5.04	4.99	4.50	4.45	4.26	4.25	4.08
RMSE	1.95	2.00	1.86	1.88	1.67	1.47	1.39	1.65	1.75	2.02
	Adaptive Manifolds [3]									
Time	1.18	1.18	1.18	1.17	1.17	1.17	1.17	1.17	1.17	1.17
RMSE	1.08	1.08	0.98	0.92	0.91	0.94	0.98	1.02	1.06	1.10
	Proposed Algorithm									
Time (serial)	2.4	1.8	1.7	0.95	0.75	0.75	0.75	0.75	0.75	0.75
Time (parallel)	1.1	0.93	0.82	0.58	0.49	0.46	0.46	0.47	0.47	0.49
RMSE	1.58	1.20	0.95	0.91	0.91	0.73	0.60	0.53	0.47	0.44

Table 3. Timings (sec) and RMSE for the Dome image for $\sigma_s = 10$.



(a) Bilateral Filter. (b) AM [3], RMSE: 1.07, 1.3 sec.

(c) Proposed, RMSE: 1.36, 1.7 sec. (d) Proposed, RMSE: 1.83, 1 sec.

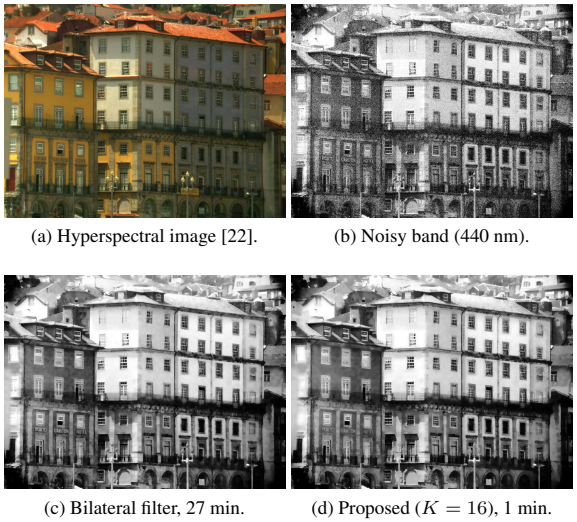
Fig. 2. Visual comparison of the outputs for the Dome image ($876 \times 584 \times 3$). The bilateral filter parameters are $\sigma_s = 10$ and $\sigma_r = 0.2$. We used 7 manifolds for AM. We have used 16 clusters in (c) and 8 clusters in (d). The reported timings are for the C++ implementations.

of Adaptive Manifolds and [8] are higher for small σ_s (Adaptive Manifolds performs better). This trend was reported in [8]. On the other hand, the timing of our algorithm is lower for small σ_s. In Table 3, we have varied σ_r keeping σ_s fixed. Again, our algorithm and Adaptive Manifolds exhibit better performance than [8]. In particular, notice that the RMSE for our algorithm tends to be better when σ_r is large. This can be explained using the error bound in (11), which has a linear dependence on L. It follows from Proposition 2.2 that, for the range kernel in (13), $L \propto \sigma_r^{-1}$. Thus, the bound in (11) is small when σ_r is large, which explains the trend in Table 3. Visual results on color and hyperspectral images are provided in Figures 2 and 3.

In non-local means, $\boldsymbol{f}$ is the input (noisy) image and $\boldsymbol{p}(\boldsymbol{i})$ is a square patch of neighboring pixels around pixel $\boldsymbol{i}$ extracted from $\boldsymbol{f}$. In particular, for a square patch of $m \times m$ pixels, $\rho = 3m^2$ for a color image. A standard choice is $m = 7$ [6]. Following the work in [25], we first form the range Θ using the extracted patches ($\rho = 3m^2$) and project it onto a lower-dimensional space ($\rho = 3$) using principal component analysis. The spatial kernel in non-local means is simply a box, that is, $\omega(\boldsymbol{x}) = 1$, and ϕ is a Gaussian, as in (13). We present a denoising result in Figure 4, where we have approximated non-local means using our algorithm and Adaptive Manifolds. Notice that our algorithm offers better tradeoff between timing and denoising performance compared to Adaptive Manifolds.

4. CONCLUSION

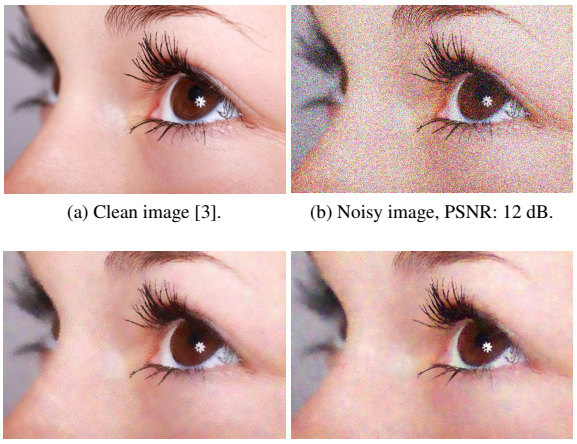
We proposed an algorithm for fast high-dimensional filtering based on clustering, which is both simple to implement and analyze. We presented some results which demonstrated that its performance for bilateral filtering and non-local means is at par with state-of-the-art algorithms. We derived a quantitative bound on the approximation error incurred by our algorithm. This translated into the guarantee that the proposed algorithm can achieve arbitrary accuracy using sufficient number of clusters. We presented an example which suggested that this guarantee is not met by Adaptive Manifolds.



(a) Hyperspectral image [22]. (b) Noisy band (440 nm).

(c) Bilateral filter, 27 min. (d) Proposed ($K = 16$), 1 min.

Fig. 3. Comparison of the proposed method and the exact bilateral filter for a hyperspectral image ($1340 \times 1017 \times 33$). The filter parameters are $\sigma_s = 5$ and $\sigma_r = 0.4$. The image in (b) shows one of the noisy bands; the same band is shown in (c) and (d) after the filtering. Note that one of the bands is shown just for visualization; the filtering was performed on the entire hyperspectral image. The reported timings are for the Matlab implementations.



(a) Clean image [3]. (b) Noisy image, PSNR: 12 dB.

(c) Proposed, 26.20 dB, 3.7 sec. (d) AM, 24.87dB, 4.7 sec.

Fig. 4. Denoising results for a natural RGB image [3]. The parameters for non-local means are $m = 7$, $\sigma_r = 0.43$ and $S = 10$. We used PCA to reduce the dimension from 147 (3×7^2) to 3 [25]. We used 32 clusters for the proposed method. We used a publicly-available code for Adaptive Manifolds [23], where the number of manifolds is set automatically. The reported timings are for the Matlab implementations [26]. We computed the PSNR as $10 \log_{10}(255^2/\text{MSE})$, where MSE is the mean squared error between the output of the denoising algorithm and the clean image.

5. REFERENCES

[1] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian kd-trees for fast high-dimensional filtering," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.

[2] A. Adams, J. Baek, and A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," *Computer Graphics Forum*, vol. 29, no. 2, pp. 753-762, 2010.

[3] E. S. Gastal and M. M. Oliveira, "Adaptive manifolds for real-time high-dimensional filtering" *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[4] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Proc. IEEE International Conference on Computer Vision*, pp. 839-846, 1998.

[5] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, *Bilateral Filtering: Theory and Applications*, Now Publishers Inc., 2009.

[6] A. Buades, B. Coll and J.-M. Morel, "A non-local algorithm for image denoising," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 60-65, 2005.

[7] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 257-266, 2002.

[8] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," *Proc. European Conference on Computer Vision*, pp. 568-580, 2006.

[9] Q. Yang, K. H. Tan, and N. Ahuja, "Real-time $O(1)$ bilateral filtering," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 557-564, 2009.

[10] K. N. Chaudhury, D. Sage, and M. Unser, "Fast $O(1)$ bilateral filtering using trigonometric range kernels," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3376-3382, 2011.

[11] K. Sugimoto and S. I. Kamata, "Compressive bilateral filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3357-3369, 2015.

[12] K. N. Chaudhury and S. Dabhade, "Fast and provably accurate bilateral filtering," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2519-2528, 2016.

[13] S. Ghosh and K. N. Chaudhury, "Fast and high-quality bilateral filtering using Gauss-Chebyshev approximation," *Proc. International Conference on Signal Processing and Communications*, pp. 1-5, 2016.

[14] P. Nair, A. Popli and K. N. Chaudhury, "A fast approximation of the bilateral filter using the discrete Fourier transform," *Image Processing On Line*, 2017.

[15] Q. Yang, K. H. Tan, and N. Ahuja, "Constant time median and bilateral filtering," *International Journal of Computer Vision*, vol. 112, no. 3, pp. 307-318, 2015.

[16] S. Ghosh and K. N. Chaudhury, "Fast bilateral filtering of vector-valued images," *Proc. IEEE International Conference on Image Processing*, pp. 1823 - 1827, 2016.

[17] M. G. Mozerov and J. van de Weijer, "Global color sparseness and a local statistics prior for fast bilateral filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5842-5853, 2015.

[18] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Education India, 2006.

[19] U. V. Luxburg and O. Bousquet, "Distance-based classification with Lipschitz functions," *Journal of Machine Learning Research*, vol. 5, pp. 669-695, 2004.

[20] R. Deriche, "Recursively implementing the Gaussian and its derivatives," *Research Report*, INRIA-00074778, 1993.

[21] I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter," *Signal Processing*, vol. 44, pp. 139-151, 1995.

[22] D. H. Foster, K. Amano, S. M. C. Nascimento and M. J. Foster, "Frequency of metamerism in natural scenes," *Journal of the Optical Society of America A*, vol. 23, pp. 2359-2372, 2006.

[23] V. Vinogradov, "C++ implementation of the adaptive manifold high dimensional filter," `https://github.com/jet47/AdaptiveManifoldFilter`, 2013.

[24] S. Paris and F. Durand, "C++ implementation of a fast approximation of the bilateral filter using a signal processing approach," `http://people.csail.mit.edu/sparis/bf/`, 2009.

[25] T. Tasdizen, "Principal neighborhood dictionaries for nonlocal means image denoising," *IEEE Transactions on Image Processing*, vol. 18, no. 12, pp. 2649-2660, 2009.

[26] E. S. Gastal and M. M. Oliveira, "Matlab implementation of the adaptive manifold high dimensional filter," `http://inf.ufrgs.br/~eslgastal/AdaptiveManifolds/`, 2013.