Fast Patch-based Denoising Using Approximated Patch Geodesic Paths

Xiaogang Chen[1,3,4], Sing Bing Kang[2], Jie Yang[1,3], and Jingyi Yu[4]

[1]Shanghai Jiao Tong University, Shanghai, China. {cxg,jieyang}@sjtu.edu.cn
[2]Microsoft Research, Redmond, WA, USA. sbkang@microsoft.com
[3]Key Laboratory of System Control and Information Processing, Ministry of Education, China.
[4]University of Delaware, Newark, DE, USA. yu@cis.udel.edu

Abstract

Patch-based methods such as Non-Local Means (NLM) and BM3D have become the de facto gold standard for image denoising. The core of these approaches is to use similar patches within the image as cues for denoising. The operation usually requires expensive pair-wise patch comparisons. In this paper, we present a novel fast patch-based denoising technique based on Patch Geodesic Paths (PatchGP). PatchGPs treat image patches as nodes and patch differences as edge weights for computing the shortest (geodesic) paths. The path lengths can then be used as weights of the smoothing/denoising kernel.

We first show that, for natural images, PatchGPs can be effectively approximated by minimum hop paths (MHPs) that generally correspond to Euclidean line paths connecting two patch nodes. To construct the denoising kernel, we further discretize the MHP search directions and use only patches along the search directions. Along each MHP, we apply a weight propagation scheme to robustly and efficiently compute the path distance. To handle noise at multiple scales, we conduct wavelet image decomposition and apply PatchGP scheme at each scale. Comprehensive experiments show that our approach achieves comparable quality as the state-of-the-art methods such as NLM and BM3D but is a few orders of magnitude faster.

1. Introduction

Image denoising is a classical inverse problem. Traditional pixel-based edge-preserving algorithms such as median filters, bilateral filters [34], total variation [33] and anisotropic diffusion [33] have long served as workhorses in denoising tasks. These approaches focus on computing the (de)similarities between pixels within a local neighborhood and are easy to implement. More recent approaches include the simple yet elegant Gaussian scale mixture (GSM) algorithm [31] and Non-Local Means (NLM) [10] that explore global image structures using patches. These powerful patch-based schemes can be interpreted as expectation-maximization (EM)-based inference on stochastic factor graphs and have shown outstanding performance.

The core of these approaches is to use patches similar to the noisy one within the image as cues. This operation usually requires expensive pair-wise patch comparisons. For example, in NLM and BM3D, denoising each patch requires computing its similarity with all other patches in a predefined search window. The similarity scores are stored as a convolution kernel for denoising. Admas et al. [4] show that one can use the K most similar patches instead of all patches within the window which is equivalent to solving the K-nearest neighbor (K-NN) problem in the high dimensional patch space. However, accelerating the K-NN algorithm remains challenging in machine learning.

In this paper, we present a novel fast patch-based denoising technique based on Patch Geodesic Paths (PatchGP). PatchGP extends the pixel geodesic paths (PixelGP) [5] by treating image patches as nodes and assigning patch differences as edge weights for computing the shortest (geodesic) paths. The path lengths can then be used as weights of the smoothing/denoising kernel. Brute-force implementation of PatchGP, however, is more expensive than NLM or BM3D. We therefore develop a class of acceleration schemes. We first show that for natural images, PatchGPs can be effectively approximated by minimum hop paths (MHPs) that generally correspond to Euclidean line paths connecting two patch nodes. To construct the denoising kernel, we further discretize the MHP search directions and use only patches along the search directions. Along each MHP, we apply a weight propagation scheme to robustly and efficiently compute the path distance. Finally, to handle noise at multiple scales, we conduct wavelet image decomposition and apply PatchGP scheme at each scale. Comprehensive experiments show that our approach achieves comparable quality as the state-of-the-art methods but is a few orders

of magnitude faster.

2. Related work

Image denoising is a long standing problem. Classical approaches attempt to filter the noisy image in the spatial or frequency domain. In recent decades, spatial domain filters have been particularly popular since they are easy to implement and can be accelerated on the GPUs [2]. These solutions have also been used in non-photorealistic Rendering [37], tone mapping [14], image/video segmentation [13], etc. We refer the readers to the recent survey [13] that compares a broad range of techniques. Most recently, Levin et al. [24] discussed the the relation between the patch complexity of natural images, patch size, and restoration errors. They also pointed out a law of diminishing return: patches that require a large increase in database size also benefit little from a larger window.

Pixel vs. Patches. A widely used class of pixel-based algorithms is edge-preserving filters such as anisotropic diffusion [28] and bilateral filters [29]. They can be viewed as convolving the noisy image with a special smoothing kernel [29] [34]:

$$\widehat{I}(i) = \frac{1}{Z(i)} \sum_{j \in \Omega_i} w(i,j) I(j), \tag{1}$$

where w is the smoothing kernel, Ω_i is the spatial support of w or a neighborhood of pixel i, and $Z(i)$ is the normalization factor as $\sum_{j \in \Omega_i} w(i,j)$. For example, in bilateral filters, w is computed as the product of two Gaussians:

$$w_{bilateral}(i,j) = G_{\sigma_r}(|I(i) - I(j)|^2) G_{\sigma_s}(|i-j|^2), \tag{2}$$

where G_{σ_r} is the range kernel and G_{σ_s} is the spatial kernel, both centered at pixel i. The range kernel can also take color into consideration [7] as $G_{\sigma_{color}}(|v(i) - v(j)|^2)$. Anisotropic diffusion uses similar local filters to successively produces a family of parameterized images where new images at each iteration are computed by applying diffusion filters to the ones from the previous iteration [29].

More recent approaches exploit the frequent occurrence of similar patches within the image. For example, NLM uses patch similarity instead of pixel similarity for constructing the smoothing kernel:

$$w_{NLM}(i,j) = G_{\sigma}(|N_I(i) - N_I(j)|^2), \tag{3}$$

where $N_I(i)$ and $N_I(j)$ represent patches centered at i and j and $|\cdot|^2$ is the sum of squared differences (SSD) between the patches. BM3D uses a similar scheme (1) except that the weight is calculated in the 3D transform domain. Instead of using the convolutional approach, LPG-PCA [40] applies principal component analysis (PCA) on similar patches. More sophisticated schemes [32] [36] [41] further utilize image statistics within patches to improve the denoising results. Compared with the pixel-based techniques, patch-based solutions are more reliable but slower.

Acceleration Schemes. Brute-force implementations of Eq. (1), whether pixel or patch based, are computational expensive. The original bilateral filters have computational complexity of $O(r^2)$ for each pixel, where r is radius of the spatial support. In the past decade, a large number of acceleration schemes have been proposed for pixel-based denoising. Durand and Dorsey [14] applied piecewise-linear approximations to the range (intensity) kernel. Yang et al. [38] developed recursive Gaussian filters to handle varying kernel sizes. Paris and Durand [27] mapped the 2D filtering process onto the 3D space so that the filter can be efficiently implemented by standard 3D Gaussian convolution. Weiss [35] and Porikli [30] used the box kernel to approximate the spatial Gaussian so that Integral Histogram can be directly used for acceleration. These acceleration schemes can also be extended to multichannel images. Most recently, Yang [37] proposed to utilize coherency between different channels to achieve linear computational complexity.

Despite great advances on pixel-based denoising, accelerating patch-based denoising remains as an open problem. This is mainly due to the high dimensionality of patch space. By far, the focus has been using smart data structures such as the KD trees to arrange the patches for quick querying [23, 9, 4]. He and Sun [19] further proposed a Propagation-Assisted KD-Tree model to further improve the performance. However, these high-dimensional structures are storage demanding and less suitable for devices with limited memory and computation resources. We, in contrast, explore the problem from the perspective of natural image patch statistics.

3. Patch Geodesic Paths

The core of our approach is to accelerate patch-based denoising by only conducting patch comparisons on the *geodesic* paths.

3.1. Pixel Geodesic Distance

In a graph, the geodesic distance between two nodes is the accumulative edge weights in a shortest path connecting them. Yatziv and Sapiro [39] introduced the geodesic distance $d(s,t)$ between two pixels s and t for an image I as:

$$d(s,t) = \min_{\Gamma} \int_0^1 |\nabla I \cdot \dot{\Gamma}(p)| dp, \tag{4}$$

where Γ denotes a path between s to t, $\dot{\Gamma}(p)$ denotes the tangent of the path (curve) Γ at pixel p, and the integral measures the accumulative directional derivative at all pixels p along the path. The pixel geodesic distance corresponds to

the shortest path in terms of image gradients, i.e., the smoothest curve in intensity.

The concept of pixel geodesic distance has been successfully applied to colorization [39], segmentation and matting [5, 18, 12], texture removal and non-photorealistic rendering [13], and most recently, denoising [13, 17]. It is a common practice to construct a graph from the image using 4-connected pixels [6] and discretize Eq. (4) to:

$$d^{GD}(s,t) = \min_{\Gamma} \sum_{i=1}^{N_\Gamma - 1} |I(\Gamma(p_{i+1})) - I(\Gamma(p_i))|, \quad (5)$$

where Γ denotes a path starting from the patch centered at s to the patch centered at t. N_Γ denotes the hops of the path Γ. $\Gamma(p_1) = s$ and $\Gamma(p_{N_\Gamma}) = t$. Dijkstra's algorithm and Euclidean distance transform [39] have been used to accelerate the search. To apply pixel geodesic distance for image denoising, one can compute the smoothing kernel with $w^{GD}(i,j) = G_\sigma(d(i,j))$. This is often referred to as Pixel Geodesic Path (PixelGP) denoising [17]. However, PixelGP shares similar issues with general pixel-based algorithms. Since PixelGP accumulates the gradients, for images with strong noise and hence large gradients the distance measurement can be unreliable.

3.2. Patch Geodesic Distance

We extend the notion of PixelGP to patches. Specifically, we treat each image patch as a node and define the geodesic distance between two patches s and t as:

$$d^{patchGP}(s,t) = \min_{\Gamma} \sum_{i=1}^{N_\Gamma - 1} ||N_I(\Gamma(p_{i+1})) - N_I(\Gamma(p_i))||, \quad (6)$$

where $N_I(x)$ is the patch centered at x, $||.||$ measures the patch differences. We call the shortest path Γ between two patches the Patch Geodesic Path (PatchGP).

Similar to PixelGP, we use PatchGP to define the smoothing kernel as $w_p(i,j) = G_\sigma(d^{patch}(i,j))$. As a patch-based scheme, PatchGP generally outperforms the pixel-based approaches (as shown in Fig. 5). However, the brute-force implementation of PatchGP is very expensive because weight computation requires pairwise patch comparisons. Our approach is to narrow down the search to a special subset of paths.

3.3. Minimal Hop Paths (MHP)

Consider two patches centered at pixel s and t. We define the Minimum Hop Path (MHP) as the path with the minimal number of hops connecting two nodes. An example is illustrated in Fig. 1(a). Among all paths connecting p and q, the diagonal red path is the corresponding MHP under 8-connectivity. In fact, for nodes lying along the 8 directions,

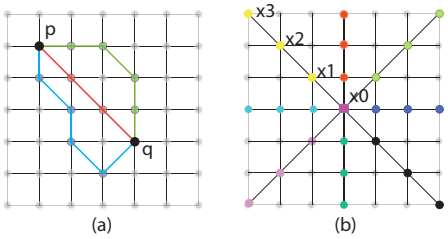

Figure 1. Left: The MHP between p and q (red). Right: MHPs under 8 search directions.

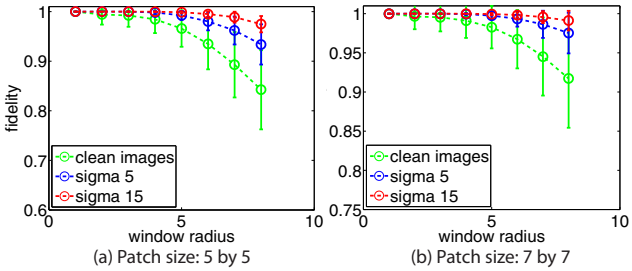

Figure 2. Minimum Hop Paths (MHP) vs. Patch Geodesic Paths (PatchGP). We randomly select 200 images from Berkeley image segmentation database and add white Gaussian noise (σ_n=5 and σ_n = 15). For each configuration (patch size, window size, noise variance), we find PatchGP between every pair of patches and verify if it is an MHP. The percentage of PatchGP being MHP is shown in (a) 5×5 patch size and (b) 7×7 patch size.

the MHPs are the same as the Euclidean Line Path (ELP) connecting the nodes.

Here is our key observation: for two relatively close patches in a natural image, we can approximate their PatchGP using the MHP. To illustrate this, we use 200 training images from Berkeley Image Segmentation Database [26]. We add white Gaussian noise to the images and test different patch sizes. For a fixed noise variance and patch size, we first compute the ground truth PatchGPs between all patches. We then verify if they correspond to MHPs. In Fig 2, the Y-axis is the percentage of MHPs being PatchGPs averaged over all 200 images and the X-axis is the spatial support (the maximum hop allowed between the nodes).

Fig 2 illustrates that PatchGP has a high probability of being MHP for small to medium support (window size). The results hold for noisy images: even with noise variance $\sigma_n = 15$ and spatial support 7 (15×15 window), over 90% PatchGPs are MHPs, as shown in Fig 2(a) and (b). However, this percentage goes down on clean images. This is because PatchGP is sensitive to small perturbations. For example, a slight inconsistency on a uniform background may alter PatchGPs. However, that also means small we only need to use small windows [10] where MHPs still effectively approximates PatchGPs. Nevertheless, for images with small noise variances, the smaller window is required for smoothing and MHP approximation is mostly reserved (over 95% for window radius 5). we can still use them for

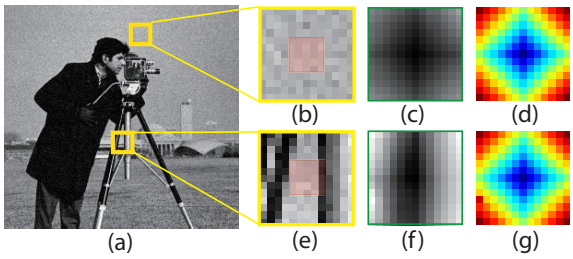

Figure 3. Patch Geodesic Distance. (a) shows the cameraman image and two central patches (5x5) we aim to denoise. For each patch, we use a spatial support (window size) of 13×13. For each patch within the window, we find its PatchGP to the central patch and compute the actual path distance. (b) and (e) show the close-up views of the two patches. (c) and (f) show the patch distance map.(d) and (g) show the color-coded path hop maps for the corresponding PatchGPs.

denoising. Fig 2 also shows that patch-sizes consistent with scene structures result in higher PatchGP-MHP consistency. In the extreme case when patch size is 1×1, PatchGP degenerates to PixelGP where MHP is not longer consistent with PixelGP. Since most patch-based denoising schemes (NLM, BM3D, etc) use a large patch size, MHPs thus still provide good approximations to PatchGPs.

Fig. 3 illustrates PatchGP between two patches in the Cameraman image. Both patches are of size 5x5. For each patch, we compute its PatchGP to all other patches within a window of 13×13. Fig. 3 (c) and (f) show the patch geodesic distance map with respect to (b) and (e) respectively. Notice that the distance maps reflect image structures: (c) exhibits a uniform structure where as (f) shows a vertical structure. Fig. 3(d) and (g) show the number of hops in the corresponding PatchGPs. They form concentric squares and are nearly identical to the MHP hop maps under 4-way connectivity.

4. Fast PatchGP Denoising

Our analysis reveals that MHPs can be used to approximate PatchGPs on noisy natural images. Finding MHPs in a lattice graph is straightforward and the results can be precomputed and stored. To further reduce storage and computation, we only consider MHPs along discrete directions: if we use 8 connectivity ($\frac{i\pi}{4}$, $i \in \{1, 2, ..., 8\}$), we only compute MHPs along each direction. Consider a diagonal MHP in Fig. 1(b), in order to denoise patch x_0, we need to compute the path distances from $N(x_0)$ to patches $N(x_1)$, $N(x_2)$, ... $N(x_r)$ as:

$$d^{patch}(x_0, x_r) \approx \sum_{t=1}^{r} ||N(x_t) - N(x_{t-1})||. \quad (7)$$

We can rewrite Eqn. (7) as:

$$d^{patch}(x_0, x_r) \approx d_p(x_0, x_{r-1}) + ||N(x_r) - N(x_{r-1})||. \quad (8)$$

This indicates the patch geodesic distance can be computed progressively: we can first compute 1-hop path distance and then propagate it to 2-hop, 3-hop, and so on. Under our direction and hop discretization, we can reformulate the denoising filter as:

$$\widehat{I}(i) = \frac{1}{z(i)} \sum_{\theta \in \Theta} \sum_{r=1}^{R} w(i, i_{\theta,r}) I(i_{\theta,r}), \quad (9)$$

where the normalization factor $z(i) = \sum_{\theta \in \Theta} \sum_{r=1}^{R} w(i, i_{\theta,r})$. $w(i, i_{\theta,r})$ denotes the weight of the pixel $i_{\theta,r}$.

There are two major advantages of using discretized MHPs. First, it greatly reduces memory usage. For each discretized direction, we only need maintain a one-hop distance value. Second, different directions can be processed in parallel. Notice though that the downside of this approach is that it can no longer cover all patches within the spatial support, i.e., patches that do not lie on the predefined directions will not be used. This can potentially affect the patch-based denoising performance as some of the missing patches may be critical for denoising the central patch. In reality, we use a relatively dense directional discretization for reducing the number of missing patches. We also implement a multi-scale denoising scheme to compensate for sparse patch sampling.

Weight Threshold. The assumption that MHPs well approximate PatchGPs generally holds as shown in Fig 3. To properly handle the outliers, we first propose a weight threshold scheme analogous to truncated threshold in graph-cut. Reusing Fig 1, if $N(x_1)$ is significantly different from $N(x_0)$ while $N(x_2)$ is highly similar to $N(x_0)$, patch $N(x_2)$ should be assigned a large weight for denoising $N(x_0)$. However, since we propagate the weight using Eqn. (8), our estimated $d^{patchGD}(x_0, x_2)$ will be large and $N(x_2)$'s weight will be small. In this case, the MHP between x_0 and x_2 is unlikely to be the PatchGP (x_2 will likely bypass x_1 to connect to x_0). To handle this issue, we adopt a threshold scheme similar to the one in the graph-cut based solutions [8]: $d^{patchGD^*}(x_0, x_r)$=

$$max\{d^{patchGD^*}(x_0, x_{r-1}), ||N(x_r) - N(x_{r-1})||\}\alpha, \quad (10)$$

where we choose α=1.2 in all examples in our paper. In Sec. 5, we show that the new distance metric is more robust in presence of strong noise. It is worth noting that BM3D and NLM do not suffer from this issue as they conduct an exhaustive search although other star-shaped filters Veksler [18] and Foi et al. [15] share the same issue.

To compute 1-hop patch distance $||N(x_r) - N(x_{r-1})||$, we can either treat pixels within the patch equally or adopt a Gaussian weighting [10]. The former (we call uniform weighting) is faster as its computation is independent of the patch size by using Integral Histogram [30]. The latter (Gaussian weighting) is more accurate but slower and

widely adopted in NLM. For all our experiments (except for Fig. 8), we use patch size 7×7 and Gaussian weighting with $\sigma_{weight} = 2$.

Fast Multi-Scale Denoising. A common challenge in pixel-based denoising is reducing low frequency noise: properly handling low frequency noise requires using ultra-large spatial support. It is not only expensive but also may destroy image structures. Fig. 4 (c) and (e) show the denoising results using the PixelGP [17] vs. our fast PatchGP respectively. Notice that the sky regions in the denoised images appear a bit splotchy.

We resolve this problem by using a coarse-to-fine denoising scheme. We first build a Laplacian Pyramid [11] of the input image and denoise the top coarsest low-frequency image in the pyramid using fast PatchGP. We then use the result to denoise the second level image and repeat this process until we process the original image. Specifically, we use Haar wavelet transformation [16] to extract low- and high- frequency components when building the Laplacian Pyramid. The Harr wavelet transformation provides two advantages: it is faster compared with the Gaussian pyramid and does not affect noise statistics. In all our examples, we construct Laplacian pyramids of three levels and the overall computational cost is slightly (1/3) higher than fast PatchGP.

5. Experiments

We ran comprehensive experiments for evaluating our approaches. We first compare the performance between the brute-force PatchGP, fast PatchGP (F-PatchGP), and fast multi-scale PatchGP (FM-PatchGP).

Fig. 4 compares the denoising results of PixelGP [17], PatchGP, F-PatchGP and FM-PatchGP. The noisy image (b) is synthesized by adding Gaussian noise with variance 20. (c, d, e, f) show the denoised results by different schemes. We observe that all three PatchGP-based algorithms achieve a higher PSNR than PixelGP, although PatchGP cannot remove noise on the sky. F-PatchGP produces a slightly more visually pleasing result since the truncated distance metric (10) effectively suppressed errors on uniform regions. FM-PatchPG produces the most visually pleasing results as well as the highest PSNR.

Next, we compare FM-PatchGP with state-of-the-art methods, in both quality and speed. Specifically, we compare FM-PatchGP against the FoE, NLM, BM3D, PixelGP, and fast bilateral filters (F-BL) [38]. We use the C-implementation of NLM [10] and F-BL [38], the Matlab implementations of FOE [1] and PixelGP [17], the BM3D kernel (already optimized using approximation strategies) with Matlab wrapper [21], and finally our C-implementation of FM-PatchGP. Notice that F-BL has two parameters (spatial and range Gaussian variances). For each image, we exhaust different parameters and record the result

Figure 4. Quality comparison between PixelGP, PatchGP, F-PatchGP, and FM-PatchGP. (a) and (b) show the latent and the noisy images.

with the highest PSNR.

Fig. 5 shows the PSNR of the denoised results on eight different images. For each image, we synthesize 6 noisy versions by adding Gaussian noise with different variances between 5 and 30. Each panel in Fig. 5 corresponds to a specific image where the X-axis is the noise variance and Y-axis the PSNR. The last panel shows the averaged PSNR curve for all 8 images. Fig. 6 compares the visual quality, the PSNR, and the processing time of different denoising algorithms on the 'man' image (with added Gaussian noise σ_n=15). The last panel of Fig. 5 reveals that our FM-PatchGP (curve in red) achieves nearly identical performance to FoE and NLM but consistently outperforms F-BL and PixelGP. BM3D is consistently a full dB better than FM-PatchGP and the rest. However, since FM-PatchGP is significantly faster, it can be run with multiple iterations to achieve comparable output quality.

Fig 9 compares the processing speed (mega-pixels/sec) w.r.t the image resolution using NLM, BM3D, F-BL and our FM-PatchGP with uniform and Gaussian weighting. All algorithms are tested on a Thinkpad X200 laptop with 2.6 GHz CPU and 2GB memory as a single-thread program. We downsample a clean image of resolution 4032×6048 from [22] to different resolutions and then add Gaussian noise $\sigma_n = 15$. FM-PatchGP, uniform or Gaussian weighting, is significantly faster than BM3D and NLM at all res-

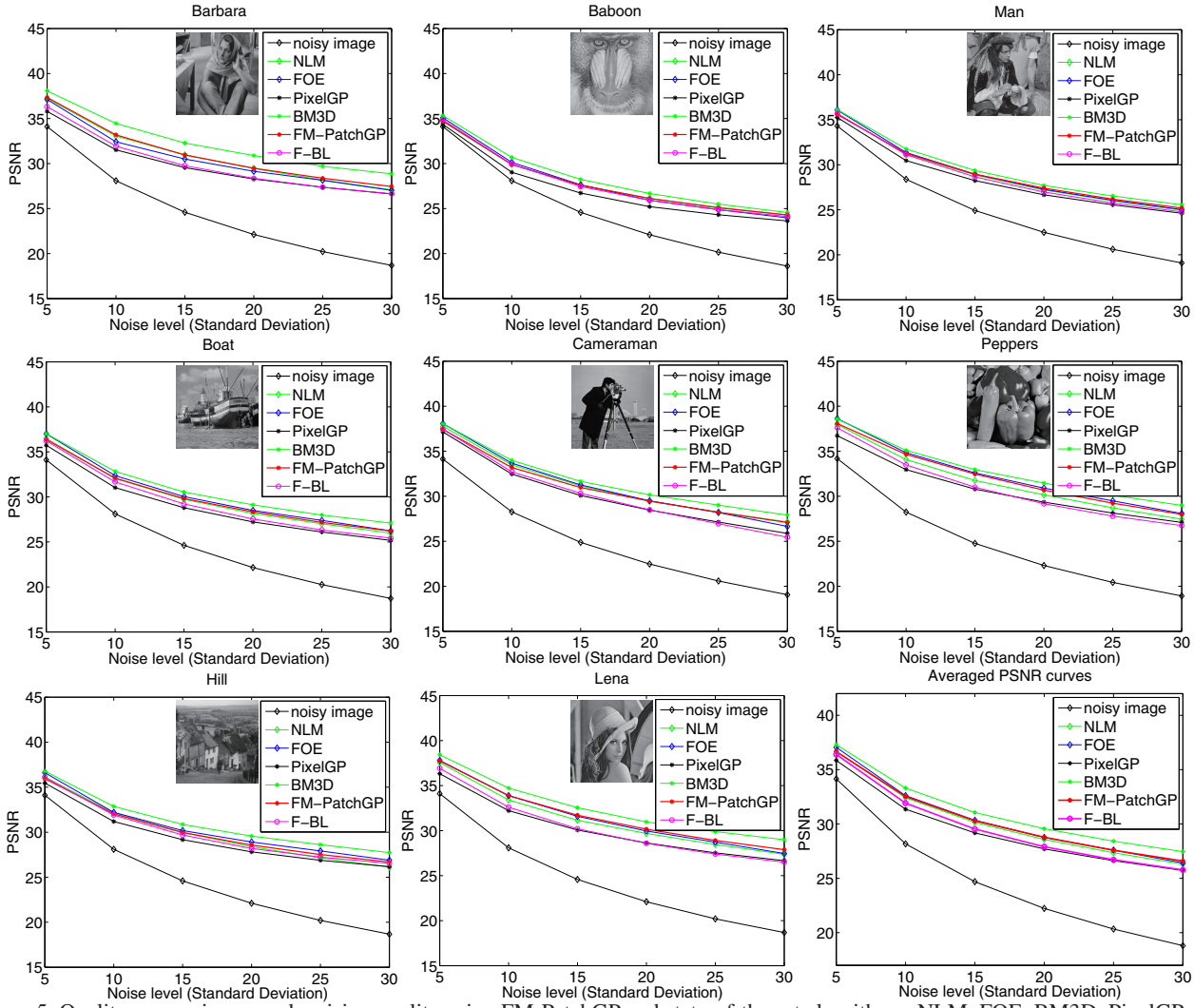

Figure 5. Quality comparisons on denoising quality using FM-PatchGP and state-of-the-art algorithms: NLM, FOE, BM3D, PixelGP, and F-BL. Gaussian noise at different levels (σ_n ranges from 5 to 30) is added 8 images.

olutions and is about 10-50% faster than F-BL. Although beyond the scope of this paper, FM-PatchGP can be further accelerated via parallel processing. Eq. (9) shows that our algorithm can independently compute MHPs at different directions. For example, we can assign a different thread to each direction and utilize CPU-related vector operators such as the Intel SSE.

Fig. 7 compares FM-PatchGP, NLM, and BM3D on a flower image of (190×190). We use 3 different noise levels (σ_n from 10 to 30) and patch size 7×7. FM-PatchGP produces competitive results to BM3D and NLM but at a much faster speed (0.03 sec. vs. 0.59 sec. vs. 1.60 sec.). Our experiments suggest that FM-PatchGP can be potentially used for real-time denoising on mobile devices with relatively low computational power.

Finally, we compare FM-PatchGP with two commercial denoising tools "Neat Image" [2] and "Noise Ninja" [3]. Both tools automatically estimate the noise profile to account for intensity-dependent noise variances [25]. We then use their identified 'uniform' regions to estimate noise variance σ_n to determine the window size and then apply FM-PatchGP with uniform weighting. Figure 8 compares the denoised results on an input noisy image (600×600) captured a Canon 400D with ISO 1600 in low light. Our FM-PatchGP result achieve comparable quality as the two commercial tools. For the computational time (on single channel), Noise Ninja takes 0.26s and FM-PatchGP takes 0.33s. Neat Image does not report the processing time although it performs at about the same speed. Notice though that FM-PatchGP is currently implemented as a single-thread program without any acceleration whereas Noise Ninja has been optimized by exploiting advanced features on the Pentium 4 and G5 CPU processors as shown in its User's Guide. The results illustrate the significant potential of FM-PatchGP. Additional comparisons can be found at `http://graphics.cis.udel.edu/denoise`.

Figure 6. Denoising results on the 'man' image with Gaussian noise σ_n =15. Our result is comparable state-of-the-art but is one to three orders of magnitude faster.

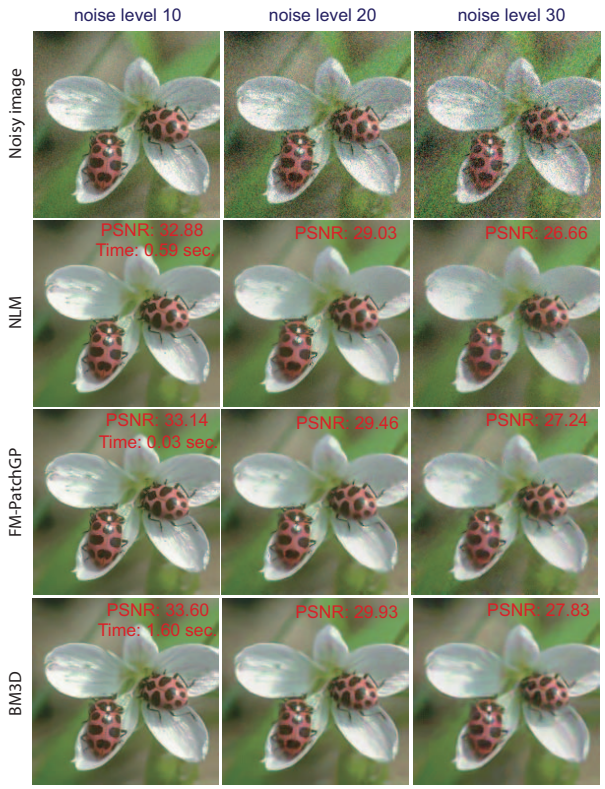

Figure 7. Quality and time comparisons between NLM, FM-PatchGP and BM3D on a high resolution image. The noise level σ_n ranges from 10 to 30. The processing time, independent of the noise level, are shown in the middle column.

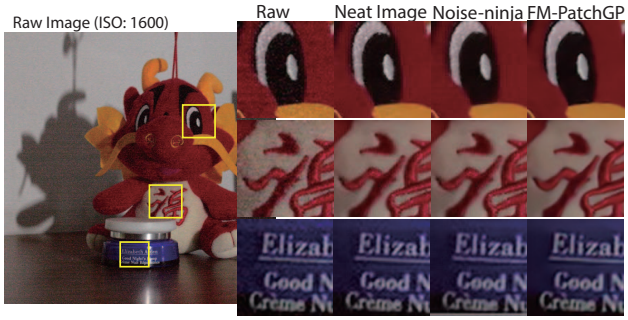

Figure 8. Comparisons between FM-PatchGP and two commercial denoising tools Noise Ninja [3] and Neat Image [2] on real images.

6. Limitations and Future work

We have presented a new patch-based image denoising algorithm based on the observation that patch geodesic paths (PatchGP) can be approximated by the minimal hop paths (MHP). Comprehensive experiments on a broad range of natural images demonstrate that our new fast multi-scale PatchGP or FM-PatchGP is comparable to or outperforms state-of-the-art algorithms in terms of quality, and is orders of magnitude faster. We plan to compare FM-PatchGP with recent patch-based techniques [20, 41].

Similar to most denoising schemes, FM-PatchGP requires using good parameters, e.g., the patch size, the window size, the discretized search directions, etc. Similar to BM3D and NLM, we usually fix the search directions and patch sizes and exhaust different window sizes. An important future direction thus is to develop automatic parameter tuning methods by exploring image statistics of natural images, e.g., the noise statistics to model the Noise Level Function (NLF) [25]. In addition, our evaluations by far have been restricted to Gaussian noise. Recent studies have shown that patch-based schemes can potentially handle Poisson noise. For example, FM-PatchGP can be used to quickly locate similar patches for conducting PCA-based denoising [40]. Finally, in our solution, we separately denoise each color channel and then combine the three channels. In the future, we plan to investigate efficient multi-channel image processing scheme as shown [37] by exploiting color correlations.

Acknowledgement. This project was partially supported by the National Science Foundation (US) under grants IIS-CAREER-0845268 and IIS-RI-1016395; NSFC China (No:61273258, 61105001, 61075012); Ph.D. Programs Foundation of Ministry of Education of China (No.20120073110018) and Committee of Science and technology, Shanghai (No:11530700200). Most of this work was done while the first author was a visiting scholar at the University of Delaware.

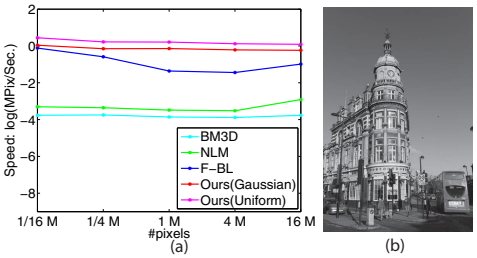

Figure 9. Processing speed comparisons between: BM3D, NLM, fast bilateral filter (F-BL) [38], and our FM-PatchGP with uniform and Gaussian weighting at different image resolutions.

References

[1] http://www.gris.informatik.tudarmstadt.de/sroth/research/foe.

[2] http://www.neatimage.com.

[3] http://www.picturecode.com.

[4] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph.*, 28:21:1–21:12, 2009.

[5] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV'07*.

[6] X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *Int. J. Comput. Vision*, 82(2):113–132, Apr. 2009.

[7] D. Barash. A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Trans. PAMI.*, 24(6):844–847, 2002.

[8] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *ICCV*, 2001.

[9] T. Brox, O. Kleinschmidt, and D. Cremers. Efficient nonlocal means for denoising of textural patterns. *IEEE Trans. on Imag. Proc.*, 17(7):1083–1092, 2008.

[10] A. Buades and B. Coll. A non-local algorithm for image denoising. In *CVPR*, 2005.

[11] P. J. Burt and E. H. Adelson. *Readings in computer vision: issues, problems, principles, and paradigms*. 1987.

[12] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. In *ECCV*, 2008.

[13] A. Criminisi, T. Sharp, C. Rother, and P. P'erez. Geodesic image and video editing. *ACM Trans. Graph.*, 29, 2010.

[14] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. 2002.

[15] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images, 2006.

[16] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., USA, 2006.

[17] J. Grazzini and P. Soille. Edge-preserving smoothing using a similarity measure in adaptive geodesic neighbourhoods. *Pattern Recogn.*, 42(10):2306–2316, 2009.

[18] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010.

[19] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, 2012.

[20] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *ECCV*, pages 112–125, 2012.

[21] V. K. Kostadin Dabov, Alessandro Foi and K. Egiazarian. Image denoising with block-matching and 3d filtering. In *Proc. SPIE 6064,606414 (2006)*, pages 454–467.

[22] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*. 2009.

[23] N. Kumar, L. Zhang, and S. Nayar. What is a good nearest neighbors algorithm for finding similar patches in images? In *ECCV*, 2008.

[24] A. Levin, B. Nadler, F. Durand, and W. T. Freeman. Patch complexity, finite pixel correlations and optimal denoising. In *ECCV*, 2012.

[25] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang. Noise estimation from a single image. In *CVPR*, 2006.

[26] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[27] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *IJCV*, 81, 2006.

[28] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. PAMI.*, 1990.

[29] G. A. Pierre Charbonnier, Laure Blanc-Feraud and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Trans. on Imag. Proc.*, 6(6), 1997.

[30] F. Porikli. Constant time O(1) bilateral filtering. In *CVPR*, 2008.

[31] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Imag. Proc.*, 2003.

[32] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *CVPR*, pages 860–867, 2005.

[33] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60:259 – 268, 1992.

[34] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[35] B. Weiss. Fast median and bilateral filtering. *ACM Trans. Graph.*, 25(3):519–526, July 2006.

[36] Y. Weiss and W. T. Freeman. What makes a good model of natural images. In *CVPR*, pages 1–8, 2007.

[37] Q. Yang. Recursive bilateral filtering. In *ECCV*, 2012.

[38] Q. Yang, K.-H. Tan, and N. Ahuja. Real-time o(1) bilateral filtering. In *CVPR*, pages 557–564, 2009.

[39] L. Yatziv, L. Yatziv, G. Sapiro, and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE Trans. on Imag. Proc.*, 15, 2006.

[40] L. Zhang, W. Dong, D. Zhang, and G. Shi. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recogn.*, 43(4):1531–1549, 2010.

[41] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.